Noname manuscript No.
(will be inserted by the editor)

Generalized Exploration in Policy Search

Herke van Hoof · Daniel Tanneberg · Jan Peters

the date of receipt and acceptance should be inserted later

Abstract To learn control policies in unknown environments, learning agents need to explore by trying actions deemed suboptimal. In prior work, such exploration is performed by either perturbing the actions at each time-step independently, or by perturbing policy parameters over an entire episode. Since both of these strategies have certain advantages, a more balanced trade-off could be beneficial. We introduce a unifying view on step-based and episode-based exploration that allows for such balanced trade-offs. This trade-off strategy can be used with various reinforcement learning algorithms. In this paper, we study this generalized exploration strategy in a policy gradient method and in relative entropy policy search. We evaluate the exploration strategy on four dynamical systems and compare the results to the established step-based and episode-based exploration strategies. Our results show, that a more balanced trade-off can yield faster learning and better final policies, and illustrate some of the effects that cause these performance differences.

Keywords Reinforcement Learning · Policy Search · Exploration

1 Introduction

Obtaining optimal behavior from experience in unknown environments is formalized in the reinforcement learning (RL) framework [43]. To learn in this manner, addressing the exploration/exploitation trade-off, that is, choosing between actions known to be good and actions that could prove to be better, is critical for improving skill performance in the long run. In fact, many reinforcement learning techniques require a non-zero

H. van Hoof
School of Computer Science, McGill University, Montreal, Canada
Intelligent Autonomous Systems Group, TU Darmstadt, Germany,
Tel.: +1-514-398-7071 ext. 00115, E-mail: herke.vanhoof@mail.mcgill.ca

D. Tanneberg
Intelligent Autonomous Systems Group, TU Darmstadt, Germany.

J. Peters
Intelligent Autonomous Systems Group, TU Darmstadt, Germany,
Max Planck Institute for Intelligent Systems. Tübingen, Germany

probability of trying each action in every state to be able to prove that the algorithm converges to the optimal policy [43].

Most tasks require agents to make a sequence of decisions over multiple time steps. Typical algorithms perform exploration by modifying the action taken at some or all of the time steps. Popular exploration heuristics include ϵ-greedy action selection (choosing a random action in a fraction ϵ of time steps), use of a stochastic controller that injects random noise at every time step, and by using a soft-max (or Boltzmann) distribution that selects actions that are deemed better more often, but not exclusively [19, 8, 20]. Another strategy is the use of parametrized controllers with a distribution over actions or parameters, and sampling from this distribution at every time step [9].

However, the paradigm of modifying actions at individual time-steps has multiple shortcomings. High-frequency exploration can show inefficient 'thrashing' behavior [40, 31, 1] and in the worst case exhibit a random walk behavior that fails to explore much of the state space [21]. At the same time, for longer horizons, the variance of policy roll-outs explodes as the results depend on an increasing number of independent decisions [28]. Furthermore, when learning controllers within a certain function class, perturbing single time-steps can result in trajectories that are not reproducible by any noise-free controller in that function class [8].

Skill learning in robotics and other physical systems is a prominent application domain for reinforcement learning. In this domain, reinforcement learning offers a strategy for acquiring skills when, for example, parts of the robot or parts of the environment cannot be modeled precisely in advance [19, 20]. High-frequency exploration can cause additional problems when applied to robot systems. Namely, high-frequency exploration causes high jerks, that can damage robots [26, 8, 21, 49]. Furthermore, real robots exhibit non-Markov effects such as dead-band, hysteresis, stiction, and delays due to processing and communication delays and inertia [20]. These effects make it hard to precisely measure the effects of the perturbations. Such problems could be addressed by including a history of actions in the state-space, but this would make the dimensionality of the reinforcement learning problem larger and thereby increase the complexity of the problem exponentially [20].

In this paper, we focus on addressing these problems in policy search methods employing parametrized controllers. Such methods, that are popular in e.g. robotics applications, tend to yield stable updates that result in safe robot behavior [20, 8]. Parametrized policies are also easily applicable in environments with continuous state-action spaces. In these methods, perturbing individual actions can be realized by perturbing the policy parameters in each time step independently. We will refer to this strategy as *time-step-based exploration*.

The problems of high-frequency exploration in policy search methods can be addressed by exploiting that data for learning tasks through reinforcement learning is usually gathered in multiple roll-outs or episodes. One roll-out is a sequence of state-action pairs, that is ended when a terminal state is reached or a certain number of actions have been performed. One can thus perturb the controller parameters at the beginning of a policy roll-out, and leave it fixed until the episode has ended [36, 38, 21, 45, 42].

The advantage of this *episode-based exploration* approach is that random-walk behavior and high jerks are avoided due to the *coherence* of the exploration behavior. The disadvantage, however, is that in each episode, only one set of parameters can be evaluated. Therefore, such techniques usually require more roll-outs to be performed, which can be time-consuming on a robotic system.

We think of the time-step-based and episode-based exploration strategies as two extremes, with space for many different intermediate trade-offs. In this paper, we *provide a unifying view* on time-step-based and episode-based exploration and *propose intermediate trade-offs* that slowly vary the controller parameters during an episode, rather than independent sampling or keeping the parameters constant. Formally, we will sample parameters at each time step in a manner that depends on the previous parameters, thereby defining a Markov chain in parameter space. Our experiments compare such intermediate trade-offs to existing step-based and episode-based methods.

In the rest of this section, we will describe related work, and after that describe our unified view on time-step-based and episode-based exploration and our problem statement. Then, in the subsequent sections, we describe our approach for generalized exploration formally and provide the details of the set-up and results of our experiments. We conclude with a discussion of the results and future work.

1.1 Related Work

Numerous prior studies have addressed the topic of temporal coherence in reinforcement learning, although most have not considered finding trade-offs between fully temporally correlated and fully independent exploration. In this section, we will first discuss temporal coherence through the use of options and macro-actions. Then, the possibility of temporal coherence through the use of parametrized controllers such as central pattern generators and movement primitives is discussed. Considering that different forms of sampling parameters are central to the difference between step-based and episode-based methods, we will conclude by discussing other approaches for sampling exploratory actions or policies.

1.1.1 Temporal Coherence through Options

Hierarchical reinforcement learning has been proposed to scale reinforcement learning to larger domains, especially where common subtasks are important [18, 39]. These early studies allowed choosing higher-level actions at every time step, and are thus time-step based strategies. Later approaches tended to have a higher-level policy which select a lower-level policy that takes control for a number of time steps, for example, until the lower-level policy reaches a specific state, or when a certain number of time steps has passed [34, 32, 10, 44]. Choosing such a lower-level policy to be executed for multiple time steps makes the subsequent exploration decisions highly correlated. In addition to choosing which lower-level policy to execute, coherent explorative behavior can also be obtained by stochastic instantiation of the parameters of lower-level policies [47]. Moreover, this hierarchical framework allows learning to scale up to larger domains efficiently [44, 18, 32, 10]. In such a hierarchical framework, the temporal coherence of exploration behavior contributes to this success by requiring fewer correct subsequent decisions for reaching a desired, but faraway, part of the state space [44].

Much of this work has considered discrete Markov decision processes (MDPs), and does not naturally extend to robotic settings. Other work has focused on continuous state-action spaces. For example, Morimoto and Doya [27] study an upper level policy that sets sub-goals that provide a reward for lower-level policies. This method was used to learn a stand-up behavior for a three-link robot. A similar set-up was used in [12], where the agent could choose between setting a sub-goal and executing a primitive

action. Local policies are often easier to learn than global policies. This insight was used in [23] in an option discovery framework, where a chain of sub-policies is build such that each sub-policy terminates in an area where its successor can be initiated. Another option discovery method is described in [5], where probabilistic inference is used to find reward-maximizing options for, among others, a pendulum swing-up task.

1.1.2 Episode Based Exploration and Pattern Generators

The option framework is a powerful approach for temporally correlated exploration in hierarchical domains. However, option-based methods usually require the options to be pre-defined, require additional information such as the goal location, demonstrations, or knowledge of the transition dynamics, or are intrinsically linked to specific RL approaches. Another approach to obtaining coherent exploration employs parametrized controllers, where the parameters are fixed for an entire episode. Such an approach is commonly used with pattern generators such as motion primitives.

Such episode-based exploration has been advocated in a robotics context in previous work. For example, [36, 38] describe a policy gradient method that explores by sampling parameters in the beginning of each episode. This method is shown to outperform similar policy gradient methods which use independent Gaussian noise at each time step for exploration. One of the proposed reasons for this effect is that, in policy gradient methods, the variance of gradient estimates increases linearly with the length of the history considered [28]. Similarly, the PoWER method that uses episode-based exploration [21] outperforms a baseline that uses independent additive noise at each time step. Furthermore, path-integral based methods have been shown to benefit from parameter-based exploration [45, 42], with episode-based exploration conjectured to produce more reliable updates [42].

Episode-based exploration has been shown to have very good results where policies have a structure that fits the task. For example, in [22], a task-specific parametrized policy was learned for quadrupedal locomotion using a policy gradient method. Dynamic movement primitives have proven to be a popular policy parametrization for a wide variety of robot skills [37]. For example, reaching, ball-in-a-cup, under actuated swing-up and many other tasks have been learned in this manner [21, 37, 20]. In case different initial situations require different controllers, a policy can be found that maps initial state features to controller parameters [3, 4].

However, episode-based exploration also has disadvantages. Notably, in every roll-out only a single set of parameters can be evaluated. Compared to independent per-step exploration, many more roll-outs might need to be performed. Performing such roll-outs can be time-consuming and wear out the mechanisms of the robot. One solution would be to keep exploration fixed for a number of time steps, but then choose different exploration parameters. Such an approach was proposed in [28]. A similar effect can be reached by sequencing the execution of parametrized skills, as demonstrated in [41, 4]. However, suddenly switching exploration parameters might again cause undesired high wear and tear in robot systems [26]. Instead, slowly varying the exploration parameters is a promising strategy. Such a strategy is touched upon in [8], but has remained largely unexplored so far.

1.1.3 Sampling for Reinforcement Learning

In this paper, we propose building a Markov chain in parameter space to obtain coherent exploration behavior. Earlier work has used Markov chain Monte Carlo (MCMC) methods for reinforcement learning, but usually in a substantially different context. For example, several papers focus on sampling models or value functions. In case models are sampled, actions are typically generated by computing the optimal action with respect to the sampled model [1, 40, 29, 6, 11]. By preserving the sampled model for multiple time steps or an entire roll-out, coherent exploration is obtained [40, 1]. Such methods cannot be applied if the model class is unknown. Instead, samples can be generated from a distribution over value functions [52, 7, 31] or Q functions [30]. Again, preserving the sample over an episode avoids dithering by making exploration coherent for multiple time-steps [31]. Furthermore, [31] proposes a variant where the value function is not kept constant, but is allowed to vary slowly over time.

Instead, in this paper, we propose sampling policies from a learned distribution. Earlier work has used MCMC principles to build a chain of policies. This category of work includes [17] and [24], who use the estimated value of policies as re-weighting of the parameter distribution, [51], where structured policies are learned so that experience in one state can shape the prior for other states, and [48], where a parallel between such MCMC methods and genetic algorithms is explored. In those works, every policy is evaluated in an episode-based manner, whereas we want an algorithm that is able to explore during the course of an episode.

Such a method that explores during the course of an episode was considered in [13], where a change to a single element of a tabular deterministic policy is proposed at every time-step. However, this algorithm does not consider stochastic or continuous policies that are needed in continuous-state, continuous-action MDPs.

The work that is most closely related to our approach, is the use of auto-correlated Gaussian noise during exploration. This type of exploration was considered for learning robot tasks in [49, 27]. In a similar manner, Ornstein-Uhlenbeck processes can be used to generate policy pertubations [25, 16]. However, in contrast to the method we propose, these approaches perturb the actions themselves instead of the underlying parameters, and can therefore generate actions sequences that cannot be followed by the noise-free parametric policy.

1.2 Notation in Reinforcement Learning and Policy Search

Reinforcement-learning problems can be formalized as Markov decision processes. A Markov decision process is defined by a set of states $\mathcal{S}$, a set of actions $\mathcal{A}$, the probability $p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a})$ that executing action $\mathbf{a}$ in state $\mathbf{s}_t$ will result in state $\mathbf{s}_{t+1}$ at the next time step, and a reward function $r(\mathbf{s}, \mathbf{a})$. The time index t here denotes the time step within an episode. In our work, we will investigate the efficacy of our methods in various dynamical systems with continuous state and action spaces, $\mathbf{s}_t \in \mathcal{S} \subset \mathbb{R}^{D_s}$ and $\mathbf{a}_t \in \mathcal{A} \subset \mathbb{R}^{D_a}$, where D_s and D_a are the dimensionality of the state and action space, respectively. Also, the transition distribution $p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a})$ is given by the physics of the system, and will thus generally be a delta distribution.

Our work focuses on policy search methods to find optimal controllers for such systems. In policy search methods, the policy is explicitly represented. Often, this policy is parametrized by a parameter vector $\boldsymbol{\theta}$. The policy can be deterministic or

stochastic given these parameters. Deterministic policies will be denoted as a function $\mathbf{a} = \pi(\mathbf{s}; \boldsymbol{\theta})$, whereas stochastic policies will be denoted as a conditional distribution $\pi(\mathbf{a}|\mathbf{s}; \boldsymbol{\theta})$.

1.3 Unifying View on Step- and Episode-based Exploration

In this paper, we will look at parameter-exploring policy search methods. Existing methods in this category have almost exclusively performed exploration by either performing exploration at the episode level or performing exploration at the step-based level. A unifying view on such methods is, that we have a (potentially temporally coherent) policy of the form

$$\mathbf{a}_t = \pi(\mathbf{s}_t; \boldsymbol{\theta}_t) \tag{1}$$

$$\boldsymbol{\theta}_t \sim \begin{cases} p_0(\cdot) & \text{if } t = 0 \\ g(\cdot|\boldsymbol{\theta}_{t-1}) & \text{otherwise,} \end{cases} \tag{2}$$

where $\boldsymbol{\theta}_t$ is the vector of parameters at time t, $\mathbf{a}_t$ is the corresponding action taken in state $\mathbf{s}_t$, π is a policy conditioned on the parameters. Furthermore, p_0 is the distribution over parameters that is drawn from at the beginning of each episode, and $g(\cdot|\boldsymbol{\theta}_t)$ the conditional distribution over parameters at every time step thereafter. The familiar step-based exploration algorithms correspond to the specific case where $g(\boldsymbol{\theta}_t|\boldsymbol{\theta}_{t-1}) = p_0(\boldsymbol{\theta}_t)$, such that $\boldsymbol{\theta}_t \perp \boldsymbol{\theta}_{t-1}$. Episode-based exploration is another extreme case, where $g(\boldsymbol{\theta}_t|\boldsymbol{\theta}_{t-1}) = \delta(\boldsymbol{\theta}_t - \boldsymbol{\theta}_{t-1})$, where δ is the Dirac delta, such that $\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1}$. Note, that in both cases

$$\forall t : \int p(\boldsymbol{\Theta}_t = \boldsymbol{\theta}|\boldsymbol{\Theta}_0 = \boldsymbol{\theta}')p_0(\boldsymbol{\Theta}_0 = \boldsymbol{\theta}')d\boldsymbol{\theta}' = p_0(\boldsymbol{\Theta}_0 = \boldsymbol{\theta}), \tag{3}$$

where $\boldsymbol{\Theta}$ is used to explicitly indicate random variables[1]. That is, the marginal distribution is equal to the desired sampling distribution p_0 regardless of the time step. Besides these extreme choices of $g(\cdot|\boldsymbol{\theta}_{t-1})$, many other exploration schemes are conceivable. Specifically, in this paper we address choosing $g(\boldsymbol{\theta}_t|\boldsymbol{\theta}_{t-1})$ such that the $\boldsymbol{\theta}_t$ is neither independent of nor equal to $\boldsymbol{\theta}_{t-1}$ and Eq. (3) is satisfied. Our reason for enforcing Eq. (3) is that in time-invariant system, the resulting time-invariant distributions over policy parameters are suitable.

2 Generalizing Exploration

Equation (2) defines a Markov chain on the policy parameters. To satisfy Eq. (3), p_0 should be a stationary distribution of this chain. A sufficient condition for this property to hold, is that detailed balance is satisfied [15]. Detailed balance holds, if

$$\frac{p_0(\boldsymbol{\Theta}_0 = \boldsymbol{\theta})}{p_0(\boldsymbol{\Theta}_0 = \boldsymbol{\theta}')} = \frac{g(\boldsymbol{\Theta}_{t+1} = \boldsymbol{\theta}|\boldsymbol{\Theta}_t = \boldsymbol{\theta}')}{g(\boldsymbol{\Theta_{t+1}} = \boldsymbol{\theta}'|\boldsymbol{\Theta}_t = \boldsymbol{\theta})}. \tag{4}$$

[1] Following common practice, where the random variable is clear from the context, we will not explicitly mention it, writing $p_0(\boldsymbol{\theta}_0)$ for $p_0(\boldsymbol{\Theta}_0 = \boldsymbol{\theta}_0)$, for example.

Given a Gaussian policy[2] $p_0 = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Lambda}^{-1})$, this constraint can easily be satisfied[3]. For example, a reasonable proposal distribution could be obtained by taking a weighted average of the parameters $\boldsymbol{\theta}_t$ at the current time step and a sample from a Gaussian centered on $\boldsymbol{\mu}$. Since averaging lowers the variance, this Gaussian will need to have a larger variance than $\boldsymbol{\Lambda}^{-1}$. As such, we consider a proposal distribution of the form

$$\boldsymbol{\theta}_{t+1} = \beta\tilde{\boldsymbol{\theta}} + (1-\beta)\boldsymbol{\theta}_t, \quad \tilde{\boldsymbol{\theta}} \sim N(\boldsymbol{\mu}, f(\beta)^2\boldsymbol{\Lambda}^{-1}), \tag{5}$$

where β is the weighting of the average and $f(\beta)$ governs the additional scaling of the covariance. This scaling needs to be set such that the detailed balance criterion in Eq. (4) is satisfied. The detailed balance criterion can most easily be verified by comparing the logarithms of the left- and right hand side of Eq. (4). For the left hand side, we obtain the simple expression

$$\log\left(\frac{p_0(\boldsymbol{\theta})}{p_0(\boldsymbol{\theta'})}\right) = -\frac{\boldsymbol{\theta}^T\boldsymbol{\Lambda}\boldsymbol{\theta}}{2} + \boldsymbol{\theta}^T\boldsymbol{\Lambda}\boldsymbol{\mu} + \frac{\boldsymbol{\theta'}^T\boldsymbol{\Lambda}\boldsymbol{\theta'}}{2} - \boldsymbol{\theta'}^T\boldsymbol{\Lambda}\boldsymbol{\mu}. \tag{6}$$

For the right hand side of Eq. (4), we can insert $g(\boldsymbol{\theta'}|\boldsymbol{\theta}) = N((1-\beta)\boldsymbol{\theta} + \beta\boldsymbol{\mu}, \tilde{\boldsymbol{\Lambda}}^{-1})$, with $\tilde{\boldsymbol{\Lambda}} = f(\beta)^{-2}\beta^{-2}\boldsymbol{\Lambda}$, and vice versa for $g(\boldsymbol{\theta}|\boldsymbol{\theta'})$. The resulting log-ratio is given as

$$\begin{aligned}
\log\left(\frac{g(\boldsymbol{\theta}|\boldsymbol{\theta'})}{g(\boldsymbol{\theta'}|\boldsymbol{\theta})}\right) =& -\frac{\boldsymbol{\theta}^T\tilde{\boldsymbol{\Lambda}}\boldsymbol{\theta}}{2} - (1-\beta)\beta\boldsymbol{\theta'}^T\tilde{\boldsymbol{\Lambda}}\boldsymbol{\mu} - \frac{(1-\beta)^2\boldsymbol{\theta'}^T\tilde{\boldsymbol{\Lambda}}\boldsymbol{\theta'}}{2} + \beta\boldsymbol{\theta}^T\tilde{\boldsymbol{\Lambda}}\boldsymbol{\mu} \\
&+ \frac{\boldsymbol{\theta'}^T\tilde{\boldsymbol{\Lambda}}\boldsymbol{\theta'}}{2} + (1-\beta)\beta\boldsymbol{\theta}^T\tilde{\boldsymbol{\Lambda}}\boldsymbol{\mu} + \frac{(1-\beta)^2\boldsymbol{\theta}^T\tilde{\boldsymbol{\Lambda}}\boldsymbol{\theta}}{2} - \beta\boldsymbol{\theta'}^T\tilde{\boldsymbol{\Lambda}}\mu \\
=& \left(2\beta-\beta^2\right)\left(-\tfrac{1}{2}\boldsymbol{\theta}^T\tilde{\boldsymbol{\Lambda}}\boldsymbol{\theta} + \boldsymbol{\theta}^T\tilde{\boldsymbol{\Lambda}}\boldsymbol{\mu} + \tfrac{1}{2}\boldsymbol{\theta'}^T\tilde{\boldsymbol{\Lambda}}\boldsymbol{\theta'} - \boldsymbol{\theta'}^T\tilde{\boldsymbol{\Lambda}}\boldsymbol{\mu}\right) \\
=& \frac{2\beta-\beta^2}{f(\beta)^2\beta^2}\log\left(\frac{p_0(\boldsymbol{\theta})}{p_0(\boldsymbol{\theta'})}\right),
\end{aligned}$$

where we inserted Eq. (6) in the last line. Now, we can identify, that for

$$f(\beta)^2 = (2\beta-\beta^2)/\beta^2 = 2/\beta - 1,$$

detailed balance is satisfied. Thus, $\tilde{\boldsymbol{\Lambda}}^{-1} = (2\beta-\beta^2)\boldsymbol{\Lambda}^{-1}$.

In principle, such generalized exploration can be used with different kinds of policy search methods. However, integrating coherent exploration might require minor changes in the algorithm implementation. In the following two sections, we will consider two types of methods: policy gradient methods and relative entropy policy search.

2.1 Generalized Exploration for Policy Gradients

In policy gradient methods, as the name implies, the policy parameters are updated by a step in the direction of the estimated gradient of the expected return over T time

[2] Such Gaussian policies are a typical choice for policy search methods [8], and have been used in diverse approaches such as parameter-exploring policy gradients [36], CMA-ES [14], PoWER [21], PI2 [45], and REPS [33].

[3] In more general systems, the Metropolis-Hastings acceptance ratio [15] could be used to satisfy Eq. (4).

steps $J_{\boldsymbol{\mu}} = \mathbb{E}\left[\sum_{t=0}^{T-1} r(\mathbf{s}_t, \mathbf{a}_t)\right]$ with respect to the meta-parameters $\boldsymbol{\mu}$ that govern the *distribution* over the policy parameters $\boldsymbol{\theta} \sim p_0 = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Lambda}^{-1})$. Formally,

$$\boldsymbol{\mu}_{k+1} = \boldsymbol{\mu}_k + \alpha \nabla_{\boldsymbol{\mu}} J_{\boldsymbol{\mu}},$$

where α is a user-specified learning rate [50]. The gradient $\nabla_{\boldsymbol{\mu}} J_{\boldsymbol{\mu}}$ can be determined from the gradient of the log-policy [50, 2]

$$\nabla_{\boldsymbol{\mu}} J_{\boldsymbol{\mu}} = \mathbb{E}\left[\sum_{t=0}^{T-1} \nabla_{\boldsymbol{\mu}} \log \pi_{\boldsymbol{\mu}} \left(\mathbf{a}_0, \ldots, \mathbf{a}_t | \mathbf{s}_0, \ldots, \mathbf{s}_t\right) \left(r_t - b_t\right)\right],$$

considering that the action can depend on the previous actions when using the generalized exploration algorithm. In this equation, b is a baseline that can be chosen to reduce the variance. Here, we will use the form of policy proposed in Eqs. (1)-(2). In this case, the conditional probability of a sequence of actions is given by

$$\pi_{\boldsymbol{\mu}} \left(\mathbf{a}_0, \ldots, \mathbf{a}_t | \mathbf{s}_0, \ldots, \mathbf{s}_t\right) = \mathbb{E}_{\boldsymbol{\theta}_0 \ldots \boldsymbol{\theta}_t}\left[\prod_{j=0}^{t} \pi\left(\mathbf{a}_j | \mathbf{s}_j; \boldsymbol{\theta}_j\right)\right], \tag{7}$$

$$p\left(\boldsymbol{\theta}_0, \ldots, \boldsymbol{\theta}_t\right) = p_0(\boldsymbol{\theta}_0; \boldsymbol{\mu}, \boldsymbol{\Lambda}^{-1}) \prod_{j=1}^{t} g\left(\boldsymbol{\theta}_j | \boldsymbol{\theta}_{j-1}; \boldsymbol{\mu}, \boldsymbol{\Lambda}^{-1}\right).$$

If $\beta = 1$, $p(\boldsymbol{\theta}_t | \boldsymbol{\theta}_{t-1}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) = p_0(\boldsymbol{\theta}_t; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ and Eq. (7) can be written as

$$\pi_{\boldsymbol{\mu}} \left(\mathbf{a}_0, \ldots, \mathbf{a}_t | \mathbf{s}_0, \ldots, \mathbf{s}_t\right) = \prod_{j=0}^{t} \tilde{\pi}\left(\mathbf{a}_j | \mathbf{s}_j; \boldsymbol{\mu}, \boldsymbol{\Lambda}^{-1}\right),$$

$$\text{with } \tilde{\pi}\left(\mathbf{a}_j | \mathbf{s}_j; \boldsymbol{\mu}, \boldsymbol{\Lambda}^{-1}\right) = \int p_0\left(\boldsymbol{\theta}_0; \boldsymbol{\mu}, \boldsymbol{\Lambda}^{-1}\right) \pi\left(\mathbf{a}_j | \mathbf{s}_j; \boldsymbol{\theta}\right) d\boldsymbol{\theta}.$$

When $\beta = 1$, this identity makes the gradient of the log-policy equal to the gradient of $\tilde{\pi}$ computed using G(PO)MDP [2]. In our paper, we will focus on learning the mean $\boldsymbol{\mu}$ of a distribution over parameters of a linear Gaussian policy: $\mathbf{a} = \mathbf{s}^T \boldsymbol{\theta}$ with $\boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Lambda}^{-1})$. In that case, the required gradient is given by

$$\nabla_{\boldsymbol{\mu}} \log \pi_{\boldsymbol{\mu}} \left(\mathbf{a}_0, \ldots, \mathbf{a}_t | \mathbf{s}_0, \ldots, \mathbf{s}_t\right) = \left[\mathbf{s}_0 \ldots \mathbf{s}_t\right]^T \boldsymbol{\Sigma}^{-1} \left(\left[\mathbf{a}_0 \ldots \mathbf{a}_t\right]^T - \left[\mathbf{s}_0 \ldots \mathbf{s}_t\right]^T \boldsymbol{\mu}\right),$$

where the elements of the covariance matrix $\boldsymbol{\Sigma}$ over correlated sequences of actions are given by

$$\boldsymbol{\Sigma}_{jk} = {\mathbf{s}_j}^T \boldsymbol{\Lambda}^{-1} \mathbf{s}_k (1 - \beta)^{|j-k|}.$$

However, when the coherency $\beta = 0$ and t is larger than D_{s} (the dimensionality of $\mathbf{s}$), $\boldsymbol{\Sigma}$ is not invertible. Instead, the gradient of Eq. (7) can be computed as

$$\nabla_{\boldsymbol{\mu}} \log \pi_{\boldsymbol{\mu}} \left(\mathbf{a}_0, \ldots, \mathbf{a}_t | \mathbf{s}_0, \ldots, \mathbf{s}_t\right) = \boldsymbol{\Lambda}^{-1} (\boldsymbol{\theta}_0 - \boldsymbol{\mu}),$$

making the algorithm equivalent to PEPG [38] on the distribution over policy parameters $\mathcal{N}(\boldsymbol{\theta}; \boldsymbol{\mu}, \boldsymbol{\Lambda}^{-1})$. Setting $0 < \beta < 1$ yields intermediate strategies that trade off the advantages of G(PO)MDP and PEPG, i.e., of step-based and episode-based exploration.

2.2 Generalized Exploration for Relative Entropy Policy Search

In relative entropy policy search (REPS), the goal is to take larger steps than policy gradient methods while staying close to the previous sampling policy in information-theoretic terms [33, 46]. This objective is reached by solving the optimization problem

$$\max_{\pi,\mu_\pi} \quad \iint_{\mathcal{S}\times\mathcal{A}} \pi(\mathbf{a}|\mathbf{s})\mu_\pi(\mathbf{s})r(\mathbf{s},\mathbf{a})d\mathbf{a}d\mathbf{s}, \tag{8}$$

$$\text{s. t.} \quad \iint_{\mathcal{S}\times\mathcal{A}} \pi(\mathbf{a}|\mathbf{s})\mu_\pi(\mathbf{s})d\mathbf{a}d\mathbf{s} = 1, \tag{9}$$

$$\forall s' \iint_{\mathcal{S}\times\mathcal{A}} \pi(\mathbf{a}|\mathbf{s})\mu_\pi(\mathbf{s})p(\mathbf{s}'|\mathbf{s}\mathbf{a})d\mathbf{a}d\mathbf{s} = \mu_\pi(\mathbf{s}'), \tag{10}$$

$$\mathrm{KL}(\pi(\mathbf{a}|\mathbf{s})\mu_\pi(\mathbf{s})||q(\mathbf{s},\mathbf{a})) \leq \epsilon, \tag{11}$$

where $\mu_\pi(\mathbf{s})$ is the steady-state distribution under $\pi(\mathbf{a}|\mathbf{s})$, as enforced by Eq. (10), and $\pi(\mathbf{a}|\mathbf{s})\mu_\pi(\mathbf{s})$ is the reward-maximizing distribution as specified by Eqs. (8-9). Equation (11) specifies the additional information-theoretic constraints, where q is a reference distribution (e.g., the previous sampling distribution), and KL denotes the Kullback-Leibler divergence [33].

Earlier work [33] detailed how to derive the solution to the optimization problem in Eqs. (8-11). Here, we will just give a brief overview of the solution strategy. The optimization problem is first approximated by replacing the steady-state constraint[4] in Eq. (9) by

$$\iint_{\mathcal{S}\times\mathcal{S}\times\mathcal{A}} \pi(\mathbf{a}|\mathbf{s})\mu_\pi(\mathbf{s})p(\mathbf{s}'|\mathbf{s},\mathbf{a})\boldsymbol{\phi}(\mathbf{s}')d\mathbf{a}d\mathbf{s}d\mathbf{s}' = \int_{\mathcal{S}} \mu_\pi(\mathbf{s}')\boldsymbol{\phi}(\mathbf{s}')d\mathbf{s}', \tag{12}$$

using features $\boldsymbol{\phi}$ of the state. Furthermore, the expected values in Eqs. (8-11) are approximated by sample averages. Since we will look at deterministic dynamical systems[5], the expected features under the transition distribution $p(\mathbf{s}'|\mathbf{s},\mathbf{a})$ are simply given by the subsequent state in the roll-out [33]. Subsequently, Lagrangian optimization is used to find the solution to the approximated optimization problem, which is takes the form of a re-weighting $w(\mathbf{s},\mathbf{a})$ of the reference distribution q, with $\pi(\mathbf{a}|\mathbf{s})\mu_\pi(\mathbf{s}) = w(\mathbf{s},\mathbf{a})q(\mathbf{s},\mathbf{a})$, as derived in detail in [33].

The re-weighting coefficients $w(\mathbf{s},\mathbf{a})$ can only be calculated at sampled state-action pairs $(\mathbf{s},\mathbf{a})$. To find a generalizing policy that is defined at all states, the sample-based policy can be generalized by optimizing a maximum likelihood objective

$$\arg\max_{\boldsymbol{\mu},\mathbf{D}} \mathbb{E}_{\pi,\mu_\pi} \log p\left(\mathbf{a}_{1:N}\,|\mathbf{s}_{1:N};\boldsymbol{\mu},\mathbf{D},\sigma\right) \tag{13}$$

where $\mathbf{s}_{1:N}$ and $\mathbf{a}_{1:N}$ are the sequences of states and actions encountered during one episode. The hyper-parameters, consisting of $\boldsymbol{\mu}$ and the entries of diagonal covariance matrix $\mathbf{D}$, govern a distribution $p(\boldsymbol{\theta}|\boldsymbol{\mu},\mathbf{D})$ over policy parameters $\boldsymbol{\theta}$ for policies of the form $\mathbf{a} = \mathbf{f}(\mathbf{s})^T\boldsymbol{\theta}$, where $\mathbf{f}(\mathbf{s})$ are features of the state[6]. Earlier work has focused on

[4] This approximation is equivalent to approximating the function-valued Lagrangian multiplier for the continuum of constraints in Eq. (10) by a function linear in the features ϕ [46].

[5] For stochastic systems, a learned trasition model could be used [46].

[6] Here, we work with the diagonal policy covariance $\mathbf{D}$ rather than policy precision $\boldsymbol{\Lambda}$ used in the policy gradient section. The notational difference serves to stress an important difference: we will optimize over the entries of $\mathbf{D}$ rather than fixing the covariance matrix to a set value.

the case where actions during an episode are chosen independently of each other [33, 46]. However, with coherent exploration, policy parameters are similar in subsequent time-steps and, thus, this assumption is violated. Here, instead we define the likelihood terms as

$$p\left(\mathbf{a}_{1:N}\,|\mathbf{s}_{1:N};\boldsymbol{\mu},\mathbf{D},\sigma\right) = \int_{\Theta^N} p(\mathbf{a}_{1:N}|\mathbf{s}_{1:N},\boldsymbol{\theta}_{1:N},\sigma)p(\boldsymbol{\theta}_{1:N}|\boldsymbol{\mu},\mathbf{D})d\boldsymbol{\theta}_{1:N}, \tag{14}$$

with

$$p\left(\mathbf{a}_{1:N}\,|\mathbf{s}_{1:N},\boldsymbol{\theta}_{1:N},\sigma\right) = \mathcal{N}\left(\mathbf{f}\left(\mathbf{s}_{1:N}\right)^T\boldsymbol{\theta}_{1:N},\sigma^2\mathbf{I}\right), \tag{15}$$

where $\boldsymbol{\theta}_{1:N}$ denotes a sequence of parameters and σ is a regularization term that can be understood as assumed noise on the observation. Under the proposal distribution of Eq. (5), the distribution over these parameters is given by

$$p(\boldsymbol{\theta}_{1:N}|\boldsymbol{\mu},\mathbf{D}) = p(\boldsymbol{\theta}_1|\,\boldsymbol{\mu},\mathbf{D})\prod_{j=1}^{N} g\big(\boldsymbol{\theta}_j\big|\,\boldsymbol{\theta}_{j-1},\boldsymbol{\mu},\mathbf{D}\big) = \mathcal{N}(\boldsymbol{\theta}_{1:N}|\,\tilde{\boldsymbol{\mu}},\mathbf{D}\otimes\mathbf{E}). \tag{16}$$

In this equation, $\tilde{\boldsymbol{\mu}} = [\boldsymbol{\mu}^T,\ldots,\boldsymbol{\mu}^T]^T$, $[\mathbf{E}]_{jk} = (1-\beta)^{|j-k|}$, and $\otimes$ denotes the Kronecker product. Inserting (15) and (16) into (14) yields the equation

$$p\left(\mathbf{a}_{1:N}\,|\mathbf{s}_{1:N},\boldsymbol{\theta}_{1:N},\sigma\right) = \mathcal{N}\left(\mathbf{a}_{1:N};\mathbf{f}(\mathbf{s}_{1:N})^T\boldsymbol{\mu},\boldsymbol{\Sigma}+\sigma^2 I\right), \tag{17}$$

where the elements of the covariance matrix $\boldsymbol{\Sigma}$ over correlated sequences of actions are given by

$$\boldsymbol{\Sigma}_{jk} = \mathbf{f}(\mathbf{s}_j)^T\mathbf{D}\,\mathbf{f}(\mathbf{s}_k)(1-\beta)^{|j-k|}.$$

We could approximate the expectation in Eq. (13) using samples $(\mathbf{a},\mathbf{s}) \sim \pi(\mathbf{a}|\mathbf{s})\mu_\pi(\mathbf{s})$, however, we have samples $(\mathbf{a},\mathbf{s}) \sim q$ from the sampling distribution and re-weighting factors $w_j = p(a_j,s_j)/q(a_j,s_j)$. We can thus use importance weighting, meaning that we maximize the weighted log-likelihood

$$\sum_{i=1}^{M}\sum_{j=1}^{N} w_j^{(i)}\log\mathcal{N}\left(\mathbf{a}_j^{(i)};\mathbf{f}\left(\mathbf{s}_{1:N}^{(i)}\right)^T\boldsymbol{\mu},\boldsymbol{\Sigma}_{jj}+\sigma^2\right),$$

where we sum over M rollouts with each N time steps. Weighting the samples is, up to a proportionality constant, equivalent to scaling the covariance matrix. Since $\boldsymbol{\Sigma}_{jk} = \rho_{jk}\sqrt{\boldsymbol{\Sigma}_{jj}\boldsymbol{\Sigma}_{kk}}$, where ρ_{jk} is the correlation coefficient, re-scaling $\boldsymbol{\Sigma}_{jj}$ by $1/w_j$ means that $\boldsymbol{\Sigma}_{jk}$ has to be scaled by $1/\sqrt{w_j}$ accordingly, such that we define

$$\tilde{\boldsymbol{\Sigma}}_{jk}^{(i)} = \mathbf{f}(\mathbf{s}_j)^T\mathbf{D}\,\mathbf{f}(\mathbf{s}_k)(1-\beta)^{|j-k|}\left(w_j^{(i)}w_k^{(i)}\right)^{-\frac{1}{2}}.$$

We can now solve $\arg\max_{\boldsymbol{\mu}}\prod_i L_i$ in closed form, yielding

$$\boldsymbol{\mu}^* = \left(\sum_{i=1}^{M}\mathbf{f}\left(\mathbf{s}_{1:N}^{(i)}\right)\tilde{\boldsymbol{\Sigma}}^{(i)}\,\mathbf{f}\left(\mathbf{s}_{1:N}^{(i)}\right)^T\right)^{-1}\sum_{i=1}^{M}\mathbf{f}\left(\mathbf{s}_{1:N}^{(i)}\right)\tilde{\boldsymbol{\Sigma}}^{(i)}\mathbf{a}_{1:N}^{(i)}. \tag{18}$$

However, there is no closed-form solution for the elements of $\mathbf{D}$, so we solve

$$\mathbf{D}^* = \arg\max_{\mathbf{D}}\prod_{i=1}^{M} p\left(\mathbf{a}_{1:N}^{(i)}\,\middle|\mathbf{s}_{1:N}^{(i)};\boldsymbol{\mu}^*,\mathbf{D}\right),$$

by using a numerical optimizer. The variance σ^2 of the action likelihood term in Eq. (15) is set to 1 in our experiments. This variance is small relative to the maximum action, and acts as a regularizer in Eq. (18). The KL divergence bound ϵ was set to 0.5 in our experiments.

3 Experiments and Results

In this section, we employ the generalized exploration algorithms outlined above to solve different reinforcement learning problems with continuous states and actions. In our experiments, we want to show that generalized exploration can be used to obtain better policies than either the step-based or the episode-based exploration approaches found in prior work. We will also look more specifically into some of the factors mentioned in Section 1 that can explain some of the differences in performance. First, we evaluate generalized exploration in a policy gradient algorithm on a linear control task. Then, we will evaluate generalized exploration in relative entropy policy search on three tasks: an inverted pendulum balancing task with control delays; an underpowered pendulum swing-up task; and an in-hand manipulation task in a realistic robotic simulator.

3.1 Policy Gradients in a Linear Control Task

In the first experiment, we consider a dynamical system where the state $\mathbf{s} = [x, \dot{x}]^T$ is determined by the position and velocity of a point mass of $m = 1$kg. The initial state of the mass is distributed as a Gaussian distribution with $x \sim \mathcal{N}(-7.5, 5^2)$ and $\dot{x} \sim \mathcal{N}(0, 0.5^2)$. The position and velocity of the mass are limited to $-20 \leq x \leq 20$, $-10 \leq \dot{x} \leq 10$ by clipping the values if they leave this range. The goal of the controller to bring the mass to the phase-space origin is defined by the reward function

$$r(\mathbf{s}) = -\frac{3}{200}\mathbf{s}^T\mathbf{s} + \exp\left(-\frac{\mathbf{s}^T\mathbf{s}}{8}\right).$$

As action, a force can be applied to this mass. Furthermore, friction applies a force of $-0.5\dot{x}$N. The actions are chosen according to the linear policy $a = \boldsymbol{\theta}^T\mathbf{s}$, with $\boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{\mu}, 1)$, where $\boldsymbol{\mu}$ is initialized as $\mathbf{0}$ and subsequently optimized by the policy gradient algorithm outlined in Section 2.1. Every episode consists of 50 time-steps of 0.1 s. As baseline for the policy gradients, we use the average reward of that iteration.

The rationale for this task is, that it is one of the simplest task where consistent exploration is important, since the second term in the reward function will only yield non-negligible values as the point mass gets close to the origin. Our proposed algorithm is a generalization of the G(PO)MDP and PEPG algorithms, we obtain those algorithms if we choose $\beta = 1$ (GPOMDP) or $\beta = 0$ (PEPG). We will compare those previous algorithms to other settings for the exploration coherence term β. Besides analyzing the performance of the algorithms in terms of average reward, we will look at how big a range of positions is explored by the initial policy for the various settings.

For every condition, 20 trials were performed. In each trial, 15 iterations were performed that consist of a data-gathering step and a policy update step. Seven episodes were performed in each iteration, as seven is the minimum number of roll-outs required to fit the baseline parameters in a numerically stable manner. As different

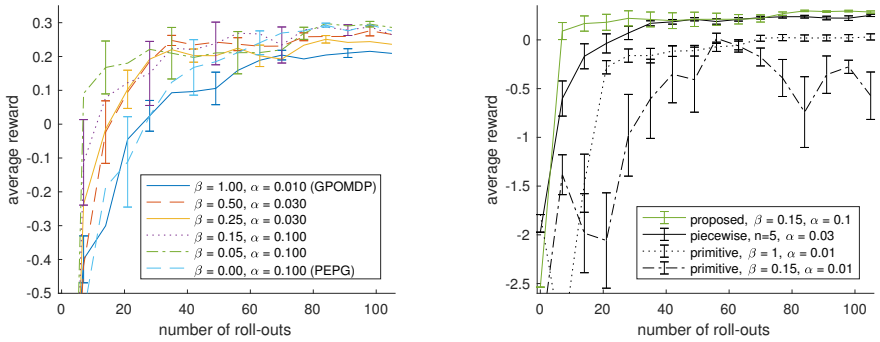
values of the coherence parameter β require a different step size for optimal performance within the 15 available iterations, we ran each condition for each step size $\alpha \in \{0.1, 0.03, 0.01, 0.003, 0.001\}$, and used the step-size that yielded maximal final average reward.

We compared the proposed method to three baselines. First, we look at a coherent and an in-coherent strategy that perform non-coherent exploration directly on the primitive actions. These strategies use the same coherency trade-off β, but applied to an additive Gaussian noise term. The third baseline is a piecewise constant policy that repeats the same parameters for n time steps. The policy gradient for these three methods was derived using the same approach as in Section 2.1. The best number of repeats n and the step-size were found using a grid search.

3.2 Results and Discussion of the Linear Control Experiment

The results of the linear control experiment are shown in Figures 1 and 2. The average rewards obtained for different coherency settings of the proposed algorithm are shown in Figure 1a, where the best step size α for each value of the trade-off parameter β is used. In this figure, we can see that for intermediate values of the temporal coherence parameter β, the learning tends to be faster. Furthermore, lower values of β tended to yield slightly better performance by the end of the experiment. Suboptimal performance for PEPG ($\beta = 0$) can be caused by the fact that PEPG can only try a number of parameters equal to the number of roll-outs per iteration, which can lead to high-variance updates. Suboptimal performance for G(PO)MDP ($\beta = 1$) can be caused by the 'washing out' due to the high frequency of policy perturbations.

A comparison to the baselines discussed in Section 3.1 is shown in Figure 1b. Non-coherent and coherent exploration on primitive action do not find policies that are as good as any parameter-based exploration strategy. Coherent exploration directly on primitive actions tended to yield extremely high-variance updates independent of the



(a) Average reward in the linear control task with policy gradient methods. Error bars show the standard error over 20 trials.

(b) Comparison of the proposed method to three baselines. Error bars show the standard error over 20 trials.

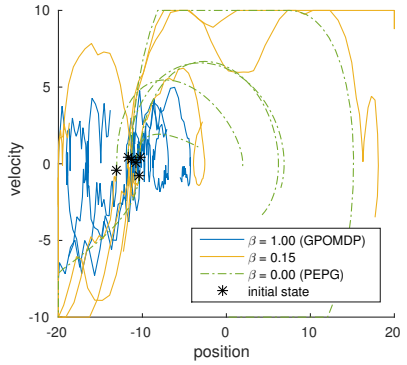
Fig. 1: Learning progress for our proposed method and three baselines. Note that the scale on the y-axis differs between the plots.

step size that was used (the variant which reached the highest performance is shown). We think parameter-based exploration performs better as it can adapt to the region of the state-space, i.e., in the linear control experiment, exploration near the origin would automatically become more muted. Furthermore, coherent exploration on the individual actions can easily overshoot the target position. The piecewise constant policy that repeats the same parameters for n time-steps finds roughly similar final strategies as the proposed method, but takes longer to learn this strategy as the step-size parameter needs to be lower to account for higher-variance gradient estimates. Another disadvantage of the piecewise strategy is that the behavior on a real system would be more jerky and cause more wear and tear.

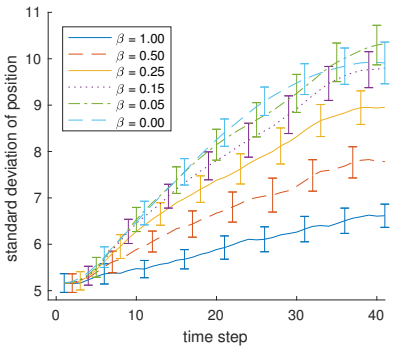
To investigate this possible cause, in Figure 2, we show example trajectories as well as the evolution of the standard deviation of the position x. In Figure 2a, example trajectories under the initial policies are shown. Here, the difference between coherent exploration and high-frequency perturbations are clearly visible. Figure 2b shows that, from the initial standard deviation, low values of β yield a higher increase in variance over time, indicating those variants explore more of the state-space. This difference is likely to be caused by those methods exploring different 'strategies' that visit different parts of the state-space, rather than the high-frequency perturbations for high β that tend to induce random walk behavior. The growth of the standard deviation slows down in later time steps as the position limits of the system are reached.

3.3 REPS for Inverted Pendulum Balancing with Control Delays

In this experiment, we consider the task of balancing a pendulum around its unstable equilibrium by applying torques at its fulcrum. The pendulum we consider has a mass $m = 10\text{kg}$ and a length $l = 0.5\text{m}$. Furthermore, friction applies a force of $0.36\dot{x}\text{Nm}$. The pendulum's state is defined by its position and velocity $\mathbf{s} = [x, \dot{x}]^T$, where the angle of the pendulum is limited $-1 < x < 1$. The chosen action $-40 < a < 40$ is a torque



(a) Example trajectories under different settings of the coherency parameter. Coherent trajectories explore more globally.



(b) Standard deviation of positions reached. Error bars show the standard error over 20 trials.

Fig. 2: Example trajectories and distribution statistics under the initial policy using G(PO)MDP ($\beta = 1$) and PEPG ($\beta = 0$) as well as other settings for β.

to be applied at the fulcrum for a time-step of 0.05s second. However, in one of our experimental conditions, we simulate control delays of 0.025s, such that the actually applied action is $0.5a_t + 0.5a_{t-1}$. This condition breaks the Markov assumption, and we expect that smaller values of the trade-off parameter β will be more robust to this violation. The action is chosen according to a linear policy $a = \boldsymbol{\theta}^T \mathbf{s}$. The parameters are chosen from a normal distribution $\boldsymbol{\theta} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{D})$, which is initialized using $\boldsymbol{\mu} = \mathbf{0}$, and $\mathbf{D}$ a diagonal matrix with $D_{11} = 120^2$ and $D_{22} = 9^2$. Subsequently, $\boldsymbol{\mu}$ and $\mathbf{D}$ are updated according to the generalized REPS algorithm introduced in Section 2.2. We use the quadratic reward function $r(x, \dot{x}, a) = 10x^2 + 0.1\dot{x}^2 + 0.001a^2$.

Roll-outs start at a position $x \sim \mathcal{N}(0, 0.2^2)$ with a velocity $\dot{x} \sim \mathcal{N}(0, 0.5^2)$. At every step, there is a fixed probability of 10% of terminating the episode [46]. As such, each episode contains 10 time steps on average. Initially, 60 roll-outs are performed. At every iteration, the 20 oldest roll-outs are replaced by new samples. Then, the policy is updated using these samples. The sampling distribution q is, thus, a mixture of state-action distributions under the previous three policies. For the features ϕ_i in Eq. (12), we use 100 random features that approximate the non-parametric basis in [46]. These random features $\boldsymbol{\Phi}$ are generated according to the procedure in [35], using manually specified bandwidth parameters, resulting in

$$\phi_i(\mathbf{s}) = 50^{-1/2} \cos\left([\cos(x), \sin(x), \dot{x}]\boldsymbol{\omega}_i + b_i\right), \tag{19}$$

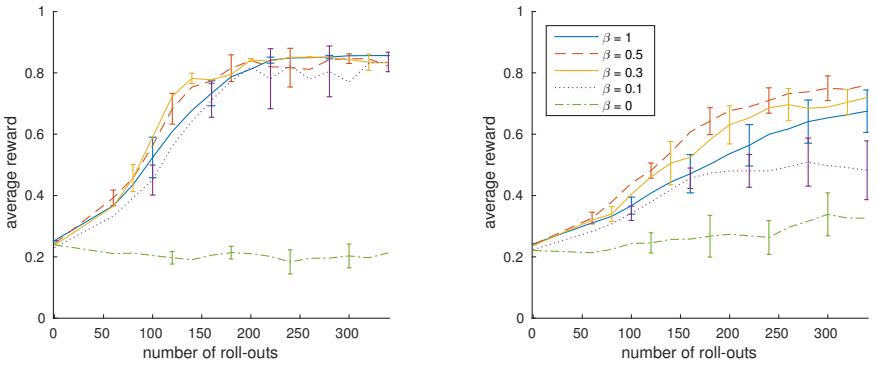
where b is a uniform random number $b \in [0, 2\pi]$ and $\omega_i \sim \mathcal{N}(\mathbf{0}, \mathbf{B}^{-1})$, where $\mathbf{B}$ is a diagonal matrix with the squared kernel bandwidth for each dimension. Thus, every features ϕ_i is defined by a randomly draw vector ω_i and a random scalar b_i. In our experiments, the bandwidths are 0.35, 0.35, and 6.5, respectively.

In our experiment, we will compare different settings of the coherence parameter β under a condition without delays and a condition with the half time-step delay as explained earlier in this section. In this condition, we want to test the assumption that a lower value of β makes the algorithm more robust against non-Markov effects. For $\beta = 1$, we obtain the algorithm described in [33, 46]. We will compare this previous step-based REPS algorithm to other settings of the coherence trade-off term β.

3.4 Results of the Pendulum Balancing Experiment

The results of the inverted pendulum balancing task are shown in Figure 3. The results on the standard balancing task, without control delays, are shown in 3a. This figure shows that, generally, values of the consistency trade-off parameter β of at least 0.3 result in better performance than setting $\beta = 0.1$. Setting $\beta = 0$ results in the algorithm being unable to improve the policy. Being able to try only one set of parameters per roll-out could be one cause, but the procedure described in Section 2.2 might also struggle to find a distribution that matches all weighted samples while keeping the parameter values constant for the entire trajectory. Between the different settings with $\beta \geq 0.3$ small differences exist, possibly because the standard version of REPS with time-independent exploration ($\beta = 1$) suffers from 'washing out' of exploration signals like in the policy gradient experiment in Section 3.1.

In a second experimental condition, we simulate control delays, resulting in the applied action in a certain time step being a combination of the actions selected in the previous and current time steps. This violation of the Markov assumption makes the task harder. As expected, Figure 3b shows that the average reward drops for all



(a) Inverted pendulum balancing without delays.

(b) Inverted pendulum balancing with a delay of half a time-step.

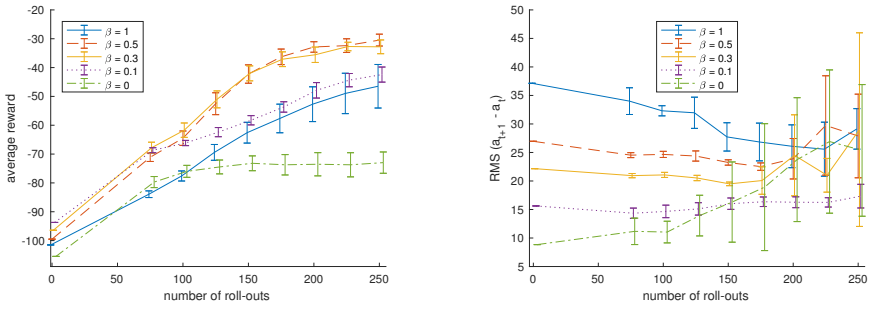
Fig. 3: Inverted pendulum balancing tasks with control delays using relative entropy policy search. Error bars show twice the standard error over 10 trials, and are shown at selected iterations to avoid clutter.

conditions. For $\beta = 1$, the decrease in performance is much bigger than for $\beta = 0.5$ or $\beta = 0.3$. However, unexpectedly $\beta = 0.5$ seems to yield better performance than smaller values for the trade-off parameter. We suspect this effect to be caused by a too sparse exploration of the state-action space for each set of policy parameters, together with a possible difficulty in maximizing the resulting weighted likelihood as discussed in the previous paragraph.

3.5 REPS for Underpowered Swing-up

In the underpowered swing-up task, we use the same dynamical system as in the previous experiment with the following modifications: the pendulum starts hanging down close to the stable equilibrium at $x = \pi$, with $x_0 \sim \mathcal{N}(\pi, 0.2^2)$ with $\dot{x}_0 = 0$. The episode was re-set with a probability of 2% in this case, so that the average episode length is fifty time steps. The pendulum position is in this case not limited, but projected on $[-0.5\pi, 1.5\pi]$. Actions are limited between -30 and 30N. A direct swing-up is consequently not possible, and the agent has to learn to make a counter-swing to gather momentum first.

Since a linear policy is insufficient, instead, we use a policy linear in exponential radial basis features with a bandwidth of 0.35 in the position domain and 6.5 in the velocity domain, centered on a 9×7 grid in the state space, yielding 63 policy features. Optimizing 63 entries of the policy variance matrix $\mathbf{D}$ would slow the learning process down drastically, so in this case we used a spherical Gaussian with $\mathbf{D} = \lambda \mathbf{I}$, so that only a single parameter λ needs to be optimized. We found that setting the regularization parameter $\sigma = 0.05\lambda$ in this case made the optimization of λ easier and resulted in better policies. In this experiment, we used 25 new roll-outs per iteration, using them together with the 50 most recent previous rollouts, to account for the higher dimensionality of the parameter vector. The rest of the set-up is identical to that in Section 3.3. Notably, the same random features are used for the steady-state constraint.



(a) Average reward for the underpowered swing-up task.

(b) Root-mean square of differences between subsequent actions indicates applied jerks.

Fig. 4: Pendulum swing-up task using relative entropy policy search. Error bars show twice the standard error over 10 trials, and are slightly offset to avoid clutter.

Besides evaluating the average rewards obtained using different values of the exploration coherence trade-off term β, we evaluate the typical difference between subsequent actions as measured by the root-mean-squared difference between subsequent actions. Actions correspond to applied torques in this system, and the total torque (from applied actions and gravity) is directly proportional to the rotational acceleration. Thus, a big difference in subsequent actions can cause high jerks, which causes wear and tear on robotic systems. As such, in most real systems we would prefer the typical difference between subsequent actions to be low.

3.6 Results of the Underpowered Swing-up Experiment

The results on the pendulum swing-up task are shown in Figure 4. With $\beta = 0$, performance is rather poor, which could be due to the fact that in this strategy rather few parameter vectors are tried in each iteration. Figure 4a shows, that setting the trade-off parameter to an intermediate value yields higher learning speeds than setting $\beta = 1$ as in the original REPS algorithm. Again, the washing out of high-frequency exploration could be a cause of this effect.

Figure 4b shows another benefit of setting the exploration coherence parameter to an intermediate value. The typical difference between chosen actions is more than 50% higher initially for the original REPS algorithm ($\beta = 1$), compared to setting $\beta = 0.3$. This behavior will cause higher jerks, and thus more wear and tear, on robot systems where these controllers are applied. The typically higher difference between actions persist even as the algorithm gets close to an optimal solution after 175 roll-outs. After that, the typical difference tends to go up for all methods, as the hyper-parameter optimization finds more extreme values as the policy gets close to convergence.

3.7 REPS in an in-hand manipulation task

In this experiment, we aim to learn policies for switching between multiple grips with a simulated robotic hand based on proprioceptive and tactile feedback. Grips are changed

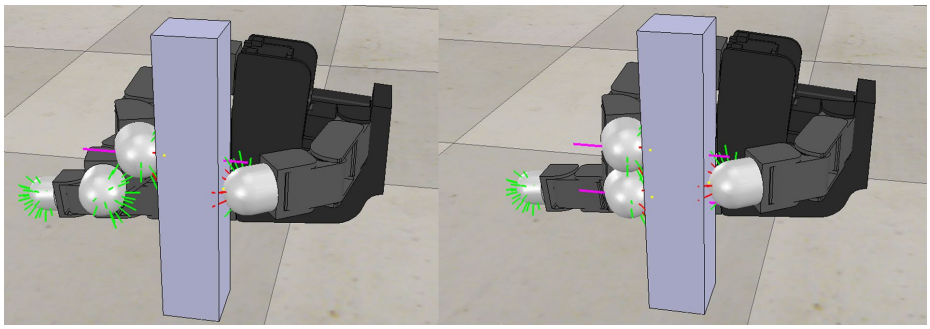


Fig. 5: Illustration of the in-hand manipulation task. Two of the goal positions are shown; the task for the robot is to transfer stably between such goal positions without dropping the held block. Colored lines show activation of the sensor array as well as the contact normals.

by adding or removing fingers in contact with the object. Consequently, the force that needs to be applied by the other fingers changes, requiring co-ordination between fingers. We use three different grips: one three-finger grip where the thumb is opposed to two other fingers, and two grips where one of the opposing fingers is lifted. For each of these three grips, we learn a sub-policy for reaching this grip while maintaining a stable grasp of the object. All three sub-policies are learned together within one higher-level policy. This task is illustrated in Figure 5.

The V-REP simulator is used to simulate the dynamics of the Allegro robot hand. We additionally simulate a tactile sensor array with 18 sensory elements on each fingertip, where the pressure value at each sensor is approximated using a radial basis function centered at the contact location multiplied by the contact force, yielding values between 0 and 7. The internal PD controller of the hand runs at 100Hz, the desired joint position is set by the learned policy at 20Hz. We control the proximal joints of the index and middle finger, while the position of the thumb is kept fixed. Thus, the state vector $\mathbf{s}$ consists of the proximal joint angles of the index and middle finger. The resulting state representation is then transformed into the feature vector $\boldsymbol{\phi}(\mathbf{s})$ using 500 random Fourier features described in Section 3.3. We take the Kronecker product of those 500 features with a one-hot encoding of the current goal grip, resulting in 1500 features that were used for both the value function and the policy. Since only the features for the active goal are non-zero, the resulting policy represents a combination of several sub-policies. Other settings are the same as described in Section 3.5.

Each goal grip is defined by demonstrated joint- and sensor space configurations. Using these demonstrations as target positions, we define the reward using the squared distance in sensor and joint space with an additional penalty for wrong finger contact configurations. The precise reward function is given by

$$r(\mathbf{s}, \mathbf{a}) = -(\mathbf{z}(\mathbf{s}, \mathbf{a}) - \mathbf{z}_d)^2 - w_j(\mathbf{j}(\mathbf{s}, \mathbf{a}) - \mathbf{j}_d)^2 - w_c,$$

where $\mathbf{z}(\mathbf{s}, \mathbf{a})$ is the sensor signal for all three fingers resulting from applying action $\mathbf{a}$ in state $\mathbf{s}$, and $\mathbf{z}_d$ is the desired sensor signal. Coefficient w_j is the weight for the joint distance and is set to 3000, $\mathbf{j}(\mathbf{s}, \mathbf{a})$ and $\mathbf{j}_d$ are the current and desired joint angle configuration for all three fingers, and w_c is a penalty term for wrong contact configurations

with the object and is set to 150 for each finger that is in contact while it should not be, or vice versa.

We performed 5 trials for each setting where each trial consists of 20 iterations. Initially, 90 roll-outs are performed. In each subsequent iteration, 30 new roll-outs are sampled and used together with the 60 most recent previous roll-outs. At the start of each roll-out, a start grip and target grip are selected such that each target grip is used equally often. Each rollout consists of 30 time steps (1.5 s of simulated time).

3.8 Results of the in-hand manipulation experiment

The result of the in-hand manipulation experiment are shown in Figure 6. In all cases, the controller improved upon the initial controller, becoming better at the task of switching between two- and three-fingered grip while keeping the object stable. However, choosing an intermediate value of $\beta = 0.75$ lead to markedly better improvement compared to step-based REPS ($\beta = 1$), or lower values of the trade-off parameter β. Learning such a policy with a pure episode-based method ($\beta = 0$) failed to learn the task at all. We suspect that an impracticable number of roll-outs would be necessary to learn the 1500 dimensional parameter vector with this method.

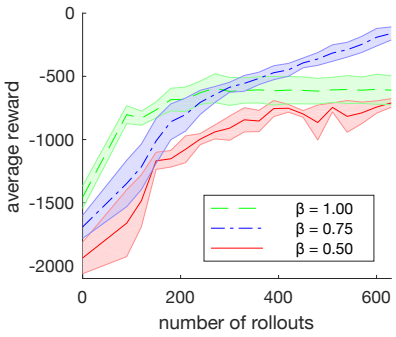


Fig. 6: Average reward in the in-hand manipulation experiment with REPS, using a 1500-dimensional parameter vector. The shaded area indicates twice the standard error over 5 trials.

With $\beta = 0.75$, the resulting learned policies were able to safely switch between the two- and the three-finger grips in both directions while keeping the object stable in the robot's hand. Although the target end-points of the movement were demonstrated, the robot autonomously learned a stable strategy to reach them using tactile and proprioceptive feedback.

4 Discussion and Future Work

In this paper, we introduced a generalization of step-based and episode-based exploration of controller parameters. This generalization allows different trade-offs to be made between the advantages and disadvantages of temporal coherence in exploration. Whereas independent perturbation of actions at every time step allows more parameter values to be tested within a single episode, fully coherent (episode-based) exploration has the advantages of, among others, avoiding 'washing out' of explorative perturbations, being robust to non-Markovian aspects of the environment, and inducing lower jerk, and thus less strain, on experimental platforms.

Our experiments confirm these advantages of coherent exploration, and show that intermediate strategies between step-based and episode-based exploration provide a trade-off between these advantages. In terms of average reward, as expected, for many

systems intermediate trade-offs between completely independent, step-based exploration and complete correlated, episode exploration, provides the best learning performance.

Many of the benefits of consistent exploration are especially important on robotic systems. Our experiment on a simulated robotic manipulation task shows, that use of the trade-off parameter can in fact help improve learning performance on such systems. Since the advantage of using an intermediate strategy seemed most pronounced on this more complex task, we expect similar benefits on tasks with even more degrees of freedom.

Our approach introduces a new open hyper-parameter β that needs to be set manually. Based on our experience, we have identified some problem properties that influence the best value of β, which provide some guideline for setting it. When the number of roll-outs that can be performed per iteration is low, β should be set to a relatively high value. However, if abrupt changes are undesirable, if the system has delays, or if a coherent sequence of actions is required to observe a reward, β should be set to a relatively low value. Intermediate values of β allow trading off between these properties.

A couple of questions are still open: In future work, we want to consider how smoother non-Markov processes over parameters could be used for exploration, and investigate methods to learn the coherency trade-off parameter β from data.

Acknowledgement

This work has been supported in part by the TACMAN project, EC Grant agreement no. 610967, within the FP7 framework programme. Part of this research has been made possible by the provision of computing time on the Lichtenberg cluster of the TU Darmstadt.

References

1. J. Asmuth, L. Li, M. L. Littman, A. Nouri, and D. Wingate. A Bayesian sampling approach to exploration in reinforcement learning. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 19–26. AUAI Press, 2009.
2. J. Baxter and P. L. Bartlett. Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research*, 15:319–350, 2001.
3. B. C. da Silva, G. Konidaris, and A. G. Barto. Learning parameterized skills. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1679–1686, 2012.
4. C. Daniel, G. Neumann, O. Kroemer, and J. Peters. Hierarchical relative entropy policy search. *Journal of Machine Learning Research*, 2016.
5. C. Daniel, H. van Hoof, J. Peters, and G. Neumann. Probabilistic inference for determining options in reinforcement learning. *Machine Learning*, 104:337–357, 2016.
6. R. Dearden, N. Friedman, and D. Andre. Model based Bayesian exploration. In *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 150–159, 1999.
7. R. Dearden, N. Friedman, and S. Russell. Bayesian Q-learning. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 761–768, 1998.
8. M. P. Deisenroth, G. Neumann, and J. Peters. A survey on policy search for robotics. *Foundations and Trends in Robotics*, pages 388–403, 2013.
9. M. P. Deisenroth, C. E. Rasmussen, and J. Peters. Gaussian process dynamic programming. *Neurocomputing*, 72(7):1508–1524, 2009.
10. T. G. Dietterich. Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of Artificial Intelligence Research*, 13:227–303, 2000.

11. F. Doshi-Velez, D. Wingate, N. Roy, and Tenenbaum J. B. Nonparametric Bayesian policy priors for reinforcement learning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 532–540, 2010.
12. M. Ghavamzadeh and S. Mahadevan. Hierarchical policy gradient algorithms. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 226–233, 2003.
13. M. Guo, Y. Liu, and J. Malec. A new Q-learning algorithm based on the Metropolis criterion. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 34(5):2140–2143, 2004.
14. N. Hansen, S. D. Müller, and P. Koumoutsakos. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). *Evolutionary computation*, 11(1):1–18, 2003.
15. W. K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.
16. M. Hausknecht and P. Stone. Deep reinforcement learning in parameterized action space. In *Proceedings of the International Conference on Learning Representations*, 2016.
17. M. Hoffman, A. Doucet, N. D. Freitas, and A. Jasra. Bayesian policy learning with trans-dimensional MCMC. In *Advances in Neural Information Processing Systems (NIPS)*, pages 665–672, 2007.
18. L. P. Kaelbling. Hierarchical learning in stochastic domains: Preliminary results. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 167–173, 1993.
19. L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.
20. J. Kober, J. A. Bagnell, and J. Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 11(32):1238–1274, 2013.
21. J. Kober and J. Peters. Policy search for motor primitives in robotics. In *Advances in Neural Information Processing Systems (NIPS)*, pages 849–856, 2009.
22. N. Kohl and P. Stone. Policy gradient reinforcement learning for fast quadrupedal locomotion. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, volume 3, pages 2619–2624, 2004.
23. G. Konidaris and A. Barto. Skill discovery in continuous reinforcement learning domains using skill chaining. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1015–1023, 2009.
24. P. Kormushev and D. G. Caldwell. Direct policy search reinforcement learning based on particle filtering. In *Proceedings of the European Workshop on Reinforcement Learning (EWRL)*, 2012.
25. T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning. In *Proceedings of the International Conference on Learning Representations*, 2016.
26. H. J. Meijdam, M. C. Plooij, and W. Caarls. Learning while preventing mechanical failure due to random motions. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 182–187, 2013.
27. J. Morimoto and K. Doya. Acquisition of stand-up behavior by a real robot using hierarchical reinforcement learning. *Robotics and Autonomous Systems*, 36(1):37–51, 2001.
28. R. Munos. Policy gradient in continuous time. *Journal of Machine Learning Research*, 7:771–791, 2006.
29. P. A. Ortega and D. A. Braun. A minimum relative entropy principle for learning and acting. *Journal of Artificial Intelligence Research*, 38:475–511, 2010.
30. I. Osband, C. Blundell, A. Pritzel, and B. Van Roy. Deep exploration via bootstrapped dqn. In *Advances in Neural Information Processing Systems*, pages 4026–4034, 2016.
31. I. Osband, B. Van Roy, and Z. Wen. Generalization and exploration via randomized value functions. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 2377–2386, 2016.
32. R. Parr and S. Russell. Reinforcement learning with hierarchies of machines. *Advances in Neural Information Processing Systems (NIPS)*, pages 1043–1049, 1998.
33. J. Peters, K. Mülling, and Y. Altün. Relative entropy policy search. In *Proceedings of the National Conference on Artificial Intelligence (AAAI), Physically Grounded AI Track*, pages 1607–1612, 2010.
34. D. Precup. *Temporal Abstraction in Reinforcement Learning*. PhD thesis, University of Massachusetts Amherst, 2000.

35. A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1177–1184, 2007.
36. T. Rückstieß, F. Sehnke, T. Schaul, D. Wierstra, Y. Sun, and J. Schmidhuber. Exploring parameter space in reinforcement learning. *Paladyn, Journal of Behavioral Robotics*, 1(1):14–24, 2010.
37. S. Schaal, J. Peters, J. Nakanishi, and A. Ijspeert. Learning movement primitives. In *International Symposium on Robotics Research.*, pages 561–572, 2005.
38. F. Sehnke, C. Osendorfer, T. Rückstieß, A. Graves, J. Peters, and J. Schmidhuber. Parameter-exploring policy gradients. *Neural Networks*, 23(4):551–559, 2010.
39. S. Singh. Transfer of learning by composing solutions of elemental sequential tasks. *Machine Learning*, 8(3):323–339, 1992.
40. M. Strens. A Bayesian framework for reinforcement learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 943–950, 2000.
41. F. Stulp and S. Schaal. Hierarchical reinforcement learning with movement primitives. In *Proceedings of the IEEE International Conference on Humanoid Robots (Humanoids)*, pages 231–238, 2011.
42. F. Stulp and O. Sigaud. Path integral policy improvement with covariance matrix adaptation. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2012.
43. R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT press, 1998.
44. R. S. Sutton, D. Precup, and S. Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1):181–211, 1999.
45. E. Theodorou, J. Buchli, and S. Schaal. A generalized path integral control approach to reinforcement learning. *The Journal of Machine Learning Research*, 11:3137–3181, 2010.
46. H. van Hoof, J. Peters, and G. Neumann. Learning of non-parametric control policies with high-dimensional state features. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AIstats)*, pages 995–1003, 2015.
47. A. Vezhnevets, V. Mnih, S. Osindero, A. Graves, O. Vinyals, J. Agapiou, et al. Strategic attentive writer for learning macro-actions. In *Advances in Neural Information Processing Systems*, pages 3486–3494, 2016.
48. C. Watkins and Y. Buttkewitz. Sex as Gibbs sampling: a probability model of evolution. Technical Report 1402.2704, ArXiv, 2014.
49. P. Wawrzyński. Control policy with autocorrelated noise in reinforcement learning for robotics. *International Journal of Machine Learning and Computing*, 5:91–95, 2015.
50. R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256, 1992.
51. D. Wingate, N. D. Goodman, D. M. Roy, L. P. Kaelbling, and J. B. Tenenbaum. Bayesian policy search with policy priors. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2011.
52. J. Wyatt. *Exploration and Inference in Learning from Reinforcement*. PhD thesis, University of Edinburgh, College of Science and Engineering, School of Informatics, 1998.