EXPLORATION VIA SAMPLE-EFFICIENT SUBGOAL DESIGN

Yijia Wang[1], **Brian Bell**[2], **Matthias Poloczek**[2,3], **Daniel R. Jiang**[1]
[1]University of Pittsburgh, [2]The University of Arizona, [3]Uber AI Lab
yiw94@pitt.edu, bwbell@math.arizona.edu,
poloczek@email.arizona.edu, drjiang@pitt.edu

ABSTRACT

The problem of exploration in unknown environments continues to pose a challenge for reinforcement learning algorithms. In this paper, we consider a new problem domain where an agent faces an unknown task in the future, assumed to be drawn from an unknown distribution of Markov decision processes, that it must learn within a small number of samples. Prior to this, the agent has opportunities to "practice" in related tasks from the same distribution. We propose a sample-efficient Bayesian approach for subgoal design to maximize the expected performance over a distribution of tasks, given a limited number of interactions with the environment.

1 INTRODUCTION

We study the problem of exploration in reinforcement learning (RL), where agents face the often expensive problem of exploring various regions of the state space in order to find a reasonable operating policy for the Markov decision process (MDP). In this paper, we examine the issue of exploration through a new lens, by focusing on problems with the following unique features: (1) there is a distribution of possible environments (or tasks) that are related through common state and action spaces; (2) the agent must attain a sparse and delayed reward, such as reaching a goal; and (3) interactions with the environment(s) are limited or expensive, in both training and testing.

Specifically, we consider a *training phase*, where the agent is given a fixed number of opportunities to train in randomly drawn environments (henceforth, they are referred to as training environments). Thereafter, the agent enters a random *test environment* and must learn a policy within a limited number of interactions. The idea is that the agent can learn an exploration strategy that works well on average, across the distribution of test environments, by practicing in related settings.

The motivation for this problem comes from a need to apply RL in real-world settings where fast and cheap interactions with the environment are unavailable and only a limited number of experiments can be tried. The following examples illustrate two potential application domains: (1) Autonomous robotic systems have long been used to explore unknown or dangerous terrains (Apostolopoulos et al., 2001; Ferguson et al., 2004). Matthies et al. (1995) describe the design of a rover for the Mars Pathfinder mission, where a human designates waypoints to navigate a rocky terrain and reach a goal. To train for the eventual Mars mission, the engineers utilized an "indoor arena" filled with specially selected rocks to act as obstacles, which approximates the eventual test environment. (2) Our second example is the adaptive controllers for *carbon nanotube growth* (Nikolaev et al., 2014; Dee et al., 2018) in materials science. Material scientists perform extensive experiments to find growth recipes in order to optimize the growth rate of carbon nanotube forests. It is time-consuming and is costly. And each experiment may take place in slightly different laboratory environments.

Our proposed Bayesian approach for finding optimal subgoals and intrinsic reward shaping schemes consists of two components. One is a tailored probabilistic model that learns the typical performance of subgoals from observations. The other is an efficient, Bayes optimal one-step policy to select subgoals. We point out that while dynamic programming could in principle be used to obtain an optimal solution to the problem, it is impractical due to the curse of dimensionality (Powell, 2007).

Related Work. RL has recently been tremendously successful in complex sequential decision-making problems, with applications in games (Silver et al., 2017), robotics (Hanna & Stone, 2017),

healthcare (Prasad et al., 2017) and many other areas. State-of-art algorithms (e.g. Grondman et al. (2012); Mnih et al. (2013)) inherently require a large number of observations from the environment and converge slowly in complex problems. Moreover, the problem of exploration in unknown environments continues to pose a challenge for many RL algorithms (Osband et al., 2013; 2016; Fortunato et al., 2017). Many algorithms employ optimism approaches to encourage exploration in poorly-understood states and actions (Kearns & Singh, 2002; Stadie et al., 2015; Tang et al., 2017).

Intrinsic reward helps robots learn increasingly complex behavior in a self-motivated way (Huang & Weng, 2002; Pathak et al., 2017). Potential-based reward shaping (PBRS) (Ng et al., 1999) provides a way to modify the reward that simultaneously maintains the optimal policy and potentially accelerates learning. The agent's performance highly depends on an informative reward signal; however, manual reward shaping (RS) can be a labor-intensive process requiring a deep domain expertise.

To overcome the variation of the environment in real-world application, RL researchers are recently interested in leveraging the experience from previous tasks to improve the agent's performance in a new task, including continual learning, transfer learning and meta learning. The idea has been realized by parameter initialization (Tanaka & Yamamura, 1997; Konidaris & Barto, 2006), parameter distinguishing (Vuorio et al., 2018), goal-conditioned value function (Mankowitz et al., 2018), and policy learning (Fernández & Veloso, 2013; Deisenroth et al., 2014).

Our approach of designing subgoals follows the Bayesian optimization (BO) paradigm, a powerful technique for optimizing black-box functions, in particular for tuning ML models and design of experiments (Brochu et al., 2010; Snoek et al., 2012; Herbol et al., 2018). Our work bears resemblance and generalizes methods for network architecture search and optimization with multiple information sources (Swersky et al., 2013; Feurer et al., 2015; Domhan et al., 2015; Klein et al., 2016). Their work differs in that they may observe directly the loss curve during training, whereas our setting only allows to observe the score of the policy after the underlying exploration process has completed.

2 Problem Formulation

Let $\xi \in \Xi$ be a random variable that parameterizes the set of possible environments. Each environment is modeled by an MDP $\langle \mathcal{S}, \mathcal{A}, g_\xi, R_\xi, \gamma \rangle$, where $\mathcal{S}$ and $\mathcal{A}$ are the state and action spaces, $g_\xi : \mathcal{S} \times \mathcal{A} \times \mathcal{W} \to \mathcal{S}$ is a transition function, $\mathcal{W}$ is the space associated with noise w, $R_\xi : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the reward function and $\gamma \in (0, 1)$ is the discount factor. Our model assumes common state and action spaces across the distribution of MDPs (i.e., they are independent of ξ), while the reward and transition functions can vary. Given $\mathcal{S}$ and $\mathcal{A}$, a *policy* is a mapping such that $\pi(\cdot|s)$ is a distribution over $\mathcal{A}$ for any $s \in \mathcal{S}$. For any MDP $\langle \mathcal{S}, \mathcal{A}, g_\xi, R_\xi, \gamma \rangle$, define the *value function* of policy π as

$$V_\xi^\pi(s) = \mathbb{E}\big[\textstyle\sum_{t=1}^{\infty} \gamma^{t-1} R_\xi(s_t, a_t, s_{t+1}) \,|\, \pi, s\big], \tag{1}$$

where s is the initial state, $a_t \sim \pi(\cdot|s_t)$, and $\mathbb{E}$ is over noise w and stochastic policy π. The optimal value and policy is $V_\xi^*(s) = \sup_\pi V^\pi(s)$ and $\pi_\xi^*(s) \in \arg\max_{a \in \mathcal{A}} \mathbb{E}[R_\xi(s, a, s') + \gamma V_\xi^*(s') \,|\, s, a]$. We refer to the sequence of training environment realizations by the realizations of the parameter ξ, denoted by $\xi^1, \ldots, \xi^N$, where N is the total number of practice opportunities given to the agent.

2.1 Subgoals with Intrinsic Reward Shaping

The reward function R_ξ is the *extrinsic reward*, which we aim to maximize in a cumulative fashion. However, these rewards are often sparse and delayed, providing little learning signal. We propose to use subgoals with intrinsic RS to provide an artificial reward signal, that if property designed, can direct the agent toward useful parts of the state space to explore. Let $\theta \in \Theta$ be a parameter describing the locations and rewards associated with a specific set of "shaped subgoals." Specifically, we have:

$$\theta = (k, \{\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_k\}, \{R_1, R_2, \ldots, R_k\}), \tag{2}$$

where k is the number of subgoals. The set $\mathcal{S}_j \subseteq \mathcal{S}$ contains the states associated with subgoal j (i.e., if the agent lands in states in $\mathcal{S}_j$, then subgoal j is "completed"), and R_j is a ***PBRS function*** for subgoal j; see Ng et al. (1999) for details regarding PBRS, which we have adapted for the subgoal case. Concretely, $R_j(s, s') = \gamma \Phi_j(s') - \Phi_j(s)$, where Φ_j is a potential function over $\mathcal{S}$.

Each choice of θ introduces an auxiliary state $i \in \mathcal{I}_\theta := \{0, 1, \ldots, k\}$, representing the number of subgoals reached by the agent so far. Initially, $i_0 = 0$. The state of the new system is $(s, i) \in \mathcal{S} \times \mathcal{I}_\theta$

and the transitions are $s' = g_\xi(s, a, w)$ and $i' = i + \mathbf{1}_{\{s' \in \mathcal{S}_{i+1}\}} =: h_\theta(s, a, w, i)$, where $\mathcal{S}_{k+1} = \varnothing$. With auxiliary state i, the intrinsic reward is $R_{i+1}(s, s')$. Let $R_{k+1} \equiv 0$. Intuitively, a different shaped intrinsic reward appears as we complete each subgoal, directing agent toward the next subgoal.

Let $R_\theta(s, i, s') = R_{i+1}(s, s')$ denote the additional intrinsic reward supplemented to the agent by subgoal parameter θ. Our *augmented* MDP has reward function $R_{\xi,\theta}(s, i, a, s') = R_\xi(s, a, s') + R_\theta(s, i, s')$, and we define the value function for the new MDP as

$$V^\pi_{\xi,\theta}(s, i) = \mathbb{E}\big[\textstyle\sum_{t=1}^\infty \gamma^{t-1} R_{\xi,\theta}(s_t, i_t, a_t, s_{t+1}) \,|\, \pi, s, i\big], \tag{3}$$

where, with a slight abuse/reuse of notation, π is a policy acting on the new state space $\mathcal{S} \times \mathcal{I}_\theta$. Our goal is to find a set of parameters θ that can incentivize the agent to intelligently explore environments drawn from the distribution of ξ. The hope is that at test time, the agent is able to quickly learn a good policy after observing only a small number of samples from the test environment.

2.2 Optimizing Subgoals for Exploration

The problem of selecting subgoals depends on the agent's learning algorithm, which could be any RL algorithm. In Section 4, our agent learns via Q-learning (Watkins & Dayan, 1992). However, for the time being, we do not restrict the RL algorithm and simply define $\pi^\tau_{\xi,\theta}$ to be the policy, with state space $\mathcal{S} \times \mathcal{I}_\theta$, attained after τ interactions in environment ξ augmented by shaped subgoals θ. Thus, the RL algorithm is optimizing $V^\pi_{\xi,\theta}$ in (3). While our original objective is V^π_ξ as in (1) without the subgoal rewards. To bridge the gap, we define the value function associated with an augmented policy $\pi(\cdot \,|\, s, i)$ that only incurs extrinsic rewards: $\tilde{V}_\xi(\pi) = \mathbb{E}\big[\sum_{t=1}^\infty \gamma^{t-1} R_\xi(s_t, a_t, s_{t+1}) \,|\, \pi\big]$. Finally, let τ_{max} be the number of interactions available in test environment.

Combining all of the pieces, we have the following optimization problem: $\max_{\theta \in \Theta} \; u(\theta, \tau_{\text{max}})$, where $u(\theta, \tau) := \mathbf{E}\big[\tilde{V}_\xi(\pi^\tau_{\xi,\theta})\big]$. The expectation is taken over the environment ξ and the stochasticity of policy $\pi^{\tau_{\text{max}}}_{\xi,\theta}$. The interpretation of the objective is: we are looking for a subgoal design θ that incentivizes the agent to explore random environments ξ in a way that maximizes the expected performance of a policy learned by the fixed RL algorithm in τ_{max} interactions.

As discussed above, we do not assume the ability to compute the expectation in $\mathbf{E}\big[\tilde{V}_\xi(\pi^\tau_{\xi,\theta})\big]$ and can only observe the performance of policies in a sequence of environment realizations $\xi^1, \xi^2, \ldots, \xi^N$. Let us now move on to discuss the costly training aspect of the model. Two decisions are made at the beginning of training opportunity $n \in \{1, 2, \ldots, N\}$: a subgoal design θ^n and the number of interactions τ^n to use for θ^n. The idea is that the performance of the policy trained for fewer interactions could be informative of that trained for τ_{max} interactions. Let $\mathcal{T}$ be the set of possible values of τ and let $\mathcal{Z} = \Theta \times \mathcal{T}$ be the decision space. We then observe $y^n = u(\theta^n, \tau^n) + \varepsilon^n$, where ε^n is a standard normal random variable that captures the sampling noise due to ξ^n, the noise in $\pi^{\tau^n}_{\xi,\theta^n}$ due to a sample run of the RL algorithm, and the evaluation noise due to an inability to exactly compute $\tilde{V}_\xi$ (even when ξ is fixed). After training opportunity N, we need to output a subgoal design θ^{N+1} for the test MDP ξ^{N+1} with a budget of τ_{max} interactions for the agent to spend.

3 The Algorithm – BESD

Our Bayesian approach for designing subgoals consists of two components: a tailored probabilistic model and an algorithm for selecting the next subgoal and interaction budget to observe. We call the overall approach BESD, an acronym for *Bayesian exploratory subgoal design*.

The Model. We model the expected performance of the underlying RL policy when executed with subgoals θ for $\tau \in \mathbb{N}$ interactions by the latent function u. We suppose observations $y(\theta, \tau) \sim \mathcal{N}(g(\theta, \tau), \lambda(\theta, \tau))$ when evaluating subgoals θ for τ steps, where λ is the variance due to randomness in ξ and the underlying policy. Suppose λ is finite and known, although in practice it is learned from data, e.g., via a maximum likelihood estimate (MLE) or via Gaussian process (GP) regression.

We propose a generative model that gives a GP prior f on the latent function u with mean function $\mu : \mathcal{Z} \to \mathbb{R}$ and covariance function $k : \mathcal{Z} \times \mathcal{Z} \to \mathbb{R}_+$. We set μ to the mean of an initial sample set and use a multidimensional product kernel $k((\theta, \tau), (\theta', \tau')) = k_\theta(\theta, \theta')\, k_\tau(\tau, \tau')$, where k_θ is the

Algorithm 1 Bayesian Exploratory Subgoal Design (BESD)

1: Estimate hyperparameters of the GP prior using initial samples. Set iteration counter $n = 0$.
2: Compute next decision $(\theta^{n+1}, \tau^{n+1})$ according to the acquisition function given in (4).
3: Train the agent in environment ξ^{n+1} based on objective (3) using $(\theta^{n+1}, \tau^{n+1})$. Observe y^{n+1}.
4: Update the posterior distribution with the new data $\{(\theta^{n+1}, \tau^{n+1}), y^{n+1}\}$ and increment counter n. If $n < N$, return to Step 2.
5: Return a subgoal recommendation θ^{N+1} that maximizes $\mu^N(\theta, \tau_{\max})$.

$(5/2)$-Matérn kernel $k_\theta(\theta, \theta') = \frac{\lambda(\theta,\tau)}{r}\left(1 + \frac{\sqrt{5}d}{\rho} + \frac{5d^2}{3\rho^2}\right)\exp\left(-\frac{\sqrt{5}d}{\rho}\right)$ with $d^2 = (\theta - \theta')^\intercal(\theta - \theta')$ and hyperparameter $\rho \geq 0$, $k_\tau(\tau, \tau') = \phi(\tau)^\intercal \Sigma_\phi \phi(\tau')$ is a polynomial kernel with $\phi(\tau) = (1, \tau)^\intercal$ and hyperparameters Σ_ϕ. Note that kernels are closed under multiplication, we may use standard GP machinery to analytically compute the posterior distribution conditioned on the history after n steps: $H^n = (\theta^1, \tau^1, y^1, \dots \theta^n, \tau^n, y^n)$. See (Rasmussen & Williams, 2006, Ch. 2.2) for details.

Bayesian Exploratory Subgoal Design. Suppose the training budget is used up after N steps. The optimal risk-neutral decision is to use subgoals on the test MDP ξ^{N+1} that have maximum expected score under the posterior. The expected score of this choice is μ_N^* where $\mu_n^* := \max_{\theta'} \mu^n(\theta', \tau_{\max})$ and $\mu^n(\theta, \tau) = \mathbb{E}_n[f(\theta, \tau)]$, where $\mathbb{E}_n$ is conditioned on the history H^n. The proposed algorithm proceeds in iterations, designing one set of subgoals θ to be evaluated in each iteration. Thus, we take a myopic approach, i.e., we suppose that this is the last training MDP before the test MDP. If we evaluate the subgoals θ next for τ steps, then the *expected gain in score* (GiS) on the testing MDP is $\mathrm{GiS}^n(\theta, \tau) = \mathbb{E}_n\left[\mu_{n+1}^* \,|\, \theta^{n+1} = \theta, \tau^{n+1} = \tau\right] - \mu_n^*$. Therefore, the one-step optimal strategy is to choose the next subgoals θ^{n+1} and budget τ^{n+1} so that the expected gain GiS is maximized. However, this strategy would generally allocate steps $\tau_{\max}$ for the evaluation of the next subgoal design, as observing $\tau_{\max}$ during training is most informative of the test conditions (where the agent exhaust the entire budget). That is, this strategy does not consider the cost of training. Hence, we propose an acquisition function that maximizes the *gain in score per effort* by dividing the GiS function by the budget τ^{n+1}, resulting in an algorithm that selects

$$(\theta^{n+1}, \tau^{n+1}) \in \arg\max_{\theta,\tau} \mathrm{GiS}^n(\theta, \tau)/\tau. \tag{4}$$

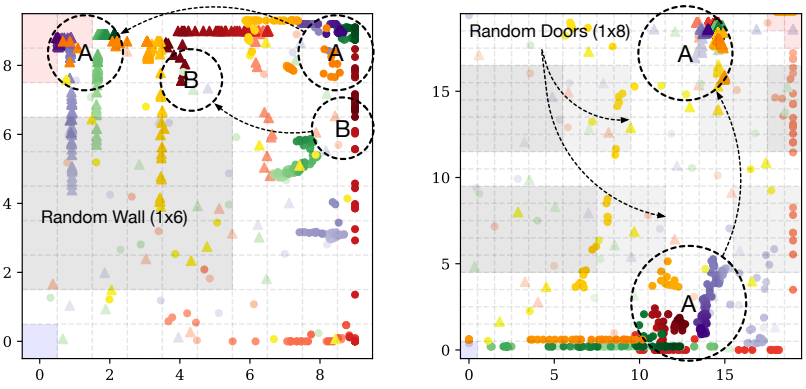
In Appendix B, we detail how to find a maximizer $(\theta^{n+1}, \tau^{n+1})$ efficiently.

4 Numerical Experiments

We evaluate the performance of BESD and the following baseline algorithms on a number of domains: standard Q-learning (QL) (Watkins, 1989), Hyperband (HB) (Li et al., 2016), the expected improvement algorithm (EI) (Močkus, 1975; Jones et al., 1998), and the lower confidence bound algorithm (LCB) (Cox & John, 1992). The details of baselines are in Appendix C. BESD was implemented in Python 2.7 using MOE (Clark et al., 2014). It is given three choices for the interaction budget: $\tau_{\min}$, τ_{mid}, and $\tau_{\max}$. To ensure a fair comparison, in particular with other BO methods, we fixed the number of replications to $r = 20$ for all algorithms. The observational noise λ was set to an MLE. The initial data size is ten for each value of τ. We design two subgoals in all experiments. The potential function at state (s, j) is $\Phi_j(s) = w_1 \exp[(s - j)^2/w_2]$, where $w_1 = 0.2$, $w_2 = 10$.

Two-Room Gridworlds (GW10). We first consider a distribution of 10×10 gridworlds with two rooms separated by a wall. As shown in Fig. 1a, the wall is randomly located in the middle rows (dark gray), with a door located on four grid squares on its right. the goal is to reach the upper right squares (red-shaded) to collect a reward of one. The agent starts from the lower-left squares (blue-shaded) and the action space is the four compass directions. The agent will stay still when it hits the wall. The movement of the agent has a 0 to 0.02 probability of being disturbed by "wind" and moves in a random direction (thus, ξ determines the wall location and the wind probability). The three budgets are 200, 600, and 1000, respectively. Subgoal locations are limited to the continuous subset of $\mathbb{R}^2$ which contains the grid, e.g. $\Theta = ([0, 10] \times [0, 10])^2$.

Let $\theta_i^{*,n}$ be the subgoal recommendation in iteration n of algorithm $i \in \{\text{BESD}, \text{HB}, \text{EI}, \text{LCB}\}$. (For the baselines, $\theta_i^{*,n}$ is the subgoal with the highest score up until iteration n. Note that this makes their performance only look stronger.) To show the performance of algorithm i, we take its recommendation



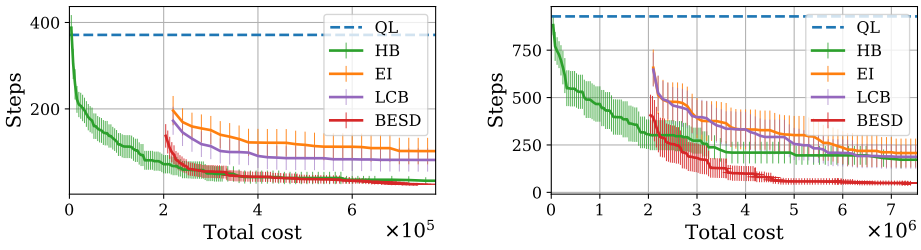
(a) GW10 Domain (b) GW20 Domain

Figure 1: Recommendation paths

in each iteration and test it on a sample of MDPs from the distribution group. The reported result is the mean performance over this sample. Fig. 1a displays four realizations of the initial data and the "recommendation paths" of BESD, defined as $(\theta^{*,1}_{\texttt{BESD}}, \ldots, \theta^{*,n}_{\texttt{BESD}})$ for GW10. Each color corresponds to one realization, and the color becomes deeper as n increases, with the most light points being the initial samples. The circles and triangles represent the first and second subgoals respectively.

We point out two "subgoal-pairs of interest" using 'A' and 'B' labels, which are commonly recommended pairs of subgoals in later iterations of BESD (darker circle and triangle pairs). Generally speaking, the first subgoal "path" moves toward the upper right corner, which motivates the agent to bypass the random wall. The second subgoal has a trend toward the upper left corner, directing the agent towards the goal after bypassing the wall. Note that the agent itself follows the standard RL paradigm and does not have any knowledge about the model of the gridworld.

Three-Room Gridworlds (GW20). The next domain is a distribution of 20×20 gridworlds with three rooms separated by two walls. As shown in Fig. 1b, the walls are randomly located in the middle rows (dark gray). A door of size 8 is randomly located somewhere within the wall (light gray). The budgets τ_{min}, τ_{mid} and τ_{max} are 4000, 7000, and 10000, respectively. Recommendation paths are shown in Fig. 1b. As more environments are observed, the first subgoal moves downward, toward the entrance of the first door. The second subgoal converges toward the exit of the second door, moving the agent near the goal.



(a) GW10 Domain (b) GW20 Domain

Figure 2: Performance curve as a function of cost

We report in Figure 2 the mean performance of the best subgoal design found by an algorithm as a function of the total cumulative cost. The error bars indicate $\pm$ 2 standard errors of the mean. We note that BESD is able to learn even using small budgets (and indeed, it relies on these low-cost observations often); therefore, it is much cheaper than training with budget τ_{max}. The initial sample cost of BESD is the same as that of EI and LCB in our simulated environments, since we can get the observations of τ_{min}, τ_{mid}, and τ_{max} from a sample with τ_{max}. BESD outperforms EI and LCB significantly in both the expected score and the stability of the recommendations. On the other hand, the initial sample is expensive when compared to HB, making HB competitive at the beginning. As the training iteration grows, BESD quickly outperforms HB in the long run.

REFERENCES

Joshua Achiam and Shankar Sastry. Surprise-based intrinsic motivation for deep reinforcement learning. *arXiv preprint arXiv:1703.01732*, 2017.

Haitham Bou Ammar, Eric Eaton, Paul Ruvolo, and Matthew Taylor. Online multi-task learning for policy gradient methods. In *International Conference on Machine Learning*, pp. 1206–1214, 2014.

Dimitrios S Apostolopoulos, Liam Pedersen, Benjamin N Shamah, Kimberly Shillcutt, Michael D Wagner, and William L Whittaker. Robotic antarctic meteorite search: Outcomes. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 4, pp. 4174–4179. IEEE, 2001.

Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems*, pp. 1471–1479, 2016.

Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

Eric Brochu, Vlad M Cora, and Nando De Freitas. A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599*, 2010.

Yuri Burda, Harri Edwards, Deepak Pathak, Amos Storkey, Trevor Darrell, and Alexei A Efros. Large-scale study of curiosity-driven learning. *arXiv preprint arXiv:1808.04355*, 2018.

Scott Clark, Eric Liu, Peter Frazier, JiaLei Wang, Deniz Oktay, and Norases Vesdapunt. Moe: A global, black box optimization engine for real world metric optimization. `https://github.com/Yelp/MOE`, 2014.

Dennis D Cox and Susan John. A statistical method for global optimization. In *Proceedings of 1992 IEEE International Conference on Systems, Man, and Cybernetics*, pp. 1241–1246. IEEE, 1992.

Nicholas T Dee, Mostafa Bedewy, Abhinav Rao, Justin Beroz, Byeongdu Lee, Eric R Meshot, Cécile AC Chazot, Piran R Kidambi, Hangbo Zhao, Thomas Serbowicz, et al. In situ mechanochemical modulation of carbon nanotube forest growth. *Chemistry of Materials*, 2018.

Marc Peter Deisenroth, Peter Englert, Jan Peters, and Dieter Fox. Multi-task policy search for robotics. In *2014 IEEE International Conference on Robotics and Automation (ICRA 2014)*, 2014.

Tobias Domhan, Jost Tobias Springenberg, and Frank Hutter. Speeding up automatic hyperparameter optimization of deep neural networks by extrapolation of learning curves. In *IJCAI*, volume 15, pp. 3460–8, 2015.

David Ferguson, Aaron Morris, Dirk Haehnel, Christopher Baker, Zachary Omohundro, Carlos Reverte, Scott Thayer, Charles Whittaker, William Whittaker, Wolfram Burgard, et al. An autonomous robotic system for mapping abandoned mines. In *Advances in Neural Information Processing Systems*, pp. 587–594, 2004.

Fernando Fernández and Manuela Veloso. Learning domain structure through probabilistic policy reuse in reinforcement learning. In *Progress in Artificial Intelligence*, volume 2, pp. 13–27. Springer, 2013.

Matthias Feurer, Jost Tobias Springenberg, and Frank Hutter. Initializing Bayesian hyperparameter optimization via meta-learning. In *AAAI*, pp. 1128–1135, 2015.

Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Ian Osband, Alex Graves, Vlad Mnih, Remi Munos, Demis Hassabis, Olivier Pietquin, et al. Noisy networks for exploration. *arXiv preprint arXiv:1706.10295*, 2017.

Peter Frazier, Warren Powell, and Savas Dayanik. The knowledge-gradient policy for correlated normal beliefs. *INFORMS Journal on Computing*, 21(4):599–613, 2009.

J González. GPyOpt: A Bayesian optimization framework in Python. `http://github.com/SheffieldML/GPyOpt`, 2016.

Ivo Grondman, Lucian Busoniu, Gabriel AD Lopes, and Robert Babuska. A survey of actor-critic reinforcement learning: Standard and natural policy gradients. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(6):1291–1307, 2012.

Xiaoxiao Guo, Satinder Singh, Honglak Lee, Richard L Lewis, and Xiaoshi Wang. Deep learning for real-time atari game play using offline Monte-Carlo tree search planning. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger (eds.), *Advances in Neural Information Processing Systems 27*, pp. 3338–3346. Curran Associates, Inc., 2014.

Xiaoxiao Guo, Satinder Singh, Richard Lewis, and Honglak Lee. Deep learning for reward design to improve Monte Carlo tree search in atari games. *arXiv preprint arXiv:1604.07095*, 2016.

Abhishek Gupta, Russell Mendonca, YuXuan Liu, Pieter Abbeel, and Sergey Levine. Meta-reinforcement learning of structured exploration strategies. *arXiv preprint arXiv:1802.07245*, 2018.

Josiah P Hanna and Peter Stone. Grounded action transformation for robot learning in simulation. In *AAAI*, pp. 3834–3840, 2017.

Henry C Herbol, Weici Hu, Peter Frazier, Paulette Clancy, and Matthias Poloczek. Efficient search of compositional space for hybrid organic–inorganic perovskites via Bayesian optimization. *NPJ Computational Materials*, 4(1):51, 2018.

Xiao Huang and John Weng. Novelty and reinforcement learning in the value system of developmental robots. In *2nd International Workshop on Epigenetic Robotics: Modeling Cognitive Development in Robotic Systems*. Lund University Cognitive Studies, 2002.

Daniel R Jiang, Emmanuel Ekwedike, and Han Liu. Feedback-based tree search for reinforcement learning. In *International Conference on Machine Learning*, pp. 2284–2293, 2018.

Donald R Jones, Matthias Schonlau, and William J Welch. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455–492, 1998.

Frédéric Kaplan and Pierre-Yves Oudeyer. Maximizing learning progress: An internal reward system for development. In *Embodied Artificial Intelligence*, pp. 259–270. Springer, 2004.

Michael Kearns and Satinder Singh. Near-optimal reinforcement learning in polynomial time. *Machine Learning*, 49(2-3):209–232, 2002.

Aaron Klein, Stefan Falkner, Simon Bartels, Philipp Hennig, and Frank Hutter. Fast Bayesian optimization of machine learning hyperparameters on large datasets. *arXiv preprint arXiv:1605.07079*, 2016.

George Konidaris and Andrew Barto. Autonomous shaping: Knowledge transfer in reinforcement learning. In *Proceedings of the 23rd International Conference on Machine Learning*, pp. 489–496. ACM, 2006.

Guillaume Lample and Devendra Singh Chaplot. Playing fps games with deep reinforcement learning. In *AAAI*, pp. 2140–2146, 2017.

Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: Bandit-based configuration evaluation for hyperparameter optimization. In *ICLR 2017*, 2016.

Daniel J Mankowitz, Augustin Žídek, André Barreto, Dan Horgan, Matteo Hessel, John Quan, Junhyuk Oh, Hado van Hasselt, David Silver, and Tom Schaul. Unicorn: Continual learning with a universal, off-policy agent. *arXiv preprint arXiv:1802.08294*, 2018.

Larry Matthies, Erann Gat, Reid Harrison, Brian Wilcox, Richard Volpe, and Todd Litwin. Mars microrover navigation: Performance evaluation and enhancement. *Autonomous robots*, 2(4): 291–311, 1995.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

Jonas Močkus. On Bayesian methods for seeking the extremum. In *Optimization Techniques IFIP Technical Conference*, pp. 400–404. Springer, 1975.

Erica EM Moodie, Bibhas Chakraborty, and Michael S Kramer. Q-learning for estimating optimal dynamic treatment rules from observational data. *Canadian Journal of Statistics*, 40(4):629–645, 2012.

Igor Mordatch, Nikhil Mishra, Clemens Eppner, and Pieter Abbeel. Combining model-based policy search with online model learning for control of physical humanoids. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pp. 242–248. IEEE, 2016.

Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*, volume 99, pp. 278–287, 1999.

Pavel Nikolaev, Daylond Hooper, Nestor Perea-Lopez, Mauricio Terrones, and Benji Maruyama. Discovery of wall-selective carbon nanotube growth conditions via automated experimentation. *ACS nano*, 8(10):10214–10222, 2014.

Junhyuk Oh, Xiaoxiao Guo, Honglak Lee, Richard L Lewis, and Satinder Singh. Action-conditional video prediction using deep networks in atari games. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 28*, pp. 2863–2871. Curran Associates, Inc., 2015.

Ian Osband and Benjamin Van Roy. Why is posterior sampling better than optimism for reinforcement learning? In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 2701–2710. JMLR. org, 2017.

Ian Osband, Daniel Russo, and Benjamin Van Roy. (More) Efficient reinforcement learning via posterior sampling. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger (eds.), *Advances in Neural Information Processing Systems 26*, pp. 3003–3011. Curran Associates, Inc., 2013.

Ian Osband, Benjamin Van Roy, and Zheng Wen. Generalization and exploration via randomized value functions. *arXiv preprint arXiv:1402.0635*, 2014.

Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped DQN. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 29*, pp. 4026–4034. Curran Associates, Inc., 2016.

Ian Osband, Daniel Russo, Zheng Wen, and Benjamin Van Roy. Deep exploration via randomized value functions. *arXiv preprint arXiv:1703.07608*, 2017.

Georg Ostrovski, Marc G Bellemare, Aaron van den Oord, and Rémi Munos. Count-based exploration with neural density models. *arXiv preprint arXiv:1703.01310*, 2017.

Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International Conference on Machine Learning (ICML)*, volume 2017, 2017.

Matthias Poloczek, Jialei Wang, and Peter Frazier. Multi-information source optimization. In *Advances in Neural Information Processing Systems*, pp. 4288–4298, 2017.

Warren B Powell. *Approximate Dynamic Programming: Solving the curses of dimensionality*, volume 703. John Wiley & Sons, 2007.

Niranjani Prasad, Li-Fang Cheng, Corey Chivers, Michael Draugelis, and Barbara E Engelhardt. A reinforcement learning approach to weaning of mechanical ventilation in intensive care units. *arXiv preprint arXiv:1704.06300*, 2017.

Jette Randløv and Preben Alstrøm. Learning to drive a bicycle using reinforcement learning and shaping. In *ICML*, volume 98, pp. 463–471. Citeseer, 1998.

Carl Edward Rasmussen and Christopher K.Ĩ. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006. ISBN 0-262-18253-X.

Mark Bishop Ring. *Continual learning in reinforcement environments*. PhD thesis, University of Texas at Austin, 1994.

Daniel Russo and Benjamin Van Roy. Learning to optimize via posterior sampling. *Mathematics of Operations Research*, 39(4):1221–1243, 2014.

Andrei A Rusu, Matej Vecerik, Thomas Rothörl, Nicolas Heess, Razvan Pascanu, and Raia Hadsell. Sim-to-real robot learning from pixels with progressive nets. *arXiv preprint arXiv:1610.04286*, 2016.

Wolfram Schultz. Predictive reward signal of dopamine neurons. *Journal of Neurophysiology*, 80(1): 1–27, 1998.

Warren Scott, Peter Frazier, and Warren Powell. The correlated knowledge gradient for simulation optimization of continuous parameters using gaussian process regression. *SIAM Journal on Optimization*, 21(3):996–1026, 2011.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354, 2017.

Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical Bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems*, pp. 2951–2959, 2012.

Jonathan Sorg, Richard L Lewis, and Satinder P Singh. Reward design via online gradient ascent. In *Advances in Neural Information Processing Systems*, pp. 2190–2198, 2010.

Jonathan Sorg, Satinder P Singh, and Richard L Lewis. Optimal rewards versus leaf-evaluation heuristics in planning agents. In *AAAI*, 2011.

Bradly C Stadie, Sergey Levine, and Pieter Abbeel. Incentivizing exploration in reinforcement learning with deep predictive models. *arXiv preprint arXiv:1507.00814*, 2015.

Kevin Swersky, Jasper Snoek, and Ryan P Adams. Multi-task bayesian optimization. In *Advances in Neural Information Processing Systems*, pp. 2004–2012, 2013.

Kevin Swersky, Jasper Snoek, and Ryan Prescott Adams. Freeze-thaw Bayesian optimization. *arXiv preprint arXiv:1406.3896*, 2014.

Fumihide Tanaka and Masayuki Yamamura. An approach to lifelong reinforcement learning through multiple environments. In *6th European Workshop on Learning Robots*, pp. 93–99, 1997.

Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, OpenAI Xi Chen, Yan Duan, John Schulman, Filip DeTurck, and Pieter Abbeel. # exploration: A study of count-based exploration for deep reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 2753–2762, 2017.

Ana C Tenorio-Gonzalez, Eduardo F Morales, and Luis Villaseñor-Pineda. Dynamic reward shaping: Training a robot by voice. In *Ibero-American Conference on Artificial Intelligence*, pp. 483–492. Springer, 2010.

Risto Vuorio, Dong-Yeon Cho, Daejoong Kim, and Jiwon Kim. Meta continual learning. *arXiv preprint arXiv:1806.06928*, 2018.

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

Christopher John Cornish Hellaby Watkins. *Learning from delayed rewards*. PhD thesis, King's College, Cambridge, 1989.

Aaron Wilson, Alan Fern, Soumya Ray, and Prasad Tadepalli. Multi-task reinforcement learning: A hierarchical Bayesian approach. In *Proceedings of the 24th International Conference on Machine Learning*, pp. 1015–1022. ACM, 2007.

Chao Yu, Minjie Zhang, Fenghui Ren, and Guozhen Tan. Emotional multiagent reinforcement learning in spatial social dilemmas. *IEEE Transactions on Neural Networks and Learning Systems*, 26(12):3083–3096, 2015.

Zeyu Zheng, Junhyuk Oh, and Satinder Singh. On learning intrinsic rewards for policy gradient methods. In *Advances in Neural Information Processing Systems*, pp. 4649–4659, 2018.

A HYPERPARAMETER ESTIMATION

The hyperparameters of the covariance function k are set via *maximum a posteriori* (MAP) estimation. Recall that a MAP estimate is the mode under the log-posterior obtained as the sum of the log-marginal likelihood of the observations and the logarithm of the probability under a hyper-prior. We focus on describing the hyper-prior, since the log-marginal likelihood follows canonically; see (Rasmussen & Williams, 2006, Ch. 5) for details. The proposed prior extends the hyper-prior for the multi-task GP model used in (Poloczek et al., 2017). We set the mean function μ and the noise function λ to constants. For the covariance function we need to estimate $d+5$ hyperparameters: the signal variance, one length scale for every subgoal parameter in θ and the four parameters associated with k_τ. We suppose a normal prior for these parameters. For the signal variance, the prior mean is given by the variance of the observations, after subtracting the above estimate for the observational noise. Here we use the independence of observational noise that we argued above. For any length scale, we set the prior mean to the size of the interval that the associated parameter is chosen in. Having determined a prior mean μ_ψ for each hyperparameter ψ, we may then set the variance of the normal prior to $\sigma_\psi^2 = (\mu_\psi/2)^2$.

B COMPUTATION OF THE ACQUISITION VALUE

This section details how to find the maximum GiS per effort on the testing MDP in period n efficiently. This approach follows (Scott et al., 2011). Note that any finite set of subgoals $((\theta,\tau))_n$ has a joint multivariate normal distribution under the posterior given the history at time H^n, thus $\mu^*_{n+1}(\theta) = \mathbb{E}_{n+1}[f(\theta,\cdot)]$ can be written as

$$\mu^*_{n+1}(\theta) = \mu^*_n(\theta) + \tilde{\sigma}_n(\theta, \theta^{n+1}, \tau^{n+1}) \cdot Z^{n+1},$$

where Z^{n+1} is a standard normal random variable, $\tilde{\sigma}^2_n(\theta, \theta^{n+1}, \tau^{n+1}) = Var_n[g(\theta,\tau_{\max})] - \mathbb{E}_n\big[Var_{n+1}[g(\theta,\tau_{\max})] \mid \theta^{n+1}, \tau^{n+1}\big]$ quantifies the effect that the observation at $(\theta^{n+1},\tau^{n+1})$ has on the posterior distribution of θ. Let $\mathbf{\Theta}'$ be a discrete set that $\mathbf{\Theta}' \subseteq \mathbf{\Theta}$, $|\mathbf{\Theta}'| = L < \infty$. We take the maximum in GiS per effort over these L points as an approximation:

$$\begin{aligned}\mathrm{GiS}^n(\theta,\tau) &\approx \mathbb{E}_n\big[\max_{\theta'\in\mathbf{\Theta}'} \mu^n(\theta',\tau_{\max}) + \tilde{\sigma}_n(\theta,\theta^{n+1},\tau^{n+1})Z^{n+1}\big] - \max_{\theta'\in\mathbf{\Theta}'} \mu(\theta',\tau_{\max}) \\ &= h\big(\mu^n(\mathbf{\Theta}',\tau_{\max}), \tilde{\sigma}_n(\mathbf{\Theta}',\theta^{n+1},\tau^{n+1})\big),\end{aligned}$$

where

$$\begin{aligned}\mu^n(\mathbf{\Theta}',\tau_{\max}) &= \big(\mu^n(\theta_i,\tau_{\max})\big)_{i=1}^L \\ \tilde{\sigma}_n(\mathbf{\Theta}',\theta^{n+1},\tau^{n+1}) &= \big(\tilde{\sigma}_n(\theta_i,\theta^{n+1},\tau^{n+1})\big)_{i=1}^L,\end{aligned}$$

and function $h : \mathbb{R}^L \times \mathbb{R}^L \to \mathbb{R}$ is defined by $h(a,b) = \mathbb{E}[\max_i a_i + b_i Z] - \max_i a_i$, where a and b are any deterministic vectors, and Z is a one-dimensional standard normal random variable. Denote $\mu^n(\theta_i,\tau_{\max})$ and $\tilde{\sigma}_n(\theta_i,\theta^{n+1},\tau^{n+1})$ by $\dot{\mu}_i$ and $\dot{\sigma}_i$ respectively for short. Based on Algorithm 1 in (Frazier et al., 2009) and the evaluation of $h(a,b)$, we can get a set of k indices $\{j_1, j_2, \ldots, j_k\} \in \{1,2,\ldots,L\}$ such that

$$\mathrm{GiS}^n(\theta,\tau) = \sum_{i=1}^{k-1} \big(\dot{\sigma}_{j_{i+1}} - \dot{\sigma}_{j_i}\big) f\left(-\Big|\frac{\dot{\mu}_{j_{i+1}} - \dot{\mu}_{j_i}}{\dot{\sigma}_{j_{i+1}} - \dot{\sigma}_{j_i}}\Big|\right),$$

where $f(z) = \varphi(z) + z\Phi(z)$ with φ and Φ being the standard normal cdf and pdf. This shows how to compute the gain in score. To get the GiS per effort, we divide both sides by the budget τ^{n+1}.

B.1 GAIN IN SCORE GRADIENT COMPUTATION

If the sum has only a single term, i.e., $k = 1$ holds, then

$$\begin{aligned}\mathrm{GiS}^n(\theta,\tau) &= 0 \\ \nabla\mathrm{GiS}^n(\theta,\tau) &= 0\end{aligned}$$

If $k > 1$, then by direct computation, we have

$$\nabla \mathrm{GiS}^n(\theta, \tau) = \sum_{i=1}^{k-1} \big(-\nabla \dot{\sigma}_{j_{i+1}} + \nabla \dot{\sigma}_{j_i}\big) \varphi\left(\left| \frac{\dot{\mu}_{j_{i+1}} - \dot{\mu}_{j_i}}{\dot{\sigma}_{j_{i+1}} - \dot{\sigma}_{j_i}} \right| \right)$$

Thus we must compute $\nabla \dot{\sigma}_{j_i}, \forall i \in \{1, ..., k\}$. Again, by direct computation we have

$$\nabla \tilde{\sigma}_n(\theta_{j_i}, \theta^{n+1}, \tau^{n+1}) = \beta_1 \beta_2 - \frac{1}{2} \beta_1^3 \beta_2 [\beta_5 - \beta_4]$$

Where

$$\begin{aligned}
\beta_1 &= [k((\theta^{n+1}, \tau^{n+1}), (\theta^{n+1}, \tau^{n+1})) - \gamma^T A_n^{-1} \gamma]^{-1/2} \\
\beta_2 &= B(\theta, n+1) - [B(\theta, 1) \cdots B(\theta, n)] A_n^{-1} \gamma \\
\beta_3 &= (\nabla B(\theta, n+1) - \nabla(\gamma^T) A_n^{-1} \begin{bmatrix} B(\theta, 1) \\ \vdots \\ B(\theta, n) \end{bmatrix} \\
\beta_4 &= 2 \nabla(\gamma^T) A_n^{-1} \gamma \\
\beta_5 &= \nabla k((\theta^{n+1}, \tau^{n+1}), (\theta^{n+1}, \tau^{n+1})) \\
\gamma &= \begin{pmatrix} k((\theta^{n+1}, \tau^{n+1}), (\theta^1, \tau^1)) \\ \vdots \\ k((\theta^{n+1}, \tau^{n+1}), (\theta^n, \tau^n)) \end{pmatrix} \\
A_n &= (k((\theta^i, \tau^i), (\theta^j, \tau^j))^n + \mathrm{diag}((\tilde{\sigma}_n^2(\theta^i, \tau^i))_{j=1}^n) \\
B(\theta, i) &= \int k((\theta, \tau), (\theta^i, \tau^i) p(\tau) d\tau
\end{aligned}$$

Following (Scott et al., 2011), we compute the partial derivatives of GiS^n with respect to θ and τ and use the gradient based optimizer `L-BFGS-B` with restarts to find a global optimizer of the aquisition criterion.

C Description of Baseline Algorithms

1. The first baseline is the Q-Learning (`QL`) algorithm of Watkins (1989) with no subgoals or reward shaping: that is, we directly run `QL` on environment ξ^{N+1} for $\tau_{\max}$ interactions. The reported results are average performances over 200 realizations of ξ^{N+1} for each domain.
2. The popular Hyperband (`HB`) algorithm of Li et al. (2016) treats hyperparameter optimization as a pure-exploration infinite-armed bandit problem; it uses sophisticated techniques for adaptive resource allocation and early-stopping to concentrate its learning efforts on promising designs. Setting $\eta = 3$ (the default value) and $R = 81$, `HB` consists of $\lfloor \log_\eta R \rfloor$ rounds. The first round starts with R samples of subgoal designs θ from a Latin hypercube sample. Each θ is evaluated for $\tau_{\min}$ steps and then only the best $1/\eta$-fraction designs are kept for the next round. In round i, Hyperband samples R/η^{i-1} subgoal designs to evaluate for $\tau_{\min}\, \eta^{i-1}$ steps.
3. The expected improvement (`EI`) algorithm (Močkus, 1975; Jones et al., 1998) is the most popular BO method. `EI` allocates one sample in each round, selecting a point that maximizes the expected improvement beyond currently sampled points: $\theta^{n+1} = \arg\max_\theta \mathbb{E}_n \big[[y(\theta, \tau_{\max}) - \mu_n^*]^+ \big]$. In each iteration, we evaluate the `EI` selection using $\tau_{\max}$ iterations. `EI` is implemented in `Python` 2.7 using the `GPyOpt` package (González, 2016).
4. The lower-confidence-bound (`LCB`) algorithm (Cox & John, 1992) controls the exploration-exploitation trade-off using a "bonus term" proportional to the standard deviation at each point: $\mathtt{LCB}(\theta) = \mu(\theta, \tau_{\max}) - \kappa \sqrt{\lambda(\theta, \tau_{\max})}$. The parameter κ is set to 2. `LCB` is also implemented in `Python` 2.7 using the `GPyOpt` package (González, 2016)

D BENEFITS OF BESD RECOMMENDATION

Table 1 displays the ratio of the scores of an agent using subgoals versus an agent learning from scratch. The implemented subgoal is one particular recommendation of BESD. GW20, KEY and MC are measured per 1000, 500 and 1000 steps respectively. The agent's task in these environments is to reach the goal as quickly as possible. Performance is measured by the expected number of steps the agent takes from start to goal so that a small ratio reflects the advantage of the recommended subgoal design θ.

Table 1: Performance ratio of subgoal agent versus standard agent

No.	GW20	KEY	MC
1	0.779	0.604	0.980
2	0.492	0.643	1.048
3	0.234	0.448	0.949
4	0.224	0.471	0.896
5	0.108	0.506	0.987
6	0.088	0.498	0.878
7	0.068	0.184	1.077
8	0.075	0.260	0.877
9	0.059	0.232	0.974
10	0.058	0.332	0.869

E DETAILED REVIEW OF RELATED WORK

Reinforcement learning has recently been tremendously successful in complex sequential decision-making problems, with applications in a number of domains, such as games (Mnih et al., 2013; Silver et al., 2017), robotics (Rusu et al., 2016; Mordatch et al., 2016; Hanna & Stone, 2017), healthcare (Moodie et al., 2012; Prasad et al., 2017) and other areas (Yu et al., 2015; Bojarski et al., 2016). State-of-art algorithms (e.g. Grondman et al. (2012); Mnih et al. (2013); Guo et al. (2014)), however, inherently require a large number of observations from the environment and converge slowly in complex problems, effectively limiting their application to problems where a fast simulation is available. Moreover, the problem of exploration in unknown environments continues to pose a challenge for many RL algorithms (Osband et al., 2013; 2014; 2016; Fortunato et al., 2017).

A core challenge in RL is the trade-off between exploration and exploitation. The former one seeks out novel states and actions and may improve future performance, while the latter one maximizes short-term gains. Naive exploration strategies such as ϵ-greedy can lead to exponentially large data requirements. Many algorithms employ optimism approaches that encourage exploration in poorly-understood states and actions by assigning an optimistic bonus (Kearns & Singh, 2002; Tang et al., 2017), parametrizing density estimates for state visits and utilizing pseudo-counts (Bellemare et al., 2016; Ostrovski et al., 2017) and learning the dynamics and choosing action that leads to states that are poorly-explored (Stadie et al., 2015) or most dissimilar to recent states (Oh et al., 2015). Some works focus on posterior sampling (Russo & Van Roy, 2014; Osband & Van Roy, 2017). Others leverage values by randomizing value functions (Osband et al., 2014; 2017) and modeling the Q-value distribution via the bootstrap (Osband et al., 2016). Fortunato et al. (2017) adds parametric noise to the weights of neural networks and Gupta et al. (2018) develops a gradient based method to learn exploration strategies from prior experience.

The implementation of *intrinsic reward* (also called *intrinsic motivation*) is inspired by the response of dopamine neurons to sensory stimuli (Schultz, 1998). Intrinsic reward helps robots learn increasingly complex behavior in a self-motivated way. There are many different sources of intrinsic reward, including novelty (Huang & Weng, 2002), learning progress (Kaplan & Oudeyer, 2004), curiosity (Pathak et al., 2017; Burda et al., 2018), and surprise (Achiam & Sastry, 2017). Sorg et al. (2010); Guo et al. (2016) treat the intrinsic reward as parameters that influence the outcome of the planning process and train it via gradient ascent. Sorg et al. (2011) uses intrinsic reward as an alternative to the leaf-evaluation heuristic approach and extends Policy Gradient for Reward Design (PGRD) to

learn the optimal reward. Zheng et al. (2018) designs an algorithm to learn the intrinsic rewards for policy-gradient based learning agents. Potential-based reward shaping (PBRS) (Ng et al., 1999) provides a way to modify the reward function that simultaneously maintains the optimal policy and potentially accelerates learning. PBRS has been used in a wide range of areas, including driving a bicycle (Randløv & Alstrøm, 1998), robotic training (Tenorio-Gonzalez et al., 2010), and video game artificial intelligence (Lample & Chaplot, 2017; Jiang et al., 2018). The agent's performance is highly dependent on an informative reward signal (Ng et al., 1999); however, manual reward shaping can be a labor-intensive process that often requires a deep domain expertise.

To overcome the variation of the environment in real-world application, RL researchers are recently interested in leveraging the experience accumulated from previous tasks to improve the agent's performance in a new task, including continual learning, transfer learning and meta learning. The idea has been realized by hierarchically learning from an easy task to a complicated task (Ring, 1994), learning the relationship among the MDPs (Wilson et al., 2007), initializing the parameters of the new task by using the information from previous tasks (Tanaka & Yamamura, 1997; Konidaris & Barto, 2006), probabilistic policy reuse (Fernández & Veloso, 2013), learning a parametrized policy that generalizes across tasks (Deisenroth et al., 2014), multi-task policy gradient method (Ammar et al., 2014), utilizing goal-conditioned value function (Mankowitz et al., 2018), and distinguishing the important parameters to previous tasks (Vuorio et al., 2018).

Our approach of designing subgoals follows the Bayesian optimization (BO) paradigm that has recently emerged as powerful technique for the optimization of black-box functions, in particular for tuning ML models and design of experiments (Brochu et al., 2010; Snoek et al., 2012; Herbol et al., 2018). Our work bears resemblance and generalizes methods for network architecture search and optimization with multiple information sources (Swersky et al., 2013; 2014; Feurer et al., 2015; Domhan et al., 2015; Li et al., 2016; Klein et al., 2016; Poloczek et al., 2017). Their work differs in that they may observe directly the whole curve of the loss during training, whereas our setting only allows to observe the score of the policy after the underlying exploration process has completed (as motivated by the real-world applications described above).