Dynamic Subgoal-based Exploration via Bayesian Optimization

Anonymous authors
Paper under double-blind review

Abstract

Policy optimization in unknown, sparse-reward navigation environments with expensive and limited interactions is challenging and poses a need for effective exploration. Motivated by complex navigation tasks that require real-world training (when cheap simulators are not available), we consider an agent that faces an unknown distribution of environments and must decide on an exploration strategy, through a series of training environments, that can benefit policy learning in a test environment drawn from the environment distribution. Most existing approaches focus on fixed exploration strategies, while the few that view exploration as a meta-optimization problem tend to ignore the need for *cost-efficient* exploration. We propose a cost-aware Bayesian optimization (BO) approach that efficiently searches over a class of dynamic subgoal-based exploration strategies. The algorithm adjusts a variety of levers — the locations of the subgoals, the length of each episode, and the number of replications per trial — in order to overcome the challenges of sparse rewards, expensive interactions, and noise. Our experimental evaluation demonstrates that, when averaged across problem domains, the proposed algorithm outperforms the meta-learning algorithm MAML by 19%, the hyperparameter tuning method Hyperband by 23%, and BO techniques EI and LCB by 24% and 22%, respectively. We also provide a theoretical foundation and prove that the method asymptotically identifies a near-optimal subgoal design from the search space.

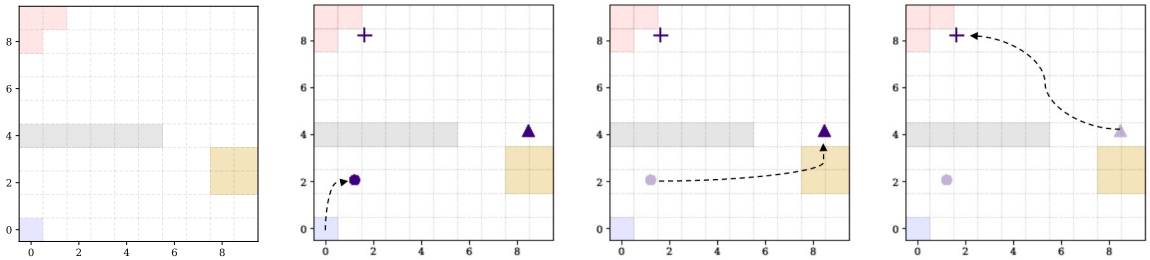
1 Introduction

Reinforcement learning (RL) is becoming the standard for approaching control problems – usually modeled by a Markov decision process (MDP) – in environments whose dynamics are unknown and learned from data. In many applications involving navigation tasks, rewards are sparse and delayed, and since most RL algorithms rely, at least initially, on random exploration, this can cause an agent to require a large, often impractical number of interactions with the environment before obtaining any rewards. Simultaneously, in real-world settings, it is often the case that fast and cheap interactions with the environment are *not available*, making it nearly impossible to apply RL algorithms. To address the two issues of sparse rewards and expensive interactions in navigation tasks, our goal in this paper is to design methods for learning better exploration policies in a *cost-efficient* manner: specifically, we propose a Bayesian optimization approach to optimize a subgoal-based exploration strategy.

An illustrative example comes from the field of robotics: autonomous systems have long been used to explore unknown or dangerous terrains (Matthies et al., 1995; Apostolopoulos et al., 2001; Ferguson et al., 2004; Thrun et al., 2004). Policies learned offline (e.g., via a simulator) are common in these situations, but it may be beneficial to introduce agents that execute an offline-learned exploration policy to guide the learning of an *online* policy that can better tailor to the details of the test environment. Matthies et al. (1995) describe the design of a rover for the Mars Pathfinder mission, where one of the main tasks is navigating the rover in a rocky terrain and reaching a goal. To train for the eventual mission, the engineers utilized an "indoor arena" that mimics the true environment. The need for cost-efficient training also arises in the setting of safe robot navigation (Oliveira et al., 2020). Existing approaches to exploration have largely ignored the need

to be cost-efficient during training process and therefore are challenging to apply in the scenarios described here (see Section 2 for a detailed discussion of related work).

In our setup, an agent is given a fixed (and small) number of opportunities to train in environments randomly drawn from a distribution Ξ (henceforth, we refer to these as "training environments"), with the caveat that each interaction in the training environment *incurs a cost*. After these opportunities are exhausted, the agent enters a random *test environment* $\xi \sim \Xi$ and executes an underlying RL algorithm to adapt to the particulars of ξ, while aided by the higher-level exploration strategy learned for Ξ. One can view this formulation as a meta-optimization problem with two levels: an upper-level problem to select an exploration strategy, represented by parameters θ, and a lower-level RL problem that explores with the help of the exploration strategy θ on an environment instance $\xi \sim \Xi$.



(a) Original (b) First subgoal (c) Second subgoal (d) Third subgoal

Figure 1: Example of a dynamic subgoal exploration strategy. The first, second, and third subgoals are denoted by the circle, triangle, and cross, respectively. The blue square is the starting location of the agent, the grey region is a wall, the yellow region is the location of the key, and the red region is the door (goal). Note that although the second subgoal is not exactly in the location of the key, it brings the agent to the correct vicinity, allowing the underlying RL algorithm (executed online) to further adapt to the environment's particular details.

We propose optimizing over a class of *dynamic subgoal exploration strategies* in the upper-level optimization problem. To illustrate this concept, consider the sparse-reward environment shown in Figure 1a, where an agent is tasked with picking up a "key" in the yellow region, in order to exit the "door" in the red region. The grey region is a wall. An RL algorithm paired with a naive exploration strategy making use of random actions (such as ϵ-greedy) requires a prohibitively large number of random actions before finding a suitable path to the door through the key, while avoiding the wall. A dynamic subgoal strategy is an *ordered* set of subgoals (along with associated rewards leading to each subgoal, omitted here for illustrative clarity) that leads the agent on a trajectory where the underlying RL algorithm is *more likely to discover* the optimal behavior. Figures 1b-1d together show an example of a dynamic subgoal exploration with three subgoals, which first leads the agent to the vicinity of the key and later towards the door. Note that the situation here in Figure 1 is simplified in that we are actually interested in finding dynamic subgoal strategies that work on average across a distribution of environments, rather than a single environment.

1.1 Our Contributions

Our contributions are as follows. We first propose a framework for *cost-efficient learning* of a dynamic subgoal exploration strategy for a distribution of environments; in other words, interactions with the environment are expensive *during training*, making most gradient-based approaches infeasible. We instead leverage the Bayesian optimization (BO) paradigm, a well-known class of sample-efficient optimization techniques (Brochu et al., 2010; Snoek et al., 2012; Herbol et al., 2018; Frazier, 2018), and propose a new acquisition function as a core ingredient of our approach. The Gaussian process (GP) surrogate model used by the BO formulation has the ability to reason about the *learning curve* of the underlying RL algorithm, enabling us to introduce two additional levers in the BO learning process to improve cost-efficiency: (1) how long to run each episode of training, (2) the number of replications to run in each training environment. These levers allow us to intelligently trade-off running a longer trial versus more replications of shorter trials; the motivation is

Figure 2: Outline of the BESD algorithm. During the training phase BESD optimizes an exploration strategy (represented as subgoals) on sampled training environments. It then utilizes the learned subgoal design as an exploration strategy in the test environment to train an effective policy within a limited number of interactions.

that, given $\tau_1 < \tau_2$, an accurate evaluation of a particular exploration strategy θ after τ_1 steps may be more informative than a noisy evaluation of θ after τ_2 steps, even though the same number of environment interactions are used in both cases. The proposed algorithm, Bayesian exploratory subgoal design (BESD), is outlined in Figure 2. We also prove an asymptotic guarantee on the quality of the solution found by our approach, compared to the best possible subgoal-based exploration strategy within a given parameterized class.

2 Related Work

Our framework of cost-efficient learning of exploration strategies through BO appears to be distinct from existing formulations in its strong focus on expensive environmental interactions *during training*, made possible through the additional control levers of episode length and number of replications. Nevertheless, our work is related to a number of distinct areas of study: Bayesian optimization, exploration for RL, intrinsic reward and reward design in RL, multi-task RL, and transfer learning. Here, we attempt to give a tour through the various strands of relevance in each field.

2.1 Bayesian Optimization

BO is a technique for optimizing black-box functions in a sample-efficient manner, in particular for tuning ML models and design of experiments (Snoek et al., 2012; Brochu et al., 2010; Frazier, 2018; Herbol et al., 2018). Our work is related to methods for optimization with multiple information sources or fidelities (Swersky et al., 2013; 2014; Feurer et al., 2015; Domhan et al., 2015; Li et al., 2017) and in particular, the ability of our approach to select the length of an RL training episode builds upon ideas from Picheny & Ginsbourger (2013), Poloczek et al. (2017), and Klein et al. (2017). The first paper proposes fitting GP to partially converged simulations, and the latter two propose acquisition functions that consider the ratio of "information gain" to cost of evaluation. Our approach also reasons about multiple replications in an environment, similar to the problem studied in Binois et al. (2019) in the context of computer experiments. Our work fills a gap in the BO literature where the length of training and number of replications are *selected jointly* in a cost-aware setting, a natural and powerful idea that has not been considered in the literature. Our theoretical analysis builds upon techniques developed in Frazier et al. (2008) and Poloczek et al. (2017) but extend them in new directions, accounting for the ability to select the number of replications and providing a characterization of the asymptotic suboptimality due to using a discretized domain.[1]

BO has also been applied in the setting of navigation planning. Martinez-Cantin et al. (2007), Martinez-Cantin et al. (2009), and Binney et al. (2013) use BO to optimize a sequence of waypoints for a robot to follow. While our method similarly optimizes a sequence of subgoals, we use the subgoals as an exploration strategy (over a

[1]This is a common computational technique used when optimizing complex acquisition functions (Scott et al., 2011; Wu & Frazier, 2016; Poloczek et al., 2017), but none of the previous works have addressed it in theoretical analyses.

distribution of environments) on top of an existing RL algorithm, rather than as a *direct specification* of the control policy. In order to allow subgoals to provide exploration in a "plug-and-play" manner for existing RL algorithms, our approach also features a novel integration of subgoals with potential-based intrinsic rewards.

In two other works, Tesch et al. (2011) and Garcia-Barcos & Martinez-Cantin (2020), BO is directly applied to optimize a parameterized policy, but this is limited to low-dimensional parameterizations of the policy: Tesch et al. (2011) tune a two-dimensional gait parameter, while Garcia-Barcos & Martinez-Cantin (2020) tune four policy parameters. In our work, we augment an underlying RL algorithm that can learn arbitrary policies with a BO-optimized low-dimensional exploration strategy, striking a balance between flexibility and cost-efficiency.

2.2 Exploration in Reinforcement Learning

Naive exploration strategies such as ϵ-greedy can lead to unreasonably large data requirements, making exploration a commonly studied topic in RL. Most existing work focus on proposing a fixed exploration strategy that is executed for a single underlying environment. For example, some previous related work employ approaches based on optimism (Kearns & Singh, 2002; Stadie et al., 2015; Bellemare et al., 2016; Tang et al., 2017) and posterior sampling (Osband et al., 2016; Russo & Van Roy, 2014; Osband & Van Roy, 2017; Morere & Ramos, 2018) to guide exploration. Others insert an active learning (Shyam et al., 2019) or experimental design (Mehta et al., 2021) perspective into the model-based RL framework.

Our work departs from these existing studies in that we formulate the problem of exploration as a meta-optimization over a parameterized class of exploration strategies and aim to find a suitable strategy for a distribution of environments. A more closely related paper is Gupta et al. (2018), which extends the model-agnostic meta-learning (MAML) approach of (Finn et al., 2017a) to the problem of exploration for a set of tasks in a way that is similar in spirit to our formulation. However, their gradient-based approach is not sample-efficient and costly environment interactions during training is not considered. In addition, Gupta et al. (2018) makes use of task-specific parameters during training, limiting their approach to a small set of environments. For a more comprehensive list of methods for exploration in RL, we refer the reader to the excellent survey of Amin et al. (2021).

2.3 Options in Reinforcement Learning

The concept of options, which are temporally extended actions represented as a policy and a termination condition, is a way to improve the efficiency of RL through the use of previously acquired "skills" (Sutton et al., 1999; Precup et al., 1998). These skills might be acquired with the help of a human, either fully user-specified (e.g., Jothimurugan et al. (2021)) or obtained from expert demonstrations (e.g., Pan et al. (2018) and Paul et al. (2019)). Of particular relevance to our work is when options are automatically discovered, a problem that is well-known to be challenging. One stream of work views option discovery to be (at least somewhat) detached from the RL reward maximization objective, using state visitation frequencies (Stolle & Precup, 2002; McGovern & Barto, 2001; Goel & Huber, 2003), clustering (Mannor et al., 2004), novelty (Şimşek & Barto, 2004), local graph partitioning (Şimşek et al., 2005), or diversity objectives (Eysenbach et al., 2018; Zhang et al., 2021), to name a few examples. Approaches that considers a joint objective for option learning RL reward maximization objective (like ours) (Kulkarni et al., 2016; Vezhnevets et al., 2016; Bacon et al., 2017; Frans et al., 2017; Veeriah et al., 2021) typically use large, neural network-based representations along with gradient-based (meta-) optimization and do not focus on cost-aware training. The method that we propose in this paper is unique from previous works in that (1) it is designed specifically for the case where cost-aware training is warranted, (2) it offers an integrated objective for subgoal-design and RL reward maximization, and (3) it uses a novel combination of subgoals and reward shaping, which has a simpler representation than a generic option.

2.4 Intrinsic Reward and Reward Design

When a particular subgoal of our proposed dynamic subgoal exploration strategy is active, we "turn on" a set of artificial rewards that incentivize the agent to move toward that subgoal (these rewards are then removed after the agent moves on to the next subgoal). Hence, the literature on intrinsic reward and reward design in

RL are also relevant. Intrinsic reward (also called *intrinsic motivation*) helps an agent learn increasingly complex behavior in a self-motivated way (Randløv & Alstrøm, 1998; Ng et al., 1999; Huang & Weng, 2002; Kaplan & Oudeyer, 2004; Şimşek & Barto, 2006; Tenorio-Gonzalez et al., 2010; Pathak et al., 2017; Achiam & Sastry, 2017; Lample & Chaplot, 2017). Several works from the *reward design* literature are most closely related to our paper. Sorg et al. (2010) and Guo et al. (2016) directly optimize the intrinsic reward parameters, via gradient ascent, to maximize the outcome of the learning process. Similarly, Zheng et al. (2018) use intrinsic rewards in policy gradient, and treat the parameters of policy as a function of the parameters of intrinsic rewards. Again, these methods differ from ours in that they do not consider the costliness of training and focus on finding intrinsic rewards for a single MDP.

2.5 Multi-task RL and Transfer Learning

Also related to our setting are methods that aim to train agents with the capability of solving (or adapting to) multiple sequential decision making tasks (Pickett & Barto, 2002; Konidaris & Barto, 2006; Wilson et al., 2007; Fernández et al., 2010; Deisenroth et al., 2014; Doshi-Velez & Konidaris, 2016; Finn et al., 2017a;b; Pinto & Gupta, 2017; Espeholt et al., 2018; Hessel et al., 2019; Vithayathil Varghese & Mahmoud, 2020); such methods generally fall under the umbrella of *multi-task RL* or *transfer learning*. As before, many of these methods require the training of large neural networks and are not designed for a cost-aware setting. Despite their stated purpose of being sample-efficient in adapting to new tasks, most multi-task RL or transfer learning approaches do not place a strong emphasis on cost-efficiency of training on existing tasks. This is an important distinction to our work. The two papers that are closest in spirit to our work are Pickett & Barto (2002), where macro-actions are extracted from previous tasks, and Konidaris & Barto (2006), where shaped rewards are learned for a set of tasks. One drawback of Pickett & Barto (2002) is that it assumes access to optimal policies for an initial set of MDPs. Konidaris & Barto (2006) directly uses previous value functions as shaped rewards (thereby requiring the agent to solve some tasks from scratch) and does not provide an avenue for cost-effective exploration.

3 Problem Formulation

This section formulates the problem mathematically, by defining the original (sparse-reward) MDPs and how a dynamic subgoal exploration strategy induces an auxiliary, "subgoal-augmented" MDPs. We then describe the iterative training process.

3.1 Original MDPs $\mathcal{M}_\xi$ with Sparse Rewards

Consider a family of MDPs $\{\mathcal{M}_\xi = \langle \mathcal{S}, \mathcal{A}, T_\xi, R_\xi, \gamma \rangle\}_\xi$ parameterized by a random variable $\xi \sim \Xi$, where $\mathcal{S}$ and $\mathcal{A}$ are the state and action spaces, T_ξ is the transition matrix, $R_\xi : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the extrinsic[2] reward function, $\gamma \in [0, 1]$ is the discount factor[3], and Ξ is the environment distribution (not assumed to be known, nor does it need to be finite or discrete).[4] A *sparse-reward environment* is an environment where R_ξ is non-zero only for a small number of "goal" states. To ensure that all quantities are well-defined, we assume that R_ξ is bounded, as is common in the reinforcement learning literature. We assume common state and action spaces across the distribution of MDPs (i.e., they are independent of ξ), while the reward and transition functions vary with ξ.

Given $\mathcal{S}$ and $\mathcal{A}$, a *policy* π is a mapping such that $\pi(\cdot \mid s)$ is a distribution over $\mathcal{A}$ for any state $s \in \mathcal{S}$. For any $\xi \sim \Xi$, define the *value function* of policy π at any state s as

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

[2]In Section 3.3, we describe how a dynamic subgoal exploration strategy supplements the extrinsic reward function with additional *intrinsic* rewards.

[3]We allow for the case of episodic MDPs, where $\gamma = 1$, provided that any policy will reach a *terminal state* with probability one. A terminal state is absorbing and any action taken in that state gives zero reward.

[4]Note that our approach also applies to the case of a *single environment* if the distribution contains only one environment.

where the notation of "conditioning" on π and s indicates that $s_0 = s$ is the initial state and $a_t \sim \pi(\cdot \mid s_t)$. For the MDP $\mathcal{M}_\xi$, its optimal value function and associated optimal policy are

$$V_\xi^*(s) = \sup_\pi V_\xi^\pi(s) \quad \text{and} \quad \pi_\xi^*(s) \in \arg\max_{a \in \mathcal{A}} \mathbb{E}\big[R_\xi(s,a,s') + \gamma V_\xi^*(s') \mid s, a\big].$$

Now that we have defined the value function, let us comment on the environment distribution Ξ. In Section 3.4, we will formulate the meta-optimization problem, which requires that the expected performance of any policy π over the environment distribution, i.e., $\mathbb{E}_\xi[V_\xi^\pi(s)]$, is well-defined. However, since we assumed bounded rewards, implying bounded performance $V_\xi^\pi(s)$, it will always be the case that this expectation exists and we do not require further assumptions on Ξ.

When the extrinsic reward function R_ξ is sparse, it produces little to no learning signal for the agent. Under most RL algorithms, the agent essentially performs random exploration and does not start learning until the first time it wanders to the goal. The time it takes to find the goal under a random exploration strategy is often prohibitively long. The ϵ-greedy, which takes a random action with probability ϵ and the best action under the current value function approximation, is an example of a random exploration strategy.

3.2 Dynamic Subgoal Exploration Strategies

An *intrinsic reward* is an artificial reward signal experienced by the agent that does not come directly from the environment. A *subgoal* is defined by a (usually small) set of states, such that when the agent lands in any of them, the subgoal is considered "completed." A *dynamic subgoal exploration strategy* is a sequence of subgoals, along with an associated reward shaping function for each subgoal, that provides an intrinsic reward signal for the agent. If the locations of the subgoals are chosen well, this strategy can help the agent explore the environment efficiently. We call this a *dynamic* strategy because the subgoals are turned on one-by-one and consequently introduces a new state into the MDP (described in detail below).

Suppose there are K subgoals and let $\theta \in \Theta$ be a parameter that fully describes a subgoal exploration strategy (i.e., subgoal locations, associated rewards, and sequencing). Let $\mathcal{G}_{\theta,k} \subseteq \mathcal{S}$ be a set of "target" states associated with the kth subgoal, for $k \in \{1, 2, \ldots, K\}$, in the sense that if the agent lands in some state in $\mathcal{G}_{\theta,k}$, then the kth subgoal is considered "completed." In addition, we define an artificial reward function $g_{\theta,k}(s, s')$ that, when activated, provides a sequence of rewards that leads the agent toward subgoal k. Concretely, we use potential-based reward shaping from Ng et al. (1999) to achieve this. Let $\Phi_{\theta,k}$ be a *potential function*, a function that assigns a value for each state in $\mathcal{S}$, with higher potential indicating a more "valuable" state. $\Phi_{\theta,k}$ should have the property that target states in $\mathcal{G}_{\theta,k}$ have the highest potential. Then, let

$$g_{\theta,k}(s, s') = \gamma \Phi_{\theta,k}(s') - \Phi_{\theta,k}(s), \tag{2}$$

for all $s, s' \in \mathcal{S}$. The definition of $g_{\theta,k}(s, s')$ in (2) can be interpreted as the difference in potential between states s' and s (with discount γ). This potential difference motivates the agent to move towards the target states (high potential) of kth subgoal. Thus, a parameterization of a set of K subgoals, which forms our exploration strategy, is fully described by

$$\left(\{\mathcal{G}_{\theta,k}\}_{k=1}^K, \{g_{\theta,j}\}_{k=1}^K\right),$$

the locations and associated reward shaping functions.

Example 1 (Key and Door Environment) *Let us consider a distribution of maze MDPs with states* $\{(i,j)\}_{1 \le i,j \le 10}$ *and a sparse reward in the upper left corner at* $(0, 10)$*. In addition, suppose that the agent needs to pick up a key in order to receive the reward at* $(0, 10)$*, but the location of the key is uncertain but likely to be in the* right half of the room*. The environment illustrated in Figure 1 can be considered to be one possible realization from this distribution of mazes. Now, let us consider a subgoal design with* $K = 3$ *subgoals. The simple parameterization* $\theta = (i_1, j_1, i_2, j_2, i_3, j_3)$*, with*

$$\mathcal{G}_{\theta,k} = \{(i_k, j_k)\} \quad \textit{and} \quad \Phi_{\theta,k}(s) = e^{-\|s - (i_k, j_k)\|^2}$$

specifies that for $k \in \{1, 2, 3\}$, the kth subgoal is located at a single state (i_k, j_k) and the artificial reward potential is a Gaussian centered at (i_k, j_k). Using Figure 1 as a visual reference, one can imagine that the subgoal design $\theta = (1, 2, 8, 4, 2, 8)$ would be useful in guiding the agent toward the vicinity of the key on the right side of the room and then toward the vicinity of the goal. Once the agent is in the correct vicinity, the underlying RL algorithm can discover the precise locations of the key and goal in the particular environment realization more quickly.

3.2.1 Subgoal Parameterization vs State Dimensionality

For the types of navigation tasks that we are concerned with in this paper, the dimension of the subgoal parameterization θ need not scale with the dimension of the state s, which would pose a potential scalability issue. Instead, one general rule-of-thumb to keep in mind is that for a dynamic subgoal exploration strategy to be effective in navigation tasks, the dimension of θ only needs to scale with the number components of s that pertain to the *spatial positioning of the agent*. The next example provides an illustration.

Example 2 (Mountain Car Environment, with $\mathbf{dim}(\theta) < \mathbf{dim}(s)$) *Consider the well-known Mountain Car problem, a continuous control task where an underpowered car, operating in a one-dimensional space, must make its way up a steep mountain (Sutton & Barto, 2018, Example 10.1). The state is two-dimensional, $s = (x, \dot{x})$, where $x \in [-1.2, 0.5]$ is the position of the agent while $\dot{x} \in [-0.07, 0.07]$ is its velocity. A possible subgoal design with $K = 2$ is $\theta = (i_1, i_2)$, with*

$$\mathcal{G}_{\theta,k} = \{(i_k, \dot{x}) \,|\, \dot{x} \in [-0.07, 0.07]\} \quad \textit{and} \quad \Phi_{\theta,k}(s) = e^{-(x-i_k)^2}$$

for each k. In other words, the agent reaches a subgoal target state if its position is i_k, for any value of its velocity. Also, the artificial reward only depends on the spatial position x rather than the full state $(x, \dot{x})$. In Section 5, we give numerical results for exactly this example.

One could imagine that the concept illustrated in Example 2 also applies to more complex robotics environments with a high-dimensional state, but where the components related to the spatial positioning is relatively small, meaning that the subgoal parameterization (and the resulting BO problem) is often of much lower dimension than that of the state itself.

3.3 Subgoal-Augmented MDPs $\mathcal{M}_{\xi,\theta}$

Now that we have described how a particular subgoal design is parameterized, the remaining question is how these are integrated in a useful way into the original, sparse-reward MDP described in Section 3.1. We propose the notion of a *subgoal-augmented*, auxiliary MDP, where the K subgoals are sequentially "activated." This way, we encode subgoal ordering[5] into the exploration strategy, meaning that the agent only moves on to the next subgoal after finishing the current one.

Let $\mathcal{M}_{\xi,\theta}$ denote an auxiliary, subgoal-augmented MDP based on an original MDP $\mathcal{M}_\xi$, except that it is includes rewards and transitions associated with the dynamic subgoal exploration strategy θ. We introduce an auxiliary state $i \in \mathcal{I} := \{0, 1, \ldots, K\}$, where i represents the number of subgoals reached by the agent so far. Initially, we have $i_0 = 0$. The state of the $\mathcal{M}_{\xi,\theta}$ is $(s, i) \in \mathcal{S} \times \mathcal{I}$ and the transition for the auxiliary state is $i' = i + \mathbf{1}_{\{s' \in \mathcal{G}_{\theta,i+1}\}}$, where we take $\mathcal{G}_{\theta,K+1} = \varnothing$. This means the auxiliary state i is updated to $i + 1$ whenever s' reaches the next subgoal. Let the intrinsic reward of the agent be:

$$G_\theta(s_t, i_t, s_{t+1}) = \sum_{k=1}^{K} \mathbf{1}_{\{k=i_t\}} \cdot g_{\theta,k+1}(s_t, s_{t+1}),$$

where the indicator function encodes the logic that if i_t subgoals have been completed so far, then the current target is subgoal $i_t + 1$ and only the rewards leading to subgoal $j + 1$ should be active. The new reward function consists of both extrinsic (R_θ) and intrinsic (G_θ) rewards:

$$\hat{R}_{\xi,\theta}(s, i, a, s') = R_\theta(s, a, s') + G_\theta(s, i, s').$$

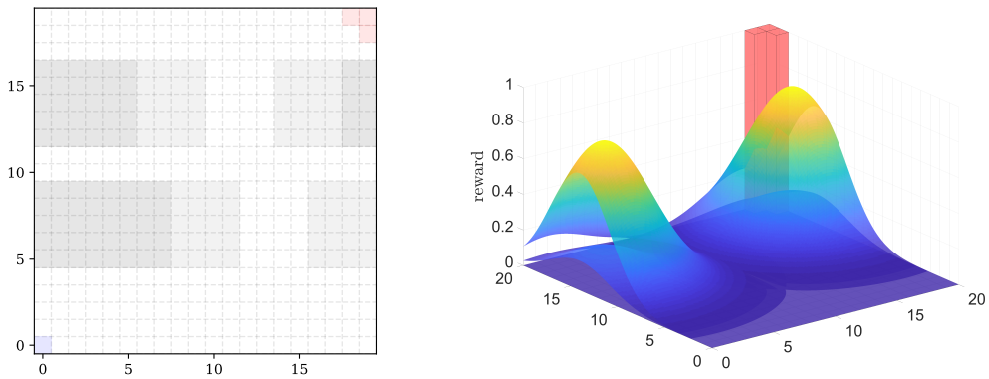
[5]Without ordering, rewards from multiple subgoals can inhibit the agent's progress.

The value function for the new MDP $\mathcal{M}_{\xi,\theta}$ is written

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

where $\hat{\pi}(\cdot|s,i)$ is now a policy defined on the new state space $\mathcal{S} \times \mathcal{I}$.

Figure 3 gives an example of the overall setup: Figure 3a shows the original MDP environment $\mathcal{M}_\xi$, where the dark gray cells are walls, the light gray represent uncertainty in the size of the "doors", and the red cells represent goal states (the sparse, extrinsic reward). Figure 3b shows the possible rewards the agent can encounter in the augmented MDP $\mathcal{M}_{\xi,\theta}$, for a random selection of subgoals θ. The original sparse reward is represented by the red bar in the corner and the first subgoal is the one that is farther from the goal. Both subgoals are singletons and the potential functions are radial basis functions centered at the subgoal locations, similar to the parameterization described in Example 1. Note that this randomly selected set of subgoals θ is not a good exploration strategy for the environment in Figure 3a (as it leads the agent toward a wall), motivating the need for optimizing their locations, as we discuss in the next section.



(a) 20×20 Gridworld Environment

(b) Goal and Subgoal Rewards

Figure 3: An example that visualizes an environment and a random dynamic subgoal exploration strategy along with the rewards of the associated subgoal-augmented MDP.

3.4 Optimizing the Exploration Strategy

Selecting the best subgoal design θ depends on the agent's underlying learning algorithm, which could in principle be any RL algorithm where intermediate rewards influence the learning process: this includes any temporal difference-based algorithm that makes use of learned value functions. In the numerical results of Section 5, our agent learns via Q-learning Watkins & Dayan (1992). We refer to the underlying RL algorithm as `RL-ALGO`. Let us use the notation `RL-ALGO`$[\tau, \mathcal{M}]$ to refer to the policy learned by `RL-ALGO` on MDP $\mathcal{M}$ after τ training interactions. We remind the reader that the subgoal-based exploration strategy is fixed *before* the test environment is revealed, so that the sequence of events in the test phase is as follows:

1. A subgoal design θ for exploration is selected.
2. The agent is placed in a new environment ξ.
3. The agent uses the subgoal-augmented MDP $\mathcal{M}_{\xi,\theta}$ and an RL algorithm with a budget of $\tau_{\max}$ interactions to learn a policy `RL-ALGO`$\big[\tau_{\max}, \mathcal{M}_{\xi,\theta}\big]$.
4. The agent's policy is evaluated using only the extrinsic reward function R_ξ of the original MDP.

Our goal is to find an exploration strategy $\theta \in \Theta$ such that a policy trained using θ behaves well in the original MDP:

$$\max_{\theta \in \Theta} \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^{t-1} R_\xi(s_t, a_t, s_{t+1}) \,\middle|\, \hat{\pi}_{\xi,\theta}^{\tau_{\max}}, (s_0, i_0)\right] \text{ with } \hat{\pi}_{\xi,\theta}^{\tau_{\max}} = \texttt{RL-ALGO}\big[\tau_{\max}, \mathcal{M}_{\xi,\theta}\big], \tag{4}$$

where (s_0, i_0) is the initial augmented state. The interpretation of the objective in (4) is as follows: Evaluate $\hat{\pi}_{\xi,\theta}^{\tau_{\max}}$ (a policy for the subgoal-augmented MDP) using the same dynamics as the subgoal-augmented MDP, but *without* the rewards associated with the subgoals. In other words, the policy internally considers subgoals, but is not rewarded for it explicitly. This explains why we need to consider a starting state (s_0, i_0), rather than just s_0 (note however that the reward does not depend on the auxiliary state i_t).

The expectation in (4) is taken over the random choice of a test environment ξ, the stochastic dynamics within $\mathcal{M}_\xi$, and the stochasticity of the learning algorithm itself. It is convenient to explicitly define the following:

$$u(\theta, \tau) = \mathbb{E}\left[\sum_{t=1}^{\infty} \gamma^{t-1} R_\xi(s_t, a_t, s_{t+1}) \,\middle|\, \hat{\pi}_{\xi,\theta}^{\tau}, s_0, i_0\right] \text{ with } \hat{\pi}_{\xi,\theta}^{\tau} = \texttt{RL-ALGO}\big[\tau, \mathcal{M}_{\xi,\theta}\big].$$

Note that the objective function in (4) is $u(\theta, \tau_{\max})$, but the notation $u(\theta, \tau)$ will be useful in Section 4, where we discuss using fewer than $\tau_{\max}$ interactions to learn about $u(\theta, \tau_{\max})$ as a way of reducing cost.

3.5 Iterative Training and Additional Cost-Reduction Levers

In our setting, we observe the performance of exploration strategies and the resulting policies in a sequence of training environment realizations $\xi^1, \xi^2, \ldots, \xi^N$ drawn from the MDP distribution Ξ. By default, each complete evaluation of the objective function in (4) $u(\theta, \tau_{\max})$ for a fixed θ requires running `RL-ALGO` for $\tau_{\max}$ interactions. Since each interaction in the training environments is expensive (e.g., in robotics applications, this could involve time, labor, and equipment), we want to consider ways to reduce the number of training interactions. To do so, we propose two additional levers:

1. **Maximum number of interactions.** For each training environment ξ^n, we allow the specification of a maximum number of interactions τ^n, chosen from a finite set $\mathcal{T}$. In episodic tasks, if an episode finishes before τ^n interactions are used, we start a new episode and continue in this manner until exactly τ^n environment interactions are exhausted. In the next section, we describe our probabilistic model of the RL training curve, which allows observations of short episodes to be informative about the final performance. This also can reduce the risk of spending too many interactions with an unpromising exploration strategy.
2. **Multiple replications.** We can reduce the variance of performance observations by averaging over the observed cumulative reward over q^n i.i.d. replications, for a total of $\tau^n q^n$ interactions in training environment ξ^{n+1}. We suppose that q^n is chosen from a finite set $\mathcal{Q}$. The idea here is that even with the same number of total interactions, a lower variance observation of a "preliminary" result could be more informative than a higher variance observation of the "full" result.

To summarize, three decisions are made at the beginning of each training opportunity n: (1) a choice of subgoal design θ^n, (2) the maximum number of interactions τ^n, and (3) the number q^n of independent replications to use for this particular θ^n. For each of the q^n replications, we obtain a policy

$$\hat{\pi}_{\xi^{n+1},\theta^n}^{\tau^n} = \texttt{RL-ALGO}\big[\tau^n, \mathcal{M}_{\xi^{n+1},\theta^n}\big],$$

before observing a estimate of its performance. After the q^n training replications are complete, we compute the average performance over the q^n replications. Written more succinctly, our observation in episode n takes the form

$$y^{n+1}(\theta^n, \tau^n, q^n) = u(\theta^n, \tau^n) + \varepsilon_{\text{env}}^{n+1} + \varepsilon_{\text{rep}}^{n+1}(q^n),$$

where $\varepsilon_{\text{env}}^{n+1}$ represents the deviation from the $u(\theta^n, \tau^n)$ due to the random environment ξ^{n+1}, while the observation noise $\varepsilon_{\text{rep}}^{n+1}(q^n)$ represents the noise that can be reduced via multiple replications, i.e., the noise

in $\hat{\pi}^{\tau^n}_{\xi^{n+1},\theta^n}$ due to a sample run of `RL-ALGO`. Thus, $\varepsilon^{n+1}_{\text{rep}}(q^n)$ depends on the number of replications q^n. Naturally, a larger number of replications implies a smaller observation noise. Note that the observations $\{y^n\}$ are i.i.d., since a new MDP is sampled in each iteration. The total training cost incurred is cumulative number of interactions: $\sum_{n=0}^{N-1} \tau^n q^n$.

After training opportunities $0, 1, \ldots, N-1$, we reach the *test phase* and commit to a final subgoal design θ^N_{rec}. This design is evaluated on the test MDP $\xi^{N+1} \sim \Xi$ with an agent that has a full budget of $\tau_{\max}$ interactions.

4 Bayesian Optimization for Cost-Efficient Exploration

The setup for BO typically consists of two components: (1) a probabilistic surrogate model (usually a Gaussian process) for modeling the objective function, and (2) an acquisition function, a strategy for selecting the next point to sample that encodes a way to balance exploration with exploitation. The BO "loop" repeats the following: sample a point (using a combination of the current surrogate model and acquisition function), observe new data, and update the surrogate model. For a detailed tutorial, we refer readers to Frazier (2018).

For our setting of learning a dynamic subgoal exploration strategy, we use a tailored probabilistic model for the RL learning curve and an acquisition function for selecting the next subgoal design, the maximum episode length, and the number of replications to run. Although shorter episodes and smaller number of replications are more cost-efficient, they also decrease the chance of reaching the goal and produce higher observation noise; the acquisition function must carefully trade off these downsides with the cost of interactions. We call this the *Bayesian Exploratory Subgoal Design* (`BESD`) acquisition function.

4.1 Surrogate Model

In order to enable the ability to dynamically select the maximum episode length of training, as described in Section 3.5, our approach uses a GP surrogate model over $u(\theta, \tau)$, rather than $u(\theta, \tau_{\max})$. In other words, our model is a function of both θ and τ rather than just θ, enabling it to capture the performance of a policy trained with subgoals θ, for a variety of episode lengths. Assume that $\Theta \subseteq \mathbb{R}^m$. We place a GP prior f on the latent function u with mean function $\mu : \Theta \times \mathcal{T} \to \mathbb{R}$ and covariance function $k : (\Theta \times \mathcal{T}) \times (\Theta \times \mathcal{T}) \to \mathbb{R}_+$. More precisely, to capture the structure of the RL learning curve, we set μ to the mean of an initial set of samples and use a multidimensional product kernel, based on the kernel used in Klein et al. (2017), which is known to work well in modeling ML performance as a function of parameters and time:

$$k\big((\theta, \tau), (\theta', \tau')\big) = k_\theta(\theta, \theta')\, k_\tau(\tau, \tau'), \tag{5}$$

where the first kernel k_θ is the (5/2)-Matérn kernel and k_τ is a polynomial kernel $k_\tau(\tau, \tau') = \phi(\tau)^\intercal\, \Sigma_\phi\, \phi(\tau')$ with $\phi(\tau) = (1, \tau)^\intercal$ and hyperparameters Σ_ϕ. Note that the covariance under k is large only if the covariance is large under both k_θ and k_τ. We make the modeling assumption that $\varepsilon^{n+1}_{\text{env}}$ and $\varepsilon^{n+1}_{\text{rep}}(q^n)$ are independent, zero mean, and normally distributed[6] with variances σ^2_{env} and $\sigma^2_{\text{rep}}/q^n$, respectively. This allows us to take advantage of standard GP machinery to analytically compute the posterior on f conditioned on the history after n steps. This posterior is another GP, whose mean and kernel functions are denoted $\mu^n(\theta, \tau)$ and $k^n((\theta, \tau), (\theta', \tau'))$; the exact expressions can be found in, e.g., Rasmussen (2004).

We remind the reader that the dimensionality of the GP surrogate model is $\dim(\Theta) + 1$, i.e., the dimension of the subgoal parameterization, along with an additional dimension for τ. As illustrated in Example 2, it will often be the case for navigation domains that the dimension of the subgoal parameterization is smaller than that of the state space of the underlying RL problem (due to the relatively small number of spatial components of the state). Therefore, dynamic subgoal exploration strategies can be tractably modeled and optimized for broad classes of navigation problems, even with vanilla GPs.[7]

[6] Although the assumption of normality is commonplace in BO for tractability of the posterior (Frazier, 2018), other noise distributions can be used through an appropriate likelihood function (but this is often difficult to know a priori).

[7] Of course, when the need arises to optimize for high dimensional subgoal parameterizations, one may opt for scalable extensions of the model and optimization formulation (e.g., Chen et al. (2012); Djolonga et al. (2013); Wang et al. (2016); Mutny & Krause (2018) along with many more recent papers). We leave extensions in this direction to future work and focus on a more standard setting.

Algorithm 1 Bayesian Exploratory Subgoal Design

1. Set $n = 0$. Estimate hyperparameters of the GP prior f using initial samples.
2. Compute next decision (θ^n, τ^n, q^n) according to the acquisition function (7).
3. Train in environment ξ^{n+1} augmented with θ^n ($\mathcal{M}_{\xi^{n+1},\theta^n}$) using levers (τ^n, q^n).
4. Observe $y^{n+1}(\theta^n, \tau^n)$ and update posterior on f.
5. If $n < N$, increment n and return to Step 2.
6. Return a subgoal recommendation θ^N_{rec} that maximizes $\mu^N(\theta, \tau_{\max})$.

4.2 Acquisition Function

As described above, our framework proceeds in iterations, selecting one set of subgoals θ^n along with τ^n and q^n, to be evaluated in each training environment. We now propose the acquisition function for making these evaluation decisions. An overview of the BO setup is given in Algorithm 1.

Suppose the training budget is used up after training iterations $0, 1, \ldots, N-1$. Then, the optimal risk-neutral decision is to use subgoals on the test MDP ξ^{N+1} that have maximum expected performance under the posterior. The expected score of this choice is μ^n_* where

$$\mu^n_* := \max_\theta \mu^n(\theta, \tau_{\max}), \tag{6}$$

where $\mu^n(\theta, \tau_{\max}) = \mathbb{E}_n[f(\theta, \tau_{\max})]$. Here $\mathbb{E}_n$ is the conditional expectation with respect to the history after the first n observations: $(\theta^0, \tau^0, q^0, y^1, \ldots, \theta^{n-1}, \tau^{n-1}, q^{n-1}, y^n)$. Note that although we are allowed to use fewer than $\tau_{\max}$ interactions in training environments to reduce cost, the agent uses its full budget for the test MDP ξ^{N+1}.

We take the knowledge gradient, one-step lookahead approach (Frazier et al., 2008; 2009), i.e., we imagine for each training MDP that it is the last opportunity before the test MDP and act optimally. Full lookahead approaches require solving an intractable dynamic programming problem; however, we show that nonetheless, the one-step approach is asymptotically optimal in Theorem 1 and Theorem 2. If we evaluate (θ, τ, q), i.e., the subgoals θ for τ steps and q replications, then the expected gain in performance in the test MDP of the recommended exploration strategy after the evaluation, based on (6), with respect to the current best is

$$\nu^n(\theta, \tau, q) = \mathbb{E}_n\left[\mu^{n+1}_* \mid \theta^n = \theta, \tau^n = \tau, q^n = q\right] - \mu^n_*.$$

Therefore, the one-step optimal strategy is to choose the next subgoals θ^n, maximum episode length τ^n, and number of replications q^n so that ν^n is maximized. However, this strategy would generally allocate a maximum number of steps $\tau_{\max}$ and replications $q_{\max}$ for the evaluation of the next subgoal design, as observing $\tau_{\max}$ during training is most informative, and repeating for $q_{\max}$ replications reduces the noise maximally. In other words, this strategy does not consider the cost of training.

Hence, we propose an acquisition strategy that maximizes the *gain in performance per effort*:

$$(\theta^n, \tau^n, q^n) \in \arg\max_{\theta,\tau,q} \frac{\nu^n(\theta, \tau, q)}{q\tau}. \tag{7}$$

The optimization problem (7) is challenging when the domain Θ is continuous, so we take the approach of replacing it with a discrete domain $\bar{\Theta} \subseteq \Theta$ (for example, this could be selected by a Latin Hypercube design). This approach has been applied successfully in other knowledge gradient style acquisition functions (Scott et al., 2011; Wu & Frazier, 2016; Poloczek et al., 2017). Unlike previous work however, we provide a novel theoretical guarantee on the asymptotic suboptimality of a discretized optimization domain; see Theorem 2 in the next section.

4.3 Theoretical Analysis

We now provide our main theoretical results on the asymptotic optimality of BESD. Detailed proofs can be found in Appendix A. For convenience, we suppose $\mu(\theta, \tau) = 0$ for all (θ, τ), and further assume that the kernel $k(\cdot, \cdot)$ has continuous partial derivatives up to the fourth order. Recall that $\theta^N_{\text{rec}} \in \bar{\Theta}$ is the final recommendation made in iteration N:

$$\theta^N_{\text{rec}} \in \arg\max_{\theta \in \bar{\Theta}} \ \mu^N(\theta, \tau_{\max}).$$

Our first theorem is concerned with the finite, discretized optimization domain $\bar{\Theta}$.

Theorem 1 *The acquisition function described in (7) has the property of asymptotic optimality with respect to $\bar{\Theta}$, i.e.,*

$$\lim_{N \to \infty} f(\theta^N_{rec}, \tau_{max}) = \max_{\theta \in \bar{\Theta}} f(\theta, \tau_{max}),$$

almost surely. That is, the recommended design θ^N_{rec} becomes optimal as $N \to \infty$.

If the optimization domain $\bar{\Theta} = \Theta$, then Theorem 1 suffices. Unfortunately, for many applications, the subgoal parameterizations will naturally be continuous. Next, we provide an additive bound on the difference between the solution of BESD in $\bar{\Theta}$ and the unknown optimum in Θ, as the number of iterations N tends to infinity.

We use a probabilistic Lipschitz constant of a GP from Lederer et al. (2019) to quantify the performance with respect to the full, continuous subgoal parameter space. We make use of the fact that the derivative $df(\theta, \tau_{\max})/d\theta_i$ is another GP with covariance

$$k^{\partial i}(\theta, \theta') = \frac{\partial^2}{\partial \theta_i \partial \theta'_i} \, k\big((\theta, \tau_{\max}), (\theta', \tau_{\max})\big),$$

for all $i = 1, 2, \ldots, m$ (Rasmussen & Williams, 2006, Section 9.4). See also Ghosal et al. (2006) and Wu et al. (2017) for other uses of this property. For each $i = 1, 2, \ldots, m$, define the constant

$$L^i_\delta = k^\partial_{\max} \sqrt{2 \log\Big(\frac{2m}{\delta}\Big)} + 12\sqrt{6m} \max\Big\{k^\partial_{\max}, \sqrt{L^{\partial i}_k \max_{\theta, \theta' \in \Theta} \operatorname{dist}(\theta, \theta')}\Big\}, \tag{8}$$

where $L^{\partial i}_k$ be a Lipschitz constant of the kernel $k^{\partial i}$ and $k^\partial_{\max} = \max_{\theta \in \Theta} \sqrt{k^{\partial i}(\theta, \theta)}$.

Theorem 2 *The acquisition function of (7) has bounded asymptotic suboptimality with respect to the original domain Θ in the sense that with probability at least $1 - \delta$, it holds that*

$$\lim_{N \to \infty} f(\theta^N_{\text{rec}}, \tau_{\max}) \geq \max_{\theta \in \Theta} f(\theta, \tau_{\max}) - d \cdot \|L_\delta\|$$

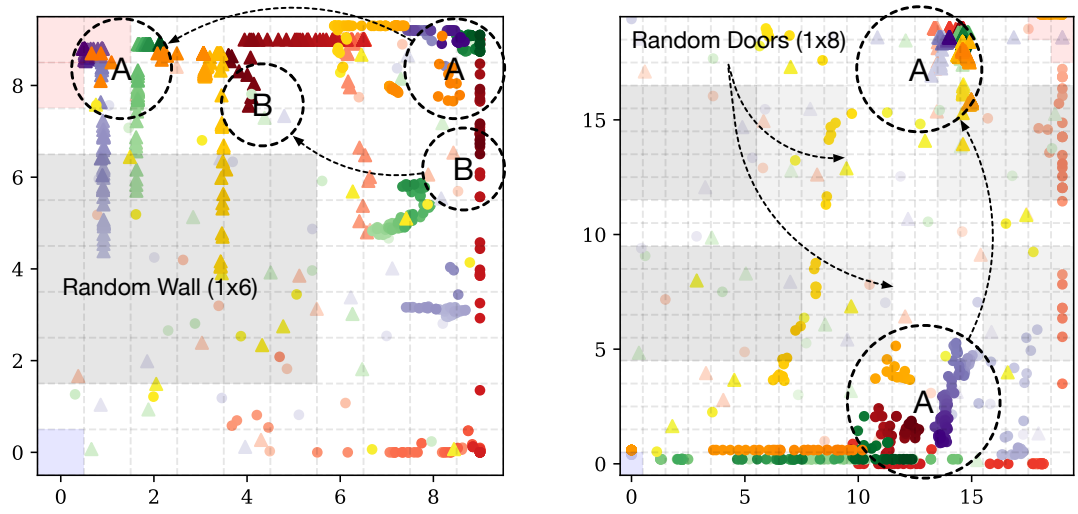
where $d = \max_{\theta \in \Theta} \min_{\theta' \in \bar{\Theta}} \operatorname{dist}(\theta, \theta')$ is a measure on the "coarseness" of the discretization and L_δ is the vector $(L^1_\delta, L^2_\delta, \ldots, L^m_\delta)$, with each L^i_δ defined as in (8).

5 Numerical Experiments

We now provide numerical experiments to demonstrate the cost-effectiveness of the BESD framework. BESD was implemented using the MOE package (Clark et al., 2014) and will be open-sourced upon acceptance of the manuscript. In the experiments that follow, we use the proposed BESD approach to optimize dynamic subgoal exploration strategies consisting of two or three subgoals.

BESD is given a few choices for the interaction length τ and number of replications q (values reported for each benchmark below). Each replication of the BESD is given an initial set of 10 observations for each value of τ (these initial observations incur interaction costs just like future observations). The potential function at state s with the jth subgoal activated is $\Phi_j(s) = w_1 \exp[-0.5(s - j)^2/w_2]$, where the "height" is $w_1 = 0.2$ and "width" is $w_2 = 10$.

In the next few subsections, we first introduce each of the environments and along the way, show some qualitative results obtained by BESD, focusing on providing intuition. Later, in Section 5.6, we introduce the baseline algorithms used for an empirical comparison, and in Section 5.7, provide a discussion of those results.



(a) GW10 Domain

(b) GW20 Domain

Figure 4: Recommendation paths for GW10 and GW20. The blue and red shaded regions denote the starting points and goals, respectively. Dark and light gray regions possible locations of walls and doors, respectively. Each plot displays four realizations of the "recommendation paths" of BESD. Each color corresponds to one sample realization, and the color becomes darker as n increases, with the lightest points being the initial samples. The circles and triangles represent the first and second subgoals, respectively, of the exploration strategy. The 'A' and 'B' labels point out two example sets of subgoals displaying notable behaviors.

5.1 Windy Gridworlds with Walls

The first set of environments (GW10) is a distribution over 10×10 gridworlds, where the goal is to reach the upper left square that is shaded red in Figure 4a to collect a reward of one. The agent starts from the lower-left grid square shaded in blue and may in each step choose an action from the action space consisting of the four compass directions. Each gridworld is partitioned by a wall into two rooms. The wall, randomly located in one of the middle five rows in the gridworld, has a door located on four grid squares on its right. The agent will stay in the current location when it hits the wall.

There is a small amount of "wind" or noise in the transition: the agent moves in a random direction with a probability that is itself uniformly distributed between 0 and 0.02 (thus, a particular environment instance drawn from the distribution has a random wall location and wind probability).

We use $\mathcal{T} = \{200, 600, 1000\}$ for the possible values of τ and $\mathcal{Q} = \{5, 20\}$ for the possible values of q. We parameterize the exploration strategy using two subgoals, whose locations are optimized. Subgoal locations are limited to the continuous subset of $\mathbb{R}^2$ which contains the grid, i.e., $\Theta = ([0, 10] \times [0, 10])^2$ for GW10.

5.1.1 Recommendation Paths for GW10

In order to visualize the qualitative behavior of BESD, we show in Figure 4a the evolution of the recommended subgoals over time (iterations), a concept that we refer to as a *recommendation path*. The plot displays four recommendation path realizations of BESD using distinct colors. Within each color, the lightest points are the initial samples while the darker points represent recommendations for larger n. Also within each color, the circles represent the first subgoal of the exploration strategy, while the triangles represent the second subgoal. We point out two types of exploration behaviors discovered by BESD in Figure 4a:

- Behavior 'A': The pairs of regions labeled 'A' are the final recommendations of the orange, green, and purple sample paths. The strategy leads the agent toward the upper right corner (away from the wall), and then after that, directly towards the goal.
- Behavior 'B': The final recommendation of the red sample path is labeled by 'B.' Note that in behavior 'A', a direct path to the first subgoal (upper right corner) is blocked by the random wall for some realizations of the environment. Behavior 'B' might be interpreted as a slight remedy of this situation by targeting a lower region of the right edge, creating a more direct path around the wall.

Both strategies appear to be reasonable ways for the agent to avoid the door and head to the goal.

5.2 Larger, Three-Room Windy Gridworlds

The second domain (GW20) is a distribution of larger 20×20 gridworlds with three rooms separated by two walls. As shown in Figure 4b, the walls are randomly located in the middle rows (dark gray). A door of size 8 is randomly located somewhere within the wall, shaded in light gray. The starting location is the blue square in the lower left and the goal is displayed in red in the upper right. As in GW10, we optimize the locations of a two-subgoal exploration strategy, with $\Theta = ([0, 20] \times [0, 20])^2$. The noise due to wind is the same as in GW10. In this experiment, we consider the case of only allowing `BESD` to select the maximum episode length from $\mathcal{T} = \{4000, 7000, 10000\}$, while keeping $q = 20$ fixed.

5.2.1 Recommendation Paths for GW20

Recommendation paths are shown in Figure 4b. Unlike the case of GW10, all four of the realizations converge to roughly the same exploration strategy, labeled by 'A.' Focusing on the lighter red and orange circles, we can notice a trend of the first subgoal initially being placed (naively) near the goal, but as learning progresses, they move 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.

Regarding the placement of the first subgoal near the goal and inducing a direct path, it is worth pointing out this strategy might work for *some* environments (i.e., those where the first door is at its leftmost position and the second door is at its rightmost position). However, `BESD` learns that in order to perform well *across the distribution* of environments, the strategy of first moving rightward is better.

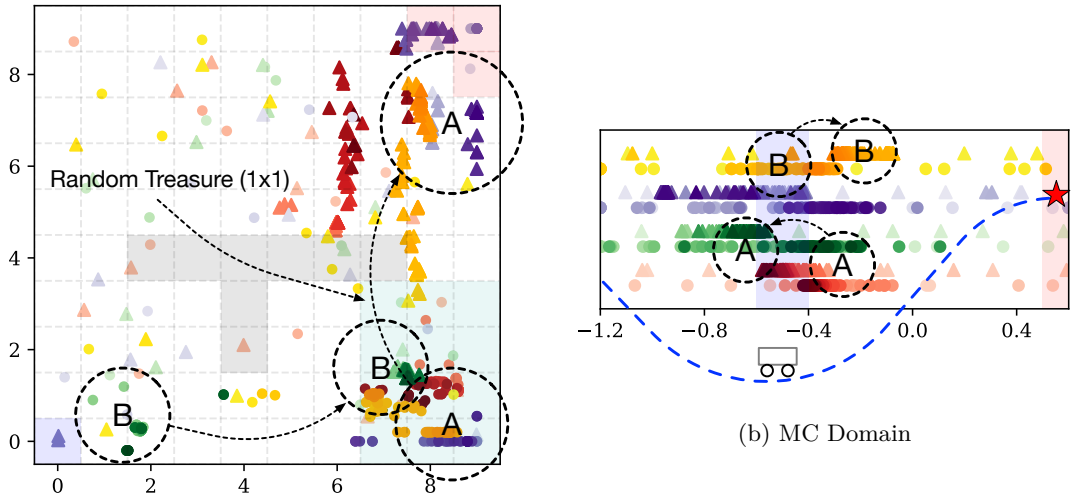
5.3 Treasure-in-Room

The third domain (TR) is a distribution of 10×10 gridworlds with a "treasure" hidden in a small room; see Figure 5a. The light green area shows the possible positions of the treasure. The agent gets a reward of 10 upon entering the square with treasure, and a reward of 10 upon reaching the goal. The cumulative reward, however, is zero if the agent does not find the goal within the interaction budget. The discount factor is set to $\gamma = 0.98$ to encourage policies that collect the reward earlier. We set $\mathcal{T} = \{400, 1200, 2000\}$ and $\mathcal{Q} = \{5, 20\}$.

5.3.1 Recommendation Paths for TR

The recommendation paths for TR are shown in Figure 5a. We observe that two strategies were discovered by `BESD` across these four realizations:

- Behavior 'A': This appears to be the ideal behavior and was discovered in the orange, purple, and red sample paths: first lead the agent to the treasure and then toward the goal through the upper right. It is also notable that the first subgoal is located at the *bottom* of the room, meaning that wherever the treasure turns out to be, the agent can pick it up without backtracking.
- Behavior 'B': The green sample path's final recommendation coincides with the (apparently suboptimal) exploration strategy denoted by 'B' simply leads the agent to the treasure, but does not provide any guidance toward the goal. We highlight that this is an instance where `BESD`'s learning is not yet complete, evidenced by the fact that behavior 'B' is often recommended in *earlier iterations of the orange sample path*. In that case however, `BESD` eventually discovers behavior 'A' in later iterations.



(a) TR Domain

(b) MC Domain

Figure 5: Recommendation paths for TR and MC. The first panel, Figure 5a, largely follows the same design as Figures 4a and 4b, except that the green squares represent possible location of the treasure. In the second panel, Figure 5b, since the location of the mountain-car is one-dimensional, we visualize the four recommendation paths by spacing them vertically to avoid crowding (therefore, the vertical axis has no meaning and is simply an expansion of the horizontal axis). The initial location of the car is colored in blue, while the goal is in red, corresponding to the overlay of the mountain.

5.4 The Mountain Car Problem (MC)

The mountain car (MC) domain, as we introduced in Example 2, is a commonly used RL benchmark environment that tests an agent's ability to explore, as it is required to go in the opposite direction of the goal in order to reach the top of the mountain; see, e.g., (Sutton & Barto, 2018, Example 10.1). For this experiment, we created a distribution of environments Ξ by randomizing the starting location of the agent, which is chosen uniformly from $[-0.6, -0.4]$. Here, we set $\mathcal{T} = \{4000, 7000, 10000\}$ and $\mathcal{Q} = \{10, 50\}$.

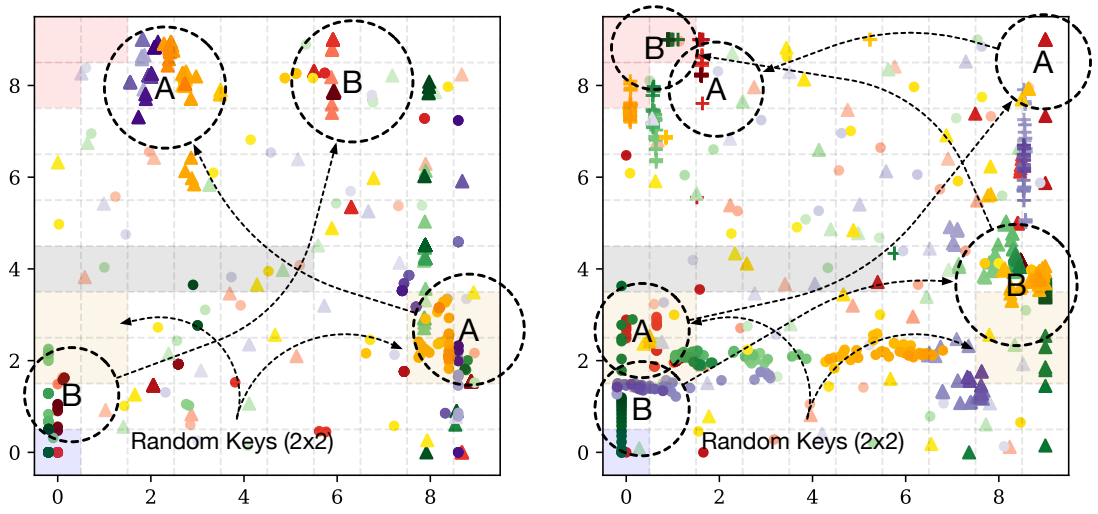
5.4.1 Recommendation Paths for MC

The subgoal-pairs discovered by `BESD` are shown in Figure 5b; they tend to be on opposite sides of the agent's starting location, thereby creating back-and-forth movement needed to generate momentum and move up the mountain. It is worth noting that the symmetric behaviors of going from left to right (Behavior 'B' in Figure 5b, for the orange sample path) and going from right to left (Behavior 'A', exhibited by the green, red, and purple sample paths) can both be found in the results of `BESD`.

5.5 Key-Door with Highly Varying Key Locations (KEY2 and KEY3)

In our last experiment, we test for the situation where the distribution of environments Ξ contains environments that might vary dramatically from one another. We also consider how the exploration behavior changes when we add an additional subgoal to the strategy.

In domains KEY2 (with two subgoals) and KEY3 (with three subgoals), we consider a 10×10 gridworld with one wall, where a "key" needs to be picked up before opening a closed door at the upper-right corner of the grid. The location of the key, however, is highly varying and is either near the left wall or the right wall. The environment is visualized in Figures 6a and 6b. We set $\mathcal{T} = \{400, 700, 1000\}$ and $\mathcal{Q} = \{5, 20\}$.



(a) KEY2 Domain

(b) KEY3 Domain

Figure 6: Recommendation paths. The blue and red shaded regions denote the starting points and goals, respectively. Dark and light gray regions possible locations of walls and doors, respectively. Each plot displays four realizations of the "recommendation paths" of BESD. Each color corresponds to one sample realization, and the color becomes darker as n increases, with the lightest points being the initial samples. The circles, triangles, and crosses represent the first, second, and third subgoals, respectively. The 'A' and 'B' labels point out two example sets of subgoals displaying notable behaviors.

5.5.1 Recommendation Paths for KEY2/KEY3

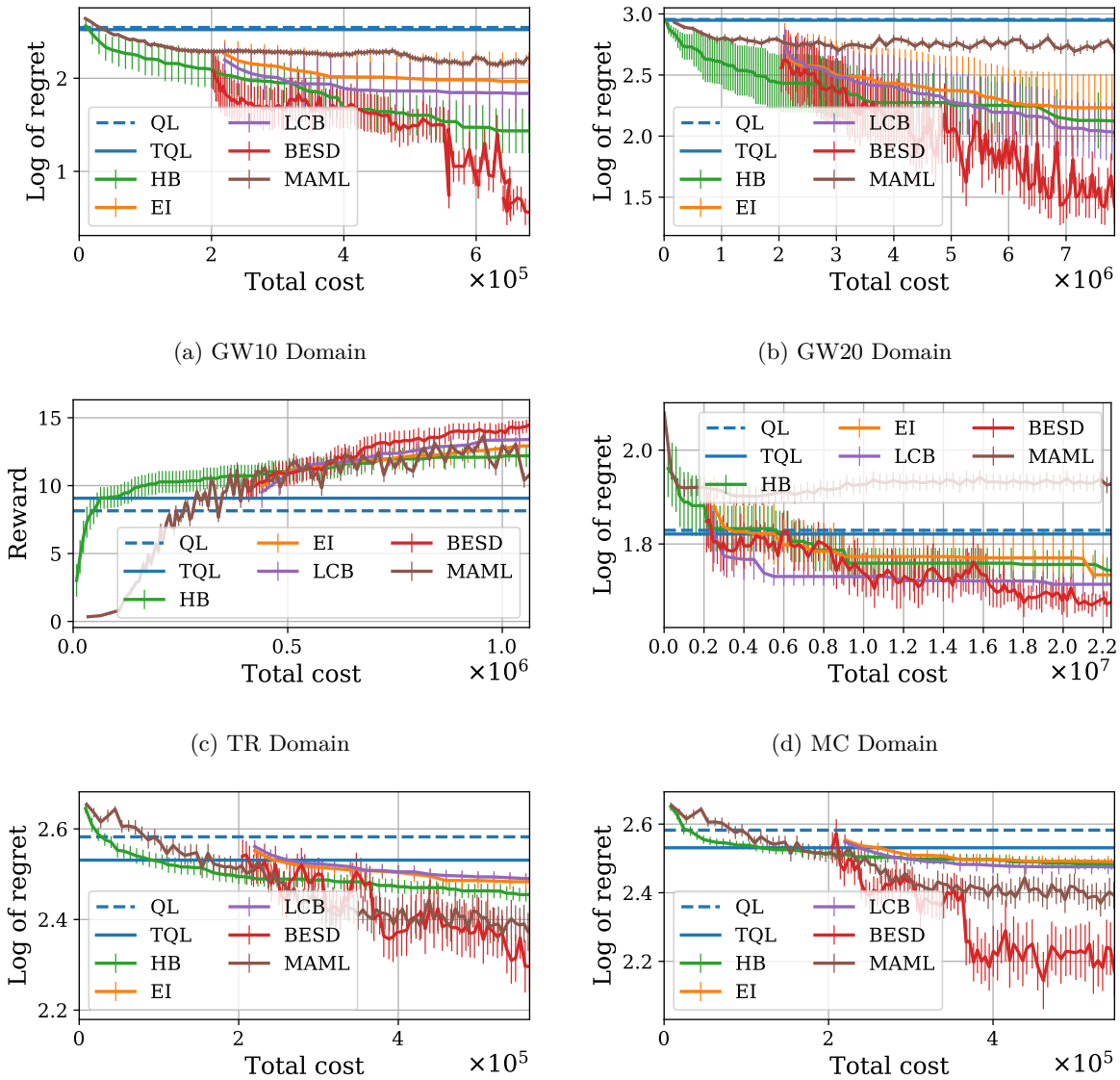
It is important that the agent moves in the *vicinity of both keys* in order for it to perform well across the distribution of environments. We now discuss how this is achieved by the two- and three-subgoal exploration strategies, using the annotations in Figures 6a and 6b.

- Behavior 'A' in KEY2 (Figure 6a): In the first exploration behavior discovered by BESD, the agent is first directed to the right-most key location and then towards the door. This is behavior is reasonable in the sense that the agent's initial location is near the left-most key location; hence, the naive exploration (e.g., ϵ-greedy) "built-in" to RL-ALGO would likely find the key (if it is there) without additional subgoal rewards.

- Behavior 'B' in KEY2 (Figure 6a): The second exploration behavior that we highlight takes a similar approach. This strategy incentivizes the agent to first check the left-most key location (going upwards from the initial location). Interestingly, the second subgoal is neither the other key location nor the goal: instead, the agent is directed toward the upper edge of the environment, slightly right of center. Upon examination, one might conclude that this path *compromises* between the second key location and the goal. On its way from the first to second subgoal, the agent enters the vicinity of the second key location and also ends up not far from the goal. In other words, the exploration strategy puts the agent in a position such that RL-ALGO's naive exploration is more likely to be successful.

- Behavior 'A' in KEY3 (Figure 6b): With an additional subgoal to work with, BESD is able to find more flexible exploration strategies. For behavior 'A', we see that the first subgoal is near the left-most key location, the second subgoal indirectly leads the agent toward the vicinity of the right-most key location, and the third subgoal is at the goal. The placement of the second subgoal is reminiscent of behavior 'B' of KEY2, but this time, a third subgoal allows BESD to directly lead the agent towards the goal

- Behavior 'B' in KEY3 (Figure 6b): This strategy is more intuitive (indeed, more replications converge to behavior 'B' than behavior 'A') and leads the agent to check each of the possible key locations (the closer one first) and then sends the agent directly toward the goal.



(a) GW10 Domain

(b) GW20 Domain

(c) TR Domain

(d) MC Domain

(e) KEY2 Domain

(f) KEY3 Domain

Figure 7: Performance as a function of the total training costs. The curves are averaged over 50 replications of the meta-optimization problem and the error bars indicate ± 2 standard errors of the mean. For each replication, to assess the performance at a particular point in the process, we take its latest recommendation and test it by averaging its performance on a random sample of 200 test MDPs (i.e., ξ^N) The x-axis is the cumulative cost including the initial sampling cost. The y-axis is typically the log regret, where regret is defined as the number of additional steps needed to reach the goal when compared to the optimal policy. The exception is in (c), where the y-axis is discounted reward (since in TR, the performance is measured by both reward and steps). Note that the curves associated with the BO methods, `BESD`, `LCB`, `EI`, start later due to the use of a set of initial points for initializing the GP model.

5.6 Baseline Algorithms

Given the somewhat unique positioning of the BESD framework, it is important for us to compare against from several streams of literature. Due to our strong focus on cost-efficiency, non-gradient-based approaches from the BO literature are particularly relevant. Two of the most common approaches are *expected improvement* (Močkus, 1975; Jones et al., 1998) and *lower/upper confidence bound*,[8] often called "GP-UCB" when used with a Gaussian process model (Cox & John, 1992; Srinivas et al., 2010). Expected improvement (EI) allocates one sample in each round, selecting a point that maximizes the expected improvement beyond currently sampled points:

$$\texttt{EI}(\theta) = \mathbb{E}_n\big[\big(\min\{y^1, \ldots, y^n\} - y^{n+1}(\theta, \tau_{\max})\big)^+\big].$$

In each iteration, we evaluate the EI selection using $\tau_{\max}$ iterations. Lower confidence bound (LCB) controls the exploration-exploitation trade-off using a "bonus term" proportional to the standard deviation at each point:

$$\texttt{LCB}(\theta) = \mu^n(\theta, \tau_{\max}) - \kappa\sqrt{k^n((\theta, \tau_{\max}), (\theta, \tau_{\max}))}.$$

The parameter κ is set to 2. Both EI and LCB are implemented using the GPyOpt package González (2016).

We also compare against two "default RL" baselines, that do not incorporate an aspect of tuning the exploration strategy. The first baseline is the Q-learning algorithm (QL) (Watkins, 1989) with no subgoals or reward shaping: that is, we directly run QL on environment ξ^N for $\tau_{\max}$ interactions. The second one is a heuristic based on the approximate Q-values learned by QL, which we call "transfer" Q-learning (TQL): for the test instance, we initialize the Q-values using the previously stored results from a randomly chosen training environment. This heuristic is inspired by the idea of *policy reuse* proposed in Fernández et al. (2010) for transferring learned strategies to new tasks.

An alternative to applying BO or bandit algorithms to hyperparameter optimization is the idea of *adaptive configuration evaluation*, which focuses on improving the throughput of configuration evaluation by quickly eliminating ones that are not promising. From this line of thinking, the Hyperband algorithm (HB) of Li et al. (2017) stands out as a popular and representative approach. It 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. Following HB's motivation of early-stopping unpromising designs, each θ is evaluated for $\tau_{\min}$ steps. 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.

Finally, we chose a representative algorithm from the multi-task RL literature, the well-known *model-agnostic meta-learning* algorithm (MAML) proposed by Finn et al. (2017a). MAML consists of two optimization loops. The outer loop provides an initialization to the inner loop, and the inner loop solves new tasks with a small number of examples. As with QL and TQL, MAML does not make use of subgoal exploration and uses neural network representations as in the original paper (Finn et al., 2017a). It utilizes stochastic gradient descent in both loops to optimize the parameters. MAML is implemented in Python 3.6 using Deleu (2018). To keep consistent with other baselines, the batch size of the outer loop is 1.

A detailed empirical comparison of BESD to baseline algorithms for all environments are given in Figure 7.

5.7 Takeaways from Baseline Comparisons in Figure 7

We now offer some observations and takeaways from the performance plots of Figures 7a-7f, where BESD is compared to a variety of baseline approaches. In these figures, the x-axis shows the total cost, defined to be the total number of environment interactions, and the y-axis shows the performance measure (either regret or reward, depending on the environment) when averaged across environments drawn from Ξ. In all environments, BESD (in red) achieves either the lowest regret or highest reward.

[8] "Lower" when minimizing the objective and "upper" when maximizing.

1. **Comparison to MAML.** BESD significantly outperforms MAML in all domains except KEY2, where performance is similar. We also see that in some domains (e.g., GW10, GW20, MC), MAML is unable to make much progress at all within the interaction budgets that we considered. This is not surprising as MAML relies on an abundance of data for gradient-based updates *during training* (despite the fact that it is designed for sample-efficient adaptation in the test environment). In addition, we note that since MAML's default hyperparameters worked even more poorly – we tuned the learning rate and batch sizes to improve performance. Note that BESD's "hyperparameters" (subgoal parameterization) are relatively more intuitive, especially given some domain knowledge. Importantly, there are no learning rates.

2. **Comparison to Hyperband.** HB is reasonably competitive against BESD on two of the easier domains, GW10 and TR. In particular, we notice that HB tends to have good performance early on (as it is able to use early stopping to quickly eliminate inferior subgoal strategies). However, as the interaction budget grows, we see that in most domains, BESD is eventually able to make better use of its evaluations, likely explained by BESD's use of a tailored surrogate model.

3. **Comparison to other BO methods.** The popular BO methods EI and LCB tend to perform similarly to each other in all domains. Compared to BESD, however, they are less cost-efficient. Since all three approaches make use of underlying GP surrogate models, but EI and LCB are constrained in always using $q_{\max}\tau_{\max}$ interactions, this is evidence that being able to reduce the episode lengths and the number of replications is valuable.

4. **Impact of more subgoals.** Lastly, we point out that Figures 7e and 7f show that although a two-subgoal exploration strategy achieves better results than the baselines, a three-subgoal strategy performs even better. This demonstrates the benefit of expanding the dimension of the parameterization in certain environments. Choosing the number of subgoals to use in a particular set of environments is not an exact science; in general, a higher dimensional subgoal parameterization makes the BO meta-optimization problem more challenging and each acquisition function optimization is also more time-consuming. We recommend the following guidelines: (1) Consider the total interaction budget across all training iterations. A rule-of-thumb is that a d-dimensional subgoal parameterization should have $2d-1$ random initial points. The interaction cost of the initial points should be less than 1/3 of the total budget in order to give BESD adequate time to make progress (if the cost of initial points is too high, then one might want to reduce d). (2) Optimizing the acquisition function becomes more time consuming as d increases, so d should be small enough such that (7) can be computed in one's allotted per-iteration time budget for acquisition function optimization.

5.8 Dynamic Subgoal Exploration Strategy vs. Learning From Scratch at Test Time

In Section 5, Figures 4, 5, and 6 gave visual intuition about the types of exploration behaviors that were discovered by BESD. In this section, we show how the final dynamic subgoal strategy θ^N_{rec} recommended by BESD is able to speed up learning in the test environment, by comparing it to "learning from scratch" (i.e., running BESD directly in the sparse reward environment, with no subgoals). Let

$$\pi^T_\xi = \texttt{RL-ALGO}\big[T, \mathcal{M}_\xi\big] \quad \text{and} \quad \pi^T_{\xi,\,\theta^N_{\text{rec}}} = \texttt{RL-ALGO}\big[T, \mathcal{M}_{\xi,\,\theta^N_{\text{rec}}}\big]$$

be the policy learned using RL-ALGO on the original, sparse-reward environment (i.e., no subgoals) and the policy learned by RL-ALGO with the aid of the subgoal strategy found by BESD after T test-time interactions. The performance ratio that we are interested in is

$$\text{performance ratio}(T) = \mathbb{E}\Big[V^{\pi^T_{\xi,\theta^N_{\text{rec}}}}(s_0)\Big] \;/\; \mathbb{E}\big[V^{\pi^T_\xi}(s_0)\big],$$

which, stated simply, represents the ratio "performance with subgoals / performance without subgoals." On GW10, GW20, MC, KEY2, and KEY3, a smaller performance ratio indicates a more effective exploration strategy.For TR, we measure performance using rewards instead of costs, so a larger performance ratio is better. Table 1 displays the performance ratios as a function of the number of interactions used in the test

environment. We can see that an optimized exploration strategy corresponds to dramatic improvements, ranging from roughly 3x in the worst cases (MC, KEY2, and KEY3) to nearly 20x in the best cases (GW10, GW20, and TR). Note that due to the varying difficulty between environments, we use a scaling factor m to show how the performance over time. The takeaway here is that using a good exploration strategy can lead to dramatic improvements at test-time.

Table 1: Performance ratios as a function of interactions in the test environment. GW10, GW20 TR, MC, KEY2, and KEY3 are evaluated every $m = 100, 1000, 200, 1000, 500, 500$ steps respectively.

τ	GW10	GW20	TR	MC	KEY2	KEY3
m	0.458	0.779	0.436	0.980	1.456	1.025
$2m$	0.218	0.492	2.823	1.048	0.736	0.940
$3m$	0.086	0.234	2.823	0.949	1.277	0.698
$4m$	0.080	0.224	0.917	0.896	0.704	0.788
$5m$	0.070	0.108	6.723	0.987	1.355	0.531
$6m$	0.086	0.088	8.939	0.878	0.856	0.503
$7m$	0.080	0.068	9.908	1.077	0.920	0.623
$8m$	0.087	0.075	10.216	0.877	0.883	0.532
$9m$	0.069	0.059	23.2936	0.512	0.232	0.566
$10m$	0.069	0.058	18.011	0.354	0.332	0.361

6 Conclusion and Future Work

The problem of finding exploration strategies for a distribution of environments with a strong focus on *cost-awareness during training* has not been adequately studied in the literature. This can be a deterrent to applying RL in real-world settings where interactions with the environment are limited and expensive (and where cheap simulators are not available). This paper proposes a solution based on Bayesian optimization; in a cost-aware manner, our approach finds subgoals with an intrinsic shaped reward that aids the agent in scenarios with sparse and delayed rewards, thereby reducing the number of interactions needed to obtain a good solution. We hope that this approach can help RL become more applicable in real world settings. An experimental evaluation demonstrates that `BESD` achieves considerably better solutions than a comprehensive field of baseline methods on a variety of benchmark problems. Moreover, an examination of its "recommendation paths" shows that `BESD` discovers solutions that induce interesting exploration strategies. There are several exciting avenues for extensions of this paper:

- **Richer BO formulations.** Extensions to the BO formulation could be made in various ways. For example, one interesting direction is to allow the acquisition function to determine the *number of subgoals* as an additional lever. Based on a few informal observations, such a formulation is likely only interesting in settings where *more subgoals incur additional experimentation cost*.[9] Alternatively, the acquisition function itself could be extended with additional features, such as encouraging successive subgoal evaluations to be nearby previous ones (i.e., to reduce setup cost) or the ability to reason about (known) symmetries in the domain. Such advanced features might be enabled by dynamic programming formulations *of the BO problem itself*, which can be tackled using multi-step lookahead BO (Lam et al., 2016; González et al., 2016; Jiang et al., 2020; Lee et al., 2020). Other possiblities include the ability to handle expensive-to-evaluate constraints (Gardner et al., 2014; Gelbart et al., 2014; Letham et al., 2019) or total cost budgets (Astudillo et al., 2021; Lee et al., 2021).

- **Case study in an application domain.** Our experiments gave proof-of-concept results on benchmarks where the RL training itself did not use prohibitive amounts of computation, in order

[9] We ran a small number of informal experiments where we allowed BO to select the number of subgoals, but found that `BESD` almost immediately gravitates to the largest number of subgoals (as subgoals come at no cost). Since in the applications that we have in mind, subgoal cost was not a primary concern, we did not pursue this direction as it did not bring any particularly strong insights for the standard case.

for us to stay within a reasonable computational budget. This is because statistically distinguishable results for baseline algorithms require many replications of the meta-optimization problem (i.e., the BO routines), each of which require many iterations of RL training. One immediate area of future work is to "productionize" the dynamic subgoal exploration strategies in a real-world application involving a navigation task.

- **The task-aware setting.** Finally, our problem formulation does not include "labels" for environments, as we were motivated by the case where the randomness of the test environments is due to the decision maker's uncertainty in its parameters. The situation often studied in the multi-task RL setting, however, often comes with task identifiers, where the agent knows that it is operating in particular task. An extension to this setting might be useful for certain applications, where exploration strategies that are good for one task (e.g., biking through an environment) are also useful for other tasks (e.g., walking through the same environment).

References

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

Susan Amin, Maziar Gomrokchi, Harsh Satija, Herke van Hoof, and Doina Precup. A survey of exploration methods in reinforcement learning. *arXiv preprint arXiv:2109.00157*, 2021.

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

Raul Astudillo, Daniel Jiang, Maximilian Balandat, Eytan Bakshy, and Peter Frazier. Multi-step budgeted bayesian optimization with unknown evaluation costs. *Advances in Neural Information Processing Systems*, 34, 2021.

Pierre-Luc Bacon, Jean Harb, and Doina Precup. The option-critic architecture. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

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.

Jonathan Binney, Andreas Krause, and Gaurav S Sukhatme. Optimizing waypoints for monitoring spatiotemporal phenomena. *The International Journal of Robotics Research*, 32(8):873–888, 2013.

Mickaël Binois, Jiangeng Huang, Robert B Gramacy, and Mike Ludkovski. Replication or exploration? sequential design for stochastic simulation experiments. *Technometrics*, 61(1):7–23, 2019.

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.

Bo Chen, Rui Castro, and Andreas Krause. Joint optimization and variable selection of high-dimensional gaussian processes. *arXiv preprint arXiv:1206.6396*, 2012.

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 *International Conference on Systems, Man, and Cybernetics*, pp. 1241–1246. IEEE, 1992.

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

Tristan Deleu. Model-Agnostic Meta-Learning for Reinforcement Learning in PyTorch, 2018. Available at: https://github.com/tristandeleu/pytorch-maml-rl.

Josip Djolonga, Andreas Krause, and Volkan Cevher. High-dimensional gaussian process bandits. *Advances in neural information processing systems*, 26, 2013.

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

Finale Doshi-Velez and George Konidaris. Hidden parameter Markov decision processes: A semiparametric regression approach for discovering latent task parametrizations. In *International Joint Conferences on Artificial Intelligence*, pp. 1432. NIH Public Access, 2016.

Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Vlad Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *International Conference on Machine Learning*, pp. 1407–1416. PMLR, 2018.

Benjamin Eysenbach, Abhishek Gupta, Julian Ibarz, and Sergey Levine. Diversity is all you need: Learning skills without a reward function. *arXiv preprint arXiv:1802.06070*, 2018.

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, Javier García, and Manuela Veloso. Probabilistic policy reuse for inter-task transfer learning. *Robotics and Autonomous Systems*, 58(7):866–871, 2010.

Matthias Feurer, Jost Tobias Springenberg, and Frank Hutter. Initializing Bayesian hyperparameter optimization via meta-learning. In *Association for the Advancement of Artificial Intelligence*, pp. 1128–1135, 2015.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pp. 1126–1135. JMLR. org, 2017a.

Chelsea Finn, Tianhe Yu, Tianhao Zhang, Pieter Abbeel, and Sergey Levine. One-shot visual imitation learning via meta-learning. In *Conference on Robot Learning*, pp. 357–368, 2017b.

Kevin Frans, Jonathan Ho, Xi Chen, Pieter Abbeel, and John Schulman. Meta learning shared hierarchies. *arXiv preprint arXiv:1710.09767*, 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.

Peter I Frazier. A tutorial on Bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018.

Peter I Frazier, Warren B Powell, and Savas Dayanik. A knowledge-gradient policy for sequential information collection. *SIAM Journal on Control and Optimization*, 47(5):2410–2439, 2008.

Javier Garcia-Barcos and Ruben Martinez-Cantin. Robust policy search for robot navigation with stochastic meta-policies. *arXiv preprint arXiv:2003.01000*, 2020.

Jacob R Gardner, Matt J Kusner, Zhixiang Eddie Xu, Kilian Q Weinberger, and John P Cunningham. Bayesian optimization with inequality constraints. In *ICML*, volume 2014, pp. 937–945, 2014.

Michael A Gelbart, Jasper Snoek, and Ryan P Adams. Bayesian optimization with unknown constraints. *arXiv preprint arXiv:1403.5607*, 2014.

Subhashis Ghosal, Anindya Roy, et al. Posterior consistency of Gaussian process prior for nonparametric binary regression. *The Annals of Statistics*, 34(5):2413–2429, 2006.

Sandeep Goel and Manfred Huber. Subgoal discovery for hierarchical reinforcement learning using learned policies. In *FLAIRS Conference*, pp. 346–350, 2003.

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

Javier González, Michael Osborne, and Neil Lawrence. Glasses: Relieving the myopia of bayesian optimisation. In *Artificial Intelligence and Statistics*, pp. 790–799. PMLR, 2016.

Xiaoxiao Guo, Satinder Singh, Richard Lewis, and Honglak Lee. Deep learning for reward design to improve Monte Carlo tree search in ATARI games. In *International Joint Conference on Artificial Intelligence*, pp. 1519–1525, 2016.

Abhishek Gupta, Russell Mendonca, YuXuan Liu, Pieter Abbeel, and Sergey Levine. Meta-reinforcement learning of structured exploration strategies. In *Advances in Neural Information Processing Systems*, pp. 5302–5311, 2018.

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.

Matteo Hessel, Hubert Soyer, Lasse Espeholt, Wojciech Czarnecki, Simon Schmitt, and Hado van Hasselt. Multi-task deep reinforcement learning with PopArt. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 3796–3803, 2019.

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.

Shali Jiang, Daniel Jiang, Maximilian Balandat, Brian Karrer, Jacob Gardner, and Roman Garnett. Efficient nonmyopic bayesian optimization via one-shot multi-step trees. *Advances in Neural Information Processing Systems*, 33:18039–18049, 2020.

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.

Kishor Jothimurugan, Osbert Bastani, and Rajeev Alur. Abstract value iteration for hierarchical reinforcement learning. In *International Conference on Artificial Intelligence and Statistics*, pp. 1162–1170. PMLR, 2021.

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. In *Artificial Intelligence and Statistics*, pp. 528–536, 2017.

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

Tejas D Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. *Advances in neural information processing systems*, 29, 2016.

Remi Lam, Karen Willcox, and David H Wolpert. Bayesian optimization with a finite budget: An approximate dynamic programming approach. *Advances in Neural Information Processing Systems*, 29, 2016.

Guillaume Lample and Devendra Singh Chaplot. Playing FPS games with deep reinforcement learning. In *Association for the Advancement of Artificial Intelligence*, pp. 2140–2146, 2017.

Armin Lederer, Jonas Umlauft, and Sandra Hirche. Uniform error bounds for Gaussian process regression with application to safe control. In *Advances in Neural Information Processing Systems*, pp. 657–667, 2019.

Eric Lee, David Eriksson, David Bindel, Bolong Cheng, and Mike Mccourt. Efficient rollout strategies for bayesian optimization. In *Conference on Uncertainty in Artificial Intelligence*, pp. 260–269. PMLR, 2020.

Eric Hans Lee, David Eriksson, Valerio Perrone, and Matthias Seeger. A nonmyopic approach to cost-constrained bayesian optimization. In *Uncertainty in Artificial Intelligence*, pp. 568–577. PMLR, 2021.

Benjamin Letham, Brian Karrer, Guilherme Ottoni, and Eytan Bakshy. Constrained bayesian optimization with noisy experiments. *Bayesian Analysis*, 14(2):495–519, 2019.

Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. Hyperband: A novel bandit-based approach to hyperparameter optimization. *The Journal of Machine Learning Research*, 18(1): 6765–6816, 2017.

Shie Mannor, Ishai Menache, Amit Hoze, and Uri Klein. Dynamic abstraction in reinforcement learning via clustering. In *Proceedings of the twenty-first international conference on Machine learning*, pp. 71, 2004.

Ruben Martinez-Cantin, Nando de Freitas, Arnaud Doucet, and José A Castellanos. Active policy learning for robot planning and exploration under uncertainty. In *Robotics: Science and systems*, volume 3, pp. 321–328, 2007.

Ruben Martinez-Cantin, Nando De Freitas, Eric Brochu, José Castellanos, and Arnaud Doucet. A bayesian exploration-exploitation approach for optimal online sensing and planning with a visually guided mobile robot. *Autonomous Robots*, 27:93–103, 2009.

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.

Amy McGovern and Andrew G Barto. Automatic discovery of subgoals in reinforcement learning using diverse density. In *International Conference on Machine Learning*, 2001.

Viraj Mehta, Biswajit Paria, Jeff Schneider, Stefano Ermon, and Willie Neiswanger. An experimental design perspective on model-based reinforcement learning. *arXiv preprint arXiv:2112.05244*, 2021.

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

Philippe Morere and Fabio Ramos. Bayesian RL for goal-only rewards. In *Conference on Robot Learning*, 2018.

Mojmir Mutny and Andreas Krause. Efficient high dimensional bayesian optimization with additivity and quadrature fourier features. *Advances in Neural Information Processing Systems*, 31, 2018.

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

Rafael Oliveira, Lionel Ott, Vitor Guizilini, and Fabio Ramos. Bayesian optimisation for safe navigation under localisation uncertainty. In *Robotics Research*, pp. 489–504. Springer, 2020.

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

Ian Osband, Benjamin Van Roy, and Zheng Wen. Generalization and exploration via randomized value functions. In *International Conference on Machine Learning*, pp. 2377–2386, 2016.

Xinlei Pan, Eshed Ohn-Bar, Nicholas Rhinehart, Yan Xu, Yilin Shen, and Kris M Kitani. Human-interactive subgoal supervision for efficient inverse reinforcement learning. *arXiv preprint arXiv:1806.08479*, 2018.

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

Sujoy Paul, Jeroen van Baar, and Amit K Roy-Chowdhury. Learning from trajectories via subgoal discovery. *arXiv preprint arXiv:1911.07224*, 2019.

Victor Picheny and David Ginsbourger. A nonstationary space-time gaussian process model for partially converged simulations. *SIAM/ASA Journal on Uncertainty Quantification*, 1(1):57–78, 2013.

Marc Pickett and Andrew G Barto. Policyblocks: An algorithm for creating useful macro-actions in reinforcement learning. In *International Conference on Machine Learning*, volume 19, pp. 506–513, 2002.

Lerrel Pinto and Abhinav Gupta. Learning to push by grasping: Using multiple tasks for effective learning. In *2017 IEEE international conference on robotics and automation (ICRA)*, pp. 2161–2168. IEEE, 2017.

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

Doina Precup, Richard S Sutton, and Satinder Singh. Theoretical results on reinforcement learning with temporally abstract options. In *European conference on machine learning*, pp. 382–393. Springer, 1998.

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

Carl Edward Rasmussen. Gaussian processes in machine learning. In *Advanced Lectures on Machine Learning*, pp. 63–71. Springer, 2004.

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

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

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.

Pranav Shyam, Wojciech Jaśkowski, and Faustino Gomez. Model-based active exploration. In *International conference on machine learning*, pp. 5779–5788. PMLR, 2019.

Özgür Şimşek and Andrew G Barto. Using relative novelty to identify useful temporal abstractions in reinforcement learning. In *International Conference on Machine Learning*, pp. 95, 2004.

Özgür Şimşek and Andrew G Barto. An intrinsic reward mechanism for efficient exploration. In *Proceedings of the 23rd international conference on Machine learning*, pp. 833–840, 2006.

Özgür Şimşek, Alicia P Wolfe, and Andrew G Barto. Identifying useful subgoals in reinforcement learning by local graph partitioning. In *International Conference on Machine Learning*, pp. 816–823, 2005.

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.

Niranjan Srinivas, Andreas Krause, Sham Kakade, and Matthias Seeger. Gaussian process optimization in the bandit setting: No regret and experimental design. In *International Conference on Machine Learning*, pp. 1015–1022, 2010.

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

Martin Stolle and Doina Precup. Learning options in reinforcement learning. In *International Symposium on abstraction, reformulation, and approximation*, pp. 212–223. Springer, 2002.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Richard S Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.

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.

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.

Matthew Tesch, Jeff Schneider, and Howie Choset. Adapting control policies for expensive systems to changing environments. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 357–364. IEEE, 2011.

Sebastian Thrun, Scott Thayer, William Whittaker, Christopher Baker, Wolfram Burgard, David Ferguson, Dirk Hahnel, D Montemerlo, Aaron Morris, Zachary Omohundro, et al. Autonomous exploration and mapping of abandoned mines. *Robotics & Automation Magazine*, 11(4):79–91, 2004.

Vivek Veeriah, Tom Zahavy, Matteo Hessel, Zhongwen Xu, Junhyuk Oh, Iurii Kemaev, Hado P van Hasselt, David Silver, and Satinder Singh. Discovery of options via meta-learned subgoals. *Advances in Neural Information Processing Systems*, 34, 2021.

Alexander Vezhnevets, Volodymyr Mnih, Simon Osindero, Alex Graves, Oriol Vinyals, John Agapiou, et al. Strategic attentive writer for learning macro-actions. *Advances in neural information processing systems*, 29, 2016.

Nelson Vithayathil Varghese and Qusay H Mahmoud. A survey of multi-task deep reinforcement learning. *Electronics*, 9(9):1363, 2020.

Ziyu Wang, Frank Hutter, Masrour Zoghi, David Matheson, and Nando de Feitas. Bayesian optimization in a billion dimensions via random embeddings. *Journal of Artificial Intelligence Research*, 55:361–387, 2016.

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 *International Conference on Machine Learning*, pp. 1015–1022. ACM, 2007.

Jian Wu and Peter Frazier. The parallel knowledge gradient method for batch bayesian optimization. *Advances in neural information processing systems*, 29, 2016.

Jian Wu, Matthias Poloczek, Andrew G Wilson, and Peter Frazier. Bayesian optimization with gradients. In *Advances in Neural Information Processing Systems*, pp. 5267–5278, 2017.

Jesse Zhang, Haonan Yu, and Wei Xu. Hierarchical reinforcement learning by discovering intrinsic options. *arXiv preprint arXiv:2101.06521*, 2021.

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 Proofs

A.1 Proof of Theorem 1

The proof is based on theoretical results of Poloczek et al. (2017). Our result, however, includes the ability to select the number of replications q. Denote $\lambda(\theta,\tau,q) = \sigma^2_{\text{env}} + \sigma^2_{\text{rep}}/q$. Also, let $\mathscr{F}^n$ denote the σ-algebra generated by the history H^n. The expectation $\mathbb{E}_n := \mathbb{E}[\,\cdot\,|\mathscr{F}^n]$ is taken with respect to $\mathscr{F}^n$. Recall that μ^n and k^n are the mean and covariance matrix of the time n belief on f. Define the quantities

$$Z^{n+1} = \frac{y^{n+1}(\theta,\tau) - \mu^n(\theta,\tau)}{\sqrt{\operatorname{Var}\big[y^{n+1}(\theta,\tau) - \mu^n(\theta,\tau)\,|\,\mathscr{F}^n\big]}},$$

and

$$\tilde{\sigma}^n_q\big((\theta',\tau'),(\theta,\tau)\big) = \frac{k^n\big((\theta',\tau'),(\theta,\tau)\big)}{\sqrt{\lambda(\theta,\tau,q) + k^n\big((\theta,\tau),(\theta,\tau)\big)}}.$$

Observe that Z^{n+1} is standard normal (conditional on $\mathscr{F}^n$). We have the following recursive updating equation for μ^{n+1}:

$$\mu^{n+1}(\theta,\tau) = \mu^n(\theta,\tau) + \tilde{\sigma}^n_{q^{n+1}}\big((\theta,\tau),(\theta^{n+1},\tau^{n+1})\big)\,Z^{n+1}, \tag{9}$$

and another recursive formula k^{n+1}:

$$\begin{aligned} k^{n+1}\big((\theta',\tau'),(\theta,\tau)\big) &= k^n\big((\theta',\tau'),(\theta,\tau)\big) \\ &\quad - \tilde{\sigma}^n_{q^{n+1}}\big((\theta',\tau'),(\theta^{n+1},\tau^{n+1})\big)\,\big[\tilde{\sigma}^n_{q^{n+1}}\big((\theta,\tau),(\theta^{n+1},\tau^{n+1})\big)\big]^\top. \end{aligned} \tag{10}$$

These updating equations are based on the Sherman-Woodbury identity; see Frazier et al. (2009) for a full derivation. The objective of the acquisition function is thus:

$$\begin{aligned} \frac{\nu^n(\theta,\tau,q)}{q\tau} &= \frac{1}{q\tau}\,\mathbb{E}_n\big[(\mu^{n+1}_* - \mu^n_*)\,|\,(\theta^n,\tau^n,q^n) = (\theta,\tau,q)\big] \\ &= \frac{1}{q\tau}\,\mathbb{E}_n\Big[\max_{\theta'}\big\{\mu^n(\theta',\tau_{\max}) + \tilde{\sigma}^n_q\big((\theta',\tau_{\max}),(\theta,\tau)\big)\,Z^{n+1}\big\} \\ &\qquad\qquad - \max_{\theta'}\mu^n(\theta',\tau_{\max})\,\Big|\,(\theta^n,\tau^n,q^n) = (\theta,\tau,q)\Big]. \end{aligned} \tag{11}$$

We also define the quantity

$$V^n(\theta,\tau,\theta',\tau') = \mathbb{E}_n[f(\theta,\tau)\cdot f(\theta',\tau')] = k^n\big((\theta,\tau),(\theta',\tau')\big) + \mu^n(\theta,\tau)\cdot\mu^n(\theta',\tau'). \tag{12}$$

Next, we restate a useful technical lemma from Poloczek et al. (2017).

Lemma 1 (Restatement of Lemma 1 of (Poloczek et al., 2017)) *Let $\tau,\tau' \in \mathcal{T}$ and $\theta,\theta' \in \Theta$. The limits of the series $\{\mu^n(\theta,\tau)\}_n$ and $\{V^n(\theta,\tau,\theta',\tau')\}_n$ exist. Denote them by $\mu^\infty(\theta,\tau)$ and $V^\infty(\theta,\tau,\theta',\tau')$ respectively. We have*

$$\lim_{n\to\infty}\mu^n(\theta,\tau) = \mu^\infty(\theta,\tau), \tag{13}$$

$$\lim_{n\to\infty}V^n(\theta,\tau,\theta',\tau') = V^\infty(\theta,\tau,\theta',\tau') \tag{14}$$

almost surely. If (θ',τ') is sampled infinitely often, then

$$\lim_{n\to\infty}V^n(\theta,\tau,\theta',\tau') = \mu^\infty(\theta,\tau)\cdot\mu^\infty(\theta',\tau').$$

Fix a sample path ω, which corresponds to a particular path of measurements and observations

$$\{(\theta^n,\tau^n,q^n,y^{n+1}(\theta^n,\tau^n,q^n))\}_n.$$

By the finiteness of $\bar{\Theta}$, $\mathcal{T}$, and $\mathcal{Q}$, there must exist a configuration (θ',τ',q') that is visited infinitely often on sample path ω. The following lemma states the asymptotic behavior of $\nu^n(\theta',\tau',q')/(q'\tau')$ for $n\to\infty$ as a function of $\mu^n(\cdot,\cdot)$ and $\tilde{\sigma}^n_{\cdot}\big((\cdot,\cdot),(\cdot,\cdot)\big)$.

Lemma 2 *Consider the sample path ω and (θ', τ', q') described above. Then, on that sample path ω, it holds that*

$$\lim_{n\to\infty} \tilde{\sigma}^n_{q'}\big((\theta'', \tau_{max}), (\theta', \tau')\big) = 0$$

for every $\theta'' \in \Theta$. Also, the acquisition value tends to zero: $\lim_{n\to\infty} \nu^n(\theta', \tau', q')/(q'\tau') = 0$

Proof It follows from Lemma 1 that

$$k^n\big((\theta, \tau), (\theta', \tau')\big) = \mathbb{E}_n[f(\theta, \tau) \cdot f(\theta', \tau')] - \mu^n(\theta, \tau) \cdot \mu^n(\theta', \tau') \xrightarrow{n\to\infty} 0$$

for any $\theta \in \Theta$, $\tau \in \mathcal{T}$. Then for all $\theta'' \in \bar{\Theta}$, we have

$$\lim_{n\to\infty} \tilde{\sigma}^n_{q'}\big((\theta'', \tau_{\max}), (\theta', \tau')\big) = \lim_{n\to\infty} \frac{k^n\big((\theta'', \tau_{\max}), (\theta', \tau')\big)}{\sqrt{\lambda\big(\theta', \tau', q'\big) + k^n\big((\theta', \tau'), (\theta', \tau')\big)}} = 0.$$

Note that we made use of the fact that the observation noise $\lambda(\theta', \tau', q') > 0$ for any q'. From the proof of Lemma 1 of Poloczek et al. (2017), it is shown that for any $\theta'' \in \bar{\Theta}$,

$$\big\{\mu^n(\theta'', \tau_{\max})\big\}_n \quad \text{and} \quad \big\{\tilde{\sigma}^n_{q'}\big((\theta'', \tau_{\max}), (\theta', \tau')\big)\big\}_n$$

are uniformly integrable (u.i.) families of random variables that converge almost surely to their respective limits $\mu^\infty(\theta'', \tau_{\max})$ and $\tilde{\sigma}^\infty_{q'}\big((\theta'', \tau_{\max}), (\theta', \tau')\big) = 0$. Note that the family of random variables $\big\{\tilde{\sigma}^n_{q'}\big((\theta'', \tau_{\max}), (\theta', \tau',)\big) Z^{n+1}\big\}_n$ is also uniformly integrable since Z^{n+1} is independent of $\tilde{\sigma}^n_{q'}\big((\theta'', \tau_{\max}), (\theta', \tau')\big)$. Let Z be a standard normal random variable (independent from all other quantities). It holds that

$$\begin{aligned}
&\lim_{n\to\infty} \frac{\nu^n(\theta', \tau', q')}{q'\tau'} \\
&\quad = \frac{1}{q'\tau'}\Big[\int_{-\infty}^{+\infty} \phi(Z) \max_{\theta''\in\bar{\Theta}} \Big\{\mu^\infty(\theta'', \tau_{\max}) + \tilde{\sigma}^\infty_{q'}\big((\theta'', \tau_{\max}), (\theta', \tau')\big) Z\Big\} dZ \\
&\qquad\qquad - \max_{\theta''\in\bar{\Theta}} \mu^\infty(\theta'', \tau_{\max})\Big] \\
&\quad = 0.
\end{aligned} \tag{15}$$

The first equality is due to (11) and the fact that the operations of summing and taking maximum over a finite set of uniform integrable random variables maintains uniform integrability. ■

From (7), we know that in each iteration n, the configuration (θ^n, τ^n, q^n) is selected from according to $\arg\max_{\theta,\tau,q} \nu^n(\theta, \tau, q)/(q\tau)$. Now, for the sake of contradiction, suppose that there exists some configuration $(\breve{\theta}, \breve{\tau}, \breve{q})$ such that $\lim_{n\to\infty} \nu^n(\breve{\theta}, \breve{\tau}, \breve{q})/(\breve{q}\breve{\tau}) > 0$. This immediately leads to a contradiction, since then it cannot be the case that (θ', τ', q') is visited infinitely often.

Since the sample path ω was arbitrary, we conclude that

$$\lim_{n\to\infty} \nu^n(\theta, \tau, q)/(q\tau) = 0 \quad \text{a.s.} \tag{16}$$

for all $\theta \in \bar{\Theta}$, $\tau \in \mathcal{T}$, and $q \in \mathcal{Q}$.

Lemma 3 *Given that (16) holds, we have that*

$$\arg\max_{\theta\in\bar{\Theta}} \mu^\infty(\theta, \tau_{max}) = \arg\max_{\theta\in\bar{\Theta}} f(\theta, \tau_{max})$$

almost surely.

Proof We can conclude from (12) and Lemma 1 that

$$\lim_{n\to\infty} k_n\big((\theta,\tau_{\max}),(\theta,\tau_{\max})\big) = k^{\infty}\big((\theta,\tau_{\max}),(\theta,\tau_{\max})\big) \quad \text{a.s.}$$

for all $\theta \in \bar{\Theta}$. In the case that the posterior variance $k^{\infty}((\theta,\tau_{\max}),(\theta,\tau_{\max})) = 0$ for all $\theta \in \bar{\Theta}$, then the maximizer is known perfectly and we are done.

If not, then we define $\hat{\Theta} = \big\{\theta \in \bar{\Theta} \,|\, k^{\infty}((\theta,\tau_{\max}),(\theta,\tau_{\max})) > 0\big\}$ and consider some $\hat{\theta} \in \hat{\Theta}$ where the posterior variance is positive. Fix any $\hat{q} \in \mathcal{Q}$. We now argue that

$$\tilde{\sigma}^{\infty}_{\hat{q}}\big((\hat{\theta},\tau_{\max}),(\hat{\theta},\tau_{\max})\big) = \tilde{\sigma}^{\infty}_{\hat{q}}\big((\theta'',\tau_{\max}),(\hat{\theta},\tau_{\max})\big) \tag{17}$$

for all $\theta'' \in \bar{\Theta}$. Suppose, for the sake of contradiction, that there exist some $\theta_1, \theta_2 \in \bar{\Theta}$ with

$$\tilde{\sigma}^{\infty}_{\hat{q}}\big((\theta_1,\tau_{\max}),(\hat{\theta},\tau_{\max})\big) \neq \tilde{\sigma}^{\infty}_{\hat{q}}\big((\theta_2,\tau_{\max}),(\hat{\theta},\tau_{\max})\big). \tag{18}$$

Recall (15) and note that it can be rewritten as

$$\lim_{n\to\infty} \frac{\nu^n(\theta',\tau',q')}{q'\tau'} = \frac{1}{q'\tau'}\Big[\mathbb{E}\big[h(Z)\big] - \max_{\theta''\in\bar{\Theta}} \mu^{\infty}(\theta'',\tau_{\max})\Big], \tag{19}$$

where $h(z) = \max_{\theta''\in\bar{\Theta}}\Big\{\mu^{\infty}(\theta'',\tau_{\max}) + \tilde{\sigma}^{\infty}_{q'}\big((\theta'',\tau_{\max}),(\theta',\tau')\big)\, z\Big\}$. Since $\bar{\Theta}$ is finite and each function within the maximization in h is affine in z, the $h(z)$ is convex[10] and piecewise linear. Since h is convex, there is an affine function l such that

$$l(0) = h(0), \quad l(z) \leq h(z) \;\text{ for all }\; z \in \mathbb{R}.$$

The assumption we made in (18), which effectively says that h is created by taking maximum over affine functions of *differing slopes*, implies h cannot itself be affine (and indeed, must consist of various "pieces"). Therefore, there exists an interval $\mathcal{I}$, either of the form (z_0,∞) or $(-\infty,z_0)$, such that $l(z) < h(z)$ for $z \in \mathcal{I}$. It follows that $\mathbb{E}[l(Z)] < \mathbb{E}[h(Z)]$. By the linearity of l, we have

$$\mathbb{E}[l(Z)] = l(\mathbb{E}[Z]) = l(0) = h(0) = \max_{\theta''\in\bar{\Theta}} \mu^{\infty}(\theta'',\tau_{\max}) < \mathbb{E}\big[h(Z)\big].$$

This implies that (19) is strictly positive, contradicting (16). We thus conclude that (17) holds, which is equivalent to

$$\frac{k^{\infty}\big((\theta'',\tau_{\max}),(\hat{\theta},\tau_{\max})\big)}{\sqrt{\lambda(\hat{\theta},\tau_{\max},\hat{q}) + k^{\infty}\big((\hat{\theta},\tau_{\max}),(\hat{\theta},\tau_{\max})\big)}} = \frac{k^{\infty}\big((\theta''',\tau_{\max}),(\hat{\theta},\tau_{\max})\big)}{\sqrt{\lambda(\hat{\theta},\tau_{\max},\hat{q}) + k^{\infty}\big((\hat{\theta},\tau_{\max}),(\hat{\theta},\tau_{\max})\big)}},$$

for all $\theta'', \theta''' \in \bar{\Theta}$. Moreover, since $\hat{\theta}$ was chosen from $\hat{\Theta}$, we know that

$$\lambda(\hat{\theta},\tau_{\max},\hat{q}) + k^{\infty}\big((\hat{\theta},\tau_{\max}),(\hat{\theta},\tau_{\max})\big) > 0,$$

and hence $k^{\infty}\big((\theta''',\tau_{\max}),(\hat{\theta},\tau_{\max})\big) = k^{\infty}\big((\theta'',\tau_{\max}),(\hat{\theta},\tau_{\max})\big)$ for all $\theta'', \theta''' \in \bar{\Theta}$.

This means the covariance matrix of $\{f(\theta,\tau_{\max}) \,|\, \theta \in \bar{\Theta}\}$ is proportional to the all-ones matrix, and that draws from $f(\theta,\tau_{\max}) - \mu^{(\infty)}(\theta,\tau_{\max})$ are *constant* across $\theta \in \bar{\Theta}$. Therefore, $\arg\max_{\theta\in\bar{\Theta}} \mu^{(\infty)}(\theta,\tau_{\max}) = \arg\max_{\theta\in\bar{\Theta}} f(\theta,\tau_{\max})$ and the statement of the theorem holds. ■

[10] Pointwise maximum of convex functions is convex.

A.2 Proof of Theorem 2

In Theorem 2, we establish an additive bound on the loss of the solution obtained by BESD, $f(\bar{\theta}, \tau_{\max})$, with respect to the unknown optimum $f(\theta^{\mathrm{OPT}}, \tau_{\max})$, as the number of iterations $N \to \infty$. Recall that we suppose $\mu(\theta, \tau) = 0$ for all θ, τ, and that the kernel $k(\cdot, \cdot)$ has continuous partial derivatives up to the fourth order. According to Theorem 3.2 of Lederer et al. (2019), for any $\delta \in (0, 1]$, with probability at least $1 - \delta$, the quantity

$$\|L_\delta\| = \left\|(L_\delta^1, L_\delta^2, \cdots, L_\delta^m)\right\|$$

is a Lipschitz constant of f on Θ , i.e., it holds that

$$|f(\theta, \tau_{\max}) - f(\theta', \tau_{\max})| \leq \|L_\delta\| \cdot \mathrm{dist}(\theta, \theta'),$$

where $\theta, \theta' \in \Theta$. By the definition of d, there exists a $\bar{\theta} \in \bar{\Theta}$ such that $\mathrm{dist}(\bar{\theta}, \theta^{\mathrm{OPT}}) \leq d$. Therefore, it follows that the suboptimality due to optimizing in $\bar{\Theta}$ is bounded by

$$f(\theta^{\mathrm{OPT}}, \tau_{\max}) - f(\bar{\theta}, \tau_{\max}) \leq \|L_\delta\| \cdot d. \tag{20}$$

Theorem 1 completes the proof of Theorem 2 since (20) holds with probability $1 - \delta$.

B GP 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 $\sigma_\psi^2 = (\mu_\psi/2)^2$.