Receding Horizon Curiosity

Matthias Schultheis, Boris Belousov, Hany Abdulsamad, Jan Peters[†]
Department of Computer Science, Technische Universität Darmstadt, Germany
[†]Robot Learning Group, Max Planck Institute for Intelligent Systems, Tübingen, Germany
matthias.schultheis@gmail.com, {belousov, abdulsamad, peters}@ias.tu-darmstadt.de

Abstract: Sample-efficient exploration is crucial not only for discovering rewarding experiences but also for adapting to environment changes in a task-agnostic fashion. A principled treatment of the problem of optimal input synthesis for system identification is provided within the framework of sequential Bayesian experimental design. In this paper, we present an effective trajectory-optimization-based approximate solution of this otherwise intractable problem that models optimal exploration in an unknown Markov decision process (MDP). By interleaving episodic exploration with Bayesian nonlinear system identification, our algorithm takes advantage of the inductive bias to explore in a directed manner, without assuming prior knowledge of the MDP. Empirical evaluations indicate a clear advantage of the proposed algorithm in terms of the rate of convergence and the final model fidelity when compared to intrinsic-motivation-based algorithms employing exploration bonuses such as prediction error and information gain. Moreover, our method maintains a computational advantage over a recent model-based active exploration (MAX) algorithm, by focusing on the information gain along trajectories instead of seeking a global exploration policy. A reference implementation of our algorithm and the conducted experiments is publicly available[1].

Keywords: Bayesian exploration, artificial curiosity, model predictive control

1 Introduction

Learning agents are expected to constantly adapt to changing environments without an otherwise explicit task specification. A subclass of such changes relates to predicting the consequences of the agent's own actions, as captured by the forward model of the environment, which is of crucial importance for the purposes of planning and task-solving. When the forward model parameters are unknown, what sequence of actions should the agent take to reveal the most about the system, i.e., what course of action facilitates the most sample-efficient exploration? To answer this question, the information content in the observations needs to be quantified, which becomes possible if the agent maintains a probability distribution over the model parameters. The sequential decision making problem with such a probabilistic model whose parameters are not directly observable is known as the partially observable Markov decision process (POMDP) [1].

Finding an exact solution of continuous-state/action POMDPs is intractable in general [2]. Therefore, various approximations are commonly employed [3]. In particular, one-step greedy heuristics—exploration bonuses—have gained prominence recently due to the ease of implementation, wide range of applicability, and good empirical performance in game domains [4]. However, they neglect long-term effects of control actions and therefore struggle in underactuated continuous control domains, focusing on immediate curiosity satisfaction [5]. As a remedy, it was proposed that the agent should plan exploration [6], purposefully driving the system to the states that provide the largest information gain, thus performing active exploration.

Many active exploration approaches have been proposed in the past [7, 8, 9, 10, 11, 12], differing chiefly in how the model is represented and how it is used for planning. The most recent representative from this family of approaches is the model-based active exploration (MAX) algorithm [5], that

[1] https://github.com/mschulth/rhc

3rd Conference on Robot Learning (CoRL 2019), Osaka, Japan.

represents the uncertainty in the dynamics via an ensemble of deterministic networks, and uses a model-free reinforcement learning (RL) algorithm to find an exploration policy optimal with respect to the ensemble. Such approach is computationally demanding because it solves a full RL problem in the inner loop to only use the optimized policy for performing a few exploratory steps.

In this paper, we present a principled model-based active exploration method, which contrasts with the recent computationally expensive RL-based approaches. The proposed exploration algorithm represents the uncertainty in the model by a distribution over the parameters in a shallow Bayesian network and finds optimally explorative actions via trajectory optimization based on the learned model. Since rigid-body dynamics can be written as a dot product between a vector of state-action features and a vector of physics parameters, we adopt a similar model structure with generic feature functions and a Gaussian vector of unknown parameters. Due to the linear-Gaussian structure, belief space dynamics can then be obtained in closed-form and incorporated into the trajectory optimization formulation, capturing the effect of current actions on future information gain.

The proposed method, termed receding horizon curiosity (RHC), addresses several challenges involved in the design and implementation of actively exploring agents. First, the agent's beliefs must be represented and propagated in time to estimate the information gain. A combination of approximations is needed and it is not clear a priori if the resulting algorithm will work. RHC employs Gaussian beliefs and a maximum likelihood observation assumption to represent and propagate the beliefs. Second, the information gain objective needs to be evaluated and optimized. Again, a number of approximations are involved, that require empirical evaluation. RHC exploits the Gaussian-linear model structure to evaluate the information gain and it relies on trajectory optimization to maximize it. Finally, a key feature of RHC is the interleaved optimal exploration and model updating, which turns out to be sufficient for promoting efficient model-learning, as shown by the experimental evaluations. On the whole, RHC compares favorably with state-of-the-art model-free intrinsic motivation approaches in terms of the model error and downstream task performance in classical continuous control environments, and compared to MAX, it is computationally far less demanding and has lower variance over runs.

2 Foundations

In this section, the background on active learning, Bayesian linear regression, random Fourier features, model-based reinforcement learning, and multiple shooting methods is provided.

2.1 Active Learning

In supervised learning, a training set is predefined and fixed. However, if an agent is allowed to choose the instances to train on, learning can potentially progress faster and require fewer samples. Active learning [13], also known as optimal experiment design [9], is an area of statistical learning that addresses exactly the question of how to choose the data points for learning. Choosing an optimal subset of points involves a combinatorial number of possibilities. Therefore, approximations of the optimal value function, known as *acquisition functions*, are employed for query point selection.

Perhaps the most straightforward and common query framework is *uncertainty sampling* [13]. In this framework, an active learner queries the instance $x \in X$ for which the model output $y \in Y$ is least certain. With the entropy of the output $\mathbb{H}(y|x)$ as the measure of uncertainty, the following acquisition function needs to be maximized

$$\alpha_{\text{us}}(x) = \mathbb{H}(y|x).$$

Compared to other approaches, uncertainty sampling is computationally relatively light, and if the likelihood belongs to an exponential family, the entropy can even be obtained in closed-form. A drawback of uncertainty sampling is that it fails if the underlying system is stochastic, as it cannot distinguish between aleatoric and epistemic uncertainty.

To fix the shortcomings of uncertainty sampling, *expected variance reduction* [13] was proposed, that explicitly takes into account the model variance, by minimizing the acquisition function

$$\alpha_{\text{evr}}(x) = \text{var}\left(\mathcal{M} \mid \mathcal{D} \cup (x, y)\right),$$

where $\text{var}\left(\mathcal{M} \mid \mathcal{D} \cup (x, y)\right)$ denotes a measure of variance of the model $\mathcal{M}$ trained on the extended dataset $\mathcal{D}_* = \mathcal{D} \cup (x, y)$. For Bayesian linear regression, the posterior entropy and the predictive variance are commonly used as measures of the model variance—both are available in closed-form and correspond to well-known alphabetic optimal design criteria [14]. Under mild assumptions [15], closed-form expressions for the output variance can also be obtained for more flexible models, such as neural networks, Gaussian mixture models, and locally-weighted linear regression [13].

2.2 Bayesian Linear Regression

Linear regression [16] assumes that the output $y \in \mathbb{R}$ is given by a linear function $\boldsymbol{\theta}^T \boldsymbol{\phi}(\mathbf{x})$ in the features $\boldsymbol{\phi}(\mathbf{x}) \in \mathbb{R}^m$ of the input $\mathbf{x} \in \mathbb{R}^n$ and parameters $\boldsymbol{\theta} \in \mathbb{R}^m$. The output uncertainty is captured by a probability distribution, most commonly—the normal distribution,

$$p(y \mid \mathbf{x}; \boldsymbol{\theta}) = \mathcal{N}\left(y \mid \boldsymbol{\theta}^T \boldsymbol{\phi}(\mathbf{x}), \beta^{-1}\right).$$

In Bayesian linear regression [16], not only the output y but also the parameters $\boldsymbol{\theta}$ are assumed to be uncertain. Conveniently, given a Gaussian prior $p(\boldsymbol{\theta} \mid \mathcal{D}) = \mathcal{N}(\boldsymbol{\theta} \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})$, the *posterior* after observing a data point $(\mathbf{x}, y)$ is also Gaussian, $p(\boldsymbol{\theta} \mid \mathcal{D} \cup (\mathbf{x}, y)) = \mathcal{N}(\boldsymbol{\theta} \mid \boldsymbol{\mu}_*, \boldsymbol{\Sigma}_*)$, with parameters

$$\begin{aligned} \boldsymbol{\mu}_* &= \boldsymbol{\Sigma}_* \left(\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} + \beta \boldsymbol{\Phi}^T \mathbf{Y}\right), \\ \boldsymbol{\Sigma}_*^{-1} &= \boldsymbol{\Sigma}^{-1} + \beta \boldsymbol{\Phi}^T \boldsymbol{\Phi}, \end{aligned} \tag{1}$$

where the features and targets corresponding to the new data point $(\mathbf{x}, y)$ are aggregated into the design matrix $\boldsymbol{\Phi}$ and the vector of targets $\mathbf{Y}$, as described in [16]. Along with the posterior, the *predictive distribution* plays an important role,

$$\begin{aligned} p(y_* \mid \mathbf{x}_*; \mathcal{D} \cup (\mathbf{x}, y)) &= \mathcal{N}(y_* \mid \boldsymbol{\mu}_*^T \boldsymbol{\phi}(\mathbf{x}_*), \sigma_*^2(\mathbf{x}_*)), \\ \text{where} \quad \sigma_*^2(\mathbf{x}_*) &= \beta^{-1} + \boldsymbol{\phi}(\mathbf{x}_*)^T \boldsymbol{\Sigma}_* \boldsymbol{\phi}(\mathbf{x}_*). \end{aligned}$$

If data arrives sequentially, the posterior can be updated incrementally, taking the current posterior as the prior for the next data point. This procedure, known as iterative least squares [16], enables efficient processing of sequential data.

2.3 Random Fourier Features

Random Fourier features provide a powerful representation by approximating a Gaussian process with an exponentiated quadratic kernel [17]. Given an input $\mathbf{x} \in \mathbb{R}^n$, following [18], the i-th feature is defined as $\phi_i(\mathbf{x}) = \sin(\sum_{j=1}^n P_{ij} x_j / \nu_j + \varphi_i)$ where $P_{ij} \sim \mathcal{N}(0, 1)$ are normally distributed real numbers and $\varphi_i \sim \mathcal{U}[-\pi, \pi)$ are uniform random phase shifts. The bandwidth parameter $\boldsymbol{\nu} \in \mathbb{R}^n$ scales the inputs, and must be chosen with care. A rule of thumb suggested in [18] is to set it equal to the average pairwise distance between observed input vectors. An even better approach is to fit the bandwidth parameter through marginal maximum likelihood optimization.

2.4 Model-Based Reinforcement Learning

The distinction between model-free and model-based reinforcement learning is not precisely defined, but for the purposes of this paper, under model-based reinforcement learning we will understand trajectory optimization using a dynamics model obtained through system identification. In detail, having learned a forward model $s' = f_\theta(s, a)$, the agent plans a trajectory $\tau = \{s_0, a_0, s_1, a_1, \ldots, a_{T-1}, s_T\}$ from a given starting state s_0 with the goal of minimizing a pre-defined trajectory cost $J(\tau)$. Variations are possible in how exactly the model is learned, what model representation is used, whether the cost function is learned or given, etc.

2.5 Shooting Methods for Trajectory Optimization

Direct shooting methods are a class of optimal control algorithms aimed at solving planning problems with deterministic continuous dynamics models [19]. Depending on whether the dynamics are imposed as a constraint or included in the objective itself, one discerns between single shooting and multiple shooting methods. In *single shooting*, the dynamics are included in the objective function,

$$\underset{a_{0:T-1}}{\text{minimize}} \quad J(s_0, a_0, f(s_0, a_0), a_1, \ldots, a_{T-1}, f(f(\ldots f(s_0, a_0), a_{T-2}), a_{T-1})),$$

and the optimization is performed only with respect to the actions $a_{0:T-1}$. Note the iterated application of the dynamics function f. When performing gradient descent on this objective, the problem known as exploding/vanishing gradients hinders efficient first-order optimization. The reason for that is the ill conditioning of the problem: the actions in the beginning of the trajectory have a bigger impact on the final state than the actions at the end.

Multiple shooting methods aim to improve the problem conditioning by splitting the trajectory into smaller chunks, subsequently glueing them together by constraints. To enable that, the dynamics are imposed as a constraint, and the optimization is performed with respect to both actions and states,

$$\begin{aligned} \underset{a_{0:T-1}, s_{1:T}}{\text{minimize}} \quad & J(s_0, a_0, s_1, a_1, \ldots, a_{T-1}, s_T) \\ \text{subject to} \quad & s_t = f(s_{t-1}, a_{t-1}), \quad t = 1, \ldots, T. \end{aligned}$$

Multiple shooting methods converge faster and are numerically more stable than single shooting because they are better conditioned thanks to the breaking up of long-term dependencies into shorter chunks [20]. Moreover, state constraints can be straightforwardly incorporated in multiple shooting since states are optimization variables. The price for such benefits is a significant increase in the problem size and, as a consequence, in the memory requirements. This drawback, however, is offset by the fact that the problem becomes much sparser [20].

3 Receding Horizon Curiosity

Consider a dynamical system with state space $\mathcal{S} \subset \mathbb{R}^n$ and action space $\mathcal{A} \subset \mathbb{R}^k$. Denote a probabilistic model of the dynamics that an agent maintains about this system by $\mathcal{M}$. The probabilistic nature of the model enables the agent to reason about the information content in observations. The agent wants to find a sequence of actions $a_{0:T-1}$ which, when executed open-loop on the real system, provides the most informative sequence of observations $s_{0:T}$ in the sense of being useful for learning the model. This setting is an active learning problem in which the sequence of actions plays the role of a query point. Therefore, we can utilize active learning approaches from Sec. 2.1 based on uncertainty sampling and expected variance reduction to solve the exploration problem.

3.1 Uncertainty Sampling

The uncertainty sampling acquisition function proposes to query the point that the model is most uncertain about. In our planning scenario, that means selecting the sequence of actions $a_{0:T-1}$ that results in a trajectory $(a_{0:T-1}, s_{0:T})$ that has the highest entropy. If $p(s' \mid s, a)$ is the prediction given by the probabilistic model $\mathcal{M}$ trained on previously observed trajectories $\mathcal{D}$, then the optimization objective can be stated as

$$\underset{a_{0:T-1}}{\text{maximize}} \quad \sum_{t=1:T} \mathbb{E}_{s_{t-1} \sim p(s_{t-1} \mid a_{t-2}, \ldots, a_0)} \mathbb{V}_p[s_t \mid s_{t-1}, a_{t-1}],$$

with $\mathbb{V}_p[s_t \mid s_{t-1}, a_{t-1}]$ denoting the variance in the prediction of the next state, and the outer expectation being taken with respect to state marginal distributions. Note that alternative formulations are possible, e.g., where only the variance at the last time step is taken into account or where different time steps are weighted differently. Ideally, one would only consider the variance at the last time step; however, due to the fact that an approximate model is used, real trajectories rather quickly diverge from the planned ones, and therefore it is desirable to reach informative states as quickly as possible, which can be achieved by rewarding the agent for information gain at every time step.

To evaluate the uncertainty sampling objective, the state distribution needs to be propagated through the probabilistic model, which is a non-trivial problem in general. Instead, an approximate version of this problem can be solved, where only the mean of the state distribution is propagated,

$$\begin{aligned} \underset{a_{0:T-1}}{\text{maximize}} \quad & \sum_{t=1:T} \mathbb{V}_p[s_t \mid \hat{s}_{t-1}, a_{t-1}] \\ \text{subject to} \quad & \hat{s}_t = \mathbb{E}_p[s_t \mid \hat{s}_{t-1}, a_{t-1}], \quad t = 1, \ldots, T. \end{aligned} \tag{2}$$

In line with the general theory of active learning, the uncertainty sampling objective (2) is easier to optimize than the expected variance reduction objective described below. If the Bayesian linear regression model (Sec. 2.2) is used to represent the dynamics, the posterior parameter covariance matrix $\mathbf{\Sigma}_*$ remains constant and does not depend on the states and actions. Thus, both the objective and constraints in (2) are differentiable and the problem can be solved using the multiple shooting method described in Sec. 2.5. The complete optimization procedure is summarized in Alg. 1.

```
**Data:** number of episodes $N$, horizon $T$, initial model $\mathcal{M}_0$
**Result:** optimized model $\mathcal{M}_N$
**for** $i \leftarrow 1$ **to** $N$ **do**
    find actions $a_{0:T-1}$ that optimize (2) or (3) given the current model $\mathcal{M}_i$;
    execute $a_{0:T-1}$ in the environment and observe $s_{0:T}$;
    update model $\mathcal{M}_{i+1}$ via (1) using $\mathcal{M}_i$ as the prior and $(a_{0:T-1}, s_{0:T})$ as the new data;
**end**
```

Algorithm 1: Receding Horizon Curiosity. In each episode i, the most informative sequence of actions $a_{0:T-1}$ under the current model $\mathcal{M}_i$ is computed; after that, observations $s_{0:T}$ are collected and the model is updated $\mathcal{M}_i \to \mathcal{M}_{i+1}$.

3.2 Expected Variance Reduction

The uncertainty sampling heuristic rewards the agent for visiting uncertain states, but it ignores the fact that the model will become more certain once those states are visited. For example, if two states are initially equally uncertain, visiting one of them may yield a larger decrease in uncertainty because that state is more informative. Uncertainty sampling would be insensitive to this difference, whereas expected variance reduction allows for taking such information gain into account. In our setting, the expected variance reduction problem can be stated as

$$\underset{a_{0:T-1}}{\text{minimize}} \quad \mathbb{E}_p \left[\text{var}(\mathcal{M} \mid \mathcal{D} \cup (a_{0:T-1}, s_{0:T})) \right].$$

The operator var$(\mathcal{M} \mid \mathcal{D}_*)$ here stands for a measure of variance of model $\mathcal{M}$ trained on dataset $\mathcal{D}_*$. We take it to be the entropy of the posterior distribution over the model parameters $\boldsymbol{\theta} \in \mathbb{R}^m$, which is known as the D-optimality criterion in Bayesian experimental design [14],

$$\text{var}(\mathcal{M} \mid \mathcal{D}_*) = \mathbb{H}(\boldsymbol{\theta} \mid \mathcal{D}_*) = \frac{1}{2} \ln \det (\boldsymbol{\Sigma}_*) + \frac{m}{2} \ln(2\pi e).$$

Importantly, the covariance matrix $\boldsymbol{\Sigma}_*$ depends on the trajectory $(a_{0:T-1}, s_{0:T})$ through the augmented dataset $\mathcal{D}_* = \mathcal{D} \cup (a_{0:T-1}, s_{0:T})$. The exact relationship is given in (1), and this relationship allows for optimization of the expected model variance with respect to the planned trajectory.

Although superior to uncertainty sampling in information-theoretic terms, expected variance reduction is quite expensive to compute and optimize in practice [13], particularly due to probabilistic state propagation [21]. We adopt the so called maximum likelihood observations assumption [22], which amounts to propagating only the mean of the state distribution. The crudeness of this approximation is offset by the computational advantage: more frequent replanning, enabled by neglecting the expensive state uncertainty propagation, allows the agent to compensate for unforeseen deviations from the planned trajectory efficiently. The corresponding optimization problem reads

$$\begin{aligned} \underset{a_{0:T-1}}{\text{minimize}} \quad & \text{var}(\mathcal{M} \mid \mathcal{D} \cup (a_{0:T-1}, s_{0:T})) \\ \text{subject to} \quad & \hat{s}_t = \mathbb{E}_p[s_t \mid \hat{s}_{t-1}, a_{t-1}], \quad t = 1, \ldots, T. \end{aligned} \tag{3}$$

Both the objective and constraints in (3) are differentiable. Therefore, gradient-based optimization described in Sec. 2.5 can in principle be used to solve this problem. However, evaluation of the objective requires differentiation through matrix inversion, since matrix $\boldsymbol{\Sigma}_*$ depends on the inverse of the kernel matrix (1). Combined with the chain-like structure of state-action dynamics, gradient computation becomes quite expensive for larger models (e. g. $T > 100$, $M > 40$).

4 Experimental Results

We compare our receding horizon curiosity algorithm (RHC, Sec. 3) to state-of-the-art model-based and model-free exploration approaches. On the model-based side, we consider MAX [5], which optimizes a certain approximation of the information gain via model ensemble disagreement. On the model-free side, we employ soft actor-critic (SAC) [23] with popular exploration bonuses: squared prediction error (SAC PE) and information gain in the form of parameter entropy difference between successive steps (SAC IG). Two acquisition functions for RHC are considered: *uncertainty sampling* (RHC US, Sec. 3.1) and *expected variance reduction* (RHC EVR, Sec. 3.2). Additionally, we report the performance of uniform random exploration (RAND) for comparison.

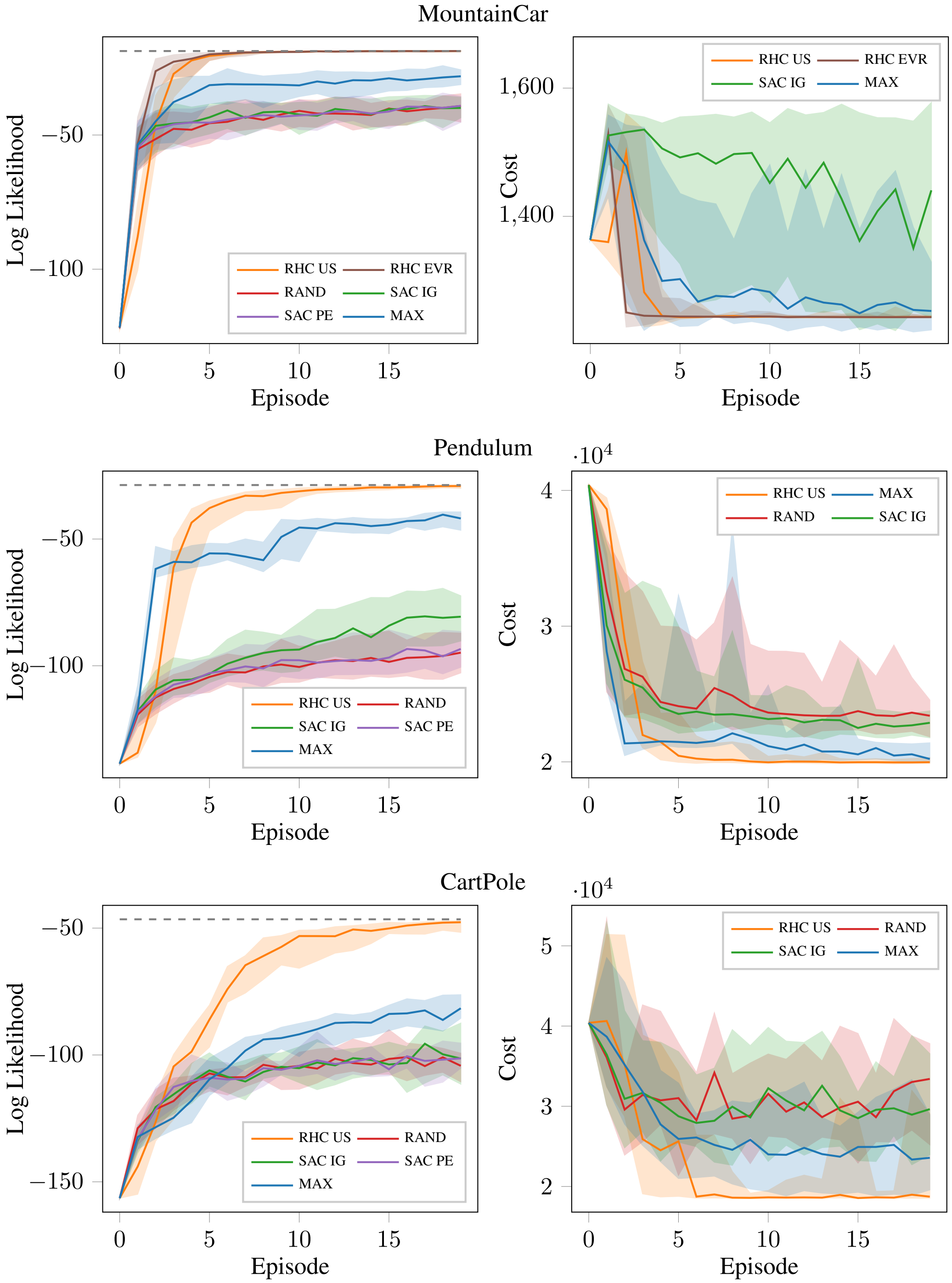

Figure 1: Evaluation of exploration methods. Log-likelihood of random 10-step trajectories (evaluated by using the learned models) is shown on the left; the plots depict the median with the 1st and 9th deciles over 20 runs. The grey dashed lines indicate the log-likelihood obtained by a model trained on 10^4 uniform transition samples from the full state-action space and therefore approximates the best achievable log-likelihood for this model class. Our proposed approach RHC reaches the highest log-likelihood the fastest, followed by MAX, and subsequently the model-free algorithms with exploration bonuses. The plots on the right show the cumulative cost (negative reward) of solving each respective control task using the learned model. The trend is similar to the model log-likelihood: RHC reaches the lowest cost the fastest, then follows MAX, and after that follow the model-free exploration approaches.

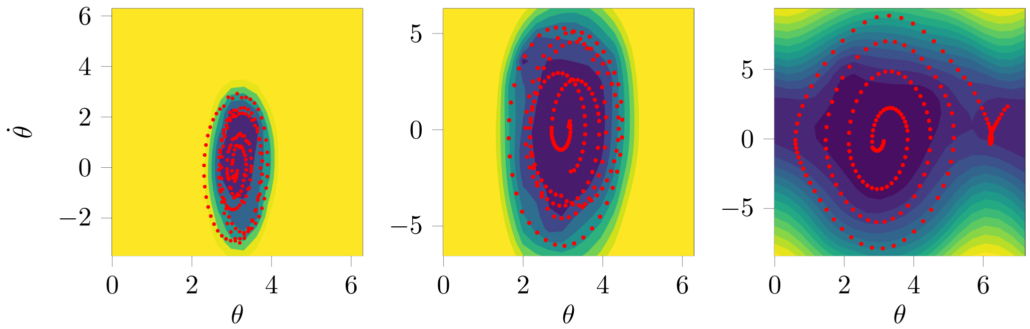

Figure 2: Exploration progress of the RHC algorithm with the uncertainty sampling objective in the pendulum environment. Episodes 2, 3, and 6 are shown from left to right. Note how the explored area of the state space is growing with iterations as the agent is trying to reach more distant states.

Experiments aim to prove the feasibility and reveal advantages and disadvantages of simultaneous feature learning combined with approximate belief-space planning under deterministic state propagation assumption for guiding exploration. Classical control environments—mountain car, pendulum, and cartpole—are used for evaluation. In these environments, exploration using current reinforcement learning algorithms with sparse rewards and random noise is insufficient. The experiments were carried out as follows. For RHC, a sequence of actions was computed in each episode using multiple shooting (Sec. 2.5) and executed open-loop in the environment (Algorithm 1). As planning with the expected variance reduction objective is only tractable for small models (Sec. 3.2), RHC with this objective was only applied to the mountain car problem. For MAX and SAC, instead of open-loop actions, the respective 'curious' policies were applied in the environment. After each episode, the Bayesian linear regression model (Sec. 2.2) with random Fourier features (Sec. 2.3) was retrained using all observations from the past episodes. The quality of the learned model was evaluated on two metrics: (i) *mean log-likelihood* on a set of test points obtained by sampling random starting states and executing random actions, and (ii) *mean return* (negative reward accumulated over transitions) on downstream learning tasks, to highlight the quality of the learned model when used in a classical planning-for-control scenario. Further experimental details can be found in the appendix, and the implementation—in the accompanying software package.

The results are shown in Fig. 1. In each environment, RHC was the only algorithm that reached the highest possible log-likelihood within 20 episodes, as indicated by the curves in the left column. When applicable, RHC EVR converged faster, as expected (Sec. 3.2). Both SAC PE and SAC IG performed on the level of random exploration (RAND), which can be traced back to their *over-commitment* behaviour [5]: in the beginning, virtually any action provides high reward because the model is uncertain, but afterwards the agent has to 'unlearn' it in order to go to more distant areas of the state-action space. MAX does not suffer from the over-commitment problem and therefore performed better than SAC and random exploration but nevertheless worse than our RHC method. Although none of the methods is real-time capable, it is worth pointing out that MAX took significantly longer than RHC because MAX solves an entire reinforcement learning problem in each episode. A table with run times is provided in the appendix.

Figure 2 shows the trajectories executed by RHC US over iterations in the pendulum environment. The background indicates the entropy of the learned forward model's output when zero action is substituted. Warm colors correspond to high entropy (uncertainty). RHC tries to find a trajectory of maximum uncertainty consistent with the learned dynamics model. Driven by curiosity, the pendulum does a full swing-up already in Episode 6 to reach further areas of the state space.

5 Related Work

In psychology, *curiosity* [24] is considered to be a type of *intrinsic motivation* [25] that drives humans to explore. In reinforcement learning, various reward signals have been proposed to promote *artificial curiosity*. An early example is the *prediction error* [26], the idea being to reward the agent whenever there is a mismatch between predicted and observed next states. Unfortunately, such approach suffers from the "noisy TV problem": if the environment is stochastic, the agent gets attracted to the source of noise. A cure was proposed in [27], which consisted in rewarding the agent for *prediction improvement* instead of prediction error. However, despite its theoretical appeal, prediction improvement is hard to compute in practice, especially with general function approximators [28].

In statistics and control engineering, the problem of 'optimal' exploration is known as *optimal input design* [7, 8] or *optimal experiment design* [9, 10]. A popular measure of novelty in these fields is the *information gain* [29]. In computer science, the problem of 'optimal' exploration is addressed by *Bayesian reinforcement learning* [3]. The general *dual control* solution [30], however, can only be obtained in very special cases [11]. Therefore, in most applications, *exploration bonuses* are employed, which stem from the "optimism in the face of uncertainty" principle [31]. *Bayesian exploration bonuses* [32] and other types of *visitation counts* [33] have been shown to be effective in video games [34, 35]. If the observation space is high-dimensional, exploration bonuses can be applied in the latent space. For example, latent-space prediction error and count-based exploration were combined in [36], while information gain was employed in [37]. *Self-supervised prediction* [28] and *random network distillation* [38] were proposed as different ways to compute the prediction error. A comprehensive study of curiosity-driven exploration methods can be found in [4].

Exploration bonuses are commonly added on top of a primary RL objective function to promote faster learning. However, such approaches do not scale to the multi-task and transfer learning settings because the knowledge gained during exploration is not reused. In contrast, model-based approaches compress the knowledge into the model and can later reuse it in any downstream task.

Our method can be seen as lying at the intersection of optimal sequential experiment design and nonlinear system identification. In the former, info-gain-maximizing strategies are well understood but for linear models; we use these insights by treating our model as linear in the last-layer parameters. In the latter, the focus is placed on numerical approaches and structured models (e.g., grey-box models such as Hammerstein-Wiener model); we use receding horizon control for numerical optimization and basis function expansion for representing the dynamics as a black box.

We stress that trajectory optimization is essential for making the problem computationally tractable. Approaches such as [12] and [10] rely on approximately solving the Bellman equation, which scales exponentially with the time horizon. On the other hand, belief space trajectory optimization scales polynomially [39], allowing for much longer horizons (e.g., we used $150 \leq T \leq 200$, whereas horizons of length $T \leq 4$ were considered in [12]).

6 Conclusion

A principled algorithm for trajectory-based active exploration in the model-based reinforcement learning setting has been proposed (Sec. 3). Two acquisition functions from active learning have been adapted to guide episodic exploration (Sec. 2.1): uncertainty sampling (US, Sec. 3.1) and expected variance reduction (EVR, Sec. 3.2). Since the acquisition functions cannot be straightforwardly evaluated due to intractability of the belief propagation over time, an approximation has been proposed, which led to a novel algorithm, called receding horizon curiosity (RHC, Algorithm 1).

The proposed RHC approach was compared to state-of-the-art model-based and model-free exploration algorithms on classical continuous control problems. Empirical evaluations showed that RHC achieves higher model likelihood and collects higher reward on downstream tasks in fewer iterations. Although not yet real-time capable, RHC was found to be computationally faster than MAX, thanks to being trajectory-based. The US objective (Sec. 4) delivered a better computation/performance trade-off, reaching the performance of EVR while being substantially easier to compute.

The experiments demonstrated that Bayesian curiosity w.r.t. last-layer parameters interleaved with nonlinear maximum likelihood feature learning can be successfully implemented and considerably improves exploration in low-dimensional classical control environments even under relatively strong deterministic state propagation assumption. Nevertheless, a number of obstacles need to be overcome to scale RHC to higher-dimensional problems. For instance, cheaper trajectory optimization methods (e.g., first-order, Hessian-free) could enable the use of larger number of features. Alternatively, dimensionality reduction techniques could allow for scaling the current approach by employing the same optimization framework but with lower-dimensional feature representations. Finally, tractable planning methods that can utilize more expressive probabilistic models, such as Bayesian neural networks, could allow for tackling even harder problems.

Acknowledgments

Calculations for this research were conducted on the Lichtenberg high performance computer of the TU Darmstadt. This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 640554 (SKILLS4ROBOTS).

References

[1] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134, 1998.

[2] C. H. Papadimitriou and J. N. Tsitsiklis. The complexity of markov decision processes. *Mathematics of operations research*, 12(3):441–450, 1987.

[3] M. Ghavamzadeh, S. Mannor, J. Pineau, A. Tamar, et al. Bayesian reinforcement learning: A survey. *Foundations and Trends® in Machine Learning*, 8(5-6):359–483, 2015.

[4] Y. Burda, H. Edwards, D. Pathak, A. Storkey, T. Darrell, and A. A. Efros. Large-scale study of curiosity-driven learning. In *ICLR*, 2019.

[5] P. Shyam, W. Jaśkowski, and F. Gomez. Model-based active exploration. *arXiv preprint arXiv:1810.12162*, 2018.

[6] Y. Sun, F. Gomez, and J. Schmidhuber. Planning to be surprised: Optimal bayesian exploration in dynamic environments. In *International Conference on Artificial General Intelligence*, pages 41–51. Springer, 2011.

[7] R. Mehra. Optimal input signals for parameter estimation in dynamic systems–survey and new results. *IEEE Transactions on Automatic Control*, 19(6):753–768, 1974.

[8] M. Gevers and X. Bombois. Input design: From open-loop to control-oriented design. *IFAC Proceedings Volumes*, 39(1):1329–1334, 2006.

[9] M. B. Zarrop. *Optimal experiment design for dynamic system identification*, volume 21. Springer, 1979.

[10] X. Huan and Y. M. Marzouk. Sequential bayesian optimal experimental design via approximate dynamic programming. *arXiv preprint arXiv:1604.08320*, 2016.

[11] E. D. Klenske and P. Hennig. Dual control for approximate bayesian reinforcement learning. *Journal of Machine Learning Research*, 17(127):1–30, 2016.

[12] C. K. Ling, K. H. Low, and P. Jaillet. Gaussian process planning with lipschitz continuous reward functions: Towards unifying bayesian optimization, active learning, and beyond. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[13] B. Settles. Active learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences, 2009.

[14] K. Chaloner and I. Verdinelli. Bayesian experimental design: A review. *Statistical Science*, pages 273–304, 1995.

[15] D. A. Cohn, Z. Ghahramani, and M. I. Jordan. Active learning with statistical models. *Journal of artificial intelligence research*, 4:129–145, 1996.

[16] C. M. Bishop. *Pattern recognition and machine learning*. springer, 2006.

[17] A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *Advances in neural information processing systems*, pages 1177–1184, 2008.

[18] A. Rajeswaran, K. Lowrey, E. V. Todorov, and S. M. Kakade. Towards generalization and simplicity in continuous control. In *Advances in Neural Information Processing Systems*, pages 6550–6561, 2017.

[19] J. Stoer and R. Bulirsch. *Introduction to numerical analysis*, volume 12. Springer Science & Business Media, 2013.

[20] J. A. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl. Casadi: a software framework for nonlinear optimization and optimal control. *Mathematical Programming Computation*, pages 1–36, 2018.

[21] M. Deisenroth and C. E. Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pages 465–472, 2011.

[22] R. Platt Jr et al. Belief space planning assuming maximum likelihood observations. In *Proceedings of the Robotics: Science and Systems Conference, 6th*, 2010.

[23] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

[24] P. J. Silvia. Curiosity and motivation. *The Oxford handbook of human motivation*, pages 157–166, 2012.

[25] R. M. Ryan and E. L. Deci. Intrinsic and extrinsic motivations: Classic definitions and new directions. *Contemporary educational psychology*, 25(1):54–67, 2000.

[26] J. Schmidhuber. A possibility for implementing curiosity and boredom in model-building neural controllers. In *Proc. of the international conference on simulation of adaptive behavior: From animals to animats*, pages 222–227, 1991.

[27] J. Schmidhuber. Curious model-building control systems. In *[Proceedings] 1991 IEEE International Joint Conference on Neural Networks*, pages 1458–1463. IEEE, 1991.

[28] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell. Curiosity-driven exploration by self-supervised prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 16–17, 2017.

[29] D. V. Lindley et al. On a measure of the information provided by an experiment. *The Annals of Mathematical Statistics*, 27(4):986–1005, 1956.

[30] A. Feldbaum. Dual control theory. i. *Avtomatika i Telemekhanika*, 21(9):1240–1249, 1960.

[31] T. L. Lai and H. Robbins. Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics*, 6(1):4–22, 1985.

[32] J. Z. Kolter and A. Y. Ng. Near-bayesian exploration in polynomial time. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 513–520. ACM, 2009.

[33] M. Kearns and S. Singh. Near-optimal reinforcement learning in polynomial time. *Machine learning*, 49 (2-3):209–232, 2002.

[34] M. Bellemare, S. Srinivasan, G. Ostrovski, T. Schaul, D. Saxton, and R. Munos. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems*, pages 1471–1479, 2016.

[35] G. Ostrovski, M. G. Bellemare, A. van den Oord, and R. Munos. Count-based exploration with neural density models. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2721–2730. JMLR. org, 2017.

[36] B. C. Stadie, S. Levine, and P. Abbeel. Incentivizing exploration in reinforcement learning with deep predictive models. *arXiv preprint arXiv:1507.00814*, 2015.

[37] R. Houthooft, X. Chen, Y. Duan, J. Schulman, F. De Turck, and P. Abbeel. Vime: Variational information maximizing exploration. In *Advances in Neural Information Processing Systems*, pages 1109–1117, 2016.

[38] Y. Burda, H. Edwards, A. Storkey, and O. Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018.

[39] S. Patil, G. Kahn, M. Laskey, J. Schulman, K. Goldberg, and P. Abbeel. Scaling up gaussian belief space planning through covariance-free trajectory optimization and automatic differentiation. In *Algorithmic foundations of robotics XI*, pages 515–533. Springer, 2015.

[40] A. Hill, A. Raffin, M. Ernestus, A. Gleave, R. Traore, P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, and Y. Wu. Stable baselines. https://github.com/hill-a/stable-baselines, 2018.

[41] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym, 2016.

[42] Y. Tassa, Y. Doron, A. Muldal, T. Erez, Y. Li, D. d. L. Casas, D. Budden, A. Abdolmaleki, J. Merel, A. Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

A Algorithms

Trajectory optimization for receding horizon curiosity (RHC) is implemented using CasADi [20], a control and auto-differentiation toolbox. The number of Fourier features in the learned dynamics model varies across environments: MountainCar 20, Pendulum 90, CartPole 80.

As a model-free baseline RL algorithm, soft actor-critic (SAC) is used [23], as implemented in Stable Baselines [40], with the following parameters across all environments

$$\begin{aligned} \gamma &= 0.99, \\ \tau &= 0.005, \\ \text{learning rate} &= 0.0003, \\ \text{buffer size} &= 50000, \\ \text{batch size} &= 64. \end{aligned}$$

The exploration bonus based on the *squared prediction error* is defined as follows

$$r_{\text{pe}}(s_t, a_t) = (s_{t+1} - \mathbb{E}_p[s_{t+1} \mid s_t, a_t])^2,$$

where p denotes the model trained on the data from previous episodes. The exploration bonus based on the *information gain* is defined as the reduction of entropy

$$r_{\text{ig}}(s_t, a_t) = \mathbb{H}(\boldsymbol{\theta} \mid \mathcal{X}_t) - \mathbb{H}(\boldsymbol{\theta} \mid \mathcal{X}_{t+1}),$$

where $\mathcal{X}_t$ denotes the set of observations until time step t, $\mathbb{H}$ is the entropy, and $\boldsymbol{\theta}$ denotes the model parameters.

B Environments

MountainCar. The implementation from OpenAI Gym [41] is modified as follows to accommodate the episodic exploration setting. Car power is set to 10^{-3} and the speed limit is removed. Upon reset, the car starts at the center of the valley with zero velocity. An episode ends when the car reaches the environment bounds or after 130 time steps. The evaluation task is to drive the car on top of the mountain as dictated by the stage cost $c = 10(x - x_{\text{goal}})^2 + 0.001a^2$, where x is the position of the car, x_{goal} is the goal location, and a the action.

Pendulum. The implementation from DeepMind Control Suite [42] with observations $[\cos\theta, \sin\theta, \dot{\theta}]$ is used with the following modifications. The pendulum is initialized handging down with zero velocity. Each episode consists of 100 time steps of 80ms duration each. The evaluation task is to swing the pendulum up as dictated by the stage cost $c = 100(1 - \cos\theta)^2 + 0.1\sin^2\theta + 0.1\dot{\theta}^2 + 0.001a^2$.

CartPole. The implementation from DeepMind Control Suite [42] with observations $[x, \cos\theta, \sin\theta, \dot{x}, \dot{\theta}]$ is used. Each episode starts with the cart at the center and the pole hanging down, both having zero velocity. The system is simulated at 50Hz. An episode ends when the cart reaches the state limits or after 100 time steps. The evaluation task is to swing the pole up, $c = 100x^2 + 100(1 - \cos\theta)^2 + 0.1\sin^2\theta + 0.1\dot{x}^2 + 0.1\dot{\theta}^2 + 0.1a^2$.

C Runtimes

Table 1 shows how long one run of each algorithm depicted in Fig. 1 on average takes. One run consists of 20 episodes (x-axis in Fig. 1). Evaluation of RHC EVR was only possible on the MountainCar environment due to its high memory demands. Evaluations were run on a machine with an Intel Xeon E5-2670 processor.

	MountainCar	Pendulum	CartPole
RHC EVR	1.51	-	-
RHC US	0.03	0.80	0.54
SAC PE	0.03	0.20	0.45
SAC IG	0.03	0.21	0.48
RAND	0.01	0.09	0.30
MAX	9.49	11.17	5.84

Table 1: Average wall-clock-time (in hours) for evaluated exploration algorithms (see Fig. 1).