LLNL-CONF-843897

Toward Multi-Fidelity Reinforcement Learning for Symbolic Optimization

F. L. Silva, J. Yang, M. Landajuela, A. Goncalves, A. Ladd, D. Faissol, B. Petersen

January 11, 2023

Workshop on Adaptive and Learning Agents
London, United Kingdom
May 29, 2023 through June 2, 2023

Toward Multi-Fidelity Reinforcement Learning for Symbolic Optimization

Felipe Leno Da Silva, Jiachen Yang, Mikel Landajuela, Andre Goncalves,
Alexander Ladd, Daniel Faissol, and Brenden Petersen
Lawrence Livermore National Laboratory
Livermore, CA, USA
{leno,yang40,landajuela1,goncalves1,ladd12,faissol1,bp}@llnl.gov

ABSTRACT

Due to its high sample-complexity, Reinforcement Learning (RL) has struggled to solve problems where sampling is prohibitively expensive or dangerous. For many of those domains, the solution consists of training RL using a simulator or surrogate model, which defines lower-fidelity estimates of the real environment. However, the learning algorithm is often unaware of the existence of the multiple fidelities, and the learning process is carried out in an ad-hoc manner, either modeling it as a transfer learning problem or trying to improve the simulator on the go. We propose to explicitly reason over the multiple fidelities. We introduce Multi-Fidelity Markov Decision Processes (MF-MDPs), where each fidelity has an associated cost, and propose two algorithms, RSEP and PGEP, to solve MF-MDPs. In this paper, we focus on solving Symbolic Optimization applications and provide an experimental evaluation in two relevant domains, Symbolic Regression and Antibody Therapeutics Optimization. We show in those challenging domains that our methods outperform baseline ways of coping with multiple fidelities and lead the way towards a whole new family of RL algorithms to solve MF-MDPs.

KEYWORDS

Multi-fidelity Learning, Symbolic Optimization, Reinforcement Learning, Risk-seeking Policy Gradients

1 INTRODUCTION

Reinforcement Learning (RL) has been used to solve numerous application problems with impressive performance. Super or expert-human performance has been achieved in varied applications [4, 33, 40, 43], showing that RL is in the vanguard of new AI developments and will be applied in ever-broader range of domains in the next years. RL has been used with remarkable success in symbolic optimization[1], where the state-of-the-art solution for symbolic regression is RL-based.

However, one common feature of those applications is that an accurate reward signal is freely available during the training process, in which the domain can often be executed in a faster-than-real-world pace. This means that, for those domains, the RL agent had access to numerous high-quality training samples, which might not be available at every domain.

[1]A discrete optimization problem used to model varied applications ranging from neural architecture search to healthcare applications. The RL-based Deep Symbolic Regression [24] is often considered to be the state-of-the-art solution.

Proc. of the Adaptive and Learning Agents Workshop (ALA 2023), Cruz, Hayes, Wang, Yates (eds.), May 9-10, 2023, Online, https://ala2023.github.io/. 2023.

This is especially challenging in domains in which exploration can be potentially harmful or very expensive, such as robotics [17], autonomous driving [16], or healthcare-related applications [31]. The most popular way to cope with expensive sampling is by building a simulator or some other kind of surrogate reward for allowing exploring the task enough to solve it. However, this often means learning from a lower fidelity, approximate, reward that often (if not always) results in a different optimal policy than the one from the original problem.

This mismatch in the reward functions has often been dealt with as a Transfer Learning problem [9, 29, 30], where the learning process is carried out using the cheap simulation and then the resulting policy is either adapted [13] or transferred directly [8, 22] to the desired domain. However, we argue that it is more effective to explicitly reason over multiple reward functions in different fidelities, allowing a more faithful modeling of the problem at hand and more effective exploration in the highest (and therefore more expensive to sample) fidelity.

However, most RL methods, including the symbolic optimization-specific algorithms, assume that only one reward is available. Despite being relatively common for Bayesian Optimization [23], most multi-fidelity methods focus on modifying the simulator, which is often not possible.

In this paper, we propose a new multi-fidelity framework to learn in the presence of multiple reward functions of different fidelities. We are mainly interested in Symbolic Optimization problems and hence primarily focus on the Symbolic Regression and Antibody Therapeutics applications (to be introduced in the next sections). Although our proposed algorithm is specific to RL-based Symbolic Optimization, our problem modeling and most of the ideas presented here are adaptable at least in the high level to any RL domain.

2 BACKGROUND

In this section, we present the needed background on Reinforcement Learning and Symbolic Optimization to understand our approach.

2.1 Reinforcement Learning

Reinforcement Learning (RL) solves sequential decision making problems where the Markov condition holds (the optimal actions can be chosen based only in the current state of the world). Those problems are formally described as Markov Decision Processes: $\langle S, A, T, R \rangle$, where S is the set of possible states, A is the set of applicable actions, $T : S \times A \rightarrow S$ is the state transition function that computes the probability of transitioning to a particular state when a given action is applied in the current state, and R is the reward function.

Because R and T are initially unknown, RL algorithms explore actions so that they can estimate the sequence of actions that results in the highest as possible sum of rewards. The RL agent aims at finding the optimal policy $\pi^* : S \longrightarrow A$, which maps each possible state to an applicable action in a way that it maximizes the sum of discounted rewards over time: $\pi^*(s_0) = \arg\max_{a_0 \in A} R(s_0, a_0) + \sum_{i=1}^{h} \gamma^i R(s_i, \pi^*(s_i))$, where γ is a discount factor and $s_i = T(s_{i-1}, a_{i-1})$. The optimal policy is often searched through iteratively learning a Deep Neural Network-based Policy Gradient model that directly predicts the action given the state. Generally, the network parametrized by $\theta \in \mathbb{R}$ is updated by following a gradient estimator: $\theta_{i+1} \longleftarrow \theta_i + \alpha \nabla_\theta J(\theta_i)$, where α is a learning rate and J is a objective function based on the expected reward return.

2.2 Deep Symbolic Optimization

In this paper, we are mainly interested in solving symbolic optimization problems. This problem consists of finding the optimal combination of discrete symbols relative to a quality score function. Let $\mathcal{L} = \{\tau^1, \ldots, \tau^t\}$ be the *library*, i.e., a set of *tokens* τ^i which define the space $\mathbb{T}$ of possible *token sequences* $\tau = \langle \tau_1, \ldots, \tau_n \rangle$ that can be built for solving the problem at hand. The reward function $R : \mathbb{T} \longrightarrow \mathbb{R}$ defines the fitness of each sequence. Any *valid* sequence τ can be "evaluated", i.e., have its reward value queried by $R(\tau)$. The solution of a symbolic optimization problem is given by:

$$\underset{n \in \mathbb{N}, \tau}{\operatorname{argmax}} [R(\tau) \text{ with } \tau = \langle \tau_1, \ldots, \tau_n \rangle, \tau_i \in \mathcal{L}. \tag{1}$$

That is, the problem is solved by finding the sequence that maximizes the reward function. The main challenge of this problem is searching the set of possible sequences, which defines a huge or infinite search space. The *Deep Symbolic Optimization* (DSO) algorithm [24] is one of the state-of-the-art solutions for the symbolic optimization problem. DSO is based on RL, where the token sequence is formed by sequentially sampling each token (action) conditioned on all tokens sampled so far (state). Although DSO successfully solved a variety of domains [10, 19, 24, 25, 31, 39], it currently cannot handle multifidelity rewards.

DSO is based on the Risk-Seeking Policy Gradient:

$$J(\theta) \coloneqq \mathbb{E}_\theta \left[R(\tau) \mid R(\tau) \geq Q_\epsilon \right] \tag{2}$$

$$\nabla_\theta J \approx \frac{1}{\epsilon N} \sum_{i=1}^{N} \left[R(\tau^{(i)}) - \tilde{R}_\epsilon(\theta) \right] \cdot \mathbf{1}_{R(\tau^{(i)}) \geq \tilde{R}_\epsilon(\theta)} \nabla_\theta \log p(\tau^{(i)} | \theta), \tag{3}$$

where Q_ϵ is the $(1-\epsilon)$-quantile of the reward distribution under the policy, $\tilde{R}_\epsilon(\theta)$ is the empirical $(1-\epsilon)$-quantile of the batch of rewards, and $\mathbf{1}_x$ returns 1 if condition x is true and 0 otherwise. Essentially, this estimator optimizes for the discovery of the *best* token sequence directly, instead of focusing on optimizing average returns. This operator is appropriate for applications in which finding higher-performing solutions faster is more important, which is the case for our evaluation domains.

3 PROBLEM STATEMENT

We introduce the Multi-Fidelity Markov Decision Process (MF-MDP) as $\langle S, A, \boldsymbol{T}^{MF}, \boldsymbol{R}^{MF} \rangle$. However, in the multi-fidelity case, we have multiple state transition and reward functions:

$$\boldsymbol{T}^{MF} = \langle T^0, T^1, \ldots, T^{n_f} \rangle$$
$$\boldsymbol{R}^{MF} = \langle R^0, R^1, \ldots, R^{n_f} \rangle$$

Each transition and reward function T^i and R^i is associated to a different source $\Xi^i = \langle T^i, R^i \rangle$, using which the agent can apply actions and explore the environment. The fidelity $f = 0$ (Ξ^0) is the "real environment", and therefore the agent objective is to maximize the reward in this fidelity:

$$\pi^* \coloneqq \underset{\pi \in \Pi}{\operatorname{argmax}} \Bigg(R^0(s_0, \pi(s_0)) + \sum_{i=1}^{h} \gamma^i R^0(T^0(s_{i-1}, a_{i-1}), \pi(T^0(s_{i-1}, a_{i-1}))) \Bigg). \tag{4}$$

where we have assumed deterministic environment dynamics, as is the case for symbolic optimization.

However, each source Ξ^i is associated to a sampling cost c, where $\forall i > 0 : c(\Xi^0) \gg c(\Xi^i)$, in a way that sampling directly from Ξ^0 until the problem is solved is often impossible. However, all other sources Ξ^i approximate the real-world source Ξ^0, and can be used to bootstrap learning and enable finding a good policy for Ξ^0 with a reduced number of samples[2]. In the general case, no specific ordering in sampling cost or accuracy of predictions is assumed for fidelities other than $f = 0$ [3]. S and A have the same definition as in the regular MDP (therefore, all fidelities are assumed to have the same state-action space).

Intuitively, MF-MDPs fit with a variety of domains of relevance, for example, Robotics [18], where Ξ^0 represents extracting trajectories from a real robot, while the other fidelities could be extracting trajectories from Robotics simulators, which are exponentially cheaper and safer to extract, but are only an approximation of the real task to be solved. Since we are here concerned with Symbolic Optimization problems, for which the rewards are computed only for whole token sequences, we are dealing with a special case where the transition function is the same for all fidelities.

4 SOLVING MULTI-FIDELITY SYMBOLIC OPTIMIZATION

Given the problem description in the last section, we propose a framework called *Multi-Fidelity Deep Symbolic Optimization* (MF-DSO). MF-DSO divides this learning problem in two stages:

(1) **Multi-fidelity Sampling**: At the start of the episode, the algorithm must choose a fidelity to sample from, so that the proper transition and reward functions are processed. In our case, since all transition functions are the same, we generate a batch $\mathcal{T}$ of

[2]In the remainder of this paper, we generally assume that the sampling cost for non-zero fidelities is negligible and does not need to be considered. However, it would also be possible to consider a joint optimization scenario where the agent aims at maximizing the policy quality while minimizing the total sampling cost.

[3]Assuming a strict ordering of the fidelities is a common assumption [7]. However, for most domains, it is more realistic to only expect that $f = 0$ is strictly a better estimate of the reward, while the other fidelities might be reasonable estimates only in portions of the state-action space.

trajectories (token sequences) and chose a fidelity or multiple fidelities for each individual sample[4].

(2) **Multi-fidelity Learning**: With a batch of sampled trajectories at hand, a proper learning objective must be defined. Given that the batch might contain a mixture of samples with rewards calculated using different fidelities, this has to be taken into account when updating the policy.

An algorithmic view of the whole training loop is depicted in Algorithm 1. First, the current policy is used to generate a batch of samples. Then, the *SAMPLING* method is used to define which fidelity to use for each of the samples. With the complete training batch defined, *LEARN* performs the policy gradient update. This process is repeated until a termination condition is achieved (for example, a budget number of samples in $f = 0$ or a total wall-clock run time). During the course of training, we save the *Hall of Fame* (HoF), defined as the set of best samples found so far.

Algorithm 1 Multi-Fidelity Deep Symbolic Optimization

```
Require: $\Gamma_\theta$: Policy network parameterized by $\theta$; $\Xi$: set of available fidelities; $n_b$: Size of the batch; SAMPLING: method for defining which fidelity to use; LEARN: method for updating the policy network.
1:  HoF ← $\emptyset$
2:  initiate network parameters $\theta$
3:  while termination condition not achieved do
4:      $\mathcal{T} \leftarrow \Gamma_\theta(n_b)$                              ▷ Generate samples
5:      $\{\mathrm{fid}(\tau)\} \leftarrow SAMPLING(\mathcal{T}, \Xi)$    ▷ Fidelity for each sample
6:      for $\forall \tau \in \mathcal{T}$ do
7:          $f \leftarrow \min(\mathrm{fid}(\tau.r), \mathrm{fid}(\tau))$        ▷ Define best fidelity
8:          $\tau.r \leftarrow R^f(\tau)$ ▷ Define reward according to chosen fidelity
9:      end for
10:     $\theta \leftarrow LEARN(\theta, \mathcal{T})$                  ▷ Update Policy Network
11:     HoF ← $hof\_update(\mathrm{HoF}, \mathcal{T})$
12: end while
13: return HoF
```

4.1 Multi-fidelity Sampling

The sampling algorithm is critical to a multi-fidelity problem, given that $c(\Xi^i)$ is spent every time a sample is evaluated in fidelity i. As shown in last section, *SAMPLING* receives as input a batch of samples $\mathcal{T}$ and has to choose a fidelity to calculate R^i for each sample. *SAMPLING* can also evaluate a particular sample in multiple fidelities, but only the highest fidelity evaluated so far, which we denote using "object.property" notation by $\tau.r$, will be used by the training method.

Our proposed sampling method is called **Elite-Pick sampling** and is based on an elitism mechanism. Similar to Cutler et al. [7], we follow the intuition that it is advantageous to sample in the highest fidelity only when we expect to have a high performance. Apart from fitting well with the risk-seeking learning method (describe in the next section), biasing the sampling towards high-performance samples might help avoid exploring trajectories with harmful consequences (for example, breaking a robot).

[4]This might include the use of conditional selection of fidelities, whereby the choice of certain fidelities depends on the sample values in other fidelities.

For this procedure, all samples are initially evaluated in the lowest fidelity R^{low}. Those estimates are used to calculate the empirical $(1-\rho)$-quantile $Q_\rho(\mathcal{T}, R^{\text{low}})$, and only the samples in the top quantile (i.e., those with $\tau.r > Q_\rho$) are evaluated in R^0:

$$SAMPLING_\rho = \begin{cases} 0 & \text{if } \tau.r \geq Q_\rho \\ \text{low} & \text{otherwise} \end{cases} : \tau \in \mathcal{T} \tag{5}$$

After sampling, $\tau.r$ is set equal to the value of the best fidelity reward sampled so far for τ. When more than two fidelities are available, one may interpret low by randomly sampling from a mixture of the non-zero fidelities for both the initial evaluation and for the return value in the case where $\tau.r < Q_\rho$.

4.2 Multi-fidelity Learning

After the sampling method is applied, the learning algorithm has to use $\mathcal{T}$ for updating the policy network. Since $\mathcal{T}$ might contain a mixture of samples in different fidelities, we propose two learning algorithms that explictly account for that.

Weighted Policy Gradient: This is a Policy Gradient algorithm where a different weight is used for each fidelity (where, naturally, a higher weight is expected to set to $fid = 0$).

$$J_{PG}(\theta) := \mathbb{E}\left[\gamma l(R^0(\mathcal{T}_0)) + \sum_{f=1}^{f=n_f} \frac{1-\gamma}{n_f - 1} l(R^f(\mathcal{T}_f)\right] \tag{6}$$

where l is a simple loss function (e.g., REINFORCE [35]), and γ is a weighting parameter. This learning algorithm with elite-pick sampling is henceforth called **PGEP**. We also consider a variation of the algorithm where all fidelities have the same weight **PGEP_u**.

Following PGEP, we introduce a more theoretically-backed algorithm for this problem:

Multi-Fidelity Risk-Seeking: Inspired by the risk-seeking policy gradient algorithm described in Section 2, we propose a multi-fidelity risk-seeking objective:

$$J_\epsilon(\theta) := \mathbb{E}_\theta\left[R^0(\tau) \mid \tau.r \geq Q^m_\epsilon\right], \tag{7}$$

where Q^m_ϵ is the top $(1-\epsilon)$-quantile of $\tau.r$ for $\tau \in \mathcal{T}$. Here, and in the following, we use superscript "m" to denote "mixture", since Q^m_ϵ is computed on a batch of samples whose $\tau.r$ may belong to different fidelities. Intuitively, we want to find a distribution π_θ such that the top ϵ fraction of samples (as evaluated using the best fidelity sampled so far) have maximum performance when evaluated using the highest fidelity reward R^0. This is motivated by the fact that fidelities $f \neq 0$ are much cheaper to evaluate during training, which justifies their use in filtering for the top ϵ samples. Crucially, note that (7) is well-defined at each iteration of the algorithm based on the fidelity of $\tau.r$ for each $\tau \in \mathcal{T}$, while the fidelities may improve after an iteration when a better fidelity is sampled. This process of improving fidelity will be justified below by Proposition 2.

The gradient of (7) is computed as:

$$\nabla_\theta J_\epsilon(\theta) = \mathbb{E}_\theta\left[\nabla_\theta \log \pi_\theta(\tau)(R^0(\tau) - R^0(\tau_\epsilon)) \mid R^m(\tau) \geq Q^m_\epsilon(\theta)\right] \tag{8}$$

where $R^m(\tau) := \tau.r$, $Q^m_\epsilon(\theta) = \inf_{\tau \in \Omega}\{R^m(\tau) \colon F^m_\theta(r) \geq 1-\epsilon\}$ denotes the $(1-\epsilon)$-quantile, and $\tau_\epsilon = \arg\inf\{R^m(\tau) \colon F^m_\theta(r) \geq$

$1-\epsilon\}$ is the sample that attains the quantile. We show how we derived the gradient in Appendix A. Hence, we can apply stochastic gradient ascent along the direction Equation (8) to find a local maximum of (7).

We call Risk-Seeking learning allied with Elite-Pick sample generation as **RSEP** henceforth. We also consider a variation of the algorithm where, after sampling is applied, all the samples currently sampled in $fid = 0$ are used to recalculate $Q_{\epsilon_2}^{fid(\tau.r)=0}$, which can further discard low-quality samples in $f = 0$. This means that additional samples might be discarded from the learning update based on their updated value of $\tau.r$ after the sampling process. We name this variation as **RSEP_0**.

4.2.1 Theoretical Analysis of RSEP. Since RSEP uses a mixture of high and low-fidelity samples to compute the quantile for filtering (i.e. selecting $\tau\colon \tau.r \geq Q_\epsilon^m$), whereas one would use only Q_ϵ^0 if this was feasible, we would like to understand the probability of wrong exclusion: the probability that a sample would have passed the high-fidelity filter Q_ϵ^0 but was wrongly rejected by the low-fidelity filter Q_ϵ^m. Proposition 1 below, derived in Appendix A, shows that the probability of wrong exclusion scales according to the cumulative distribution of the noise as a function of the difference in quantiles.

Proposition 1. *Let R^0 and R^1 be random variables related by noise $N\colon R^1 := R^0 + N$. Let Q_ϵ^0 and Q_ϵ^1 be the $(1-\epsilon)$-quantiles of the distributions of R^0 and R^1, respectively. Then we have that*

$$P(R^0 \geq Q_\epsilon^0, R^1 \leq Q_\epsilon^1) = \epsilon \mathbb{E}\left[F_N(Q_\epsilon^1 - R^0) \mid R^0 \geq Q_\epsilon^0\right] \quad (9)$$

where $F_N(r)$ is the CDF of the noise distribution.

From this, we can make two intuitive observations: 1) the smaller the ϵ used by the "true" high-fidelity risk-seeking objective, the smaller the probability of error; 2) The smaller the low-fidelity quantile Q_ϵ^1, the more likely a sample is to pass the low-fidelity filter, and hence the smaller the probability of error.

It would be also useful to understand under which conditions optimizing for the RSEP objective corresponds to optimizing for risk-seeking policy gradient. In the following, we show that the RSEP algorithm eventually maximizes the same objective as the risk-seeking policy gradient. First, we need the following assumption:

Assumption 1. *One of the following holds:*

- ***Case 1:** As part of the sampling method, we include a non-zero probability of sampling $f = 0$ for each trajectory τ regardless of its current $\tau.r$.*
- ***Case 2:** For all $\tau \in \mathcal{T}$, we have $R^0(\tau) \leq R^f(\tau)$ for $f \neq 0$.*

Case 2 corresponds to a typical scenario where lower-resolution fidelities $f \neq 0$ are overly optimistic, which might preclude the algorithm from ever sampling a trajectory in $f = 0$ as shown in the following.

Proposition 2. *Let J_{risk} be the Risk-Seeking Policy Gradient objective:*

$$J_{risk}(\theta) := \mathbb{E}_\theta\left[R^0(\tau) \mid R^0(\tau) \geq Q_\epsilon^0\right] \quad (10)$$

and J_{RSEP} be the RSEP objective (Equation 7). Given Assumption 1, optimizing for the RSEP objective, in the limit of infinite exploration, corresponds to optimizing for the risk-seeking objective.

Proof. We show that for both cases of Assumption 1, we have $\tau.r = R^0(\tau)$ and $Q_\epsilon^m = Q_\epsilon^0$ in the limit.

For Case 1: Since all sequences have a non-zero probability of being evaluated in R^0 regardless of their reward values in the lowest fidelities, in the limit of infinite exploration we have that $\forall\tau, \tau.r = R^0(\tau)$ and $Q_\epsilon^m = Q_\epsilon^0$. This holds because, eventually, all samples will be evaluated in $f = 0$ regardless of their reward values due to the random sampling component, permanently replacing $\tau.r$ with R^0 values.

For Case 2: We show that RSEP will eventually only train on samples τ that satisfy $R^0(\tau) \geq Q_\epsilon^0$. First, by Assumption 1, for any fixed batch of samples, we have the inequality among the empirical quantiles:

$$Q_\epsilon^0 \leq Q_\epsilon^m\,. \quad (11)$$

Now we enumerate all cases that may arise during the evaluation of (8).

(1) $R^0(\tau) < Q_\epsilon^0$ and $\tau.r \geq Q_\epsilon^m$. This means it mistakenly passes the multi-fidelity risk-seeking filter. However, due to passing the filter, we will have $\tau.r = R^0(\tau)$ subsequently. This means that this sample will never pass the filter on subsequent evaluations of the same batch because $R^0(\tau) < Q_\epsilon^0$ and (11).
(2) $R^0(\tau) \geq Q_\epsilon^0$ and $\tau.r \geq Q_\epsilon^m$. This case is correct since τ is supposed to contribute to the gradient and it does so by passing the filter. Also note that we will have $\tau.r = R^0(\tau)$ after the gradient computation.
(3) $R^0(\tau) < Q_\epsilon^0$ and $\tau.r < Q_\epsilon^m$. This case poses no issue since τ is not supposed to contribute to the gradient and it does not do so due to failing to pass the filter.
(4) $R^0(\tau) \geq Q_\epsilon^0$ and $\tau.r < Q_\epsilon^m$. If this case persists across training, then τ will never be used in the gradient computations even though it should. So we need to show that this case eventually stops arising. If $\tau.r = R^0(\tau)$, then there is no issue because it contributes correctly to Q_ϵ^m. So we consider the case $\tau.r = R^{f\neq 0}(\tau)$. This arises only if there exists τ' that is wrongly accepted into the quantile: i.e., $R^0(\tau') < Q_\epsilon^0$ and $\tau'.r = R^{f\neq 0}(\tau')$ and $R^{f\neq 0}(\tau') \geq Q_\epsilon^m$, which is case (1) above. However, we have shown that scenario (1) eventually does not arise, which guarantees that this scenario eventually does not arise.

Therefore, only scenario that persist are scenarios (2) and (3), which are correct. Performing a simple substitution in Equation 7:

$$\lim_{\text{training}} J_{\text{RSEP}} = \mathbb{E}_\theta\left[R^0(\tau) \mid R^0(\tau) \geq Q_\epsilon^0\right] = J_{\text{risk}} \quad (12)$$

Therefore, by learning using RSEP we are, in the limit, optimizing for the risk-seeking policy gradient objective. □

However, we expect that high-quality sequences will be found much quicker than when using a single fidelity, which will be shown in the empirical evaluation.

5 EMPIRICAL EVALUATION

In order to empirically evaluate MF-DSO, we consider two highly-relevant domains, *Symbolic Regression* and *Antibody Therapeutics Optimization.* While both domains are of practical importance and motivated the developments reported in this paper, the former provides a platform where Benchmarks are well-defined and experiments can be performed quickly, while the latter represents a challenging domain where freely sampling in the highest fidelity is completely infeasible.

5.1 Symbolic Regression

In this section we introduce *Symbolic Regression*, a problem that is particularly amenable to be solved through symbolic optimization. Symbolic regression aims to identify mathematical expressions that best fit a set of observations. This can be used, for example, to extrapolate equations that explain physical phenomena, by searching over the space of tractable (i.e. concise, closed-form) expressions.

Specifically, given a dataset $(\boldsymbol{X}, \boldsymbol{y})$, where each observation $\boldsymbol{X_i} \in \mathbb{R}^n$ is related to a target value $y_i \in \mathbb{R}$, symbolic regression aims to identify a function $f : \mathbb{R}^n \to \mathbb{R}$ that best fits the dataset, where the functional form of f is a short mathematical expression. The resulting expression can be readily interpreted and/or provide useful scientific insights simply by inspection. Naturally, a candidate expression just have to be compared against the dataset to compute an error metric.

Symbolic regression exhibits several unique features that make it an ideal domain for benchmarking symbolic optimization: (i) Well-established and challenging benchmarks are available [41]; (ii) The success criteria is clearly defined; (iii) Well-established baseline methods are available (most notably, the Eureqa algorithm [27]); and (iv) Computing the quality of candidate expressions is easy, allowing repeating experiments until statistical significance is achieved. The space of mathematical expressions is discrete (in model structure) and continuous (in model parameters), growing exponentially with the length of the expression, rendering symbolic regression a challenging machine learning problem—thought to be NP-hard [21]. For evaluation purposes, we leverage the Nguyen symbolic regression benchmark suite [38]. Nguyen is a set of 12 commonly used benchmark expressions developed and vetted by the symbolic regression community [41]. Each benchmark is defined by a ground truth expression and a set of allowed operators.

Training and test data are generated based on the ground truth expression and used to compute the error between a candidate expression and the correct one. This is Ξ^0 for us and therefore $R^0(\tau, X) = 1 - \sqrt{\frac{1}{|X|}\sum_{x \in X} |(\tau.evaluate(x) - y(x)|^2}$. Given the evaluation data $(\boldsymbol{X}, \boldsymbol{y})$, we consider other lower-fidelity rewards:

(1) Ξ^1: We add white noise to the evaluation data, in a way that the rewards are calculated using $(\boldsymbol{X}, \boldsymbol{y} + \epsilon)$.
(2) Ξ^2: Instead of using the original data, we train a simple Gaussian Process Regressor on the data $m(\boldsymbol{X}, \boldsymbol{y})$. Therefore, $R^2(\tau, X) = \frac{1}{|X|}\sum_{x \in X} 1 - |(\tau.evaluate(x) - m(x)|$, which simulates a common situation where a surrogate model is trained in real-world data to provide a faster and cheaper low-fidelity estimator (as in most of real-world scenarios, the amount of data used is not enough for the model to converge to a perfect prediction).

We show results for Nguyen 4-6, 9, 10, and 12 for all experiments in the main text of the paper. Those benchmarks were chosen because they represent the main trend of the results for both middle- and high-difficulty ground truth equations.

Due to the speed with which we can run experiments in this domain, *Symbolic Regression* is used for a wider range of comparisons, described in the following.

5.1.1 *Baseline Multi-Fidelity Performance.*

This series of experiments aim to answer the question *"Is it useful to use multi-fidelity samples?"*; and to assess the performance of simple multi-fidelity strategies. The following baselines are considered:

- **upper bound**: Equivalent to only using Ξ^0. Given unlimited samples, this baseline should be the top performer. However, we are here interested in the scenario in which samples from the highest fidelity are limited.
- **lower bound**: Only uses Ξ^1 and Ξ^2. This baseline shows the agent performance in the lower fidelities when the real world is not available.
- **sequential**: This baseline mimics a transfer learning approach. Learning is carried out without access to Ξ^0 for a number of iterations, then the agent switches to Ξ^0 and only samples from the high fidelity until the end of training.
- **shuffled**: This baseline randomly samples from different fidelities according to a fixed probability. The highest fidelity is set to around 9% of probability to be sampled from.

Figure 1 shows the best sample found per number of explored samples in $f = 0$. Although those graphs cannot be interpreted alone[5], they present a gross estimation of the learning speed of each algorithm. Table 1 shows the average quality of the hall of fame after training for this experiment, providing the extra information we needed to assess the performance. As expected, *lower bound* shows that sampling only from the lowest fidelity overfits and presents as solutions low-performing samples in the fidelity that matters to us. Although *shuffled* sometimes achieves high-performing samples (shown e.g. on Nguyen-6), the mixture of fidelities on the training batches confuses the agent, which results in low *avg* metric overall in most of the benchmarks. Despite the poor performance from those aforementioned baselines, *sequential* outlines the potential of leveraging multiple fidelities. With the same budget of samples from $f = 0$, *sequential* achieves consistently a better performance than *upper bound* (evidenced in both graph and table), showing that it is beneficial to use lower fidelities to boostrap learning.

With our first experiment indicating that there is indeed an advantage in using multiple fidelities, our next experiment will show that MF-DSO outperforms the baselines.

5.2 Symbolic Regression Evaluation

We evaluate here the performance of all of the our proposed methods: *RSEP*, *RSEP_0*, *PGEP*, and *PGEP_u*; as well as the best performing baseline method *sequential.* Table 2 shows the results for all benchmarks. For both the *max* and *avg* metrics, *RSEP* outperformed all other algorithms in, respectively, 4 and 2 of the benchmarks, which clearly makes it the best performing algorithm in this experiment. *RSEP_0*, a variant of the same algorithm, ranked best of all algorithms in 1 and 2 of benchmarks for each of the metrics. Finally, *PGEP_u* ranked best 1 and 2 times in the metrics.

[5]A good sample found by the algorithm is not necessarily stored in the hall of fame. If a sample is overestimated in another fidelity, the best sample so far might be discarded depending on the algorithm used (this happens, e.g., with *shuffled*)

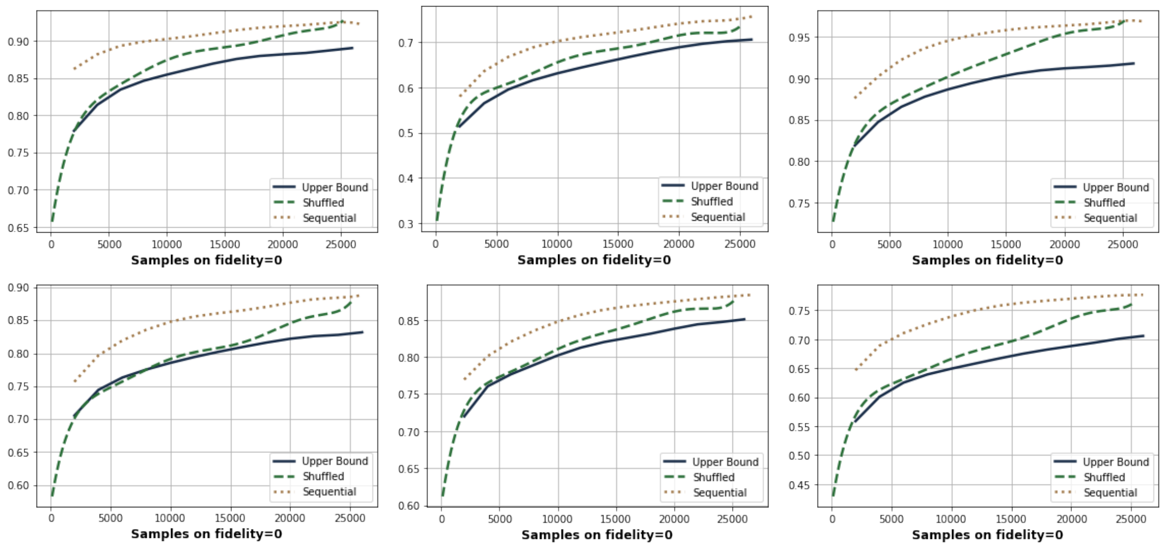

Figure 1: Average reward of best sample found so far during training (x-axis is the amount of samples from Ξ^0) across 150 repetitions. Nguyen 4-6, 9, 10, 12 are depicted from left to right, top to bottom.

Table 1: The results represent the best (max) and the average (avg) quality of samples in the hall of fame by the end of the training process. Averages across 150 **repetitions.**

Benchmark	**Lower bound**		**Upper bound**		**Shuffled**		**Sequential**	
	Max	*Avg*	*Max*	*Avg*	*Max*	*Avg*	*Max*	*Avg*
Nguyen-4	0.884	0.703	0.890	0.788	0.923	0.786	**0.925**	**0.846**
Nguyen-5	0.530	0.257	0.705	0.511	0.728	0.505	**0.754**	**0.563**
Nguyen-6	0.800	0.578	0.918	0.820	0.966	0.839	**0.969**	**0.859**
Nguyen-9	0.396	0.300	0.832	0.702	0.875	0.720	**0.889**	**0.761**
Nguyen-10	0.498	0.355	0.851	0.726	0.872	0.712	**0.883**	**0.744**
Nguyen-12	0.526	0.366	0.706	0.561	0.758	0.505	**0.777**	**0.621**

Notably, the best baseline method (sequential) was not able to outperform the multi-fidelity-specific algorithms in any of the metrics or benchmarks. The main conclusion of this experiment is that the proposed algorithms provide significant gains in multi-fidelity environments. As for choosing one of our proposed algorithms, *RSEP* was consistently the best algorithm or at least one of the top performers in all benchmarks, therefore this should be the choice when the designer does not have means to perform ablations and evaluations of different algorithms before the training process. However, the other algorithms also have their value (most notably *RSEP_0* and *PGEP_u*) and sometimes outperform *RSEP*.

5.3 Antibody Therapeutics Optimization

Antibodies are the human immune system's primary line of defense against pathogens. As a response to an unwanted substance (antigen - often a virus or bacteria) entering the body, antibodies are produced and released in the bloodstream and lymph systems. Antibodies are able to bind to specific antigens, effectively neutralizing the threat and fighting the disease. Instead of relying on the individual's immune system to generate new antibodies, modern medicine is able to develop and manufacture antibodies [5], which can be directly administered to the patient, improving survival rates. Given a target antigen we want to develop antibodies for (e.g., SARS-CoV-2), we follow a *fast response* approach. We start with an antigen that is within the same viral family of our target (e.g., SARS-CoV-1), for which we have an effective antibody (that does not bind effectively to the new target). Given that both antigens are related, we hypothesize that we need only to perform strategic mutations in the antibody for it to bind to our target, rather than developing an antibody from scratch.

Antibodies are proteins that sensitively and specifically recognize target molecules by the shape and chemistry of their complementarity determining regions (CDRs) [42]. Hence, we perform mutations on the antibody's CDR. There are 20 common amino acids, each with distinct sizes, shapes, chemistries, and other characteristics, and sequences of amino acids are conventionally written as sequences of alphabet letters. Therefore, the symbolic optimization modeling of this domain consists of sampling amino acids mutations (among the 20 choices) in the antibody CDR. Given the high cost of performing wet lab experiments, we rely on a computationally-assisted approach to train the AI. The reward for

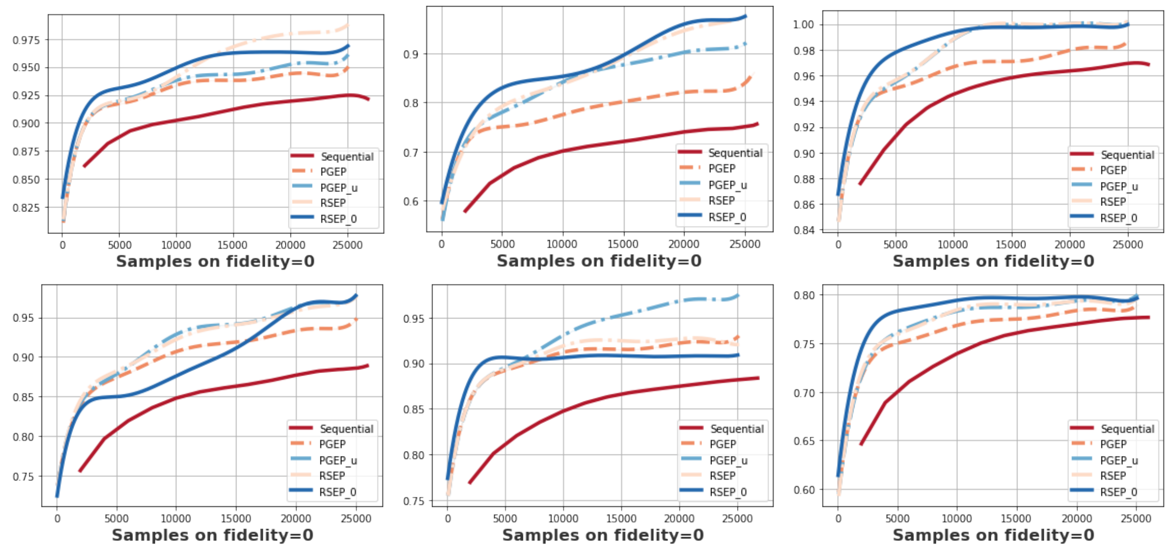

Figure 2: Average reward of best sample found so far during training (x-axis is the amount of samples from Ξ^0) across 150 repetitions. Nguyen 4-6, 9, 10, and 12 are depicted from left to right.

Table 2: The results represent the best (max) and the average (avg) quality of samples in the hall of fame by the end of the training process. Averages across 150 repetitions. Best results for each metric are highlighted in bold.

Benchmark	Sequential		PGEP		PGEP_u		RSEP		RSEP_0	
	Max	*Avg*	*Max*	*Avg*	*Max*	*Avg*	*Max*	*Avg*	*Max*	*Avg*
Nguyen-4	0.925	0.846	0.946	0.894	0.956	0.921	0.985	0.947	**0.991**	**0.961**
Nguyen-5	0.754	0.563	0.832	0.668	0.913	0.761	**0.966**	0.801	0.960	**0.823**
Nguyen-6	0.969	0.859	0.983	0.944	0.999	**0.981**	**1.000**	0.965	0.999	0.942
Nguyen-9	0.889	0.761	0.941	0.838	0.972	0.849	**0.973**	**0.858**	0.968	0.844
Nguyen-10	0.883	0.744	0.925	0.858	**0.971**	**0.901**	0.927	0.862	0.908	0.819
Nguyen-12	0.777	0.621	0.786	0.691	0.796	0.762	**0.792**	**0.776**	0.795	0.772

this domain represents the binding quality of a suggested antibody to our target, where low and high fidelities are defined as follows:

- Ξ^0: The highest fidelity uses a Rosetta Flex [20] simulation to define the binding score. Although this provides a good estimation the reward, each Rosetta reward simulation takes approximately 6 hours to execute in an HPC computer. For each mutation from the original antibody CDR, we use Rosetta to compute the change in binding free energy between the antigen and the antibody, and we define the final reward as the sum of all single-point mutation differences.
- Ξ^1: Given the excessive time to run Rosetta simulators, we train a surrogate model to use as a lower-fidelity estimation of the binding score. We train a Gaussian process (GP) model with a dataset of 122,000 pre-defined Rosetta simulations. Due to the size of the dataset, we trained an approximate sparse Gaussian Process [34] with scaled RBF kernel and hyperparameters selected via marginal likelihood maximization. The trained GP model is able to estimate rewards in a couple of seconds.

Given the long time needed to run experiments in this domain, we leverage the ablations and results from the Symbolic Regression experiments to down-select algorithms for the evaluation in this domain. We evaluate: (i) **Upper Bound**: training only on Ξ^0; (ii) **Sequential**: training on Ξ^1 for a long time then switching to Ξ^0; (iii) **RSEP**; and (iv) **PGEP_u**, the top-performing algorithms from each category in the last section.

5.4 Antibody Optimization Evaluation

The results for the antibody optimization domain are shown in Table 3 and Figure 3. The results from this domain confirm our initial results from the symbolic regression domain. *Sequential* overperforms the *upper bound* baseline, showing that performing transfer learning is better than learning directly in the highest fidelity. However, MF-MDP algorithms perform even better, with *RSEP* overperforming all other algorithms in both the *max* and *avg* metrics. Despite *PGEP_u* underperforming sequential in the *max* metric, it still performs better in *avg*, showing that the result from the multi-fidelity algorithm are more consistent. Those empirical results are very encouraging for RSEP, showing in a very complex and relevant application that it is worthy to explicitly reason over multiple fidelities in symbolic optimization tasks.

Table 3: Best and Average binding score for the antibodies in the hall of fame at the end of the training process.

Alg	*Max*	*Avg*
Upper Bound	8.351	6.311
Sequential	8.634	7.021
RSEP	**8.921**	**7.770**
PGEP_u	8.491	7.201

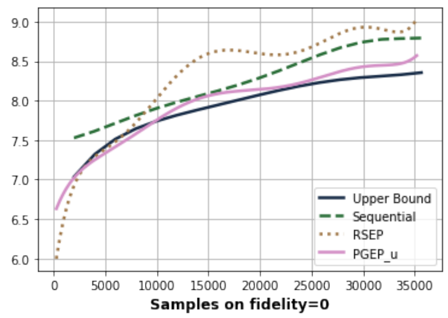

Figure 3: Average of binding score for the best antibody explored in the highest fidelity so far (x-axis is the number of samples from Ξ^0) during training.

6 RELATED WORKS

Although MF-MDPs have not been formally described before, multi-fidelity rewards have already been explored before in the literature [3, 23]. Even though the agent end goal is to optimize performance in the fidelity 0, a group of works propose ways to iteratively fine-tune lower-fidelity surrogate models to make them more realistic and enable training directly in the lower, cheaper to sample from, fidelity. A common way to handle the multiple fidelities is either through modifying lower-fidelity transition [1, 6, 11, 12] or reward [14] functions, by learning a correction factor that approximates them to the highest fidelity function. We, on the other hand, focus on explicitly using both lower and higher-fidelity estimates to learn, instead of fine tuning the lower fidelity models.

The multi-fidelity problem has been explored in an ad hoc manner as a Transfer Learning problem [29, 32], where the lower fidelity is solved, and the solution is somehow reused to learn in the highest fidelity [2]. This approach is mimicked by our *sequential* baseline and, as shown in our experiments, is not as effective and explicitly reasoning over the multiple fidelities during learning.

Some Neural Architecture Search works considered this application as a multi-fidelity problem [37, 44], because each candidate architecture can be evaluated for an arbitrarily number of training iterations, resulting in an as higher fidelity reward as longer you train the model. However, the key distinction from our method is that, in their modeling, for evaluating a sample in a given fidelity, the rewards for all lower fidelities *must* be computed, which is not the case in our modeling and applications.

Perhaps most similar to our paper are the works from Khairy and Balaprakash [15] and Cutler et al. [7]. In the former, they consider that the state space of the low-fidelity environment is an abstracted version of the high-fidelity one (and therefore smaller). We instead assume that the state space is the same and the lower fidelity simply uses a cheaper approximate way of calculating the reward. In the latter, the authors assume that the agent can estimate its epistemic uncertainty and only queries the high fidelity when the uncertainty is low, so as to avoid exploring low-quality samples in the high fidelity. While our method similarly try to bias the evaluations in the highest fidelity towards high-performing samples, we do not require uncertainty calculation, which might be difficult to do.

Outside of the RL community, several works constrain optimization within a trust-region when using lower-fidelity estimates [26]. While those methods are not directly-usable in RL problems, it might inspire TRPO-like [28] methods using our formulation.

7 CONCLUSION AND FURTHER WORK

Although many RL applications are naturally modeled as multi-fidelity problems, the literature has predominantly coped with those environment in an ad-hoc manner. Those problems are either modeled as Transfer Learning problems or as a simulation-optimization problem where the low-fidelity environment is iteratively refined but the learning algorithm is unaware of the multiple fidelities. We propose to explicitly reason over the multiple fidelities and leverage lower fidelity estimates to bias the sampling in the higher, more expensive, fidelity. We contribute the description of Multi-Fidelity MDPs (MF-MDPs), defining a new challenge to the community. We also contribute two families of algorithms for MF-MDPs: *RSEP* and *PGEP*. Moreover, we perform an empirical evaluation in the Symbolic Regression and Antibody Optimization domains, showing that MF-MDP-based algorithms outperform simple strategies in both domains. The conclusion of our experimentation is that *RSEP* is the best performing algorithm overall and should be the first choice, but since *PGEP* was the best performer in some of the symbolic regression benchmarks, it is worthy to also evaluate it in cases where this is feasible. Further work includes explicitly reasoning over the cost of sampling from each fidelity, instead of assuming that samples are free from all lower fidelities as we do in this paper. Another avenue is proposing algorithms that work for a broader class of MF-MDPs, solving more RL applications of interest.

ACKNOWLEDGMENTS

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC. LLNL-CONF-843897.

REFERENCES

[1] Pieter Abbeel, Morgan Quigley, and Andrew Y Ng. 2006. Using inaccurate models in reinforcement learning. In *Proceedings of the 23rd international conference on Machine learning*. 1–8.

[2] Roland Can Aydin, Fabian Albert Braeu, and Christian Johannes Cyron. 2019. General multi-fidelity framework for training artificial neural networks with computational models. *Frontiers in Materials* 6 (2019), 61.

[3] Philip S. Beran, Dean Bryson, Andrew S. Thelen, Matteo Diez, and Andrea Serani. 2020. *Comparison of Multi-Fidelity Approaches for Military Vehicle Design.* https://doi.org/10.2514/6.2020-3158 arXiv:https://arc.aiaa.org/doi/pdf/10.2514/6.2020-3158

[4] Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, Rafal Józefowicz, Scott Gray, Catherine Olsson, Jakub Pachocki, Michael Petrov, Henrique Pondé de Oliveira Pinto, Jonathan Raiman, Tim Salimans, Jeremy Schlatter, Jonas Schneider, Szymon Sidor, Ilya Sutskever, Jie Tang, Filip

Wolski, and Susan Zhang. 2019. Dota 2 with Large Scale Deep Reinforcement Learning. (2019). arXiv:1912.06680 https://arxiv.org/abs/1912.06680

[5] Paul J Carter. 2006. Potent antibody therapeutics by design. *Nature reviews immunology* 6, 5 (2006), 343–357.

[6] Paul Christiano, Zain Shah, Igor Mordatch, Jonas Schneider, Trevor Blackwell, Joshua Tobin, Pieter Abbeel, and Wojciech Zaremba. 2016. Transfer from simulation to real world through learning deep inverse dynamics model. *arXiv preprint arXiv:1610.03518* (2016).

[7] Mark Cutler, Thomas J Walsh, and Jonathan P How. 2014. Reinforcement learning with multi-fidelity simulators. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 3888–3895.

[8] Ruben Glatt and Anna Costa. 2017. Policy reuse in deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 31.

[9] Ruben Glatt, Felipe Leno da Silva, Reinaldo Augusto da Costa Bianchi, and Anna Helena Reali Costa. 2022. A Study on Efficient Reinforcement Learning Through Knowledge Transfer. In *Federated and Transfer Learning*. Springer, 329–356.

[10] Ruben Glatt, Felipe Leno da Silva, BUI VAN HAI, Can Huang, Lingxiao Xue, Mengqi Wang, Fangyuan Chang, Yi Murphey, and Wencong Su. 2022. Deep Symbolic Optimization for Electric Component Sizing in Fixed Topology Power Converters. In *AAAI 2022 Workshop on AI for Design and Manufacturing (ADAM)*.

[11] Florian Golemo, Adrien Ali Taiga, Aaron Courville, and Pierre-Yves Oudeyer. 2018. Sim-to-real transfer with neural-augmented robot simulation. In *Conference on Robot Learning*. PMLR, 817–828.

[12] Josiah P Hanna, Siddharth Desai, Haresh Karnan, Garrett Warnell, and Peter Stone. 2021. Grounded action transformation for sim-to-real reinforcement learning. *Machine Learning* 110, 9 (2021), 2469–2499.

[13] Josiah P Hanna and Peter Stone. 2017. Grounded action transformation for robot learning in simulation. In *Thirty-first AAAI conference on artificial intelligence*.

[14] Luca Iocchi, Fabio D Libera, and Emanuele Menegatti. 2007. Learning humanoid soccer actions interleaving simulated and real data. In *Second Workshop on Humanoid Soccer Robots*.

[15] Sami Khairy and Prasanna Balaprakash. 2022. Multifidelity reinforcement learning with control variates. *arXiv preprint arXiv:2206.05165* (2022).

[16] B Ravi Kiran, Ibrahim Sobh, Victor Talpaert, Patrick Mannion, Ahmad A Al Sallab, Senthil Yogamani, and Patrick Pérez. 2021. Deep reinforcement learning for autonomous driving: A survey. *IEEE Transactions on Intelligent Transportation Systems* (2021).

[17] Jens Kober, J Andrew Bagnell, and Jan Peters. 2013. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research* 32, 11 (2013), 1238–1274.

[18] Jens Kober, J. Andrew Bagnell, and Jan Peters. 2013. Reinforcement Learning in Robotics: A Survey. *Int. J. Rob. Res.* 32, 11 (sep 2013), 1238–1274. https://doi.org/10.1177/0278364913495721

[19] Mikel Landajuela, Brenden K Petersen, Sookyung Kim, Claudio P Santiago, Ruben Glatt, Nathan Mundhenk, Jacob F Pettit, and Daniel Faissol. 2021. Discovering symbolic policies with deep reinforcement learning. In *International Conference on Machine Learning*. PMLR, 5979–5989.

[20] Andrew Leaver-Fay, Michael Tyka, Steven M Lewis, Oliver F Lange, James Thompson, Ron Jacak, Kristian W Kaufman, P Douglas Renfrew, Colin A Smith, Will Sheffler, et al. 2011. ROSETTA3: An object-oriented software suite for the simulation and design of macromolecules. *Methods in enzymology* 487 (2011), 545–574.

[21] Qiang Lu, Jun Ren, and Zhiguang Wang. 2016. Using genetic programming with prior formula knowledge to solve symbolic regression problem. *Computational intelligence and neuroscience* 2016 (2016).

[22] Patrick MacAlpine and Peter Stone. 2018. Overlapping layered learning. *Artificial Intelligence* 254 (2018), 21–43.

[23] Benjamin Peherstorfer, Karen Willcox, and Max Gunzburger. 2018. Survey of multifidelity methods in uncertainty propagation, inference, and optimization. *Siam Review* 60, 3 (2018), 550–591.

[24] Brenden K Petersen, Mikel Landajuela, T Nathan Mundhenk, Claudio P Santiago, Soo K Kim, and Joanne T Kim. 2021. Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients. *Proceeding of the International Conference on Learning Representations (ICLR)* (2021).

[25] Jacob F Pettit, Brenden K Petersen, Chase Cockrell, Dale B Larie, Felipe Leno Silva, Gary An, and Daniel M Faissol. 2021. Learning sparse symbolic policies for sepsis treatment. In *Interpretable ML in Healthcare Workshop at ICML*.

[26] Theresa Robinson, Karen Willcox, Michael Eldred, and Robert Haimes. 2006. Multifidelity optimization for variable-complexity design. In *11th AIAA/ISSMO multidisciplinary analysis and optimization conference*. 7114.

[27] Michael Schmidt and Hod Lipson. 2009. Distilling free-form natural laws from experimental data. *science* 324, 5923 (2009), 81–85.

[28] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. 2015. Trust region policy optimization. In *International conference on machine learning*. PMLR, 1889–1897.

[29] Felipe Leno da Silva and Anna Helena Reali Costa. 2019. A survey on transfer learning for multiagent reinforcement learning systems. *Journal of Artificial Intelligence Research* 64 (2019), 645–703.

[30] Felipe Leno da Silva and Anna Helena Reali Costa. 2021. *Transfer Learning for Multiagent Reinforcement Learning Systems*. Morgan & Claypool Publishers.

[31] Felipe Leno da Silva, Andre Goncalves, Sam Nguyen, Denis Vashchenko, Ruben Glatt, Thomas Desautels, Mikel Landajuela, Brenden Petersen, and Daniel Faissol. 2022. Leveraging Language Models to Efficiently Learn Symbolic Optimization Solutions. In *Adaptive and Learning Agents (ALA) Workshop at AAMAS*.

[32] Felipe Leno da Silva, Garrett Warnell, Anna Helena Reali Costa, and Peter Stone. 2020. Agents Teaching Agents: A Survey on Inter-Agent Transfer Learning. *Autonomous Agents and Multi-Agent Systems* 34 (2020), 1–17.

[33] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. 2017. Mastering the game of go without human knowledge. *nature* 550, 7676 (2017), 354–359.

[34] Edward Snelson and Zoubin Ghahramani. 2006. Sparse Gaussian processes using pseudo-inputs. *Advances in neural information processing systems* 18 (2006), 1257.

[35] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. 1999. Policy gradient methods for reinforcement learning with function approximation. *Advances in neural information processing systems* 12 (1999).

[36] Aviv Tamar, Yonatan Glassner, and Shie Mannor. 2014. Policy gradients beyond expectations: Conditional value-at-risk. *arXiv preprint arXiv:1404.3862* (2014).

[37] Ilya Trofimov, Nikita Klyuchnikov, Mikhail Salnikov, Alexander Filippov, and Evgeny Burnaev. 2020. Multi-fidelity neural architecture search with knowledge distillation. *arXiv preprint arXiv:2006.08341* (2020).

[38] Nguyen Quang Uy, Nguyen Xuan Hoai, Michael O'Neill, Robert I McKay, and Edgar Galván-López. 2011. Semantically-based crossover in genetic programming: application to real-valued symbolic regression. *Genetic Programming and Evolvable Machines* 12, 2 (2011), 91–119.

[39] Denis Vashchenko, Sam Nguyen, Andre Goncalves, Felipe Leno da Silva, Brenden Petersen, Thomas Desautels, and Daniel Faissol. 2022. AbBERT: Learning Antibody Humanness via Masked Language Modeling. *bioRxiv* (2022).

[40] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature* 575, 7782 (2019), 350–354.

[41] David R White, James Mcdermott, Mauro Castelli, Luca Manzoni, Brian W Goldman, Gabriel Kronberger, Wojciech Jaśkowski, Una-May O'Reilly, and Sean Luke. 2013. Better GP benchmarks: community survey results and proposals. *Genetic Programming and Evolvable Machines* 14, 1 (2013), 3–29.

[42] Tai Te Wu and Elvin A. Kabat. 1970. An analysis of the sequences of the variable regions of bence jones proteins and myeloma light chains and their implications for antibody complementarity. *Journal of Experimental Medicine* 132, 2 (1970), 211–250.

[43] Peter R Wurman, Samuel Barrett, Kenta Kawamoto, James MacGlashan, Kaushik Subramanian, Thomas J Walsh, Roberto Capobianco, Alisa Devlic, Franziska Eckert, Florian Fuchs, et al. 2022. Outracing champion Gran Turismo drivers with deep reinforcement learning. *Nature* 602, 7896 (2022), 223–228.

[44] Shangshang Yang, Ye Tian, Xiaoshu Xiang, Shichen Peng, and Xingyi Zhang. 2022. Accelerating Evolutionary Neural Architecture Search via Multi-Fidelity Evaluation. *IEEE Transactions on Cognitive and Developmental Systems* (2022).

A PROOFS

Proposition 1. *Let R^0 and R^1 be random variables related by noise N: $R^1 := R^0 + N$. Let Q^0_ϵ and Q^1_ϵ be the $(1-\epsilon)$-quantiles of the distributions of R^0 and R^1, respectively. Then we have that*

$$P(R^0 \geq Q^0_\epsilon, R^1 \leq Q^1_\epsilon) = \epsilon \mathbb{E}\left[F_N(Q^1_\epsilon - R^0) \mid R^0 \geq Q^0_\epsilon\right] \tag{9}$$

where $F_N(r)$ is the CDF of the noise distribution.

Proof.

$$\begin{aligned}
&P(R^0 \geq Q^0_\epsilon \wedge R^1 \leq Q^1_\epsilon) && (13)\\
&= P(R^0 \geq Q^0_\epsilon \wedge R^0 + N \leq Q^1_\epsilon) && (14)\\
&= \int_{r \geq Q^0_\epsilon} P(R^0 = r, N \leq Q^1_\epsilon - r)dr && (15)\\
&= \int_{r \geq Q^0_\epsilon} P(R^0 = r)P(N \leq Q^1_\epsilon - r)dr && (16)\\
&= \int_{r \geq Q^0_\epsilon} P(R^0 = r)F_N(Q^1_\epsilon - r)dr && (17)\\
&= \epsilon \int_{r \geq Q^0_\epsilon} \frac{P(R^0 = r)}{P(R^0 \geq Q^0_\epsilon)} F_N(Q^1_\epsilon - r)dr && (18)\\
&= \epsilon \int_{r} \frac{P(R^0 = r, R^0 \geq Q^0_\epsilon)}{P(R^0 \geq Q^0_\epsilon)} F_N(Q^1_\epsilon - r)dr && (19)\\
&= \epsilon \mathbb{E}\left[F_N(Q^1_\epsilon - R^0) \mid R^0 \geq Q^0_\epsilon\right] && (20)
\end{aligned}$$

□

Proposition 3. *Let random variable τ have distribution π_θ, and let R^0 and R^m be two functions of τ with induced distributions p^0 and p^m. Let F^m_θ denote the CDF of p^m. Let $Q^m_\epsilon(\theta) = \inf_{\tau \in \Omega}\{R^m(\tau) \colon F^m_\theta(r) \geq 1-\epsilon\}$ denote the $(1-\epsilon)$-quantile of p^m. The gradient of*

$$J_\epsilon(\theta) := \mathbb{E}_\theta\left[R^0(\tau) \mid R^m(\tau) \geq Q^m_\epsilon(\theta)\right] \tag{21}$$

is given by

$$\nabla_\theta J_\epsilon(\theta) = \mathbb{E}_\theta\left[\nabla_\theta \log \pi_\theta(\tau)(R^0(\tau) - R^0(\tau_\epsilon)) \mid R^m(\tau) \geq Q^m_\epsilon(\theta)\right] \tag{22}$$

where $\tau_\epsilon = \arg\inf\{R^m(\tau) \colon F^m_\theta(r) \geq 1-\epsilon\}$ is the sample that attains the quantile.

Proof. First, we provide an elementary proof for the case where τ is a scalar random variable, then we provide a proof for the multi-dimensional case.

Single-dimensional case. Define the set of samples for which the mixture reward exceeds the $1-\epsilon$ quantile:

$$D_\theta := \{\tau \in \Omega \colon R^m(\tau) \geq Q^m_\epsilon(\theta)\} \tag{23}$$

We expand the definition of the objective:

$$\begin{aligned}
J_\epsilon(\theta) &= \int_\Omega R^0(\tau) f_{\theta, R^m(\tau) \geq Q^m_\epsilon(\theta)}(\tau)d\tau && (24)\\
&= \int_\Omega R^0(\tau) \frac{f_\theta(\tau, R^m(\tau) \geq Q^m_\epsilon(\theta))}{f_\theta(R^m(\tau) \geq Q^m_\epsilon(\theta)} d\tau && (25)\\
&= \frac{1}{\epsilon}\int_\Omega R^0(\tau) f_\theta(\tau, R^m(\tau) \geq Q^m_\epsilon(\theta))d\tau && (26)\\
&= \frac{1}{\epsilon}\int_{\tau \in D_\theta} R^0(\tau)\pi_\theta(\tau)d\tau && (27)
\end{aligned}$$

Assuming sufficient continuity of the reward, policy, and quantile as a function of parameter θ, we can apply the Leibniz integral rule to differentiate under the integral sign. Differentiating both sides of

$$\epsilon = \int_{\tau \in D_\theta} \pi_\theta(\tau)d\tau, \tag{28}$$

we have

$$\begin{aligned}
0 &= \nabla_\theta \int_{\tau \in D_\theta} \pi_\theta(\tau)d\tau && (29)\\
&= \nabla_\theta \int_{R^m(\tau_\epsilon(\theta))}^b p^m_\theta(r)dr && (30)\\
&= -p^m_\theta(R^m(\tau_\epsilon))\nabla_\theta R^m(\tau_\epsilon(\theta)) + \int_{R^m(\tau_\epsilon(\theta))}^b \nabla_\theta p^m_\theta(r)dr && (31)
\end{aligned}$$

Let τ_r denote the sample that satisfies $R^m(\tau) = r$. Applying the Leibniz integral rule to the objective, we have

$$\begin{aligned}
\nabla_\theta J_\epsilon(\theta) &= \nabla_\theta \frac{1}{\epsilon}\int_{R^m(\tau_\epsilon(\theta))}^b R^0(\tau_r)p^m_\theta(r)dr && (32)\\
&= -\frac{1}{\epsilon}R^0(\tau_\epsilon(\theta))p^m_\theta(R^m(\tau_\epsilon(\theta)))\nabla_\theta R^m(\tau_\epsilon(\theta)) && (33)\\
&\quad + \frac{1}{\epsilon}\int_{R^m(\tau_\epsilon(\theta))}^b R^0(\tau_r)\nabla_\theta p^m_\theta(r)dr && (34)
\end{aligned}$$

Substituting eq. (31) into eq. (33), we get

$$\begin{aligned}
\nabla_\theta J_\epsilon(\theta) &= -\frac{1}{\epsilon}R^0(\tau_\epsilon(\theta))\int_{R^m(\tau_\epsilon(\theta))}^b \nabla_\theta p^m_\theta(r)dr && (35)\\
&\quad + \frac{1}{\epsilon}\int_{R^m(\tau_\epsilon(\theta))}^b R^0(\tau_r)\nabla_\theta p^m_\theta(r)dr && (36)\\
&= \frac{1}{\epsilon}\int_{R^m(\tau_\epsilon(\theta))}^b \nabla_\theta p^m_\theta(r)\left(R^0(\tau_r) - R^0(\tau_\epsilon(\theta))\right)dr && (37)\\
&= \frac{1}{\epsilon}\int_{\tau \in D_\theta} \nabla_\theta \pi_\theta(\tau)\left(R^0(\tau_r) - R^0(\tau_\epsilon(\theta))\right)d\tau && (38)\\
&= \mathbb{E}_{\pi_\theta}\left[\nabla_\theta \log \pi_\theta(\tau)(R^0(\tau) - R^0(\tau_\epsilon(\theta))) \mid R^m(\tau) \geq Q^m_\epsilon(\theta)\right] && (39)
\end{aligned}$$

The second-to-last step implicitly uses the change-of-variables formula $p^m_\theta(r) = \pi_\theta(f(r))|\det Df(r)|$ where $f \colon R \mapsto \Omega$ is the inverse function that maps rewards to τ. But since the determinant of Jacobian does not depend on θ, the result holds.

Multi-dimensional case. For the case where τ is an n-dimensional random variable, we adapt the proof of Tamar et al. [36, Proposition 2], except for two differences: 1) the reward R^0 being optimized is different from the reward R^m used in the conditional expectation; 2) we condition on the outcomes within the top ϵ quantile, i.e. $R^m(\tau) \geq Q^m_\epsilon(\theta)$, rather than the outcomes below the ϵ-Value-at-Risk which would be $R^m(\tau) \geq Q^m_\epsilon(\theta)$. We use the same assumptions Tamar et al. [36, Assumptions 4 and 5].

Define the set $D_\theta := \{\tau \colon R^m(\tau) \geq Q^m_\epsilon(\theta)\}$. Let $\omega := \pi_\theta(\tau)R^0(\tau)d\tau$ and $\tilde{\omega} := \pi_\theta(\tau)d\tau$.

For every $\tau \in \partial D^i_\theta$, we have either (a) $R^0(\tau) = Q^m_\epsilon(\theta)$ or (b) $R^0(\tau) > Q^m_\epsilon(\theta)$. Let $\partial D^{i,a}_\theta$ and $\partial D^{i,b}_\theta$ be the subset of τ corresponding to cases (a) and (b), respectively. By the same reasoning in

Tamar et al. [36], we have

$$\int_{\partial D_\theta^{i,b}} \mathbf{v} \lrcorner \omega = 0 \,. \tag{40}$$

By definition of D_θ, we have

$$\epsilon = \int_{D_\theta} \tilde{\omega} \,. \tag{41}$$

Taking the derivative, and using eq. (40), we have

$$0 = \sum_{i=1}^{L_\theta} \left(\int_{\partial D_\theta^{i,a}} \mathbf{v} \lrcorner \tilde{\omega} + \int_{D_\theta^i} \frac{\partial \tilde{\omega}}{\partial \theta} \right) . \tag{42}$$

In the boundary case $\tau \in \partial D_\theta^{i,a}$, τ satisfies $R^m(\tau) = Q_\epsilon^m(\theta)$, so we can denote it by τ_ϵ as defined above. By definition of ω and linearity of the interior product, we have

$$\int_{\partial D_\theta^{i,a}} \mathbf{v} \lrcorner \omega = R^0(\tau_\epsilon)(\theta) \int_{\partial D_\theta^{i,a}} \mathbf{v} \lrcorner \tilde{\omega} \,. \tag{43}$$

Plugging eq. (42) into eq. (43), we get

$$\sum_{i=1}^{L_\theta} \int_{\partial D_\theta^{i,a}} \mathbf{v} \lrcorner \omega = -R^0(\tau_\epsilon) \sum_{i=1}^{L_\theta} \int_{D_\theta^i} \frac{\partial \tilde{\omega}}{\partial \theta} \,. \tag{44}$$

Our objective eq. (21) can be written as

$$\begin{aligned}
J_\epsilon(\theta) &= \mathbb{E}_\theta \left[R^0(\tau) \mid R^m(\tau) \geq Q_\epsilon^m(\theta) \right] && (45)\\
&= \int_{\tau \in \Omega} R^0(\tau) \pi_{\tau | R^m(\tau) \geq Q_\epsilon^m(\theta)}(\tau) d\tau && (46)\\
&= \frac{1}{\epsilon} \int_{\tau \in \Omega} R^0(\tau) \pi_\theta(\tau, R^m(\tau) \geq Q_\epsilon^m(\theta)) d\tau && (47)\\
&= \frac{1}{\epsilon} \int_{D_\theta} \pi_\theta(\tau) R^0(\tau) d\tau && (48)\\
&= \frac{1}{\epsilon} \sum_{i=1}^{L_\theta} \int_{D_\theta^i} \pi_\theta(\tau) R^0(\tau) d\tau \,. && (49)
\end{aligned}$$

Its gradient is

$$\nabla_\theta J_\epsilon(\theta) = \frac{1}{\epsilon} \sum_{i=1}^{L_\theta} \nabla_\theta \int_{D_\theta^i} \pi_\theta(\tau) R^0(\tau) d\tau \,. \tag{50}$$

By the Leibniz rule, we have

$$\begin{aligned}
\nabla_\theta \int_{D_\theta^i} \pi_\theta(\tau) R^0(\tau) d\tau &= \int_{\partial D_\theta^i} \mathbf{v} \lrcorner \omega + \int_{D_\theta^i} \frac{\partial \omega}{\partial \theta} && (51)\\
&= \int_{\partial D_\theta^{i,a}} \mathbf{v} \lrcorner \omega + \int_{D_\theta^i} \frac{\partial \omega}{\partial \theta} \,, && (52)
\end{aligned}$$

where the last equality follows from eq. (40). Using eq. (44) and eq. (51) in eq. (50), we get

$$\begin{aligned}
\nabla_\theta J_\epsilon(\theta) &= \frac{1}{\epsilon} \sum_{i=1}^{L_\theta} \left(\int_{D_\theta^i} \frac{\partial \omega}{\partial \theta} - R^0(\tau_\epsilon) \int_{D_\theta^i} \frac{\partial \tilde{\omega}}{\partial \theta} \right) && (53)\\
&= \frac{1}{\epsilon} \int_{D_\theta} \nabla_\theta \pi_\theta(\tau) \left(R^0(\tau) - R^0(\tau_\epsilon) \right) d\tau && (54)\\
&= \mathbb{E}_{\pi_\theta} \left[\nabla_\theta \log \pi_\theta(\tau) \left(R^0(\tau) - R^0(\tau_\epsilon) \right) \mid R^m(\tau) \geq Q_\epsilon^m(\theta) \right] && (55)
\end{aligned}$$

□