DisCo-DSO: Coupling Discrete and Continuous Optimization for Efficient Generative Design in Hybrid Spaces

Jacob F. Pettit, Chak Shing Lee, Jiachen Yang, Alex Ho, Daniel Faissol, Brenden Petersen, Mikel Landajuela*

Computational Engineering Division
Lawrence Livermore National Laboratory
Livermore, CA, 94550, USA

Abstract

We consider the challenge of black-box optimization within hybrid discrete-continuous and variable-length spaces, a problem that arises in various applications, such as decision tree learning and symbolic regression. We propose DisCo-DSO (Discrete-Continuous Deep Symbolic Optimization), a novel approach that uses a generative model to learn a joint distribution over discrete and continuous design variables to sample new hybrid designs. In contrast to standard decoupled approaches, in which the discrete and continuous variables are optimized separately, our joint optimization approach uses fewer objective function evaluations, is robust against non-differentiable objectives, and learns from prior samples to guide the search, leading to significant improvement in performance and sample efficiency. Our experiments on a diverse set of optimization tasks demonstrate that the advantages of DisCo-DSO become increasingly evident as the complexity of the problem increases. In particular, we illustrate DisCo-DSO's superiority over the state-of-the-art methods for interpretable reinforcement learning with decision trees.

Introduction

Deep learning methods have shown success in important combinatorial optimization problems (Bello et al. 2016), including generating interpretable policies for continuous control (Landajuela et al. 2021a) and symbolic regression (SR) to discover the underlying mathematical equations from the data (Petersen et al. 2021; Biggio et al. 2021; Kamienny et al. 2022; Landajuela et al. 2022). Existing approaches train a generative model that constructs a solution to the optimization problem by sequentially choosing from a set of discrete tokens, using the value of the objective function as the terminal reward for learning. However, these approaches do not jointly optimize the discrete and continuous components of such hybrid problems: Certain discrete tokens require the additional specification of an associated real-valued parameter, such as the threshold value at a decision tree node or the value of a constant token in an equation, but the learned generative model does not produce these values. Instead, they adopt the design choice of *decoupled optimization*, whereby only the construction of a discrete solution skeleton is optimized by deep learning, while the associated continuous parameters are left to a separate black-box optimizer.

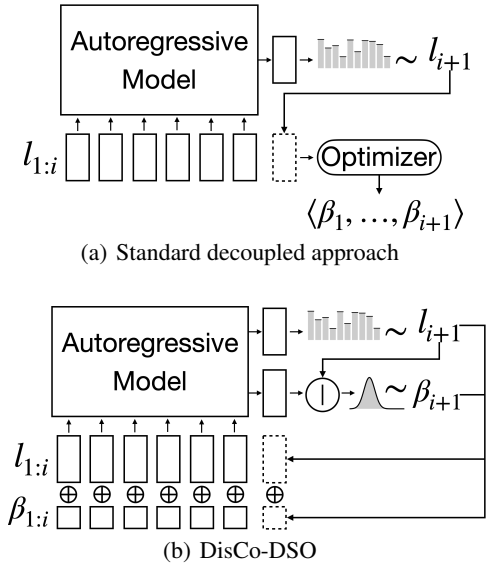


Figure 1: Comparison of the standard decoupled approach and DisCo-DSO for discrete-continuous optimization using an autoregressive model. In the decoupled approach, the discrete skeleton $\tau_{\mathrm{d}} = \langle (l_1, \cdot), \ldots, (l_T, \cdot) \rangle$ is sampled first and then the continuous parameters $\beta_1, \ldots, \beta_T$ are optimized independently. In contrast, DisCo-DSO models the joint distribution over the sequence of tokens $\langle (l_1, \beta_1), \ldots, (l_T, \beta_T) \rangle$. Here, the notation $\oplus$ stands for concatenation of vectors.

We hypothesize that a joint discrete-continuous optimization approach (Figure 1(b)) that generates a complete solution based on deep reinforcement learning (RL) (Sutton and Barto 2018) has significant advantages compared to existing decoupled approaches that employ learning only for the discrete skeleton (Figure 1(a)). In terms of efficiency, a joint approach only requires one evaluation of the objective function for each candidate solution, whereas the decoupled approach based on common non-linear black-box optimization methods such as BFGS (Fletcher 2000), simulated annealing (Xi-

*Corresponding author: landajuelala1@llnl.gov.

ang et al. 1997), or evolutionary algorithms (Storn and Price 1997) requires a significant number of function evaluations to optimize each discrete skeleton. This decoupled approach incurs a high cost for applications such as interpretable control, where each objective function evaluation involves running the candidate solution on many episodes of a high-dimensional and stochastic physical simulation (Landajuela et al. 2021a). Furthermore, joint exploration and learning on the full discrete-continuous solution space has the potential to escape from local optima and use information from prior samples to guide the subsequent search.

In this work, we consider discrete-continuous optimization problems that exhibit several key distinguishing features: (1) a black-box reward, (2) a variable-length structure of the design space, and (3) a sequential structure in the form of *prefix-dependent positional constraints*. These problems are not well-suited to existing joint optimization approaches such as Mixed Integer Programming (MIP) (Fischetti and Jo 2018; Nair et al. 2021) or Mixed Bayesian Optimization (BO) (Daxberger et al. 2019), which are designed for problems with fixed-length discrete components and do not naturally handle positional constraints in the design space. To address these challenges, we draw upon the success of deep reinforcement learning in parameterized action space Markov decision processes (Hausknecht and Stone 2016) to extend existing deep learning methods for discrete optimization (Bello et al. 2016; Zoph and Le 2017; Petersen et al. 2021; Landajuela et al. 2021a) to the broader space of joint discrete-continuous optimization. We summarize the main contributions of this paper as follows:

- We propose a novel method for joint discrete-continuous optimization using autoregressive models and deep reinforcement learning, which we call DisCo-DSO, that is suited for black-box hybrid optimization problems over variable-length search spaces with prefix-dependent positional constraints.
- We present a novel formulation for decision tree policy search in control tasks as sequential discrete-continuous optimization and propose a method for sequentially finding bounds for parameter ranges in decision nodes.
- We perform exhaustive empirical evaluation of DisCo-DSO on a diverse set of tasks, including interpretable control policies and symbolic regression. We show that DisCo-DSO outperforms decoupled approaches on all tasks.

Related Work

Hybrid discrete-continuous action spaces in reinforcement learning. The treatment of the continuous parameters as part of the action space has strong parallels in the space of hybrid discrete-continuous RL. In Hausknecht and Stone (2016), the authors present a successful application of deep reinforcement learning to a domain with continuous state and action spaces. In Xiong et al. (2018), the authors take an off-policy DQN-type approach that directly works on the hybrid action space without approximation of the continuous part or relaxation of the discrete part, but requires an extra loss function for the continuous actions. In Neunert et al. (2020), they propose a hybrid RL algorithm that uses continuous policies for discrete action selection and discrete policies for continuous action selection.

Symbolic regression with constants optimization. In the field of symbolic regression, different approaches have been proposed for addressing the optimization of both discrete skeletons and continuous parameters. Traditional genetic programming approaches and deep generative models handle these problems separately, with continuous constants optimized after discrete parameters (Topchy, Punch et al. 2001; Petersen et al. 2021; Biggio et al. 2021). Recent works aim to jointly optimize discrete constants and continuous parameters by relaxing the discrete problem into a continuous one (Martius and Lampert 2016; Sahoo, Lampert, and Martius 2018), or by tokenizing (i.e., discretizing) the continuous constants (Kamienny et al. 2022). The former approach faces challenges such as exploding gradients and the need to revert continuous values to discrete ones. The latter approach tokenizes continuous constants, treating them similarly to discrete tokens, but such quantization is problem-dependent, restricts the search space, and requires additional post-hoc optimization to refine the continuous parameters.

Decision tree policies in reinforcement learning. In the domain of symbolic reinforcement learning, where the goal is to find intelligible and concise control policies, works such as Landajuela et al. (2021a) and Sahoo, Lampert, and Martius (2018) have discretized the continuous space and used relaxation approaches, respectively, to optimize symbolic control policies in continuous action spaces. For discrete action spaces, a natural representation of a symbolic policy is a decision tree (Ding et al. 2020; Silva et al. 2020; Custode and Iacca 2023). In Custode and Iacca (2023), the authors use an evolutionary search to find the best decision tree policy and further optimized the real valued thresholds using a decoupled approach. Relaxation approaches find their counterparts within this domain in works such as Sahoo, Lampert, and Martius (2018); Silva et al. (2020); Ding et al. (2020), where a soft decision tree is used to represent the policy. The soft decision tree, which fixes the discrete structure of the policy and exposes the continuous parameters, is then optimized using gradient-based methods.

Discrete-Continuous Deep Symbolic Optimization

Notation and Problem Definition

We consider a discrete-continuous optimization problem defined over a search space $\mathcal{T}$ of sequences of tokens $\tau = \langle \tau_1, \ldots, \tau_T \rangle$, where each token τ_i belongs to a library $\mathcal{L}$ and the length T of the sequence is not fixed *a priori*. The library $\mathcal{L}$ is a set of K tokens $\mathcal{L} = \{l_1, \ldots, l_K\}$, where a subset $\hat{\mathcal{L}} \subseteq \mathcal{L}$ of them are parametrized by a continuous parameter, i.e., each token $l \in \hat{\mathcal{L}}$ has an associated continuous parameter $\beta \in \mathcal{A}(l) \subset \mathbb{R}$, where $\mathcal{A}(l)$ is the token-dependent range. To ease the notation, we define $\bar{\mathcal{L}} \stackrel{\text{def}}{=} \mathcal{L} \setminus \hat{\mathcal{L}}$ and consider a dummy range $\mathcal{A}(l) = [0, 1] \subset \mathbb{R}$ for the strictly discrete

tokens $l \in \bar{\mathcal{L}}$. Thus, we define

$$l(\beta) = \begin{cases} l & \text{if } l \in \bar{\mathcal{L}} \\ l(\beta) & \text{if } l \in \hat{\mathcal{L}} \end{cases}, \forall (l, \beta) \in \mathcal{L} \times \mathcal{A}(l).$$

In other words, the parameter β is ignored if $l \in \bar{\mathcal{L}}$. With this notation, we can write $\tau_i = l_i(\beta_i) \in \mathcal{L}, \forall i \in \{1, \dots, T\}$. In the following, we use the notation $l_i(\beta_i) \equiv (l_i, \beta_i)$ and write $\tau = \langle \tau_1, \dots, \tau_T \rangle = \langle l_1(\beta_1), \dots, l_T(\beta_T) \rangle \equiv \langle (l_1, \beta_1), \dots, (l_T, \beta_T) \rangle$.

Given a sequence τ, we define the *discrete skeleton* τ_d as the sequence obtained by removing the continuous parameters from τ, i.e., $\tau_\text{d} = \langle (l_1, \cdot), \dots, (l_T, \cdot) \rangle$. We introduce the operator $\texttt{eval} : \mathcal{T} \to \mathbb{T}$ to represent the semantic interpretation of the sequence τ as an object in the relevant design space $\mathbb{T}$. We consider problems with *prefix-dependent positional constraints*, i.e., problems for which, given a prefix $\tau_{1:(i-1)}$, there exists a possible non-empty set of unfeasible tokens $\mathcal{C}_{\tau_{1:(i-1)}} \subseteq \mathcal{L}$ such that $\texttt{eval}(\tau_{1:(i-1)} \cup \tau_i \cup \tau_{(i+1):T}) \notin \mathbb{T}$ for all $\tau_i \in \mathcal{C}_{\tau_{1:(i-1)}}$ and for all $\tau_j \in \mathcal{L}$ with $i < j \leq T$. Variable-length problems exhibiting such constraints are not well-suited for MIP solvers or classical Bayesian Optimization methods.

The optimization problem is defined by the reward function $R : \mathbb{T} \to \mathbb{R}$, which can be deterministic or stochastic. In the stochastic case, we have a reward distribution $p_R(r|t)$ conditioned on the design $t \in \mathbb{T}$ and the reward function is given by $R(t) = \mathbb{E}_{r \sim p_R(r|t)}[r]$. Note that we do not assume that the reward function R is differentiable with respect to the continuous parameters β_i. In the following, we make a slight abuse of notation and use $R(\tau)$ and $p_R(r|\tau)$ to denote $R(\texttt{eval}(\tau))$ and $p_R(r|\texttt{eval}(\tau))$, respectively. The optimization problem is to find a sequence $\tau^* = \langle \tau_1^*, \dots, \tau_T^* \rangle = \langle (l_1^*, \beta_1^*), \dots, (l_T^*, \beta_T^*) \rangle$ (where the length T is not fixed a priori) such that $\tau^* \in \arg\max_{\tau \in \mathcal{T}} R(\tau)$.

Method

Combinatorial optimization with autoregressive models. In applications of deep learning to combinatorial optimization (Bello et al. 2016), a probabilistic model $p(\tau)$ is learned over the design space $\mathcal{T}$. The model is trained to gradually allocate more probability mass to high scoring solutions. The training can be done using supervised learning, if problem instances with their corresponding solutions are available, or, more generally, using RL. In most cases, the model $p(\tau)$ is parameterized by an autoregressive (AR) model with parameters θ. The model is used to generate sequences as follows.

At position i, the model emits a vector of logits $\psi^{(i)}$ conditioned on the previously generated tokens $\tau_{1:(i-1)}$, i.e., $\psi^{(i)} = \text{AR}(\tau_{1:(i-1)}; \theta)$. The new token τ_i is sampled from the distribution $p(\tau_i | \tau_{1:(i-1)}, \theta) = \text{softmax}(\psi^{(i)})_{\mathcal{L}(\tau_i)}$, where $\mathcal{L}(\tau_i)$ is the index in $\mathcal{L}$ corresponding to node value τ_i. The new token τ_i is then added to the sequence $\tau_{1:(i-1)}$ and used to condition the generation of the next token τ_{i+1}. The process continues until a stopping criterion is met.

Different model architectures can be employed to generate the logits $\psi^{(i)}$. For instance, recurrent neural networks (RNNs) have been utilized in Petersen et al. (2021); Landajuela et al. (2021a); Mundhenk et al. (2021); da Silva et al. (2023), and transformers with causal attention have been applied in works like Biggio et al. (2021) and Kamienny et al. (2022).

Prefix-dependent positional constraints. Sequential token generation enables flexible configurations and the incorporation of constraints during the search process (Petersen et al. 2021). Specifically, given a prefix $\tau_{1:(i-1)}$, a prior $\psi_\circ^{(i)} \in \mathbb{R}^{|\mathcal{L}|}$ is computed such that ${\psi_\circ^{(i)}}_{\mathcal{L}(\tau_i)} = -\infty$ for tokens τ_i in the unfeasible set $\mathcal{C}_{\tau_{1:(i-1)}}$ and zero otherwise. The prior is added to the logits $\psi^{(i)}$ before sampling the token τ_i.

Extension to discrete-continuous optimization. Current deep learning approaches for combinatorial optimization only support discrete tokens, i.e., $\hat{\mathcal{L}} = \emptyset$, (Bello et al. 2016) or completely decouple the discrete and continuous parts of the problem, as in Petersen et al. (2021); Landajuela et al. (2021a); Mundhenk et al. (2021); da Silva et al. (2023), by sampling first the discrete skeleton τ_d and then optimizing its continuous parameters separately (see Figure 1(a)). In this work, we extend these frameworks to support *joint optimization of discrete and continuous tokens*. The model is extended to emit two outputs $\psi^{(i)}$ and $\phi^{(i)}$ for each token $\tau_i = (l_i, \beta_i)$ conditioned on the previously generated tokens, i.e., $\left(\psi^{(i)}, \phi^{(i)}\right) = \text{AR}((l, \beta)_{1:(i-1)}; \theta)$, where we use the notation $(l, \beta)_{1:(i-1)}$ to denote the sequence of tokens $\langle (l_1, \beta_1), \dots, (l_{i-1}, \beta_{i-1}) \rangle$ (see Figure 1(b)). Given tokens $(l, \beta)_{1:(i-1)}$, the i^th token (l_i, β_i) is generated by sampling from the following distribution:

$$p((l_i, \beta_i) | (l, \beta)_{1:(i-1)}, \theta) = \begin{cases} \mathcal{U}_{[0,1]}(\beta_i) \text{softmax}(\psi^{(i)})_{\mathcal{L}(l_i)} & \text{if } l_i \in \bar{\mathcal{L}} \\ \mathcal{D}(\beta_i | l_i, \phi^{(i)}) \text{softmax}(\psi^{(i)})_{\mathcal{L}(l_i)} & \text{if } l_i \in \hat{\mathcal{L}} \end{cases},$$

where $\mathcal{D}(\beta_i | l_i, \phi^{(i)})$ is the probability density function of the distribution $\mathcal{D}$ that is used to sample β_i from $\phi^{(i)}$. Note that the choice of β_i is conditioned on the choice of discrete token l_i. We assume that the support of $\mathcal{D}(\beta | l, \phi)$ is a subset of $\mathcal{A}(l)$ for all $l \in \hat{\mathcal{L}}$. Additional priors of the form $(\psi_\circ^{(i)}, 0)$ can be added to the logits before sampling the token τ_i.

Training DisCo-DSO. The parameters θ of the model are learned by maximizing the expected reward $J(\theta) = \mathbb{E}_{\tau \sim p(\tau|\theta)}[R(\tau)]$ or, alternatively, the quantile-conditioned expected reward $J_\varepsilon(\theta) = \mathbb{E}_{\tau \sim p(\tau|\theta)}[R(\tau) | R(\tau) \geq R_\varepsilon(\theta)]$, where $R_\varepsilon(\theta)$ represents the $(1-\varepsilon)$-quantile of the reward distribution $R(\tau)$ sampled from the trajectory distribution $p(\tau|\theta)$. The motivation for using $J_\varepsilon(\theta)$ is to encourage the model to focus on *best case* performance over *average case* performance (see Petersen et al. (2021)), which is the preferred behavior in optimization problems. It is worth noting that both objectives, $J(\theta)$ and $J_\varepsilon(\theta)$, serve as *relaxations* of the original $\arg\max R(\tau)$ optimization problem described above.

To optimize the objective $J_\varepsilon(\theta)$, we extend the risk-seeking policy gradient of Petersen et al. (2021) to the discrete-continuous setting. The gradient of $J_\varepsilon(\theta)$ reads as

$$\nabla_\theta J_\varepsilon(\theta) = \mathbb{E}_{\tau \sim p(\tau|\theta)} \left[A(\tau, \varepsilon, \theta) \, S((l, \beta)_{1:T}) \mid A(\tau, \varepsilon, \theta) > 0 \right],$$

where $A(\tau, \varepsilon, \theta) = R(\tau) - R_\varepsilon(\theta)$ and

$$S((l,\beta)_{1:T}) = \sum_{i=1}^{T} \begin{cases} \nabla_\theta \log p(l_i|(l,\beta)_{1:(i-1)},\theta) & \text{if } l_i \in \bar{\mathcal{L}}, \\ \nabla_\theta \log p(l_i|(l,\beta)_{1:(i-1)},\theta) & \\ \quad + \nabla_\theta \log p(\beta_i|l_{1:i},\beta_{1:i-1},\theta) & \text{if } l_i \in \hat{\mathcal{L}}. \end{cases}$$

We provide pseudocode for DisCo-DSO, a derivation of the risk-seeking policy gradient, and additional details of the learning procedure in the appendix.

Experiments

We demonstrate the benefits and generality of our approach on a diverse set of tasks as follows. Firstly, we introduce a new pedagogical task, called *Parameterized Bitstring*, to understand the conditions under which the benefits of DisCo-DSO versus decoupled approaches become apparent. We then consider two preeminent tasks in combinatorial optimization: decision tree policy optimization for reinforcement learning and symbolic regression for equation discovery.

Baselines. To demonstrate the advantages of joint discrete-continuous optimization, we compare DisCo-DSO with the following classes of methods:

- **Decoupled-RL-{BFGS, anneal, evo}**: This baseline trains a generative model with reinforcement learning to produce a discrete skeleton (Petersen et al. 2021), which is then optimized by a downstream nonlinear solver for the continuous parameters. The objective value at the optimized solution is the reward, which is used to update the generative model using the same policy gradient approach and architecture as DisCo-DSO. The continuous optimizer is either L-BFGS-B (BFGS), simulated annealing (anneal) (Xiang et al. 1997), or differential evolution (evo) (Storn and Price 1997), using the SciPy implementation (Virtanen et al. 2020).
- **Decoupled-GP-{BFGS, anneal, evo}**: This baseline uses genetic programming (GP) (Koza 1990) to produce a discrete skeleton, which is then optimized by a downstream nonlinear solver for the continuous parameters.
- **BO**: For the Parameterized Bitstring task, which has a fixed length search space and no positional constraints, we also consider a Bayesian Optimization baseline using expected improvement as acquisition function (Shahriari et al. 2015; Garrido-Merchán and Hernández-Lobato 2020).

All experiments involving RL and DisCo-DSO use a RNN with a single hidden layer of 32 units as the generative model. The GP baselines use the "Distributed Evolutionary Algorithms in Python" software[1] (Fortin et al. 2012). Additional details are provided in the appendix.

Note on baselines for symbolic regression. In the context of symbolic regression, some of the above baselines corresponds to popular methods in the literature. Specifically, **Decoupled-RL-BFGS** corresponds exactly to the method "Deep Symbolic Regression" from Petersen et al. (2021), and **Decoupled-GP-BFGS** corresponds to a standard implementation of genetic programming for symbolic regression *à la* Koza (1994) (most common approach to symbolic regression in the literature).

[1] https://github.com/DEAP/deap. LGPL-3.0 license.

Parameterized Bitstring Task

Problem formulation. We design a general and flexible *Parameterized Bitstring* benchmark problem, denoted PB(N, f, l^*, β^*), to test the hypothesis that DisCo-DSO is more efficient than the decoupled optimization approach. In each problem instance, the task is to recover a hidden string $l^* \in [0,1]^T$ of T bits and a vector of parameters $\beta^* \in \mathbb{R}^T$. Each bit l_i^* is paired with a parameter β_i^* via the reward function R, which gives a positive value based on an objective function $f(\beta_i, \beta_i^*) \in [0,1]$ only if the correct bit l_i^* is chosen at position i:

$$R(\tau, \beta) \overset{\text{def}}{=} \frac{1}{T} \sum_{i=1}^{T} \mathbf{1}_{\tau_i = \tau_i^*} \left(\alpha + (1-\alpha) f(\beta_i, \beta_i^*)\right) \quad (1)$$

The scalar $\alpha \in [0,1]$ controls the relative importance of expending computational effort to optimize the discrete or continuous parts of the reward. The problem difficulty can be controlled by increasing the length T and increasing the nonlinearity of the objective function f, such as by increasing the number of local optima. In our experiment, we tested the following objective functions, which represent objectives with multiple suboptimal local maxima (f_1) and discontinuous objective landscapes (f_2):

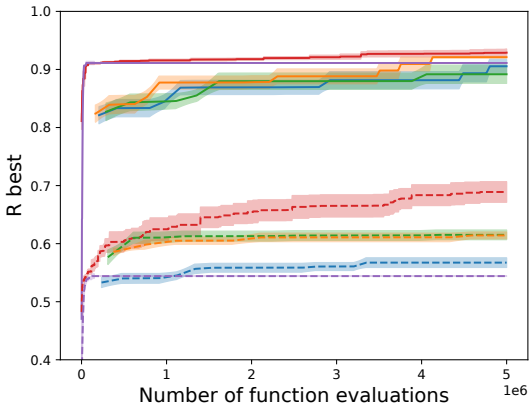
$$f_1(x, x^*) \overset{\text{def}}{=} \left| \frac{\sin(50(x - x^*))}{50(x - x^*)} \right|, \quad (2)$$

$$f_2(x, x^*) \overset{\text{def}}{=} \begin{cases} 1, & |x - x^*| \leq 0.05 \\ 0.5, & 0.05 < |x - x^*| \leq 0.1 \\ 0, & 0.1 < |x - x^*| \end{cases}. \quad (3)$$

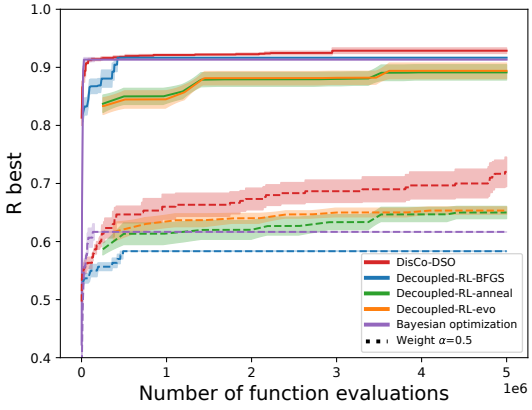
Results. Figure 2 shows that DisCo-DSO is significantly more sample efficient than the decoupled approach when the discrete solution contributes more to the overall reward. This is because each sample generated by DisCo-DSO is a complete solution, which costs only one function evaluation to get a reward. In contrast, each sample generated by the baseline decoupled methods only has a discrete skeleton, which requires many function evaluations using the downstream optimizer to get a single complete solution. As the discrete skeleton increases in importance, the relative contribution of function evaluations for continuous optimization decreases. Note that, given the same computational budget, the BO method performs less function evaluations than the rest of the methods and the final results are worse than DisCo-DSO. This is because BO has a computational complexity of $\mathcal{O}(n^3)$ (Shahriari et al. 2015), where n is the number of function evaluations. This computational complexity makes BO challenging or even infeasible for large n (Lan et al. 2022).

Decision Tree Policies for Reinforcement Learning

Problem formulation. In this section we consider the problem of discovering decision tree policies for RL. We



(a) Parameterized Bitstring with f_1



(b) Parameterized Bitstring with f_2

Figure 2: Reward of best solution versus number of function evaluations on a parameterized bitstring task, for two continuous optimization landscapes f_1 and f_2 and weights $\alpha = 0.5, 0.9$. Solid line corresponds to weight $\alpha = 0.9$, dashed line $\alpha = 0.5$. Mean and standard error over 5 seeds.

consider $\mathbb{T}$ as the space of univariate decision trees (Silva et al. 2020). Extensions to multivariate decision trees, also known as oblique trees, are possible, but we leave them for future work. Given an RL environment with observations $x_1, \ldots, x_n$ and discrete actions $a_1, \ldots, a_m$, we consider the library of Boolean expressions and actions given by $\mathcal{L} = \{x_1 < \beta_1, \ldots, x_n < \beta_n, a_1, \ldots, a_m\}$, where $\beta_1, \ldots, \beta_n$ are the values of the observations that are used in the internal nodes of the decision tree. The evaluation operator $\texttt{eval} : \mathcal{T} \to \mathbb{T}$ is defined as follows. We treat sequence τ as the pre-order traversal of a decision tree, where the decision tokens ($x_n < \beta_n$) are treated as binary nodes and the action tokens (a_n) are treated as leaf nodes. For evaluating the decision tree, we start from the root node and follow direction

$$D_{x_n<\beta_n}(x) = \begin{cases} \text{left if } x_n < \beta_n \text{ is True,} \\ \text{right if } x_n < \beta_n \text{ is False,} \end{cases}$$

for every decision node encountered until we reach a leaf node. See Figure 3 for an example. The reward function is defined as $R(t) = \mathbb{E}_{r \sim p_R(r|t)}[r]$ where $p_R(r|t)$ is the reward distribution following policy t in the environment. In practice, we use the average reward over N episodes, i.e., $R(t) = \frac{1}{N}\sum_{i=1}^{N} r_i$, where r_i is the reward obtained in episode i. Prefix-dependent positional constraints for this problem are given in the appendix.

Sampling decision nodes in decision trees. To efficiently sample decision nodes, we employ truncated normal distributions to select parameters β_i within permissible ranges. Many RL environments place boundaries on observations, and the use of the truncated normal distribution guarantees that parameters will only be sampled within those boundaries. Additionally, a decision node which is a child of another decision node cannot select parameters from the environment-enforced boundaries. This is because the threshold involved at a decision node changes the range of values which will be observed at subsequent decision nodes. In this way, a previous decision node "dictates" the bounds on a current decision node. For instance, consider the decision tree displayed in Figure 3. Assume that the observation x_1 falls within the interval $[0, 5]$ (note that in practice the RL environment provided bounds are used to determine the interval), and the tree commences with the node $x_1 < 2$. In the left child node, as $x_1 < 2$ is true, there is no need to evaluate whether x_1 is less than 4 (or any number between 2 and 5), as that is already guaranteed. Consequently, we should sample a parameter β_1 within the range (0, 2). Simultaneously, since we do not assess the Boolean expression regarding x_2, the bounds on β_2 remain consistent with those at the parent node. The parameter bounds for the remaining nodes are illustrated in Figure 3. The procedure for determining these maximum and minimum values is outlined in the appendix.

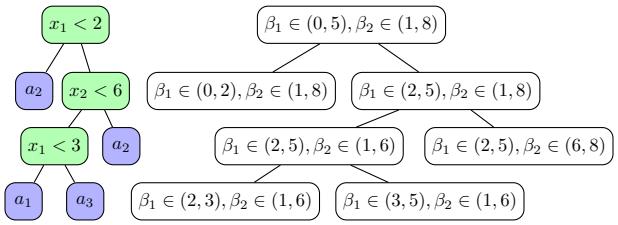


Figure 3: Left: the decision tree associated with the traversal $\langle x_1 < 2, a_2, x_2 < 6, x_1 < 3, a_1, a_3, a_2 \rangle$. Right: the corresponding bounds for the parameters during the sampling process (suppose the bounds for observations x_1 and x_2 are respectively [0, 5] and [1, 8]).

Evaluation. For evaluation, we follow other works in the field (Silva et al. 2020; Ding et al. 2020; Custode and Iacca 2023) and use the OpenAI Gym's (Brockman et al. 2016) environments MountainCar-v0, CartPole-v1, Acrobot-v1, and LunarLander-v2. We investigate the sample-efficiency of DisCo-DSO on the decision tree policy task when compared to the decoupled baselines described at the beginning of this section. We train each algorithm for 10 different random

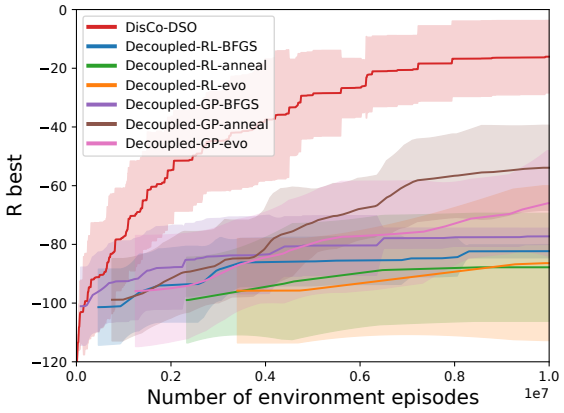


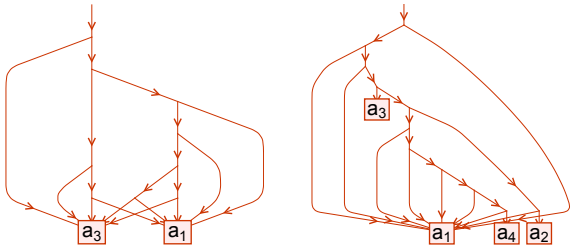
Figure 4: Reward of the best solution versus number of function evaluations on the decision tree policy task for LunarLander-v2.

seeds.

Results. In Figure 4 (see also the appendix), we report the mean and standard deviation of the best reward found by each algorithm versus number of environment episodes. These results show that DisCo-DSO dominates the baselines in terms of sample-efficiency. The trend is consistent across all environments, and is more pronounced in the more complex environments. The efficient use of evaluations by DisCo-DSO (each sample is a complete well-defined decision tree) versus the decoupled approaches, where each sample is a discrete skeleton that requires many evaluations to get a single complete solution, becomes a significant advantage in the RL environments where each evaluation involves running the environment for N episodes.

Literature comparisons. We conduct a performance comparison of DisCo-DSO against various baselines in the literature, namely evolutionary decision trees as detailed in (Custode and Iacca 2023), cascading decision trees introduced in (Ding et al. 2020), and interpretable differentiable decision trees (DDTs) introduced in (Silva et al. 2020). In addition, we provide results with a BO baseline, where the structure of the decision tree is fixed to a binary tree of depth 4 without prefix-dependent positional constraints. Whenever a method provides a tree structure for a specific environment, we utilize the provided structure and assess it locally. In cases where the method's implementation is missing, we address this by leveraging open-source code. This approach allows us to train a tree in absent environments, ensuring that we obtain a comprehensive set of results for all methods evaluated across all environments. The decision trees found by DisCo-DSO are shown in Figure 5 (see also the appendix). Comparisons are shown in Table 1. Methods we trained locally are marked with an asterisk (*). Critically, we ensure consistent evaluation across baselines by assessing each decision tree policy on an identical set of 1,000 random seeds per environment.

In Table 1 we also show the complexity of the discovered decision tree as measured by the number of parameters



(a) Acrobot-v1 (b) LunarLander-v2

Figure 5: Topology of best decision trees found by DisCo-DSO on the decision tree policy tasks for Acrobot-v1 and LunarLander-v2.

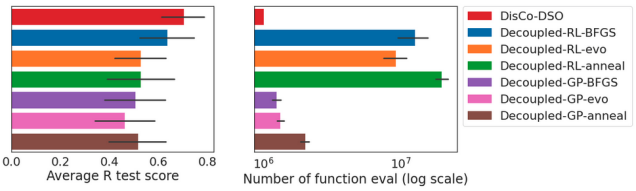


Figure 6: Average test set reward (left) and number of function evaluations (right) used across methods on the symbolic regression task. Recall that Decoupled-RL-BFGS and Decoupled-GP-BFGS correspond to the methods proposed in Petersen et al. (2021) and Koza (1994), respectively. DisCo-DSO achieves the best average reward on the test set at the lowest number of function evaluations.

in the tree. We count every (internal or leaf) node of univariate decision trees (produced by all methods except for Cascading decision trees) as one parameter. For Cascading decision trees, the trees contain feature learning trees and decision making trees. The latter is just univariate decision trees, so the same complexity measurement is used. For the leaf nodes of feature learning trees, the number of parameters is number of observations times number of intermediate features. From Table 1, we observe that the univariate decision trees found by DisCo-DSO have the best performance on all environments at a comparable or lower complexity than the other literature baselines.

Symbolic Regression for Equation Discovery

Problem formulation. Symbolic regression (SR) (Koza 1994; Bongard and Lipson 2007; Petersen et al. 2021; Landajuela et al. 2021b; de Franca et al. 2024) is a classical discrete-continuous optimization problem with applications in many fields, including robotics, control, and machine learning. In SR, we have $\mathcal{L} = \{x_1, \ldots, x_d, +, -, \times, \div, \sin, \cos, \ldots\}$ and $\hat{\mathcal{L}} = \{\text{const}(\beta)\}$, where $\text{const}(\beta)$ represents a constant with value β. The design space is a subset of the space of continuous functions, $\mathbb{T} \subset C(V^{\mathbb{R}})$, where $V \subset \mathbb{R}^d$ is the function support that depends on $\mathcal{L}$. The evaluation operator `eval` returns the function which expression tree has the sequence τ as pre-order traversal (depth-first and then left-to-right). For example,

Algorithm	Acrobot-v1		CartPole-v1		LunarLander-v2		MountainCar-v0	
	MR	PC	MR	PC	MR	PC	MR	PC
DisCo-DSO	**-76.58**	18	**500.00**	14	**99.24**	23	**-100.97**	15
Evolutionary DTs	-97.12*	5	499.58	5	-87.62*	17	-104.93	13
Cascading DTs	-82.14*	58	496.63	22	-227.02	29	-200.00	10
Interpretable DDTs	-497.86*	15	389.79	11	-120.38	19	-172.21*	15
Bayesian Optimization†	-90.99*	7	85.47*	7	-112.14*	7	-200.0*	7

Table 1: Evaluation of the best univariate decision trees found by DisCo-DSO and other baselines on the decision tree policy task. Here, MR is the mean reward earned in evaluation over a set of 1,000 random seeds, while PC represents the parameter count in each tree. For models trained in-house (*), the figures indicate the parameter count after the discretization process. †The topology of the tree is fixed for BO.

$\texttt{eval}(\langle +, \cos, y, \times, \text{const}(3.14), \sin, x\rangle) = \cos(y) + 3.14 \times \sin(x)$. Given a dataset $D = \{(x_1^{(i)}, \ldots, x_d^{(i)}, y^{(i)})\}_{i=1}^N$, the reward function is defined as the inverse of the normalized mean squared error (NMSE) between $y^{(i)}$ and $\texttt{eval}(\tau)(x_1^{(i)}, \ldots, x_d^{(i)}), \forall i \in \{1, \ldots, N\}$, computed as $\frac{1}{1+\text{NMSE}}$. SR has been shown to be NP-hard even for low-dimensional data (Virgolin and Pissis 2022). Prefix-dependent positional constraints are given in the appendix.

Evaluation. A key evaluation metric for symbolic regression is the *parsimony* of the discovered equations, i.e., the balance between the complexity of the identified equations and their ability to fit the data. A natural way to measure it is to consider the generalization performance over a test set. A SR method could find symbolic expressions that overfit the training data (using for instance overly complex expressions), but those expressions will not generalize well to unseen data. For evaluating the generalization performance of various baselines, we rely on the benchmark datasets detailed in the appendix.

Results. Results in Figure 6 demonstrate the superior efficiency and generalization capability of DisCo-DSO in the SR setting. In particular, DisCo-DSO achieves the best average reward on the test set and the lowest number of function evaluations. Note that for DisCo-DSO we have perfect control over the number of function evaluations as it is determined by the number of samples (10^6 in this case). The Decoupled-GP methods exhibit a strong tendency to overfit to the training data and perform poorly on the test set. This phenomenon is known as the *bloat* problem in the SR literature (Silva and Costa 2009). We observe that the joint optimization of DisCo-DSO avoids this problem and achieve the best generalization performance.

Literature comparisons. In Table 2, we compare DisCo-DSO against state-of-the-art methods in the SR literature. In addition to the baselines (Petersen et al. 2021; Koza 1994) described above, we compare against the methods proposed in Biggio et al. (2021) and Kamienny et al. (2022). Since the method in Biggio et al. (2021) is only applicable to ≤ 3 dimensions, we consider the subset of benchmarks with ≤ 3 dimensions. We observe that DisCo-DSO dominates all baselines in terms of average reward on the full test set. For the subset of benchmarks with ≤ 3 dimensions, DisCo-DSO achieves comparative performance to the specialized method in Biggio et al. (2021).

Algorithm	Dim ≤ 3	Dim ≥ 1
DisCo-DSO	0.6632 ± 0.3194	$\mathbf{0.7045} \pm 0.3007$
Decoupled-RL-BFGS*	0.6020 ± 0.4169	0.6400 ± 0.3684
Decoupled-RL-evo	0.0324 ± 0.1095	0.0969 ± 0.2223
Decoupled-RL-anneal	0.1173 ± 0.2745	0.1436 ± 0.3015
Decoupled-GP-BFGS**	0.5372 ± 0.4386	0.4953 ± 0.4344
Decoupled-GP-evo	0.0988 ± 0.1975	0.0747 ± 0.1763
Decoupled-GP-anneal	0.1615 ± 0.2765	0.1364 ± 0.2608
Kamienny et al. (2022)	0.6068 ± 0.1650	0.5699 ± 0.1065
Biggio et al. (2021)	$\mathbf{0.6858} \pm 0.1995$	N/A

Table 2: Comparison of DisCo-DSO against decoupled baselines and the methods proposed in Biggio et al. (2021) and Kamienny et al. (2022) on the symbolic regression task. Values are mean $\pm$ standard deviation of the reward across benchmarks (provided in the appendix). We group benchmarks because Biggio et al. (2021) is only applicable to ≤ 3 dimensions. * Petersen et al. (2021). ** Koza (1994).

Conclusion

We proposed DisCo-DSO (Discrete-Continuous Deep Symbolic Optimization), a novel approach to optimization in hybrid discrete-continuous spaces. DisCo-DSO uses a generative model to learn a joint distribution on discrete and continuous design variables to sample new hybrid designs. In contrast to standard decoupled approaches, in which the discrete skeleton is sampled first, and then the continuous variables are optimized separately, our joint optimization approach samples both discrete and continuous variables simultaneously. This leads to more efficient use of objective function evaluations, as the discrete and continuous dimensions of the design space can "communicate" with each other and guide the search. We have demonstrated the benefits of DisCo-DSO in challenging problems in symbolic regression and decision tree optimization, where, in particular, DisCo-DSO outperforms the state-of-the-art on univariate decision tree policy optimization for RL.

Regarding the limitations of DisCo-DSO, it is important to note that the method relies on domain-specific information to define the ranges of continuous variables. In cases where this information is not available and estimates are necessary, the performance of DisCo-DSO could be impacted. Furthermore, in our RL experiments, we constrain the search space to univariate decision trees. Exploring more complex search spaces, such as multivariate or "oblique" decision trees, remains an avenue for future research.

Acknowledgments

This manuscript has been authored by Lawrence Livermore National Security, LLC under Contract No. DE-AC52-07NA2 7344 with the US. Department of Energy. The United States Government retains, and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes. We thank Livermore Computing and the Laboratory Directed Research and Development program (21-SI-001) for their support. Release code is LLNL-CONF-854776.

References

Bello, I.; Pham, H.; Le, Q. V.; Norouzi, M.; and Bengio, S. 2016. Neural combinatorial optimization with reinforcement learning. *arXiv preprint arXiv:1611.09940*.

Biggio, L.; Bendinelli, T.; Neitz, A.; Lucchi, A.; and Parascandolo, G. 2021. Neural symbolic regression that scales. In *International Conference on Machine Learning*, 936–945. PMLR.

Bongard, J.; and Lipson, H. 2007. Automated reverse engineering of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 104(24): 9943–9948.

Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; and Zaremba, W. 2016. OpenAI Gym. arXiv:arXiv:1606.01540.

Custode, L. L.; and Iacca, G. 2023. Evolutionary learning of interpretable decision trees. *IEEE Access*, 11: 6169–6184.

da Silva, F. L.; Goncalves, A.; Nguyen, S.; Vashchenko, D.; Glatt, R.; Desautels, T.; Landajuela, M.; Faissol, D.; and Petersen, B. 2023. Language model-accelerated deep symbolic optimization. *Neural Computing and Applications*, 1–17.

Daxberger, E.; Makarova, A.; Turchetta, M.; and Krause, A. 2019. Mixed-variable Bayesian optimization. *arXiv preprint arXiv:1907.01329*.

de Franca, F. O.; Virgolin, M.; Kommenda, M.; Majumder, M. S.; Cranmer, M.; Espada, G.; Ingelse, L.; Fonseca, A.; Landajuela, M.; Petersen, B.; Glatt, R.; Mundhenk, N.; Lee, C. S.; Hochhalter, J. D.; Randall, D. L.; Kamienny, P.; Zhang, H.; Dick, G.; Simon, A.; Burlacu, B.; Kasak, J.; Machado, M.; Wilstrup, C.; and Cavaz, W. G. L. 2024. SRBench++: Principled Benchmarking of Symbolic Regression With Domain-Expert Interpretation. *IEEE Transactions on Evolutionary Computation*, 1–1.

Ding, Z.; Hernandez-Leal, P.; Weiguang Ding, G.; Li, C.; and Huang, R. 2020. CDT: Cascading Decision Trees for Explainable Reinforcement Learning. *arXiv preprint: arXiv:2011.07553v2*.

Fischetti, M.; and Jo, J. 2018. Deep neural networks and mixed integer linear optimization. *Constraints*, 23(3): 296–309.

Fletcher, R. 2000. *Practical methods of optimization*. John Wiley & Sons.

Fortin, F.-A.; De Rainville, F.-M.; Gardner, M.-A. G.; Parizeau, M.; and Gagné, C. 2012. DEAP: Evolutionary algorithms made easy. *The Journal of Machine Learning Research*, 13(1): 2171–2175.

Garrido-Merchán, E. C.; and Hernández-Lobato, D. 2020. Dealing with categorical and integer-valued variables in bayesian optimization with gaussian processes. *Neurocomputing*, 380: 20–35.

Hausknecht, M.; and Stone, P. 2016. Deep reinforcement learning in parameterized action space. In *International Conference on Learning Representations*.

Kamienny, P.-A.; d'Ascoli, S.; Lample, G.; and Charton, F. 2022. End-to-end symbolic regression with transformers. *arXiv preprint arXiv:2204.10532*.

Koza, J. R. 1990. *Genetic programming: A paradigm for genetically breeding populations of computer programs to solve problems*, volume 34. Stanford University, Department of Computer Science Stanford, CA.

Koza, J. R. 1994. Genetic programming as a means for programming computers by natural selection. *Statistics and computing*, 4: 87–112.

Lan, G.; Tomczak, J. M.; Roijers, D. M.; and Eiben, A. 2022. Time efficiency in optimization with a bayesian-evolutionary algorithm. *Swarm and Evolutionary Computation*, 69: 100970.

Landajuela, M.; Lee, C.; Yang, J.; Glatt, R.; Santiago, C. P.; Aravena, I.; Mundhenk, T. N.; Mulcahy, G.; and Petersen, B. K. 2022. A Unified Framework for Deep Symbolic Regression. In *Advances in Neural Information Processing Systems*.

Landajuela, M.; Petersen, B. K.; Kim, S.; Santiago, C. P.; Glatt, R.; Mundhenk, N.; Pettit, J. F.; and Faissol, D. 2021a. Discovering symbolic policies with deep reinforcement learning. In Meila, M.; and Zhang, T., eds., *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, 5979–5989. PMLR.

Landajuela, M.; Petersen, B. K.; Kim, S. K.; Santiago, C. P.; Glatt, R.; Mundhenk, T. N.; Pettit, J. F.; and Faissol, D. M. 2021b. Improving exploration in policy gradient search: Application to symbolic optimization. In *1st Mathematical Reasoning in General Artificial Intelligence Workshop, ICLR 2021*. arXiv.

Martius, G.; and Lampert, C. H. 2016. Extrapolation and learning equations. arXiv:1610.02995.

Mundhenk, T.; Landajuela, M.; Glatt, R.; Santiago, C. P.; faissol, D.; and Petersen, B. K. 2021. Symbolic Regression via Deep Reinforcement Learning Enhanced Genetic Programming Seeding. In Ranzato, M.; Beygelzimer, A.; Dauphin, Y.; Liang, P.; and Vaughan, J. W., eds., *Advances in Neural Information Processing Systems*, volume 34, 24912–24923. Curran Associates, Inc.

Nair, V.; Bartunov, S.; Gimeno, F.; von Glehn, I.; Lichocki, P.; Lobov, I.; O'Donoghue, B.; Sonnerat, N.; Tjandraatmadja, C.; Wang, P.; Addanki, R.; Hapuarachchi, T.; Keck, T.; Keeling, J.; Kohli, P.; Ktena, I.; Li, Y.; Vinyals, O.; and

Zwols, Y. 2021. Solving Mixed Integer Programs Using Neural Networks. arXiv:2012.13349.

Neunert, M.; Abdolmaleki, A.; Wulfmeier, M.; Lampe, T.; Springenberg, T.; Hafner, R.; Romano, F.; Buchli, J.; Heess, N.; and Riedmiller, M. 2020. Continuous-discrete reinforcement learning for hybrid control in robotics. In *Conference on Robot Learning*, 735–751. PMLR.

Petersen, B. K.; Landajuela, M.; Mundhenk, T. N.; Santiago, C. P.; Kim, S.; and Kim, J. T. 2021. Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Sahoo, S.; Lampert, C.; and Martius, G. 2018. Learning equations for extrapolation and control. In *International Conference on Machine Learning*, 4442–4450. PMLR.

Shahriari, B.; Swersky, K.; Wang, Z.; Adams, R. P.; and De Freitas, N. 2015. Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE*, 104(1): 148–175.

Silva, A.; Gombolay, M.; Killian, T.; Jimenez, I.; and Son, S.-H. 2020. Optimization methods for interpretable differentiable decision trees applied to reinforcement learning. In *International conference on artificial intelligence and statistics*, 1855–1865. PMLR.

Silva, S.; and Costa, E. 2009. Dynamic limits for bloat control in genetic programming and a review of past and current bloat theories. *Genetic Programming and Evolvable Machines*, 10: 141–179.

Storn, R.; and Price, K. 1997. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11: 341–359.

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement learning: An introduction*. MIT press.

Topchy, A.; Punch, W. F.; et al. 2001. Faster genetic programming based on local gradient search of numeric leaf values. In *Proceedings of the genetic and evolutionary computation conference (GECCO-2001)*, volume 155162. Morgan Kaufmann San Francisco, CA.

Virgolin, M.; and Pissis, S. P. 2022. Symbolic regression is np-hard. *arXiv preprint arXiv:2207.01018*.

Virtanen, P.; Gommers, R.; Oliphant, T. E.; Haberland, M.; Reddy, T.; Cournapeau, D.; Burovski, E.; Peterson, P.; Weckesser, W.; Bright, J.; van der Walt, S. J.; Brett, M.; Wilson, J.; Millman, K. J.; Mayorov, N.; Nelson, A. R. J.; Jones, E.; Kern, R.; Larson, E.; Carey, C. J.; Polat, İ.; Feng, Y.; Moore, E. W.; VanderPlas, J.; Laxalde, D.; Perktold, J.; Cimrman, R.; Henriksen, I.; Quintero, E. A.; Harris, C. R.; Archibald, A. M.; Ribeiro, A. H.; Pedregosa, F.; van Mulbregt, P.; and SciPy 1.0 Contributors. 2020. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17: 261–272.

Xiang, Y.; Sun, D.; Fan, W.; and Gong, X. 1997. Generalized simulated annealing algorithm and its application to the Thomson model. *Physics Letters A*, 233(3): 216–220.

Xiong, J.; Wang, Q.; Yang, Z.; Sun, P.; Han, L.; Zheng, Y.; Fu, H.; Zhang, T.; Liu, J.; and Liu, H. 2018. Parametrized deep q-networks learning: Reinforcement learning with discrete-continuous hybrid action space. *arXiv preprint arXiv:1810.06394*.

Zoph, B.; and Le, Q. V. 2017. Neural Architecture Search with Reinforcement Learning. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.