Scaling Up Unbiased Search-based Symbolic Regression

Paul Kahlmeyer[1] , **Joachim Giesen**[1] , **Michael Habeck**[2] and **Henrik Voigt**[1]
[1]Friedrich Schiller University Jena
[2]University Hospital Jena
{paul.kahlmeyer, joachim.giesen, michael.habeck, henrik.voigt}@uni-jena.de

Abstract

In a regression task, a function is learned from labeled data to predict the labels at new data points. The goal is to achieve small prediction errors. In symbolic regression, the goal is more ambitious, namely, to learn an *interpretable* function that makes small prediction errors. This additional goal largely rules out the standard approach used in regression, that is, reducing the learning problem to learning parameters of an expansion of basis functions by optimization. Instead, symbolic regression methods *search* for a good solution in a space of symbolic expressions. To cope with the typically vast search space, most symbolic regression methods make implicit, or sometimes even explicit, assumptions about its structure. Here, we argue that the only obvious structure of the search space is that it contains small expressions, that is, expressions that can be decomposed into a few subexpressions. We show that systematically searching spaces of small expressions finds solutions that are more accurate and more robust against noise than those obtained by state-of-the-art symbolic regression methods. In particular, systematic search outperforms state-of-the-art symbolic regressors in terms of its ability to recover the true underlying symbolic expressions on established benchmark data sets.

1 Introduction

Given a training set of labeled data, the goal in regression is to find a model that generalizes well beyond the training data. At its core, regression problems are search problems on some space of functions that map data points to labels. Typically, the function space is structured along two dimensions, a space of *structural frames* and a set of *parameters*. Traditionally, the search space of structural frames is kept rather simple, as a set of linear combinations of some basis functions. For a given regression problem, by considering only one frame, that is, traditionally the number and form of basis functions, the search problem can be cast as the optimization problem to find the parameters that give the best fit on the training data. Therefore, the focus shifts to the parameter space. Restricting the parameter space by adding regularization terms to the optimization problem can provably improve the ability to generalize [Hoerl and Kennard, 1970; Tibshirani, 1996].

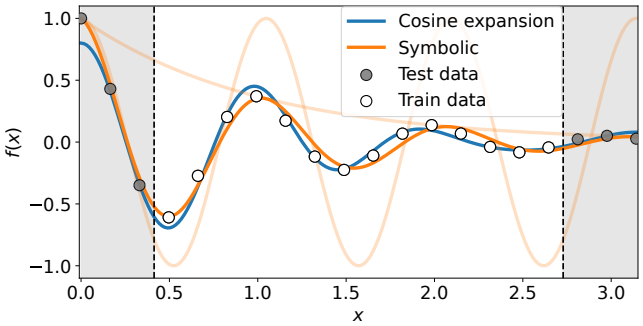

Figure 1: Interpretability goes beyond generalization. Data from a dampened pendulum are fitted by the symbolic regressor $e^{-x}\cos(6x)$ and the cosine expansion $0.06\cos(4x) + 0.13\cos(5x) + 0.30\cos(6x) + 0.23\cos(7x) + 0.08\cos(8x)$. While both models generalize well outside the training data, the symbolic regressor can be interpreted as the multiplicative composition of an exponential decay and an oscillation (light orange). The cosine expansion lacks such an interpretation.

Generalization is not the only goal of symbolic regression. The second goal is interpretability. Interpretability, facilitating fundamental insights by inspecting and interpreting the symbolic expressions, is the reason why symbolic regression has found applications in almost all areas of the natural sciences [Keren *et al.*, 2023; Liu and Tegmark, 2021; Liu and Tegmark, 2022], social sciences [Aryadoust, 2015; Truscott and Korns, 2011], and engineering [Can and Heavey, 2011; Quade *et al.*, 2016; Wang *et al.*, 2019; Landajuela *et al.*, 2021]. We illustrate this point using the example shown in Figure 1. Data points that are sampled from a dampened pendulum are fitted by two models. The first model is a linear combination of cosine basis functions (blue) and the second model is a symbolic regressor (orange). Both models generalize well beyond the training data and could be used for prediction. From the symbolic form of the polynomial, however, we do not gain further insights. The symbolic regressor provides such insights: it is composed of an exponential de-

cay and an oscillation that are coupled through multiplication, meaning that the oscillation is dampened by the decay.

Because of the second goal, interpretability, symbolic regression algorithms are evaluated differently than standard regression algorithms. Good prediction performance on test data is still important, but for symbolic regression algorithms the ability to recover known ground-truth formulas, up to symbolic equivalence, has become an accepted validation measure. Complete symbolic recovery, however, is a very strict quality measure. Therefore, we propose also a relaxed measure, namely, the successful recovery of subexpressions. For instance, recovering only the dampening or only the oscillation term of the dampened pendulum, still provides valuable insights.

Interpretability of symbolic regressors comes at the price of a much larger search space of structural frames, which consists of all mathematical expressions that are specified in a formal language, where constants are not instantiated but represented by a placeholder symbol. State-of-the-art symbolic regressors tackle the search problem by various techniques such as genetic programming, reinforcement learning, or transformers that all make some implicit assumptions to reduce the effective size of the search space. In this work, we take a step back and explore a basic, unbiased and thus assumption-free search. It turns out, that this basic approach fares well in comparison to current state-of-the-art regressors on small instances of established benchmark data sets. It does not, however, scale to moderately large expressions. For scaling up the basic search, we combine it with a variable augmentation approach that aims at identifying subexpressions in the target expression that can be eliminated from the search process. Variable augmentation itself constitutes a search problem for subexpressions, that can be addressed by the same technique as the overall symbolic regression task. Our experimental results on the standard benchmark data sets show that the combination of unbiased search and variable augmentation improves the state of the art in terms of *accuracy*, that is, its ability to recover known ground truth formulas, but also in terms of *robustness*, that is, its ability to cope with noise.

2 Related Work

The core problem in symbolic regression is searching the space of structural frames. For a given frame, the parameters are mostly estimated by minimizing a loss function on hold-out data. The space of frames is usually given in the form of expression trees for expressions from a given formal language. Symbolic regression algorithms differ in the way they search the space of expression trees.

Genetic Programming. The majority of symbolic regression algorithms follow the *genetic programming* paradigm [Koza, 1994; Holland, 1975]. The paradigm has been implemented in the seminal Eureqa system by [Schmidt and Lipson, 2009] and in gplearn by [Stephens, 2016]. More recent implementations include [La Cava *et al.*, 2016; Kommenda *et al.*, 2020; Virgolin *et al.*, 2021]. The basic idea is to create a population of expression trees, to turn the trees into symbolic regressors by estimating the parameters, and to recombine the expression trees for the best performing regressors into a new population of expression trees by (ex)changing subtrees. Here, the assumption is, that expression trees for symbolic regressors that perform well contain at least parts of the target expression. Therefore, recombining the best performing expression trees shall keep these parts within the population.

Bayesian Inference. A different approach is to use Bayesian inference for symbolic regression. In the Bayesian inference approach, MCMC (Markov Chain Monte Carlo) is used for sampling expressions from a posterior distribution. Implementations of the Bayesian approach differ in the choice of prior distribution. [Jin *et al.*, 2020] use a hand-designed prior distribution on expression trees, whereas [Guimerà *et al.*, 2020] compile a prior distribution from a corpus of 4,080 mathematical expressions that have been extracted from Wikipedia articles on problems from physics and the social sciences. Both implementations define the likelihood in terms of model fit. Here, the assumption is, that, by the choice of prior and likelihood, expression trees that are similar to well performing trees have a higher posterior probability to be sampled.

Neural Networks. [Petersen *et al.*, 2021; Mundhenk *et al.*, 2021] train a recurrent neural network for sampling expression trees token-by-token in preorder. Here, the assumption is similar to the assumption underlying the Bayesian approach, namely that the loss used during reinforcement learning shifts the token-generating distribution toward sequences of good performing expressions. The work by [Kamienny *et al.*, 2022] also falls into this category. Here, a transformer is trained in an end-to-end fashion on a large data set of regression problems to translate regression tasks, given as a sequence of input output pairs, into a sequence of tokens for an expression tree in preorder. At inference, for a given regression problem, beam search is used to return preorders with the highest likelihood. Here, the assumption is that the data set for pretraining the transformer covers the search space well, because the probability distribution favors token sequences that are similar to well-fitting sequences seen during training.

Ensemble. Recently, [Landajuela *et al.*, 2022] have combined different symbolic regressors based on genetic programming, reinforcement learning, and transformers together with problem simplification into a unified symbolic regressor. The ensemble approach is more robust with respect to the assumptions for its constituent approaches, that is, it can cope better with some of the assumptions not met. Essentially, however, it adheres to the same assumptions.

Systematic Search. The assumptions made for the different symbolic regression approaches are difficult to check. Therefore, the idea of an unbiased, more or less assumption-free search is appealing. So far, however, implementations of an unbiased search have been limited to restricted search spaces that can be searched exhaustively, namely, to rational low-degree polynomials with linear and nonlinear terms [Kammerer *et al.*, 2020], and to univariate regression problems up to a certain depth [Bartlett *et al.*, 2023]. The AIFeynman project [Udrescu and Tegmark, 2020; Udrescu *et al.*,

2020] uses a set of neural-network-based statistical property tests to scale an unbiased search to larger search spaces. Statistically significant properties, for example, additive or multiplicative separability, are used to decompose the search space into smaller parts, which are addressed by a brute-force search. The search, however, is limited by the set of available property tests. Our approach is similar in spirit to AIFeynman, but different in implementation. At the core of our approach is a succinct representation of symbolic expressions, namely expression DAGs that are, as we will show, well suited for a systematic unbiased, search-based approach.

3 DAGs as Structural Frames

The search space of structural frames in symbolic regression consists of mathematical expressions that are defined by a formal language. We provide the grammar for the formal language of mathematical expressions that we are using here in the supplemental material. The grammar, however, does not completely specify the search space. There are two issues that affect any search-based approach to symbolic regression: First, the grammar specifies the syntax of valid expressions, but does not fix the representation for its conforming expressions. Second, by the rules of arithmetic, there are syntactically different expressions that are semantically equivalent, that is, they specify exactly the same function. Therefore, it is important to find a representation for the expressions that structures the search space so that the number of functions that can be covered for a given size constraint on the search space is maximized. The most direct representation of an expression is in the form of a string of tokens, but the most commonly used representation in symbolic regression is an expression tree. We illustrate the two issues on the example of the function

$$f(x) = x^4 + x^3\,,$$

which has an expression tree with five operator nodes (addition, multiplication, and three times squaring). The same function, however, has another expression as $x^2(x^2+x)$ with an expression tree that has only four operator nodes (addition, multiplication, and two times squaring). Therefore, it is often suffcient to exhaustively search the space of small expression trees.

Moreover, it has been pointed out already by [Schmidt and Lipson, 2007] and is also well known in compiler construction [Aho *et al.*, 1986] that expression DAGs (directed acyclic graphs), where common subexpressions have been eliminated [Cocke, 1970], are an even more favorable because more succinct representation. The expression DAG for the expression $x^2(x^2 + x)$, that factors out the common subexpression x^2, has only three operator nodes (addition, multiplication, and squaring). The difference in size becomes more pronounced when we also consider the leafs, that either store variables or constants, of the expression trees and expression DAGs, respectively. We illustrate the difference between the expression tree and expression DAG representation in Figure 2.

Expression DAGs. Here, we describe the expression DAGs that we use for arithmetic expressions in more detail. We distinguish four types of nodes: variable nodes, parameter nodes, intermediary nodes, and output nodes. Variable and parameter nodes are *input* nodes. An n-variate function has n input nodes for the variables, that is, it is defined by the regression task at hand, whereas the number of input nodes for the parameters is not fixed, but part of the structural search space. Both, the intermediary and the output nodes together are *operator* nodes. The number of output nodes is also specified by the given regression task, whereas the number of intermediary nodes is not. Note that it can pay off to encode a function with several components, that is, several output nodes, into a single expression DAG, because the components can share common subexpressions. This happens frequently for systems of ordinary differential equations [Strogatz, 2000]. The DAGs are oriented from the input to the output nodes, that is, the input nodes are the roots of the DAGs and the output nodes are its leafs. Only the intermediary nodes have incoming and outgoing edges. We distinguish two types of operator nodes, namely unary and binary operator nodes. In symbolic regression, typically, the binary operators

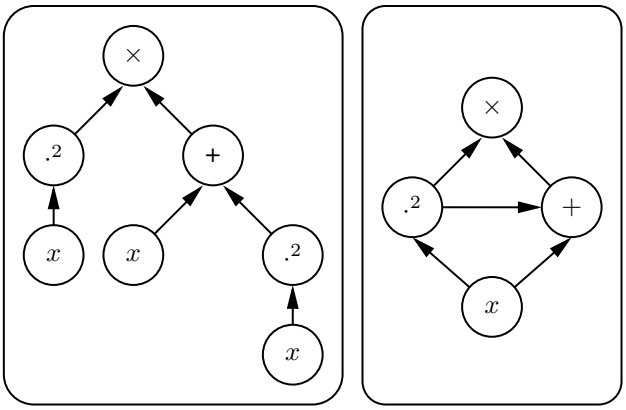

Figure 2: Representation of the expression $x^2(x^2+x)$ by an expression tree (left) and by an expression DAG (right).

$$+,\ -,\ \times,\ \text{and}\ \div,$$

and the unary operators

$$-,\ ^{-1},\ \sin,\ \cos,\ \log,\ \exp,\ ^{2},\ \text{and}\ \sqrt{\ }$$

are supported. Examples of expression DAGs are shown in Figures 2 and 3.

4 Searching the Space of Expression DAGs

As we have pointed out already, the number of variable nodes and the number of output nodes are specified by the given regression problem. It remains to parameterize the space of expression DAG by the number p of parameter nodes and by the number i of intermediary nodes. That is, the regression problem together with a tuple (p, i) defines the search space. Here, we always use $p = 1$, that is, only one parameter node. Different parameters can be expressed as functions of a single parameter, for instance $2(x^2 + 1)$ can be expressed as $(1 + 1)(x^2 + 1)$ or as $2x^2 + 2$, which only need one parameter. More details are provided in the supplemental material. We use the term DAG *skeleton* for DAGs with unlabeled operator nodes, and the term DAG *frame* for DAGs with labeled

operator nodes. Both, DAG skeletons and DAG frames, only have constant placeholders for the input nodes.

4.1 Unbiased Search of Expression DAGs

Our randomized search of the search space of expression DAGs given by $(1, i)$ is unbiased, but not exhaustive. An exhaustive search would not scale and, as it turns out (see section 6), is often not necessary. The search procedure has two phases: In the first phase, we construct a DAG skeleton, that is, a DAG without labeling of the operator nodes. In the second phase, we search over all DAG frames for a skeleton, by considering all operator node labelings from the sets of unary and binary operator symbols.

Sampling DAG Skeletons. For sampling DAG skeletons, we number the intermediary nodes from 1 to i to ensure a topological order on the nodes.

1. Unary output nodes sample its predecessor uniformly at random from all non-output nodes, and binary output nodes samples a pair of predecessors uniformly at random from all pairs of non-output nodes.
2. Unary intermediary nodes sample their predecessor uniformly at random from all input nodes and all intermediary nodes with smaller number, and binary intermediary nodes sample a pair of predecessors uniformly at random from all pairs made up from input nodes and intermediary nodes with smaller number.

Finally, we recursively remove all intermediary nodes that have no successor, that is, no outgoing edge. Note that input nodes have by definition no predecessors. Sampling of a DAG skeleton is illustrated in Figure 3.

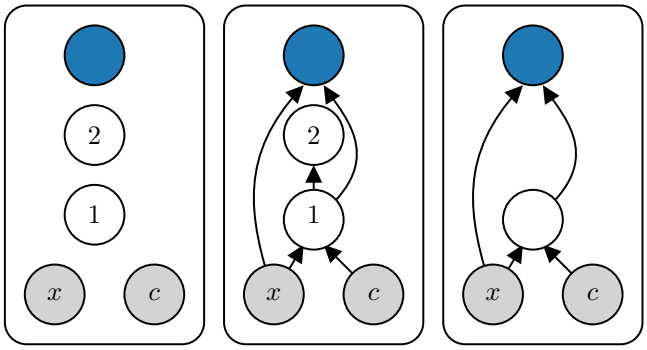

Figure 3: DAG skeletons are generated by enumerating intermediary nodes (left), selecting predecessors according to the numbering (middle), and deleting intermediary nodes without connection to the output (right). The shown DAG covers expressions such as $x(x+c)$, $x + xc$ or cx^2.

Operator Node Labeling. For a DAG skeleton, we exhaustively search the space of all DAG frames, that is, we consider all combinations of operator node labelings from the set of unary and binary operator labels for unary and binary operator nodes respectively.

4.2 Scoring Expression DAG Frames

An expression DAG frame Δ is not a regressor yet. It remains to find values for the DAG's parameter node. Here, we follow the classical approach and optimize the parameters with respect to the model fit on training data. Let $\Delta(x, \theta)$ be the function that results when the DAG's input nodes are instantiated by $x \in \mathbb{R}^n$, where n is the number of input nodes, and its parameter nodes are instantiated by the parameter vector $\theta \in \mathbb{R}^p$, where p is the number of parameter nodes in Δ. Given training data $(x_1, y_1), \ldots, (x_\ell, y_\ell) \in \mathbb{R}^{n \times m}$ for a regression problem with n input and m output variables, we compute the parameter values for the parameter nodes by minimizing the following square-loss

$$\hat{\theta} = \arg\min_{\theta \in \mathbb{R}^p} \sum_{i=1}^{\ell} \left\| \Delta(x_i, \theta) - y_i \right\|^2 =: \arg\min_{\theta \in \mathbb{R}^p} L(\Delta, \theta).$$

Every function $\Delta(\cdot, \hat{\theta})$ is a regressor. Among all the regressors $\Delta(\cdot, \hat{\theta})$ from the search space, that is, a search over Δ, we choose one that fits the data best, that is, has a minimal loss $L(\Delta, \hat{\theta})$.

5 Variable Augmentation

So far, we have argued that expression DAGs provide rather small search spaces for symbolic regression. However, even these search spaces grow so fast that they can only support very small expressions. Here, we describe our main contribution, namely, we introduce symbolic variable augmentations for simplifying symbolic regression problems.

5.1 Input Variable Augmentation

We describe the basic idea on a simple regression problem with two input and one output variable. Assume that we are given data that have been sampled from the function

$$f(x_1, x_2) = \frac{x_1 x_2^2 + x_1}{x_2}$$

that has a corresponding expression DAG with six nodes, among them four operator nodes. Using a new variable $z(x_1, x_2) = x_1/x_2$, the function can be represented by the expression

$$x_1 x_2 + z$$

that has an expression with only five nodes, among them three input nodes. That is, it is possible to find a DAG frame for the function f in a smaller search space, if we increase the number of input variables.

Given a regression problem, we call any symbolic expression in the input variables a *potential input variable augmentation*, where we only consider expressions without parameters ($p = 0$). The challenge is to identify variable augmentations that lead to smaller expression DAGs. We address the challenge by a combination of searching expression DAGs and standard, that is, non-symbolic, regression. The search of small expression DAGs is used to enumerate potential input variable augmentations, and a standard regressor is used to score the potential augmentations.

Given a regression problem, expression DAGs for potential variable augmentations are sampled using the unbiased expression DAG search from Section 4.1. Here, we sample

only expression DAGs that feature a subset of the input variables nodes and no parameter nodes. Since the sampled expression DAGs Δ do not have parameter nodes, they each describe a unique univariate function z_Δ on the corresponding selected subset of the input variables. The functions z_Δ are then scored on training data $(x_1, y_1), \ldots, (x_\ell, y_\ell) \in \mathbb{R}^{n\times m}$ for a regression problem with n input and m output variables that are augmented as,

$$(x_1, z_1, y_1), \ldots, (x_\ell, z_\ell, y_\ell) \in \mathbb{R}^{(n+1)\times m},$$

where $z_i = z_\Delta(x_i|_\Delta) \in \mathbb{R}$ and $x_i|_\Delta$ is the projection of the i-th vector of input variables onto the input variables that appear in the expression DAG Δ. For a given class $\mathcal{F}$ of standard regressors, for instance, polynomial regressors or a class of neural networks, we use a scoring function that is derived from the *coefficient of determination* R^2 [Wright, 1921]

$$\min_{f\in\mathcal{F}} \frac{\sum_{i=1}^{\ell} \|f(x_i, z_i) - y_i\|^2}{\sum_{i=1}^{\ell} \|y_i - \bar{y}\|^2} =: \min_{f\in\mathcal{F}} \big(1 - R^2(X, Y, f)\big),$$

which is frequently used in symbolic regression. Here, $\bar{y}$ is the mean of the output variables in the training data set, and X and Y are matrices of the respective input and output data vectors.

5.2 Augmented Expression DAG Search Algorithm

We have integrated the variable augmentation into an unbiased, search-based algorithm for symbolic regression that is shown in Figure 4. Given a regression problem in terms of

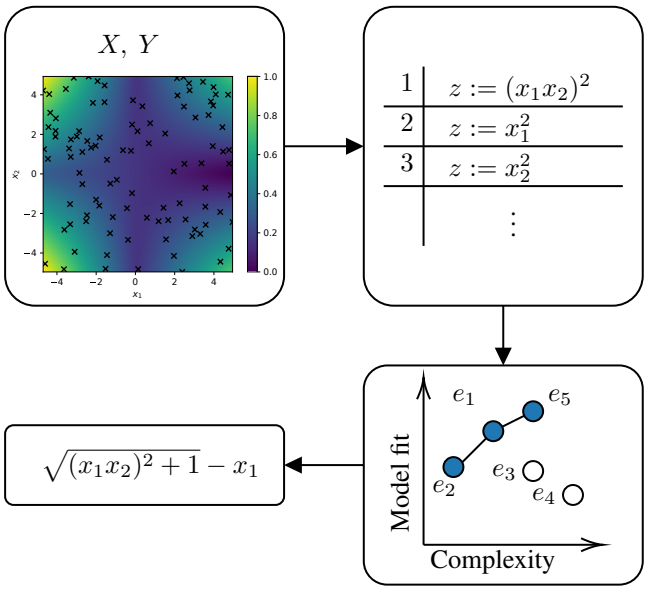

Figure 4: Conceptual sketch of the unbiased, search-based symbolic regression algorithm: From the original problem (1) we select top-k augmentations (2) and solve those problems, creating a Pareto front (3). The returned expression (4) is the best one with complexity below a threshold, or the smallest expression if the complexity of all expressions is above the threshold.

a data sample of input and corresponding output variables, our unbiased, search-based symbolic regression algorithm has two main steps:

1. **Selecting Variable Augmentations.** Given a standard family $\mathcal{F}$ of regression functions, such as linear regression, polynomial regression, neural networks, and a value k, we use the standard family of regression functions and the unbiased expression DAG search to select the top-k scoring variable augmentations.
2. **Solving Augmented Regression Problems.** For the selected variable augmentations, we compute k symbolic regressors as described in Section 4. For each regressor, we compute its *model* fit in terms of the coefficient of determination R^2 and its *model complexity* as the number of nodes in the expression DAG. Similar to [Udrescu *et al.*, 2020], we always keep the models on the Pareto front with respect to model fit and model complexity. If we have to return a single model, then we return the best fitting model with a complexity below a given threshold. If there is no such model, we simply return the smallest model.

6 Experimental Evaluation

We conducted two types of experiments. In the first experiment, we compare the performance of our unbiased, search-based approach to the state of the art in symbolic regression on the established and comprehensive SRBench test suite by [La Cava *et al.*, 2021]. In the second experiment, we evaluate the scalability advantage that variable augmentation brings to the unbiased, search-based approach.

Unless stated otherwise, our method, named UDFS (Unbiased DAG Frame Search) was used with five intermediary nodes and a maximum of 200 000 DAG skeletons. For the variable augmentation, we have used polynomial regression (UDFS + Aug) to select $k = 1$ augmentations and up to 30 nodes in the corresponding DAG search.

All experiments were run on a computer with an Intel Xeon Gold 6226R 64-core processor, 128 GB of RAM, and Python 3.10.

6.1 The SRBench Symbolic Regression Test Suite

SRBench [La Cava *et al.*, 2021] is an open-source benchmarking project for symbolic regression. It comprises 14 of the state-of-the-art symbolic regression models, a set of 252 regression problems from a wide range of applications, and a model evaluation and analysis environment. It is designed to easily include new symbolic regression models, such as the UDFS + Aug model that we propose here, for benchmarking against the state-of-the-art.

The models' performance is measured along three dimensions, namely *model fit* on test data, *complexity*, and *accuracy*, that is, the ability to recover ground truth expressions. Model fit is measured by the *coefficient of determination* R^2, and *complexity* is measured by the number of nodes in the expression tree of an expression. Most important for us, a ground truth expression f is considered as *recovered* by a model $\hat{f}$, if either $f - \hat{f}$ can be symbolically resolved to a constant or $\hat{f}$ is non-zero and $\hat{f}/f$ can be symbolically resolved to a constant. The symbolical checks are delegated

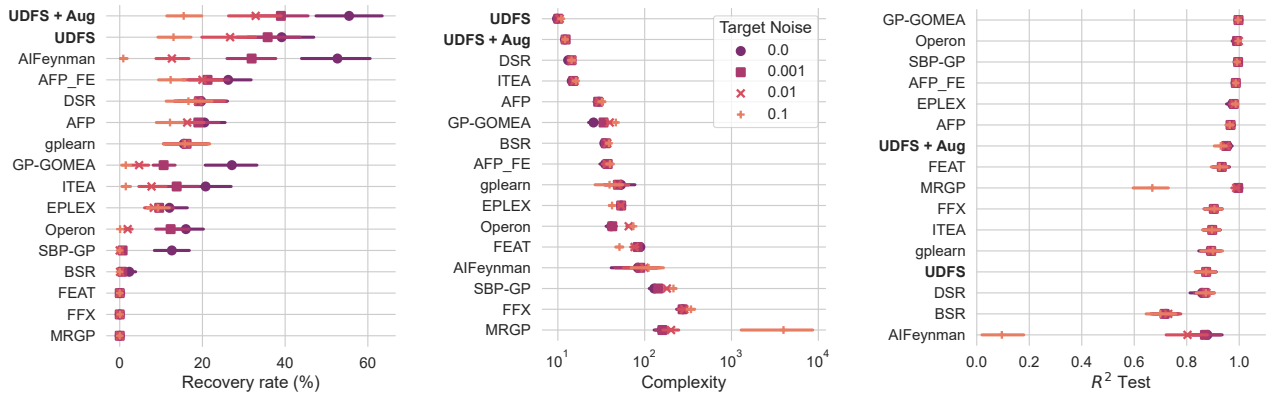

Figure 5: Left: Recovery rate, center: model complexity, right: model fit for UDFS and UDFS + Aug as scored by the SRBench test suite. The reported values average over 10 runs for the complexity and recovery measures. The reported R^2 values are medians from 10 runs, because R^2 values can vary significantly.

to the Python library `SymPy` [Meurer *et al.*, 2017]. The ground truth is only known for 130 out of the 252 problems, namely, the problems from the *Feynman Symbolic Regression Database*[1] and the *ODE-Strogatz Repository*[2]. Since we consider ground truth recovery as the most important quality measure for symbolic regressors, as it is a direct measure of interpretability, we restrict ourselves to these 130 problems that are described in the supplemental material.

The comparative experiments summarized in Figure 5 show that the unbiased search UDFS + Aug is on average more *accurate*, that is, it can recover more known ground truth expressions, and also more *robust*, that is, it can recover more ground truth expressions from noisy data, than the state of the art. Furthermore, UDFS + Aug produces small models with an average R^2- test score that is close to the optimal score of 1.0. The second-best regressor in terms of recovery, AIFeynman, gives significantly larger and worse fitting models when it cannot recover the ground truth. That is, in these cases, AIFeynman is also not a good regressor in the standard sense of regression. There is a large group of regression models that produce extremely well fitting models with an average R^2 close to 1.0. However, these models are mostly approximations of the ground truth functions, as their recovery rate is significantly lower and the model complexity is quite high. For a high R^2 score, however, one could simply resort to standard regressors, such as neural networks. But even in terms of standard model fitting performance, UDFS and UDFS+ Aug place reasonably well among the other symbolic regression methods. Figure 6 shows the different Pareto fronts with respect to the model fit and model complexity. Both, UDFS and its variable augmented extension, are on the first Pareto front.

However, both, UDFS and UDFS + Aug are not perfect. To gain a better understanding of the expressions that could not be recovered, we did a more fine-grained analysis and also looked at partial recovery of expressions. For an example,

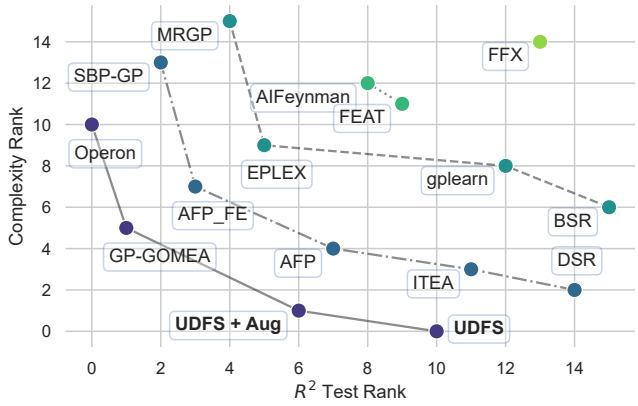

Figure 6: Pareto fronts generated by SRBench for symbolic regression models with respect to the dimensions complexity and model fit (R^2 rank).

consider the expression `Feynman_II_6_11`

$$\phi(x, y, z) = \frac{1}{4\pi\varepsilon_0}\frac{p\cos\theta}{r^2}$$

from the Feynman Lectures on Physics [Feynman *et al.*, 2011], where r, θ, and p are functions of x, y, and z, and the permittivity of free space ε_0 is a constant that here is also considered as a regression variable. Our search-based regressor returns the expression

$$\phi(x, y, z) = \frac{1}{10.44\varepsilon_0^{1.5}}\frac{p\cos\theta}{r^2},$$

which fully recovers the functional dependency on the physical variables r, θ, and p, and only misses the correct form of the dependence on ε_0. That is, the expression returned by the algorithm still provides profound insights into the physics behind the data on which it was run. Nevertheless, the recovery measure by [La Cava *et al.*, 2021] considers the problem just as not recovered. Therefore, we also consider a weaker version of recovery instead. For comparing two normalized

[1]https://space.mit.edu/home/tegmark/aifeynman.html
[2]https://github.com/lacava/ode-strogatz

expressions e_1 and e_2, let S_1 and S_2 be the corresponding sets of subexpressions. The Jaccard index [Jaccard, 1902], a similarity measure between sets,

$$J(S_1, S_2) = \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|}$$

can then be used to indirectly measure the similarity of e_1 and e_2. We have $J(S_0, S_1) \in [0, 1]$ and $J(S_0, S_1) = 1$, if e_0 and e_1 have the same subexpressions, which means that they are equal. For the example given above, we get a Jaccard index of 0.47, reflecting the partial recovery of the ground truth.

Figure 7 shows the results of the SRBench test suite using the Jaccard index instead of the full recovery rate. Note, that DSR now ranks higher than AIFeynman, because it is more robust against noise. The UDFS and the UDFS+ Aug regressors, however, are still more accurate and robust.

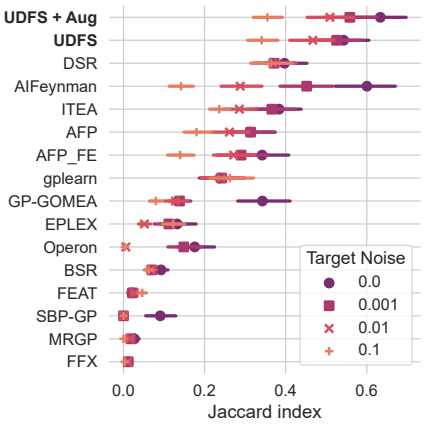

Figure 7: Results of the SRBench test suite for the Jaccard index.

6.2 Scalability

The results shown in Figure 5 show that UDFS without variable augmentation compares fairly well to the state of the art. UDFS + Aug, however, that reduces the effective size of the search space, performs even better. To see at which search space size variable augmentation does start to improve the performance of UDFS, we have compared the performance of UDFS and UDFS + Aug on the *Nguyen* problems, a collection of small but challenging regression problems that have been used in [Petersen *et al.*, 2021; Mundhenk *et al.*, 2021]. The results are shown in Figure 8.

Interestingly, the `Nguyen-2` expression $x^4 + x^3 + x^2 + x$, which has a larger expression tree than the `Nguyen-6` expression $\sin(x) + \sin(x^2 + x)$, can be recovered by UDFS, whereas `Nguyen-6` can not. UDFS however, does not recover the ground truth expression, but the semantically equivalent expression $x^2(x^2 + x) + x^2 + x$, which has a compact DAG representation with only three intermediary nodes that reuse the common subexpressions x^2 and $x^2 + x$. The complexity of UDFS, however, is not only controlled by the number of intermediary nodes within the DAG frames, but also by the number of DAG skeletons. Both, the `Nguyen-6` and `Nguyen-7` problems, have DAG representations with

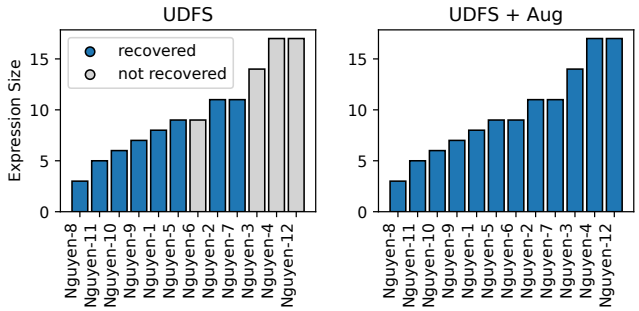

Figure 8: Recovery of Nguyen problems sorted by the number of nodes in the expression tree for the ground truth expression. The shown results are computed with five intermediary nodes and 200 000 DAG skeletons for UDFS (left) and UDFS + Aug (right).

four intermediary nodes, but only the latter has been found by UDFS. The reason why `Nguyen-6` was not recovered is that the corresponding DAG frame was not included in the search by the random sampling process. As can be seen from Figure 9 the probability to recover `Nguyen-6` increases with the number of sampled DAG skeletons, but so does the computational effort.

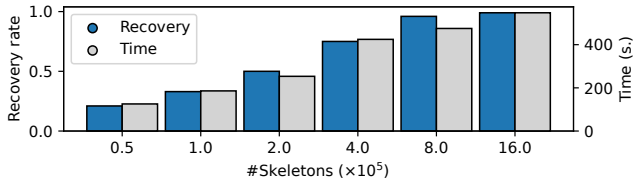

Figure 9: Recovery rate and computation time averaged over 100 runs for the `Nguyen-6` problem for an increasing number of DAG skeletons with four intermediary nodes.

For the `Nguyen-4` and `Nguyen-12` problems, that have DAG representations with more intermediary nodes, it is no longer feasible to increase the number of DAG skeletons to such a degree that they cover the search space reasonably well. Both problems, however, benefit from the variable augmentations that are used by UDFS + Aug.

More experimental results on scalability, including running times, can be found in the supplemental material.

7 Conclusions

Symbolic regression is the problem of searching for a well-fitting function in a space of symbolic function expressions. Since the search space of potential symbolic function expressions is vast, most symbolic regression approaches are biased in the sense that they make implicit, or sometimes also explicit assumptions, to reduce the effective size of the search space. Here, we have discussed how to scale up an unbiased search for symbolic regression to problem instances that are used to establish the state of the art in symbolic regression. Our unbiased, search-based regressor improves the state of the art in terms of both accuracy and robustness.

Acknowledgements

This work was supported by the Carl Zeiss Stiftung within the project "Interactive Inference". In addition, Michael Habeck acknowledges funding by the Carl Zeiss Stiftung within the program "CZS Stiftungsprofessuren".

References

[Aho *et al.*, 1986] Alfred V. Aho, Ravi Sethi, and Jeffrey D. Ullman. *Compilers: Principles, Techniques, and Tools*. Addison-Wesley, 1986.

[Aryadoust, 2015] Vahid Aryadoust. Application of evolutionary algorithm-based symbolic regression to language assessment: Toward nonlinear modeling. *Psychological Test and Assessment Modeling*, 57(3):301, 2015.

[Bartlett *et al.*, 2023] Deaglan J Bartlett, Harry Desmond, and Pedro G Ferreira. Exhaustive symbolic regression. *IEEE Transactions on Evolutionary Computation*, 2023.

[Can and Heavey, 2011] Birkan Can and Cathal Heavey. Comparison of experimental designs for simulation-based symbolic regression of manufacturing systems. *Computers & Industrial Engineering*, 61(3):447–462, 2011.

[Cocke, 1970] John Cocke. Global common subexpression elimination. In Robert S. Northcote, editor, *Proceedings of a Symposium on Compiler Optimization*, pages 20–24. ACM, 1970.

[Feynman *et al.*, 2011] Richard P. Feynman, Robert B. Leighton, and Matthew Sands. *The Feynman Lectures on Physics, Vol. II: The New Millennium Edition: Mainly Electromagnetism and Matter*. The Feynman Lectures on Physics. Basic Books, 2011.

[Guimerà *et al.*, 2020] Roger Guimerà, Ignasi Reichardt, Antoni Aguilar-Mogas, Francesco A. Massucci, Manuel Miranda, Jordi Pallarès, and Marta Sales-Pardo. A bayesian machine scientist to aid in the solution of challenging scientific problems. *Science Advances*, 6(5):eaav6971, 2020.

[Hoerl and Kennard, 1970] Arthur E. Hoerl and Robert W. Kennard. Ridge Regression: Biased Estimation for Nonorthogonal Problems. *Technometrics*, 12:55–67, 1970.

[Holland, 1975] John H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, 1975. second edition, 1992.

[Jaccard, 1902] Paul Jaccard. Lois de distribution florale dans la zone alpine. *Bulletin de la Société vaudoise des sciences naturelles*, 38:69–130, 01 1902.

[Jin *et al.*, 2020] Ying Jin, Weilin Fu, Jian Kang, Jiadong Guo, and Jian Guo. Bayesian symbolic regression, 2020.

[Kamienny *et al.*, 2022] Pierre-Alexandre Kamienny, Stéphane d'Ascoli, Guillaume Lample, and Francois Charton. End-to-end symbolic regression with transformers. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

[Kammerer *et al.*, 2020] Lukas Kammerer, Gabriel Kronberger, Bogdan Burlacu, Stephan M Winkler, Michael Kommenda, and Michael Affenzeller. Symbolic regression by exhaustive search: Reducing the search space using syntactical constraints and efficient semantic structure deduplication. *Genetic programming theory and practice XVII*, pages 79–99, 2020.

[Keren *et al.*, 2023] Liron Simon Keren, Alex Liberzon, and Teddy Lazebnik. A computational framework for physics-informed symbolic regression with straightforward integration of domain knowledge. *Scientific Reports*, 13(1):1249, 2023.

[Kommenda *et al.*, 2020] Michael Kommenda, Bogdan Burlacu, Gabriel Kronberger, and Michael Affenzeller. Parameter identification for symbolic regression using nonlinear least squares. *Genetic Programming and Evolvable Machines*, 21(3):471–501, 2020.

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

[La Cava *et al.*, 2016] William La Cava, Lee Spector, and Kourosh Danai. Epsilon-lexicase selection for regression. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, GECCO '16, page 741–748, New York, NY, USA, 2016. Association for Computing Machinery.

[La Cava *et al.*, 2021] William La Cava, Patryk Orzechowski, Bogdan Burlacu, Fabricio de Franca, Marco Virgolin, Ying Jin, Michael Kommenda, and Jason Moore. Contemporary symbolic regression methods and their relative performance. In J. Vanschoren and S. Yeung, editors, *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1. Curran, 2021.

[Landajuela *et al.*, 2021] Mikel Landajuela, Brenden K Petersen, Sookyung Kim, Claudio P Santiago, Ruben Glatt, Nathan Mundhenk, Jacob F Pettit, and Daniel Faissol. Discovering symbolic policies with deep reinforcement learning. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 5979–5989. PMLR, 18–24 Jul 2021.

[Landajuela *et al.*, 2022] Mikel Landajuela, Chak Shing Lee, Jiachen Yang, Ruben Glatt, Claudio P Santiago, Ignacio Aravena, Terrell Mundhenk, Garrett Mulcahy, and Brenden K Petersen. A unified framework for deep symbolic regression. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 33985–33998. Curran Associates, Inc., 2022.

[Liu and Tegmark, 2021] Ziming Liu and Max Tegmark. Machine Learning Conservation Laws from Trajectories. *Physical Review Letters*, 126(18):180604, May 2021.

[Liu and Tegmark, 2022] Ziming Liu and Max Tegmark. Machine-learning hidden symmetries. *Physical Review Letters*, 128(18):180201, May 2022.

[Meurer *et al.*, 2017] Aaron Meurer, Christopher P. Smith, Mateusz Paprocki, Ondřej Čertík, Sergey B. Kirpichev, Matthew Rocklin, AMiT Kumar, Sergiu Ivanov, Jason K. Moore, Sartaj Singh, Thilina Rathnayake, Sean Vig, Brian E. Granger, Richard P. Muller, Francesco Bonazzi, Harsh Gupta, Shivam Vats, Fredrik Johansson, Fabian Pedregosa, Matthew J. Curry, Andy R. Terrel, Štěpán Roučka, Ashutosh Saboo, Isuru Fernando, Sumith Kulal, Robert Cimrman, and Anthony Scopatz. Sympy: symbolic computing in python. *PeerJ Computer Science*, 3:e103, January 2017.

[Mundhenk *et al.*, 2021] T. Mundhenk, Mikel Landajuela, Ruben Glatt, Claudio Santiago, Daniel Faissol, and Brenden Petersen. Symbolic regression via neural-guided genetic programming population seeding. 11 2021.

[Petersen *et al.*, 2021] Brenden K Petersen, Mikel Landajuela Larma, Terrell N. Mundhenk, Claudio Prata Santiago, Soo Kyung Kim, and Joanne Taery Kim. Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients. In *International Conference on Learning Representations*, 2021.

[Quade *et al.*, 2016] Markus Quade, Markus Abel, Kamran Shafi, Robert K. Niven, and Bernd R. Noack. Prediction of dynamical systems by symbolic regression. *Physical Review E*, 94:012214, 2016.

[Schmidt and Lipson, 2007] Michael Schmidt and Hod Lipson. Comparison of tree and graph encodings as function of problem complexity. In *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation*, GECCO '07, page 1674–1679, New York, NY, USA, 2007. Association for Computing Machinery.

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

[Stephens, 2016] Trevor Stephens. Genetic programming in python, with a scikit-learn inspired api: gplearn, 2016.

[Strogatz, 2000] Steven H. Strogatz. *Nonlinear Dynamics and Chaos: With Applications to Physics, Biology, Chemistry and Engineering*. Westview Press, 2000.

[Tibshirani, 1996] Robert Tibshirani. Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society (Series B)*, 58:267–288, 1996.

[Truscott and Korns, 2011] Philip D. Truscott and Michael F. Korns. Detecting shadow economy sizes with symbolic regression. *Genetic Programming Theory and Practice IX*, pages 195–210, 2011).

[Udrescu and Tegmark, 2020] Silviu-Marian Udrescu and Max Tegmark. Ai feynman: A physics-inspired method for symbolic regression. *Science Advances*, 6(16):eaay2631, 2020.

[Udrescu *et al.*, 2020] Silviu-Marian Udrescu, Andrew Tan, Jiahai Feng, Orisvaldo Neto, Tailin Wu, and Max Tegmark. Ai feynman 2.0: Pareto-optimal symbolic regression exploiting graph modularity. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 4860–4871. Curran Associates, Inc., 2020.

[Virgolin *et al.*, 2021] Marco Virgolin, Tanja Alderliesten, Cees Witteveen, and Peter AN Bosman. Improving model-based genetic programming for symbolic regression of small expressions. *Evolutionary computation*, 29(2):211–237, 2021.

[Wang *et al.*, 2019] Yiqun Wang, Nicholas Wagner, and James M. Rondinelli. Symbolic regression in materials science. *MRS Communications*, 9(3):793–805, 2019.

[Wright, 1921] Sewall Wright. Correlation and causation. *Journal of Agricultural Research*, 20(3):557–585, 1921.