Call for Action: Towards the Next Generation of Symbolic Regression Benchmark

Guilherme Seidyo Imai Aldeia
guilherme.aldeia@ufabc.edu.br
Federal University of ABC
Santo André, São Paulo, BR

Hengzhe Zhang
hengzhe.zhang@ecs.vuw.ac.nz
Victoria University of Wellington
Wellington, NZ

Geoffrey Bomarito
geoffrey.f.bomarito@nasa.gov
NASA Langley Research Center
Hampton, Virginia, USA

Miles Cranmer
mc2473@cam.ac.uk
University of Cambridge
Cambridge, UK

Alcides Fonseca
me@alcidesfonseca.com
LASIGE, Faculdade de Ciências da Universidade de Lisboa
Lisboa, PT

Bogdan Burlacu
University of Applied Sciences Upper Austria
Upper Austria, AT

William G. La Cava
william.lacava@childrens.harvard.edu
Computational Health Informatics Program
Boston Children's Hospital
Harvard Medical School
Boston, Massachusetts, USA

Fabrício Olivetti de França
folivetti@ufabc.edu.br
Federal University of ABC
Santo André, São Paulo, BR

Abstract

Symbolic Regression (SR) is a powerful technique for discovering interpretable mathematical expressions. However, benchmarking SR methods remains challenging due to the diversity of algorithms, datasets, and evaluation criteria. In this work, we present an updated version of SRBench. Our benchmark expands the previous one by nearly doubling the number of evaluated methods, refining evaluation metrics, and using improved visualizations of the results to understand the performances. Additionally, we analyze trade-offs between model complexity, accuracy, and energy consumption. Our results show that no single algorithm dominates across all datasets. We propose a *call for action* from SR community in maintaining and evolving SRBench as a living benchmark that reflects the state-of-the-art in symbolic regression, by standardizing hyperparameter tuning, execution constraints, and computational resource allocation. We also propose deprecation criteria to maintain the benchmark's relevance and discuss best practices for improving SR algorithms, such as adaptive hyperparameter tuning and energy-efficient implementations.

CCS Concepts

• **Computing methodologies** → **Symbolic and algebraic algorithms**; • **General and reference** → **Empirical studies**; • **Human-centered computing** → *Visualization design and evaluation methods.*

Keywords

symbolic regression, benchmark, srbench

1 Introduction

Symbolic regression (SR) [35, 37] is a supervised machine learning technique that searches for a mathematical expression $\hat{f}(\mathbf{x}, \hat{\theta})$ that fits a set of points $\mathbf{x}_i \in \mathcal{X}, y_i \in \mathcal{Y}$ such as $y \approx \hat{f}(\mathbf{x}, \hat{\theta})$, where $\hat{\theta}$ is a vector of adjustable parameters. Searching for the function $\hat{f}$ adds a degree-of-freedom to data fitting that differs from traditional parametric models that starts from a fixed $f(x, \theta)$ and tries to find $\hat{\theta}$ that minimizes a loss function. Currently, many SR algorithms implement a variation of the original genetic programming (GP) as proposed by Koza [35, 36]. Since then, it has been successfully applied to many real-world applications, such as, scientific discovery in engineering [44], healthcare [41], physics [25], among others [10, 37, 53, 55, 60]. Its application in physics can be seen as "automating Kepler" —discovering analytical expressions through trial and error—, particularly evident in recent works that use SR to recover laws directly from experimental data [45], underscoring its potential as a tool for automated scientific reasoning.

Equation discovery is an NP-hard problem [62], and implementations rely on various approaches to solve it, such as GP [8, 10, 12, 15, 18, 20, 34, 39, 40, 51, 56, 61, 64], Neural Networks [4, 54], Transformers [29, 42, 57], Bayesian models [28], iterative methods [11, 31, 47], and even exhaustive approaches [3, 30]. In its original implementation, GP relied on generating ephemeral random constants as part of the expression, which could lead to sub-optimal exploration of the search space if these constants were not properly set. This was alleviated with the use of linear scaling [32] by scaling and translating the solution before measuring the loss function. Modern implementations rely on the optimization of the numerical parameters by either enforcing the creation of a model that is linear in the parameters [2, 8, 12, 15] or applying a non-linear optimization method to fit the parameters [18, 34].

Despite recent advances in the field of SR, La Cava et al. [38] observed a lack of consensus regarding a standardized set of datasets and benchmarking methodologies, making it challenging to accurately establish the current state-of-the-art (SotA). Another issue is the lack of a unified programming interface and clear instructions on installing and using each approach properly, creating barriers

to new experimentation and reproducibility. To address these challenges, SRBench [1] was proposed as a living benchmark for SR, comparing different SR implementations and standard ML regression models. This benchmark enforced the adoption of a common Python API and installation scripts within a *sandbox* environment, making it easier to install and evaluate different algorithms.

Running a benchmark is a laborious yet important task. Not only is it necessary to compare methods, but it also serves as a tool for measuring progress in the field. While the SRBench was a significant step toward comparing and understanding the current SotA in SR, designing effective benchmarks often requires further refinements and consideration of new factors to ensure their usefulness and longevity. An effective benchmark should contain diverse subclasses of problems, identifying which algorithms perform best in each scenario, mapping problems to algorithms rather than providing a one-size-fits-all solution. Additionally, the benchmark must challenge contemporary SR algorithms while remaining computationally feasible and representative of real-world challenges, ensuring that results extend beyond artificial test cases. SRBench falls short in these aspects, as it relies on more than 200 datasets without a detailed categorization of their specific challenges. Furthermore, it presents final results using only aggregated metrics, which can hide finer details in performance comparisons.

In this paper, we propose several improvements to the existing SRBench (hereafter referred to as SRBench 1.0), identifying challenges that must be addressed to develop a more refined version of the benchmark (SRBench 2.0). Additionally, this paper serves as a *call to action*, encouraging the community to engage in discussions on best practices for benchmarking and evaluating SR algorithms. Such involvement is crucial for the field to achieve a more mature state and drive further advancements.

The primary improvements proposed include: expanding the benchmark to incorporate new SR algorithms, bringing the total to 25; increasing the number of independent runs from 10 to 30 to enhance the statistical significance of the results; selecting 24 datasets, divided into two tracks: *(i)* black-box (drawn from SRBench 1.0) and *(ii)* phenomenological & first-principles (sourced from [10]); and refining the reporting methodology to provide a more detailed comparison of the nuances among top-performing algorithms, rather than simply ranking them and designating a single SotA method.

This *call to action* aims to highlight the challenges and issues encountered during benchmarking and to propose features that could streamline the process. Addressing these issues requires collaboration from method's authors/maintainers to develop and extend a common API that facilitates the expansion of SRBench in future iterations. To our knowledge, this is the largest study comparing SR methods under a unified experimental setup.

The paper is organized as follows: Section 2 reviews previously published benchmarks on symbolic regression, highlighting differences to our work. Section 3 presents our proposed update to SRBench, with each subsection addressing a key aspect of our update to the benchmark. Section 4 reports our results, using different levels of aggregation to gain deeper insights into the performance of individual algorithms. Section 5 discusses the results, and Section 6 presents our conclusions and call to action for advancing the benchmarking of symbolic regression methods.

[1] https://cavalab.org/srbench/

2 Related work

In 2018, Orzechowski et al. [50] surveyed many SR papers showing that methods were being benchmarked on simple problems with a lack of standardization, which could diminish the perception of SR achievements outside of its field. They also acknowledged previous efforts to conduct benchmarks for SR methods, then conducted an experiment with four SR algorithms comparing with different ML methods across more than 100 datasets of up to 1000 samples.

La Cava et al. [38] followed up the benchmark efforts and proposed SRBench, a joint-effort to achieve a large scale comparison of 14 modern SR algorithms comprising GP-based, deterministic, and deep learning based approaches. In this paper, the authors mention that determining a state-of-the-art should not be the sole focus of research — however, promising avenues of investigation cannot be well-informed without empirical evidence. In SRBench, the authors increased the number of problems and their dimensionalities, using datasets from PMLB [49, 52], an open-source library for benchmarking ML methods. A new ground-truth track was also added with more than 100 physics equations from Feynman lectures [21, 22] with data generated synthetically. They provided a more informative view of the current state of the field. Given the large number of problems and algorithms, they performed 10 individual runs for each dataset, and the budget was based on evaluations — which is hard to measure uniformly across the variety of different approaches with different computational runtime.

Further discussions emerged in top-tier conferences related to SR (such as GECCO), with criticisms about aggregated views of the results [16, 17]. de Franca et al. [14] benchmarked a smaller selection of SR algorithms, including more recent approaches, on a proposed set of artificially generated datasets to measure the performance in specific tasks: overall accuracy performance, robustness to noise, capabilities of selecting the relevant features, extrapolation performance, and interpretability. With a smaller selection of algorithms and datasets it was possible to make precise statements of the current state of SR and the current weak points that should be addressed in the future. The authors also introduced a runtime limit instead of using the number of evaluations, stimulating the efficient implementation of the algorithms. The authors noticed that even though SR shares many shortcomings with traditional ML models, they have a potential to overcome it using domain knowledge and allowing a customized experience to the user [27]. Regarding the benchmarks, the authors feel that there is still room for improvements on how to craft datasets that correctly mimic the challenges faced in the real-world.

Matsubara et al. [46] noticed that the Feynman dataset [60] used in the SRBench's ground-truth track did not adequately depict the original physical phenomena because of how the values were sampled –— the data generation was originally done by uniformly sampling standardized values. They proposed to replace the synthetic data generation process with normal or logarithmic distributions that mimic what we would empirically observe if we were to collect the data in practice, and evaluated six different SR methods.

Žegklitz and Pošík [63] proposes a benchmark with open-source and ease-of-use principles. They evaluated four SR methods on five synthetic and four real-world problems. Abdalla et al. [1] benchmarks only five SR methods focusing on benchmarking SR for learning equations in civil and construction engineering, where equations are traditionally derived using linear regression.

Thing and Koksbang [58] proposed an unified tool for benchmarking SR in cosmology, including 12 SR methods and 28 datasets representing cosmological and astroparticles physics problems, finding that most methods performed poorly in the benchmark. They argue that using standard datasets (*i.e.* PMLB) is prudent for academic comparisons, but these ensembles of datasets do not provide a representative measure for specific problems. They conducted off-the-shelf evaluations, which are useful because they do not require extra effort from the user, and, to improve reproducibility, they package the 12 methods into Docker containers.

3 Methods

3.1 Dataset selection

For this version of the benchmark, we propose two distinct tracks: *black-box* and *phenomenological & first-principles*. Table 1 lists the datasets used in our benchmark, along with the number of samples (# Rows), features (# Cols), and the range of values for the target feature (codomain). For the latter track, all codomains consist of real values, so instead we provide the data sources for these datasets. All datasets contain no missing values, have only numeric features, and are standardized before training.

Table 1: Metadata for each benchmark track. Dataset names match their corresponding PMLB entries

Black-box	**# Rows**	**# Cols**	**Codomain**
1028_SWD	1000	11	$\mathbb{Z}^+$
1089_USCrime	47	14	$\mathbb{Z}^+$
1193_BNG_lowbwt	31104	10	$\mathbb{R}^+$
1199_BNG_echoMonths	17496	10	$\mathbb{R}$
192_vineyard	52	3	$\mathbb{R}^+$
210_cloud	108	6	$\mathbb{R}^+$
522_pm10	500	8	$\mathbb{R}^+$
557_analcatdata_apnea1	475	4	$\mathbb{Z}^+$
579_fri_c0_250_5	250	6	$\mathbb{R}$
606_fri_c2_1000_10	1000	11	$\mathbb{R}$
650_fri_c0_500_50	500	51	$\mathbb{R}$
678_visualizing_environmental	111	4	$\mathbb{R}^+$
Phenomenological & first-principles	**# Rows**	**# Cols**	**Data source**
first_principles_absorption	14	2	Russeil et al. [53]
first_principles_bode	8	2	Bonnet [5]
first_principles_hubble	32	2	Hubble [26]
first_principles_ideal_gas	30	4	Generated, 10% noise
first_principles_kepler	6	2	Kepler [33]
first_principles_leavitt	26	2	Leavitt and Pickering [43]
first_principles_newton	30	4	Generated, 10% noise
first_principles_planck	100	3	Generated, 10% noise
first_principles_rydberg	50	3	Generated, 1% noise
first_principles_schechter	27	2	Generated, 20% noise
first_principles_supernovae_zr	236	2	Russeil et al. [53]
first_principles_tully_fisher	18	2	Tully and Fisher [59]

The black-box track is a selection of 12 regression datasets from the PMLB 1.0 project [52]. To make this selection, we first excluded datasets where all previously benchmarked algorithms achieved $R^2 > 0.99$, or if a simple linear regression was also capable of achieving this same R^2 threshold. Next, we created a numerical feature vector for each dataset containing the number of samples, number of features, and the performance of each algorithm tested in the SRBench 1.0. We then applied the t-SNE algorithm to reduce this metadata of the datasets to two dimensions and created 12 clusters using the k-means algorithm over the latent encodings. Finally, we picked the dataset closest to each centroid. The final selection of datasets consists of different combinations of horizontal and vertical dimensionality and, as a byproduct, different codomains —an important factor in proper model fitting, as these datasets may deviate from the traditional assumption of a normal distribution, requiring algorithms to adapt accordingly.

We should also note that in the original SRBench, half of the black-box problems were variations of the Friedman datasets [23], which introduced bias in the results, as reported by de França [16]. To mitigate this issue, we restricted the selection of these datasets, allowing a maximum of 25% of the chosen datasets to belong to this class. The goal is to ensure that the datasets provide a diverse representation of SR applications while enabling fine-grained analyses to better understand why certain methods outperform others.

The phenomenological & first-principles track includes publicly available datasets from Russeil et al. [53] and Cranmer [10]. The former provides real-world measurements used to derive *empirical relationships*, while the latter source data underlying well-known first-principles equations. All these datasets are particularly challenging because of the small number of points and the diverse and unknown natural distribution of the phenomena.

3.2 Benchmarked methods

In this benchmark, we have extended the list of methods from the original SRBench adding recently published methods, reported in Table 2, highlighting whether the algorithm supports parameter optimization, a time-limit argument[2], whether it returns multiple solutions (*e.g.* a Pareto Front (PF)) or just a single solution, the hardware the algorithm runs on, whether grid search to finetune the hyper-parameters is possible, and the programming language used in the implementation. For methods implemented in other programming languages, the authors also implemented Python bindings to facilitate their use.

A grid search was performed for all methods, except for three —Bingo, Brush, and GP-GOMEA— due to incompatibility with scikit-learn's `GridSearchCV` interface at the time. We adopted the grid search settings from the original SRBench and defined a new search space for methods that lacked one. An exception is FFX, which has no tunable hyperparameters. Additionally, we included the default (off-the-shelf) configuration as one of the candidate settings, since preliminary results suggest that, for some problems, is outperforms tuned configurations. For further details, refer to the online resources in Section 3.5.

[2]In case it does not, we adjusted the hyperparameters to reasonable values to stop at an approximate time limit.

Table 2: Algorithms evaluated, their original references, and relevant characteristics pertinent to benchmarking.

Algorithm	Const. Opt.	Time limit	Multiple solutions	Runs on	Language	Description
AFP [56]	✗	✓	✗	CPU	C++	Age-fitness Pareto (AFP) optimization, meaning model age is used as an objective, with constants randomly changed
AFP_fe	✗	✓	✗	CPU	C++	AFP with co-evolved fitness estimates
AFP_ehc [39]	✓	✓	✗	CPU	C++	AFP with epigenetic hill climbing for constants optimization as local search
Bingo [51]	✓	✓	PF	CPU	Python	Evolves acyclic graphs with non-linear optimization, using islands for managing parallel populations
Brush [9]	✓	✓	PF	CPU	C++	GP with multi-armed bandits for controlling search space exploration
BSR [28]	✗	✓	✗	CPU	Python	Bayesian model with priors for operators and coefficients is used to sample expression trees
E2E [29]	✗	✗	✗	GPU	Python	Generator using pre-trained transformers, using BFGS and subsampling for tuning parameters
EPLEX [40]	✗	✓	✗	CPU	C++	GP with ϵ-lexicase parent selection
EQL [54]	✗	✗	✗	CPU	Python	Shallow neural network using mathematical operators as activation functions, and performs a pruning to refine the network to an expression
FEAT [8]	✓	✓	PF	CPU	C++	GP algorithm with ϵ-lexicase selection and linear combination of expressions using L1-OLS
FFX [47]	✓	✗	✗	CPU	Python	Non-evolutionary, deterministic approach, that generates a set of base functions and fits a regularized OLS to combine them
Genetic Engine [20]	✗	✓	✓	CPU	Python	GP using Context-Free Grammars to guide the generation process in an efficient manner
GPGomea [61]	✓	✗	✓	CPU	C++	GP with linkage learning used to propagate patterns and avoid their disruption
GPlearn	✗	✗	✓	CPU	Python	Canonical GP implementation
GPZGD [18]	✓	✓	✗	CPU	C	GP with Z-score standardization and stochastic gradient descent for parameter optimization
ITEA [12]	✓	✗	✗	CPU	Haskell	Mutation-based algorithm with constrained representation and OLS parameter optimization
NeSymRes [4]	✓	✓	✗	GPU	Python	Pre-trained encoder-decoders generate equation skeletons, optimized with non-linear optimization
Operon [34]	✓	✓	PF	CPU	C++	GP algorithm with weighted terminals and non-linear OLS parameter optimization
Ps-Tree [64]	✗	✗	✗	CPU	Python	GP algorithm that evolves Piecewise trees with SR expressions as leaves
PySR [10]	✓	✓	✗	CPU	Julia	Evolve-simplify-optimize loop with islands to manage parallel populations
Qlattice [6]	✓	✓	✗	CPU	Python	Uses a learned probability distribution updated over iterations to sample expressions, with parameter optimization. Closed source software
Rils-rols [31]	✓	✓	✗	CPU	C++	Iterative generation of perturbations and parameter optimization with OLS and local search selection of next candidates
TIR [15]	✓	✓	PF	CPU	Haskell	GP with crossover and mutation that uses a constrained representation capable of tuning non-linear parameters with OLS
TPSR [57]	✓	✓	✗	GPU	Python	Monte-Carlo Tree Search planning with non-linear optimization wrapper for generative models E2E and NeSymRes
uDSR [42]	✓	✗	✗	GPU	Python	Unification of pre-trained transformers, GP, and linear models, into a framework that decomposes the problem

Hyper-parameter optimization is performed before each run using a 3-fold cross-validation on the training data (75%) and selecting the best configuration based on the average R^2 score. The model is then trained with optimal hyperparameters using the entire training data, and evaluated on a held-out test data (25%). We implemented a *Python* wrapper to tune every method into the same grid search pipeline using *scikit-learn*.

3.3 Performance analysis

We repeated these experiments 30 times for each dataset with fixed and distinct seeds, storing all necessary information to process the aggregated results. For SRBench 2.0, we adopted the *performance profile plot* [19], which describes the empirical distribution of the obtained results. This plot illustrates the probability of achieving a performance greater than or equal to a given R^2 threshold for all possible thresholds. In this plot, the x-axis represents a threshold value of the R^2 and the y-axis the percentage of runs that a particular algorithm obtained that value or higher. This plot can give a broader view of the likelihood of successfully achieving a high accuracy for each algorithm, while keeping all the information about each algorithm in the same plot. Using this same plot, we can calculate the area under the curve (AUC) for each individual algorithm as an aggregated measure: a value of 1.0 indicates that all 30 runs achieved the maximum $R^2 = 1$, whereas a value of 0.0 indicates that all runs resulted in $R^2 \leq 0$. The AUC provides an estimate of the method's potential to achieve high performance across multiple runs. The performance plot can be generated for each individual dataset as a result of the 30 independent runs or we can plot the aggregated performance over all datasets using an aggregation function (*e.g.*, mean, median, max, min) for each dataset. In this paper, we use the maximum value as the aggregation function, which reflects the

best-case performance of each algorithm across the 30 runs. The project website reports both individual and aggregated plots.

To measure model size, we convert the models to a SymPy [48] compatible expression and count the number of nodes. This is necessary since the internal complexity measures of each algorithm may differ from the standard way of counting nodes. The expression was not simplified (except for trivial simplifications such as constant merging) before counting the nodes, as simplification using SymPy is unreliable and can increase the model size in the process [13].

3.4 Benchmark infrastructure

Conducting extensive experiments across multiple algorithms and datasets requires substantial computational resources. Variability in hardware workload or non-homogeneous cluster nodes can make it challenging to establish a fixed computational budget or ensure that all methods are evaluated under identical conditions. Using fixed resources can help mitigate the "hardware lottery" effect [24], where an approach succeeds due to its compatibility with the available software and hardware rather than its inherent merit.

Experiments were executed on a single computing cluster. Each job was allocated 10 GB of RAM, and GPUs were provided for methods that support them (see the *runs on* column in Table 2). Jobs were not allowed to spawn subprocesses or utilize multiple cores. The time budget was set to 6 hours for hyperparameter search and 1 hour for training the algorithm with the optimal configuration, after which a `SIGALRM` signal was sent to terminate the process.

Energy consumption was estimated after hyperparameter tuning to avoid additional overhead, using the *eco2AI* library [7], which derives estimates from standard Linux commands monitoring CPU and memory usage. While not exact, this provides a rough yet informative profile of resource usage.

3.5 Data Availability

The datasets were submitted to PMLB [49, 52]. We released all experiment scripts and parameters to ensure transparency and reproducibility. We also restructured SRBench using containerized environments to ensure consistent execution across systems and simplify standalone use. The project is available at https://github.com/cavalab/srbench/tree/srbench_2025.

4 Results

Within the specified time budget, the total runtime on a single core would be 1 year, 43 weeks, and 5 days. Using eco2AI to monitor the final training phase, we report the energy and time consumption in Fig. 1. We notice that these measurements are influenced not only by the choice of programming language and implementation details but also by the internal decisions of the algorithms to terminate execution before reaching the maximum time.

In Figure 2, we present the performance plot with the AUC values based on the highest empirically observed R^2 on the test set across 30 runs, representing an optimistic perspective on model performance, assuming a user performs the same number of runs. The plot shows the empirically observed likelihood of obtaining a maximum R^2 score for black-box problems under sufficient repetitions. The probability on the y-axis, $\mathrm{P}[R^2 \geq x]$, is estimated by

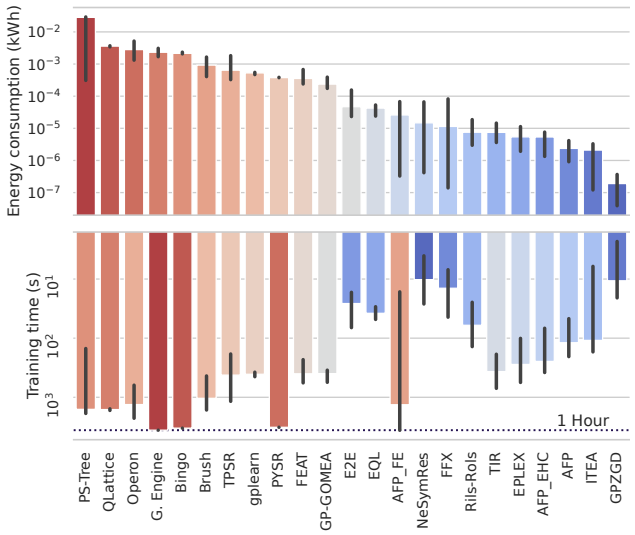

Figure 1: Median energy consumption (kWh) and training runtime for each algorithm.

calculating the proportion of runs where the maximum R^2 exceeds a given threshold x.

Figure 3 shows a cluster map of the AUC from the performance plot for each pair of dataset and algorithm on the test set. The size of each cell is proportional to the size of the best-performing final expression across the 30 runs. Higher values and smaller cells indicate better performance. At the top of this plot, we can observe a hierarchical clustering of the algorithms, grouping the methods based on performance, and the datasets based on their difficulty to solve. The best algorithm for each dataset is highlighted with a black edge color on its cell.

For the phenomenological & first-principles track, Figure 4 displays the Pareto front on the test set of the most accurate solutions obtained by SR algorithms and compared to the current accepted hypothesis (*star*). We observe that noise —whether inherent in real-world data or synthetically added in the case of generated data— can cause the ground-truth hypotheses to fall short of achieving a perfect R^2. Asterisks next to the algorithm names indicate those closest (based on Euclidean distance over the normalized axes) to the ground-truth. The final equations closest to the governing models are shown in Table 3 along with their corresponding R^2 scores and sizes.

5 Discussion

5.1 Energy consumption and execution time

Energy consumption profiles in Figure 1 reveal distinct patterns for different algorithms.

The top two most energy-hungry algorithms are Python-based, and the third, Operon, is implemented in C++ but includes an inner hyperparameter tuning step. Some methods, such as FEAT and Brush, implement a `max_stall` parameter, which halts execution

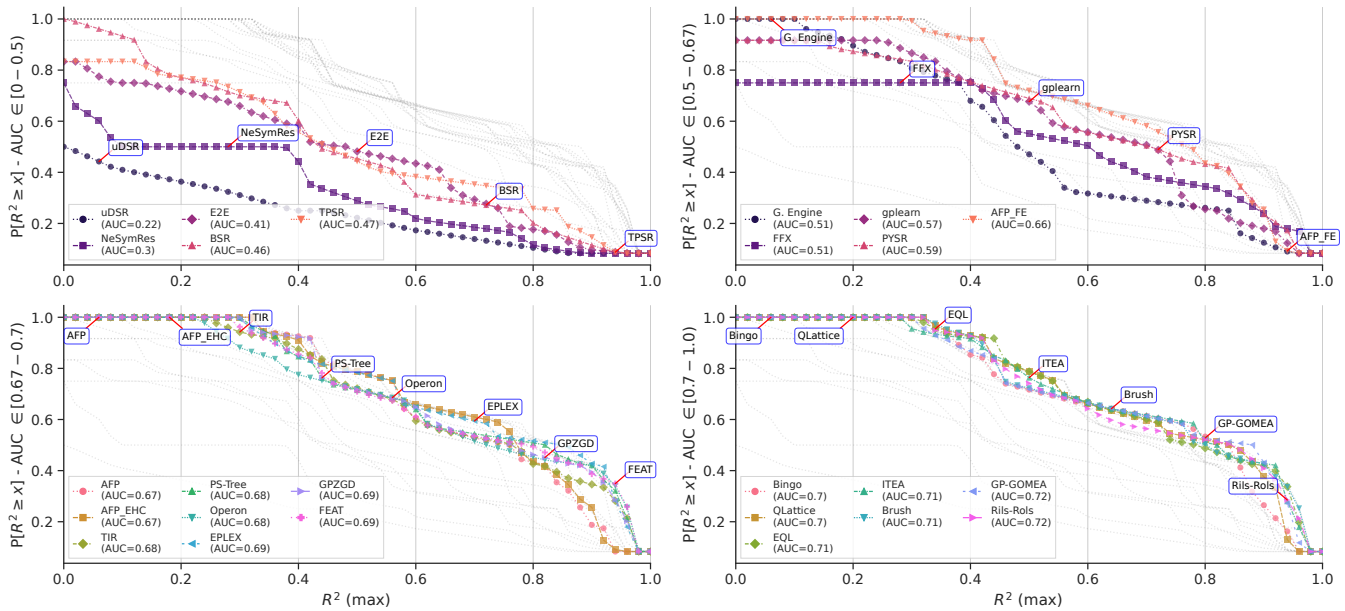

Figure 2: Performance plots for the black-box track, where the lines represent the probability of obtaining a given empirically observed R^2 value when running the experiments multiple time (i.e., max aggregation).

Table 3: Equation from the Pareto front closest to the governing models.

Dataset	R^2	Size	Symbolic model
absorption	1.0	6	$0.24 + 1.76 \tanh x$
bode	1.0	8	$0.34e^{1.51 \cdot n} - 0.87$
hubble	0.86	3	$0.090 + D$
ideal_gas	0.99	14	$0.69 \cdot \log n + 1.64 - 0.89 + 0.48 * e^{-V}$
kepler	1.00	10	$1.98 * \tanh(0.66 \cdot a - 0.56) + 0.78$
leavitt	0.97	3	$-0.94 \cdot \log P$
newton	0.90	16	$0.34 \cdot \frac{m_1}{0.83 \cdot m_2 - \frac{0.09}{m_1}} + 1.13$
planck	1.00	24	$0.023 \cdot \log(\sqrt{nu + 0.38} - 0.04) - 0.31 \cdot \frac{nu+0.3}{T+0.94} + 0.41$
rydberg	1.00	21	$1.01 \cdot e^{0.1 \cdot e^{1.73 \cdot n_1 - 1.31 \cdot n_2}} - e^{-0.69 \cdot n_1}$
schechter	1.00	13	$0.748 - 0.274 \cdot (\log(163.992 \cdot L + 106.737) + 1.414 \cdot L)$
supernovae_zr	1.00	29	$(\sin(0.2 \cdot x \cdot e^{-x}) - 0.04) \cdot (x + 0.72 \cdot \sin(5.46 \cdot x + 0.76) - 0.16) - \sin(x + 0.18)$
tully_fisher	0.93	3	$-0.93 \cdot \Delta V$

if no improvement is observed in an inner validation partition after a number of iterations. While this feature does not necessarily correlate with these methods being the best performers in terms of energy consumption, it could be further explored for more computationally demanding algorithms.

GPU-based algorithms, such as TPSR, E2E, and NeSymRes, do not show much difference in energy consumption compared to CPU-based algorithms. In particular, TPSR, which uses traditional parameter optimization methods, demonstrated higher execution times than other GPU-based methods. uDSR was not compatible with the library during the experiments.

5.2 Performance analysis of the *black-box* track

The performance curves in Figure 2 displays the expected probability of achieving maximum performance at different R^2 thresholds across a variety of problems. Some methods (shown in the top-left subplot, with AUC below 0.5) exhibit a probability lower than 1.0 of achieving a strictly positive R^2, suggesting that they may struggle to generalize; fail to capture intrinsic data structures, or produce low-quality models.

Most of the top-performing algorithms incorporate constant optimization as a local search step —using linear, gradient-based, or non-linear methods–, highlighting the importance of parameter optimization for achieving high performance. Among the highest AUC scores are GP-based algorithms such as Brush, GP-GOMEA, and GPZGD. Notably, GPZGD is a GP algorithm that standardizes the features using Z-score and performs parameter optimization. Other methods, such as Rils-Rols, also performs constant optimization.

The selection of datasets in Figure 3 reveals the "*no free lunch*" aspect of the current state of SR — there is no algorithm that performs exceptionally well on all datasets. Likewise, there is no algorithm that fails to solve all problems. Hierarchical clustering shows two main clusters of good- and bad-performing algorithms based on the AUC of the performance curves. A closer inspection of Figure 3 allows for a better understanding of challenges for existing algorithms.

Furthermore, the blue cells reveals a cluster of datasets in which the top-performing algorithms obtain a similarly good result. At the same time, we can readily see the trade-off between performance AUC and size. For example, in the *557* dataset, we can see that FFX

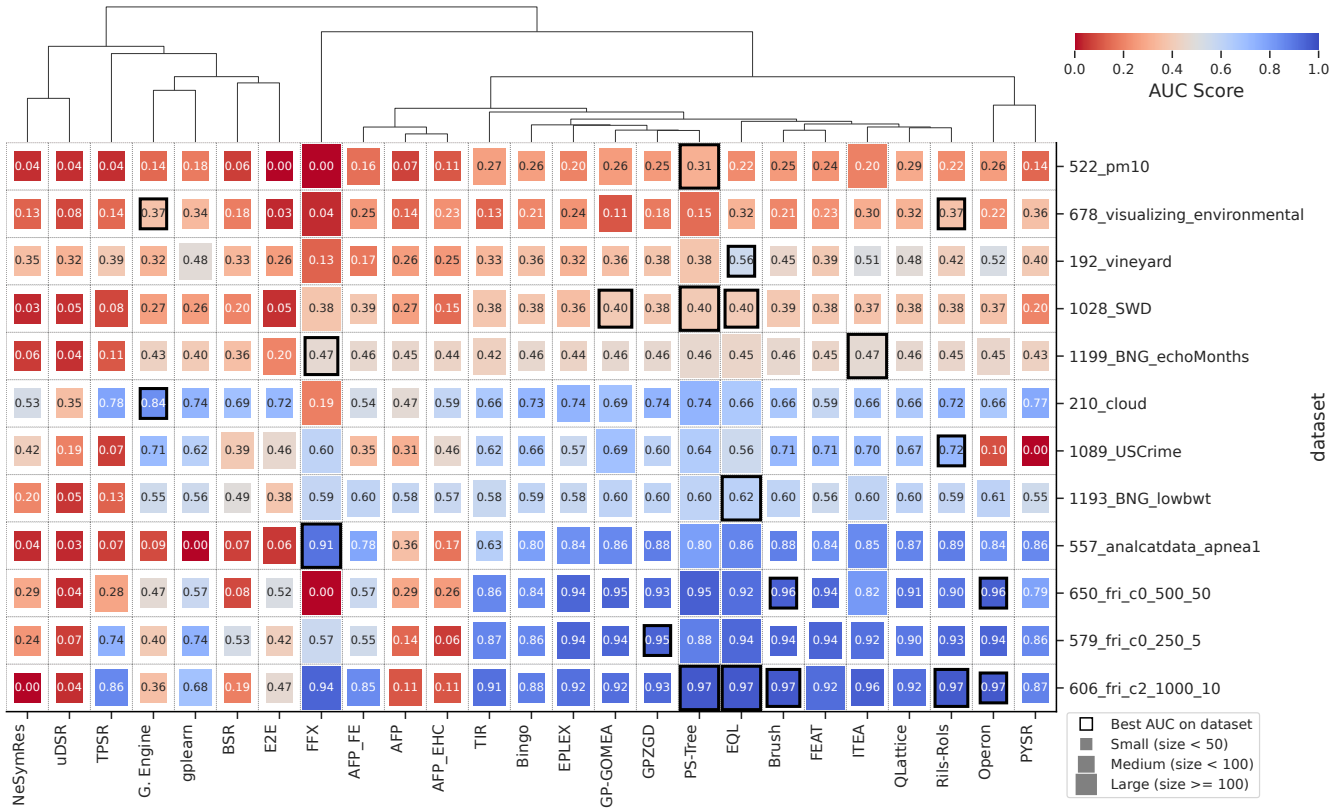

Figure 3: Cluster map of the Area Under the Curve (AUC) of Expected Performances across the 30 independent runs is segregated by algorithm and dataset. Higher values indicate better performance, while larger cells represent worse model size.

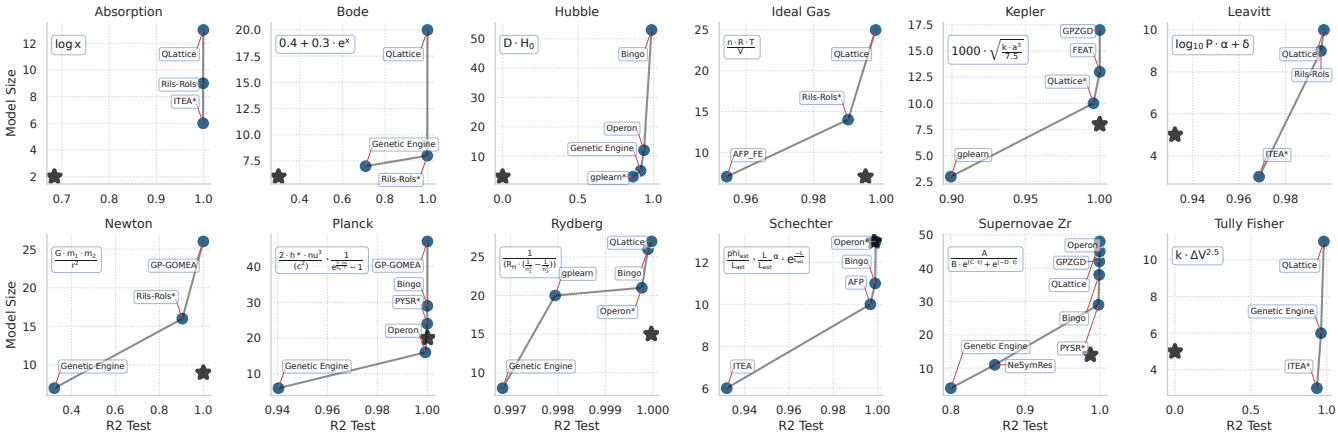

Figure 4: Pareto plots for the phenomenological & first-principles track, with model sizes on y axis, and R^2 on x axis. The *star* marker denotes the ground-truth expression performance, also denoted in the box inside each subplot. Only the first front is plotted for clarity.

obtained the best AUC score, while Rils-Rols obtained a slightly worse AUC result but with a smaller expression. Interestingly, some methods that perform poorly overall still achieve the best results on specific datasets. This highlights that no single algorithm dominates across all objectives; each has its own strengths depending on the problem characteristics.

5.3 Towards a parameter-less experience

Setting the hyper-parameters of SR algorithms often requires careful experimentation with a number of different settings that may affect the final result. For general users, this creates a significant barrier: they must either rely on default values or conduct extensive experimentation to find the best configurations. The field should move toward a parameter-less experience by developing algorithms that require as few hyperparameters as possible, unless they are intuitive and domain-related. Good practices involve carefully selecting default parameters and, if possible, integrating internal hyperparameter tuning (such as Operon with Optuna) or making parameters adaptive (as in PySR).

We propose standardizing the hyperparameter tuning by requiring each method to define a small set (*e.g.*, four) of hyperparameter configurations. A simple grid search is then applied using these configurations, each limited to a maximum runtime (*e.g.*, one hour). The off-the-shelf configuration should automatically be included by the experiment scripts, as we observed some methods perform best with their default settings.

5.4 Solving empirical and theoretical problems

The phenomenological & first-principles track provides a detailed view of the trade-off between model size and accuracy, using established models as reference points. Fig. 4 shows that, for almost every problem, no SR method could discover an expression that strictly dominates the original equation. This occurs because methods either generate a larger expression with higher accuracy or a smaller expression with lower accuracy. Table 3 further illustrates this challenge — only ITEA successfully retrieved one of the first-principles equations (Leavitt). In other cases, SR methods either produced a more complex yet more accurate expression or a simpler but less precise model. For the *Schechter* dataset, Operon found reasonable alternatives without significantly deviating from the complexity-accuracy balance of the original equation.

We notice that some first-principle equations fails to achieve perfect R^2 due to noise in the original data — notably in the Hubble and Tully-Fisher datasets. In such cases, obtaining a higher R^2 with a more complex model often indicates that the model is fitting the noise rather than the underlying relationship. This is evident in the equations found for Hubble and Tully-Fisher, which deviate from the ground-truth equation by only a small edit distance.

In contrast, the Absorption and Bode datasets represent phenomenological problems with no known ground-truth equations. A possible venue for future investigation, is the incorporation of the uncertainties information of the data, leading to a better measurement for the model accuracy.

5.5 Current difficulties and limitations

During the benchmarking process, a number of bugs had to be fixed for most evaluated algorithms. Few of the corresponding repositories are being maintained, requiring a joint decision of which algorithm to deprecate and remove from further benchmarking.

We propose establishing deprecation rules to maximize benchmarking efforts. A simple criterion could be: if an algorithm's repository is no longer actively maintained and it does not appear on at least one of the Pareto fronts in the black-box datasets, it should be deprecated. This guideline may also encourage the development of more robust algorithms that perform consistently well across diverse problems.

Another important point is the fair comparison between GPU-based and CPU-based approaches, as well as single-core versus multi-core implementations. While every algorithm with a `max_time` hyperparameter respects its limit, some terminate far earlier (*e.g.*, NeSymRes), potentially skewing performance evaluations. A more precise implementation of time management would allow for a fairer comparison of computational efficiency across different methods.

Additionally, this benchmark could provide significantly more data-driven insights for algorithm development by incorporating appropriate metadata (*e.g.*, a set of keywords), offering a direct way to link methodologies to results. Similarly, algorithms could generate post-run metadata (*e.g.*, results of internal grid search, number of local search iterations) as part of their output. This would facilitate a deeper understanding of algorithmic behavior and enable more precise fine-tuning of specific methods, ultimately leading to more informed improvements in SR techniques.

6 Conclusions and future work

This paper updates the current SRBench aiming to improve the benchmark of symbolic regression as a community accepted standard. We increased to almost twice the number of SR algorithms than the previous version, and provided an alternative and more detailed visualizations depicting the current state of SR.

To ensure a constant improvement of SRBench, we need active participation of the community to ensure compatibility with the current experiments and a correctly working implementation. We *call for action* from researchers to actively contribute to discussions, share new ideas, provide constructive criticism, and propose corrections. A continuously evolving benchmark will help drive progress in SR. Achieving a mature state for the field should be a shared priority, especially given the growing emphasis on transparency in machine learning and the increasing interest in scientific ML.

Acknowledgments

To everyone involved in the discussions to improve SRBench. To PMLB maintainers for the help. To all past researchers who somehow enabled us to use data or methods from their experiments, making science open.

Alcides Fonseca is supported by FCT through the LASIGE Research Unit, ref. UID/000408/2025. Fabricio Olivetti de Franca is supported by Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) grant 301596/2022-0. Guilherme Imai Aldeia is supported by Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) finance Code 001 and grant 88887.802848/2023-00.

References

[1] Jamal A Abdalla, MZ Naser, Saleh M Alogla, Alireza Ghasemi, and Ahmad Naser. 2024. Evaluation and Benchmarking of Performance of Machine Learning and Symbolic Regression: Datasets, Software Tools and Prediction Models. *ES General* 7 (2024), 1352.

[2] Ignacio Arnaldo, Krzysztof Krawiec, and Una-May O'Reilly. 2014. Multiple regression genetic programming. In *Proceedings of the 2014 annual conference on genetic and evolutionary computation*. 879–886.

[3] Deaglan J. Bartlett, Harry Desmond, and Pedro G. Ferreira. 2024. Exhaustive Symbolic Regression. *IEEE Transactions on Evolutionary Computation* 28, 4 (Aug. 2024), 950–964. doi:10.1109/tevc.2023.3280250

[4] Luca Biggio, Tommaso Bendinelli, Alexander Neitz, Aurelien Lucchi, and Giambattista Parascandolo. 2021. Neural Symbolic Regression that scales. In *Proceedings of the 38th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 139)*, Marina Meila and Tong Zhang (Eds.). PMLR, 936–945. https://proceedings.mlr.press/v139/biggio21a.html

[5] C. Bonnet. 1764. *Contemplation de la nature*. Number v. 2 in Contemplation de la nature. M.M. Rey. https://books.google.com/books?id=Sm8GAAAAQAAJ

[6] Kevin René Broløs, Meera Vieira Machado, Chris Cave, Jaan Kasak, Valdemar Stentoft-Hansen, Victor Galindo Batanero, Tom Jelen, and Casper Wilstrup. 2021. An Approach to Symbolic Regression Using Feyn. arXiv:2104.05417 [cs.LG] https://arxiv.org/abs/2104.05417

[7] S. A. Budennyy, V. D. Lazarev, N. N. Zakharenko, A. N. Korovin, O. A. Plosskaya, D. V. Dimitrov, V. S. Akhripkin, I. V. Pavlov, I. V. Oseledets, I. S. Barsola, I. V. Egorov, A. A. Kosterina, and L. E. Zhukov. 2022. eco2AI: Carbon Emissions Tracking of Machine Learning Models as the First Step Towards Sustainable AI. *Doklady Mathematics* 106, S1 (Dec. 2022), S118–S128. doi:10.1134/s1064562422060230

[8] William La Cava, Tilak Raj Singh, James Taggart, Srinivas Suri, and Jason Moore. 2019. Learning concise representations for regression by evolving networks of trees. In *International Conference on Learning Representations*. https://openreview.net/forum?id=Hke-JhA9Y7

[9] CavaLab. 2025. Brush: An Interpretable Machine Learning Library. https://github.com/cavalab/brush/tree/multi_armed_bandits Accessed: 2025-03-29.

[10] Miles Cranmer. 2023. Interpretable Machine Learning for Science with PySR and SymbolicRegression.jl. arXiv:2305.01582 [astro-ph.IM] https://arxiv.org/abs/2305.01582

[11] Fabrício Olivetti de França. 2018. A greedy search tree heuristic for symbolic regression. *Information Sciences* 442 (2018), 18–32.

[12] F. O. de Franca and G. S. I. Aldeia. 2021. Interaction–Transformation Evolutionary Algorithm for Symbolic Regression. *Evolutionary Computation* 29, 3 (09 2021), 367–390. doi:10.1162/evco_a_00285 arXiv:https://direct.mit.edu/evco/article-pdf/29/3/367/1959462/evco_a_00285.pdf

[13] Fabricio Olivetti de Franca and Gabriel Kronberger. 2023. Reducing Overparameterization of Symbolic Regression Models with Equality Saturation. In *Proceedings of the Genetic and Evolutionary Computation Conference* (Lisbon, Portugal) *(GECCO '23)*. Association for Computing Machinery, New York, NY, USA, 1064–1072. doi:10.1145/3583131.3590346

[14] F. O. de Franca, M. Virgolin, M. Kommenda, M. S. Majumder, M. Cranmer, G. Espada, L. Ingelse, A. Fonseca, M. Landajuela, B. Petersen, R. Glatt, N. Mundhenk, C. S. Lee, J. D. Hochhalter, D. L. Randall, P. Kamienny, H. Zhang, G. Dick, A. Simon, B. Burlacu, Jaan Kasak, Meera Machado, Casper Wilstrup, and W. G. La Cavaz. 2024. SRBench++: Principled Benchmarking of Symbolic Regression With Domain-Expert Interpretation. *IEEE Transactions on Evolutionary Computation* (2024), 1–1. doi:10.1109/TEVC.2024.3423681

[15] Fabrício Olivetti de França. 2022. Transformation-interaction-rational representation for symbolic regression. In *Proceedings of the Genetic and Evolutionary Computation Conference* (Boston, Massachusetts) *(GECCO '22)*. Association for Computing Machinery, New York, NY, USA, 920–928. doi:10.1145/3512290.3528695

[16] Fabrício Olivetti de França. 2023. Transformation-Interaction-Rational Representation for Symbolic Regression: A Detailed Analysis of SRBench Results. *ACM Trans. Evol. Learn. Optim.* 3, 2, Article 7 (June 2023), 19 pages. doi:10.1145/3597312

[17] Grant Dick. 2022. Genetic programming, standardisation, and stochastic gradient descent revisited: Initial findings on srbench. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. 2265–2273.

[18] Grant Dick, Caitlin A. Owen, and Peter A. Whigham. 2020. Feature standardisation and coefficient optimisation for effective symbolic regression. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference* (Cancún, Mexico) *(GECCO '20)*. Association for Computing Machinery, New York, NY, USA, 306–314. doi:10.1145/3377930.3390237

[19] Elizabeth D Dolan and Jorge J Moré. 2002. Benchmarking optimization software with performance profiles. *Mathematical programming* 91 (2002), 201–213.

[20] Guilherme Espada, Leon Ingelse, Paulo Canelas, Pedro Barbosa, and Alcides Fonseca. 2022. Datatypes as a More Ergonomic Frontend for Grammar-Guided Genetic Programming. In *GPCE '22: Concepts and Experiences, Auckland, NZ, December 6 - 7, 2022*, Bernhard Scholz and Yukiyoshi Kameyama (Eds.). ACM, 1.

[21] R.P. Feynman, R.B. Leighton, and M.L. Sands. 2006. *The Feynman Lectures on Physics*. Number vol. 2 in The Feynman Lectures on Physics. Pearson/Addison-Wesley. https://books.google.com.br/books?id=AbruAAAAMAAJ

[22] R.P. Feynman, R.B. Leighton, and M. Sands. 2015. *The Feynman Lectures on Physics, Vol. I: The New Millennium Edition: Mainly Mechanics, Radiation, and Heat*. Number vol. 1 in The Feynman Lectures on Physics. Basic Books. https://books.google.com.br/books?id=d76DBQAAQBAJ

[23] Jerome H. Friedman. 2001. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics* 29, 5 (Oct. 2001). doi:10.1214/aos/1013203451

[24] Sara Hooker. 2020. The Hardware Lottery. arXiv:2009.06489 [cs.CY] https://arxiv.org/abs/2009.06489

[25] Viktor Hruška, Aneta Furmanová, and Michal Bednařík. 2025. Analytical formulae for design of one-dimensional sonic crystals with smooth geometry based on symbolic regression. *Journal of Sound and Vibration* 597 (2025), 118821. doi:10.1016/j.jsv.2024.118821

[26] Edwin Hubble. 1929. A relation between distance and radial velocity among extra-galactic nebulae. *Proceedings of the National Academy of Sciences* 15, 3 (March 1929), 168–173. doi:10.1073/pnas.15.3.168

[27] Leon Ingelse and Alcides Fonseca. 2023. *Domain-Aware Feature Learning with Grammar-Guided Genetic Programming*. Springer Nature Switzerland, 227–243. doi:10.1007/978-3-031-29573-7_15

[28] Ying Jin, Weilin Fu, Jian Kang, Jiadong Guo, and Jian Guo. 2020. Bayesian Symbolic Regression. arXiv:1910.08892 [stat.ME] https://arxiv.org/abs/1910.08892

[29] Pierre-Alexandre Kamienny, Stéphane d'Ascoli, Guillaume Lample, and Francois Charton. 2022. End-to-end Symbolic Regression with Transformers. In *Advances in Neural Information Processing Systems*, Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (Eds.). https://openreview.net/forum?id=GoOuIrDHG_Y

[30] Lukas Kammerer, Gabriel Kronberger, Bogdan Burlacu, Stephan M. Winkler, Michael Kommenda, and Michael Affenzeller. 2020. *Symbolic Regression by Exhaustive Search: Reducing the Search Space Using Syntactical Constraints and Efficient Semantic Structure Deduplication*. Springer International Publishing, 79–99. doi:10.1007/978-3-030-39958-0_5

[31] Aleksandar Kartelj and Marko Djukanović. 2023. RILS-ROLS: robust symbolic regression via iterated local search and ordinary least squares. *Journal of Big Data* 10, 1 (May 2023). doi:10.1186/s40537-023-00743-2

[32] Maarten Keijzer. 2003. Improving symbolic regression with interval arithmetic and linear scaling. In *European Conference on Genetic Programming*. Springer, 70–82.

[33] Johannes Kepler. 1619. *Harmonices Mundi*. Lincii Austriæ.

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

[35] John R. Koza. 1992. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, USA.

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

[37] Gabriel Kronberger, Bogdan Burlacu, Michael Kommenda, Stephan M. Winkler, and Michael Affenzeller. 2024. *Symbolic Regression*. Chapman & Hall / CRC Press.

[38] William La Cava, Bogdan Burlacu, Marco Virgolin, Michael Kommenda, Patryk Orzechowski, Fabrício Olivetti de França, Ying Jin, and Jason H Moore. 2021. Contemporary symbolic regression methods and their relative performance. *Advances in neural information processing systems* 2021, DB1 (2021), 1.

[39] William La Cava, Kourosh Danai, and Lee Spector. 2016. Inference of compact nonlinear dynamic models by epigenetic local search. *Engineering Applications of Artificial Intelligence* 55 (2016), 292–306.

[40] William La Cava, Thomas Helmuth, Lee Spector, and Jason H. Moore. 2019. A Probabilistic and Multi-Objective Analysis of Lexicase Selection and epsilon-Lexicase Selection. *Evolutionary Computation* 27, 3 (09 2019), 377–402. doi:10.1162/evco_a_00224 arXiv:https://direct.mit.edu/evco/article-pdf/27/3/377/1858632/evco_a_00224.pdf

[41] William G. La Cava, Paul C. Lee, Imran Ajmal, Xiruo Ding, Priyanka Solanki, Jordana B. Cohen, Jason H. Moore, and Daniel S. Herman. 2023. A flexible symbolic regression method for constructing interpretable clinical prediction models. *npj Digital Medicine* 6, 1 (June 2023). doi:10.1038/s41746-023-00833-8

[42] Mikel Landajuela, Chak Shing Lee, Jiachen Yang, Ruben Glatt, Claudio P Santiago, Ignacio Aravena, Terrell Mundhenk, Garrett Mulcahy, and Brenden K Petersen. 2022. A Unified Framework for Deep Symbolic Regression. In *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (Eds.), Vol. 35. Curran Associates, Inc., 33985–33998. https://proceedings.neurips.cc/paper_files/paper/2022/file/dbca58f35bddc6e4003b2dd80e42f838-Paper-Conference.pdf

[43] Henrietta S. Leavitt and Edward C. Pickering. 1912. Periods of 25 Variable Stars in the Small Magellanic Cloud. *Harvard College Observatory Circular* 173 (March 1912), 1–3.

[44] Lianyi Liu, Sifeng Liu, Yingjie Yang, Xiaojun Guo, and Jinghe Sun. 2024. A generalized grey model with symbolic regression algorithm and its application in predicting aircraft remaining useful life. *Engineering Applications of Artificial Intelligence* 136 (2024), 108986. doi:10.1016/j.engappai.2024.108986

[45] Nour Makke and Sanjay Chawla. 2024. Interpretable scientific discovery with symbolic regression: a review. *Artificial Intelligence Review* 57, 1 (Jan. 2024). doi:10.1007/s10462-023-10622-0

[46] Yoshitomo Matsubara, Naoya Chiba, Ryo Igarashi, and Yoshitaka Ushiku. 2024. Rethinking Symbolic Regression Datasets and Benchmarks for Scientific Discovery. *Journal of Data-centric Machine Learning Research* (2024). https://openreview.net/forum?id=qrUdrXsiXX

[47] Trent McConaghy. 2011. FFX: Fast, scalable, deterministic symbolic regression technology. In *Genetic Programming Theory and Practice IX*. Springer, 235–260.

[48] 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. 2017. SymPy: symbolic computing in Python. *PeerJ Computer Science* 3 (Jan. 2017), e103. doi:10.7717/peerj-cs.103

[49] Randal S. Olson, William La Cava, Patryk Orzechowski, Ryan J. Urbanowicz, and Jason H. Moore. 2017. PMLB: a large benchmark suite for machine learning evaluation and comparison. *BioData Mining* 10, 1 (11 Dec 2017), 36. doi:10.1186/s13040-017-0154-4

[50] Patryk Orzechowski, William La Cava, and Jason H. Moore. 2018. Where are we now? a large benchmark study of recent symbolic regression methods. In *Proceedings of the Genetic and Evolutionary Computation Conference* (Kyoto, Japan) *(GECCO '18)*. Association for Computing Machinery, New York, NY, USA, 1183–1190. doi:10.1145/3205455.3205539

[51] David L Randall, Tyler S Townsend, Jacob D Hochhalter, and Geoffrey F Bomarito. 2022. Bingo: a customizable framework for symbolic regression with genetic programming. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. 2282–2288.

[52] Joseph D Romano, Trang T Le, William La Cava, John T Gregg, Daniel J Goldberg, Praneel Chakraborty, Natasha L Ray, Daniel Himmelstein, Weixuan Fu, and Jason H Moore. 2021. PMLB v1.0: an open source dataset collection for benchmarking machine learning methods. *arXiv preprint arXiv:2012.00058v2* (2021).

[53] Etienne Russeil, Fabricio Olivetti de Franca, Konstantin Malanchev, Bogdan Burlacu, Emille Ishida, Marion Leroux, Clément Michelin, Guillaume Moinard, and Emmanuel Gangler. 2024. Multiview Symbolic Regression. In *Proceedings of the Genetic and Evolutionary Computation Conference* (Melbourne, VIC, Australia) *(GECCO '24)*. Association for Computing Machinery, New York, NY, USA, 961–970. doi:10.1145/3638529.3654087

[54] Subham Sahoo, Christoph Lampert, and Georg Martius. 2018. Learning Equations for Extrapolation and Control. In *Proceedings of the 35th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 80)*, Jennifer Dy and Andreas Krause (Eds.). PMLR, 4442–4450. https://proceedings.mlr.press/v80/sahoo18a.html

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

[56] Michael Schmidt and Hod Lipson. 2010. *Age-Fitness Pareto Optimization*. Springer New York, 129–146. doi:10.1007/978-1-4419-7747-2_8

[57] Parshin Shojaee, Kazem Meidani, Amir Barati Farimani, and Chandan K. Reddy. 2023. Transformer-based Planning for Symbolic Regression. In *Thirty-seventh Conference on Neural Information Processing Systems*. https://openreview.net/forum?id=0rVXQEeFEL

[58] ME Thing and SM Koksbang. 2025. cp3-bench: a tool for benchmarking symbolic regression algorithms demonstrated with cosmology. *Journal of Cosmology and Astroparticle Physics* 2025, 01 (2025), 040.

[59] R. B. Tully and J. R. Fisher. 1977. A New Method of Determining Distance to Galaxies. *Astronomy and Astrophysics* 500 (Feb. 1977), 105–117.

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

[61] M. Virgolin, T. Alderliesten, C. Witteveen, and P. A. N. Bosman. 2021. Improving Model-Based Genetic Programming for Symbolic Regression of Small Expressions. *Evolutionary Computation* 29, 2 (2021), 211–237. doi:10.1162/evco_a_00278

[62] Marco Virgolin and Solon P. Pissis. 2022. Symbolic Regression is NP-hard. arXiv:2207.01018 [cs.NE] https://arxiv.org/abs/2207.01018

[63] Jan Žegklitz and Petr Pošík. 2020. Benchmarking state-of-the-art symbolic regression algorithms. *Genetic Programming and Evolvable Machines* 22, 1 (March 2020), 5–33. doi:10.1007/s10710-020-09387-0

[64] Hengzhe Zhang, Aimin Zhou, Hong Qian, and Hu Zhang. 2022. PS-Tree: A piecewise symbolic regression tree. *Swarm and Evolutionary Computation* 71 (2022), 101061. doi:10.1016/j.swevo.2022.101061