PARETO FRONTS FOR SYMBOLIC REGRESSION: TRAVERSAL SEARCH TOWARDS THE FRONTIER

Anonymous authors
Paper under double-blind review

ABSTRACT

Symbolic regression (SR) seeks compact mathematical expressions that balance predictive accuracy and model complexity, but the combinatorial growth of the expression space makes effective search challenging. This work makes two contributions: (i) we find Pareto fronts for SR benchmarks extended up to length 21 and (ii) we leverage the results of the Pareto front generation to develop a structured traversal SR search framework over explicitly defined topology-conditioned symbolic landscapes. For the first contribution, we combine tree topology enumeration with capped sampling of node labels to construct longer expressions, extending empirical accuracy–complexity Pareto fronts to expression lengths up to 21. For the second contribution, we develop *traversal search* over expression states within tree topologies, using multistart n-mutation search and beam search to explore mutation neighborhoods. Our central novelty is an explicit formulation of expression states and mutation neighborhoods within tree topologies, enabling traversal strategies to be evaluated against reference ranks and empirical Pareto fronts while finding high quality expressions with a small fraction of the evaluations required for exhaustive search. On representative benchmarks, our approach identifies expressions close to the empirical reference fronts, achieving competitive predictive performance with lower complexity than many SR algorithms in SRBench. We perform systematic comparisons of traversal search configurations to provide insight into effective strategies for locating highly ranked expressions and the relationship between traversal search quality and evaluation efficiency. We further examine training–test alignment to complement the evaluation of traversal search performance. Together, these contributions extend empirical accuracy–complexity frontiers to larger expression spaces and show how the resulting topology-conditioned landscapes can be exploited for efficient traversal search. By establishing an empirical reference for what is attainable within the explored search space, we provide a concrete target for algorithm development, allowing algorithms to be evaluated by how effectively they approach known high quality solutions rather than only by comparison with existing algorithms.

1 INTRODUCTION

Symbolic regression (SR) infers both the functional structure and numerical parameters of mathematical models directly from data without assuming a predefined model family (Koza, 1994). Its explicit analytical outputs make SR attractive for prediction, scientific discovery, and interpretable modeling. Early work showed that symbolic search can recover conservation laws and analytical relationships from experimental dynamical data (Schmidt & Lipson, 2009), while more recent methods have recovered physical equations using physics-inspired search (Udrescu & Tegmark, 2020), extracted force laws and Hamiltonians from learned representations (Cranmer et al., 2020), and introduced a neural approach for symbolic expression discovery (Biggio et al., 2021). Beyond physics, SR has been applied in materials science (Wang et al., 2019), healthcare and biomedical research (Christensen et al., 2022; Fong & Motani, 2024), engineering and mechanics (Bomarito et al., 2021), interpretable semantic similarity measurement (Martinez-Gil & Chaves-Gonzalez, 2020), and governing equation discovery for transport processes (Im et al., 2023). These applications have motivated SR methods including genetic programming and learning-guided approaches.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Previous work has established Pareto fronts for SR and introduced preoptimized vault of expressions (POVE) (Fong & Motani, 2025a;b). These resources reveal that existing algorithms often miss high quality short expressions, motivating more effective search strategies. In this paper, we extend the empirical accuracy-complexity Pareto fronts to expression lengths up to 21 on the same 32 regression datasets from the Penn Machine Learning Benchmarks (PMLB) (Romano et al., 2022) used in POVE. By combining tree topology enumeration with capped sampling of node labels, we construct longer-expression libraries that provide broader references for the best tradeoffs observed in the explored space.

We develop a traversal search framework that uses training fitness to guide multistart n-mutation search and beam search through mutation neighborhoods. We assess search quality and evaluation cost using reference ranks, empirical Pareto fronts, and an analytic baseline, compare predictive accuracy and expression complexity with SRBench (La Cava et al., 2021), and examine training-test alignment.

The main contributions of this paper are:

1. **A traversal search framework.** We study multistart n-mutation search and beam search over mutation neighborhoods to find high quality expressions with fewer evaluations.
2. **Extended empirical Pareto fronts.** We extend the Pareto fronts to length 21 on 32 PMLB datasets, providing broader accuracy-complexity references for evaluating SR search.
3. **Evaluation-based search complexity.** We characterize search cost in terms of candidate evaluations and derive neighborhood sizes and evaluation bounds for n-mutation search.
4. **Systematic empirical assessment.** We evaluate search performance and efficiency against analytic baseline, empirical Pareto fronts, and SRBench results, while analyzing training-test alignment to assess whether gains in search fitness translate into better generalization.

Code for Pareto front construction and the proposed search methods is available at `https://anonymous.4open.science/r/sr-fronts-search/` (anonymized link, with credits given to authors of open-source packages required).

2 Related Work

2.1 Pareto Fronts and Reference Expression Spaces

Pareto analysis is widely used in SR to characterize tradeoffs between performance and complexity (Vladislavleva et al., 2008; Kommenda et al., 2015). Fong & Motani (2025a) systematically enumerate symbolic expressions and optimize their numerical constants to construct reference Pareto fronts over performance and length on datasets from PMLB. POVE provides a reusable vault of expressions with stored fitness values, supporting efficient benchmarking and repeated algorithm evaluation (Fong & Motani, 2025b). These studies provide systematically constructed and pre-evaluated expression collections that serve as common references for SR research. In this work, we extend the coverage of these reference fronts to longer expressions and use the resulting expression spaces to investigate structured search strategies.

2.2 Symbolic Regression Strategies

SR algorithms employ a variety of strategies to explore expression spaces (Ferreira, 2001; Kammerer et al., 2020; Bartlett et al., 2024; Shojaee et al., 2023; Zhang et al., 2026). For example, a top performer in SRBench, deep symbolic regression, uses a recurrent neural network and reinforcement learning to guide expression sampling (Petersen et al., 2021), while Operon incorporates automatic differentiation into expression evaluation to support local optimization of numerical parameters using gradients (Burlacu et al., 2020). To our knowledge, we are the first to systematically study a family of explicit expression-space traversal strategies for SR, including multistart n-mutation search and beam search, within a unified framework under common evaluation rules.

2.3 TRAVERSAL SEARCH IN SR

Search mechanisms such as beam search and mutation-based neighborhood search have appeared in SR, but typically as components within larger neural or evolutionary pipelines. For example, beam search is used to decode candidate expressions from neural generators (Biggio et al., 2021), while mutation-based local search is embedded within evolutionary algorithms. In contrast, we make traversal itself the primary search mechanism, directly exploring fixed-topology expression spaces through mutation-defined neighborhoods and systematically varying search depth and breadth.

3 METHODOLOGY

3.1 CANDIDATE GENERATION WITH CAPPED SYSTEMATIC SAMPLING

The number of candidate expressions grows rapidly with expression length, making complete evaluation infeasible for the longer expressions. We therefore used a capped two-stage sampling procedure that combines exhaustive enumeration of expression tree topologies with random generation of operator and terminal assignments, followed by deterministic equal-interval subsampling.

For expression lengths below 11, the number of candidate expressions did not exceed the final cap of 10^8, and all candidates were therefore retained. When the candidate space exceeded this cap, we applied the following capped sampling procedure to each odd expression length.

For each odd expression length ℓ, we exhaustively enumerated all valid ordered full binary tree topologies. Each topology was represented by a prefix arity sequence with entries in $\{0, 2\}$, where 0 denotes a leaf node and 2 denotes a binary operator.

Let $\mathcal{T}_\ell$ denote the set of valid topologies for target length ℓ. We set a pre-cap collection target $B = 1.2 \times 10^8$ and allocated an approximately equal quota to each topology:

$$q_\ell = \left\lfloor \frac{B}{|\mathcal{T}_\ell|} \right\rfloor \tag{1}$$

The binary operator set is $\{+, -, \times, \div\}$. Within each topology, leaves were sampled independently and uniformly from the input variables and constant, while binary operators were sampled independently and uniformly from the operator set. Candidate generation continued until q_ℓ unique accepted expressions, after algebraic simplification, had been collected for that topology.

The outputs from all topologies were merged and globally deduplicated. Expressions appearing in the retained files for shorter lengths with the same number of input features were removed. The remaining expressions were sorted lexicographically.

To bound the number of expressions passed to downstream numerical optimization, we imposed a final cap $C = 10^8$. Let N_ℓ denote the number of expressions in the cleaned and lexicographically sorted candidate list generated for target length ℓ. If $N_\ell > C$, the list was divided into C approximately equal intervals, and the midpoint index of each interval was selected:

$$i_j = \left\lfloor \frac{(2j+1)N_\ell}{2C} \right\rfloor, \qquad j = 0, \ldots, C-1. \tag{2}$$

Together with approximately equal topology quotas and uniform node sampling, midpoint selection spreads candidates across the sorted list, maintaining diversity while limiting expressions to 10^8.

3.2 TRAVERSAL SEARCH

For expression length ℓ, we search label assignments within each topology $\tau \in \mathcal{T}_\ell$. A search state is a labelled preorder sequence $e = (a_1, \ldots, a_\ell)$. Internal nodes are drawn from the operator set, and leaves from input variables and constant placeholder. Let Ω_τ denote all admissible label assignments for τ. Traversal explores mutation neighborhoods in Ω_τ while preserving topology.

We write $F(e)$ for the training R^2 obtained after fitting constants of the simplified expression on the training data. In our POVE based experiments, training and test scores are precomputed for each dataset and train-test split, and $F(e)$ is retrieved from the stored training scores. For datasets outside

the vault range, the same fitness can also be computed in advance or on demand. Search decisions and final expression selection use only $F(e)$. Predictive performance is assessed using the test R^2 of the same selected expression, with its fitted constants fixed. Expression ranks are computed relative to the reference vault used in each experiment.

Single-mutation search. We perform greedy single-mutation search from independent starts (Algorithm 1). For each topology, we sample node labels independently and uniformly to obtain s starts with distinct skeletons. The neighborhood is

$$\mathcal{N}_1(e) = \{e' \in \Omega_\tau : |\{i : a'_i \neq a_i\}| = 1\}. \tag{3}$$

Thus, each neighbor is obtained by replacing a single node with an admissible alternative. At iteration t, let e_t denote the current search state. We evaluate its neighborhood and select

$$\widetilde{e}_t \in \operatorname*{arg\,max}_{e' \in \mathcal{N}_1(e_t)} F(e'). \tag{4}$$

We accept $\widetilde{e}_t$ only when $F(\widetilde{e}_t) > F(e_t)$; otherwise the trajectory terminates. Each trajectory also terminates after at most R expansions. The final output is the expression with the highest training score across all starts and topologies. Increasing s changes the initial coverage while retaining the same local search rule. Candidates are recorded by their preorder sequences within each independent trajectory. The total number of expressions evaluated is used as a measure of search complexity.

Algorithm 1 Multistart Single-Mutation Search

Require: Expression length ℓ; starts per topology s; maximum rounds R; fitness F
Ensure: Best expression $e^\star$

$e^\star \leftarrow$ None; $f^\star \leftarrow -\infty$
for $\tau \in$ GENERATETOPOLOGIES(ℓ) **do**
  for $e \in$ GENERATEUNIQUESTARTS(τ, s) **do**
    $f \leftarrow F(e)$
    for $t = 1, \ldots, R$ **do**
      $\mathcal{N} \leftarrow \mathcal{N}_1(e)$
      $\widetilde{e}_t \leftarrow \arg\max_{x \in \mathcal{N}} F(x)$
      if $F(\widetilde{e}_t) \leq f$ **then break**
      $e \leftarrow \widetilde{e}_t$; $f \leftarrow F(\widetilde{e}_t)$
    if $f > f^\star$ **then** $(e^\star, f^\star) \leftarrow (e, f)$
return $e^\star$

3.2.1 BEAM SEARCH

For each topology, we generate Gw starts using the same initialization procedure and divide them into G independent groups of width w. Each group maintains a beam $\mathcal{B}_t$ and a set $\mathcal{A}_t$ of expanded states. Initially, $\mathcal{A}_0 = \varnothing$ and $\mathcal{B}_0$ contains the group's starts. For $t \geq 1$, we form

$$\mathcal{A}_t = \mathcal{A}_{t-1} \cup \mathcal{B}_{t-1}, \qquad \mathcal{Q}_t = \left(\bigcup_{e \in \mathcal{B}_{t-1}} \mathcal{N}_1(e) \right) \setminus \mathcal{A}_t. \tag{5}$$

The next beam retains up to w candidates in $\mathcal{Q}_t$ with the highest training scores. Each group tracks its historical best while allowing the beam to traverse lower scoring expressions, and stops when $\mathcal{Q}_t = \varnothing$ or after R rounds. We return the best expression across all groups and topologies.

3.2.2 n-MUTATION SEARCH

We generalize the single-mutation by changing exactly n distinct positions jointly, where $1 \leq n \leq \ell$:

$$\mathcal{N}_n(e) = \{e' \in \Omega_\tau : |\{i : a'_i \neq a_i\}| = n\}. \tag{6}$$

All n-position combinations and all admissible replacement combinations are enumerated, and the best improving candidate is selected. A trajectory terminates when no improving candidate exists

or when the iteration limit is reached. Joint mutations allow the search to move directly between expressions that cannot be connected through strictly improving single-mutation steps, providing a broader search range.

4 Results and Discussions

4.1 Training-Test Alignment

We examined whether strong training performance corresponds to strong test performance. This analysis covered 32 datasets with 10 train-test splits each, giving 320 dataset-split pairs. For each pair, we identified the expressions attaining the highest training and test R^2 among the stored expressions of lengths up to 9, denoted by e_{tr} and e_{te}, respectively. Numerical constants were fitted on the training data and held fixed for test evaluation.

We then computed the normalized cross-ranks

$$q_{\mathrm{te}} = \frac{\mathrm{rank}_{\mathrm{te}}(e_{\mathrm{tr}})}{N}, \qquad q_{\mathrm{tr}} = \frac{\mathrm{rank}_{\mathrm{tr}}(e_{\mathrm{te}})}{N}, \tag{7}$$

where N is the number of total expressions. Expressions were ranked by their R^2 values.

The selected training and test optima shared the same expression skeleton in 49 of the 320 pairs (15.3%). Both normalized cross-ranks were within the top 0.01% in 153 pairs (47.8%). At least one cross-rank fell outside the top 10% in 42 pairs (13.1%), spanning 12 datasets. Table 1 reports the cumulative proportions at several thresholds, showing that the degree of alignment varied substantially across the evaluated pairs.

Table 1: Cumulative cross-rank counts across 320 dataset-split pairs.

Threshold p	$q_{\mathrm{te}} \leq p$	$q_{\mathrm{tr}} \leq p$	Both
0.01%	176 (55.0%)	172 (53.8%)	153 (47.8%)
0.1%	217 (67.8%)	204 (63.8%)	188 (58.8%)
1%	258 (80.6%)	231 (72.2%)	216 (67.5%)
10%	309 (96.6%)	287 (89.7%)	278 (86.9%)

To assess the magnitude of the differences in test performance, we additionally computed

$$\Delta_{\mathrm{te}} = R^2_{\mathrm{te}}(e_{\mathrm{te}}) - R^2_{\mathrm{te}}(e_{\mathrm{tr}}). \tag{8}$$

Across all 320 pairs, the median gap was 0.0295, and 92 pairs (28.8%) had a gap greater than 0.1.

Three examples illustrate why agreement in expression selection, agreement in rank, and similarity in test performance should be distinguished. On `529_pollen`, the training and test criteria selected the same expression in only 1 of 10 splits, yet every test R^2 gap was below 3×10^{-7}, indicating negligible differences. On `596_fri_c2_250_5`, both cross-ranks were within the top 0.01% in all 10 splits, while the median test R^2 gap was 0.0590, which was higher than the overall median of 0.0295. On `687_sleuth_ex1605`, at least one cross-rank fell outside the top 10% in every split, and the median gap was 0.4733, indicating a substantial difference in test performance. Different expressions can yield nearly identical test performance, while strong cross-rank alignment need not imply a small test R^2 gap. Cross-ranks and performance gaps therefore provide complementary measures of training-test alignment.

4.2 Computational Complexity

We measure search cost by candidate evaluations. Let d denote the number of features and $m = (\ell - 1)/2$ the number of operator nodes. An ordered full binary tree has $m + 1$ leaves, with $|\mathcal{T}_\ell| = \binom{2m}{m}/(m+1)$ possible topologies. With 4 binary operators and d features plus constant placeholder, each operator node has 3 replacement labels and each leaf has d. The single-mutation neighborhood therefore contains $D_1(\ell, d) := |\mathcal{N}_1(e)| = 3m + d(m + 1)$ states. With s starts per topology and at most R neighborhood expansions, the evaluation upper bound is

$$E_1^{\mathrm{cap}}(\ell, d, s, R) = s|\mathcal{T}_\ell| R D_1(\ell, d). \tag{9}$$

For $\ell = 9$, there are 14 topologies and $D_1(9, d) = 12 + 5d$. Table 2 shows that mean recorded evaluations are 33.58%-45.25% of the bound at $s = 10$ and $R = 10$.

Table 2: Evaluation counts and upper bounds for single-mutation search. Each $\bar{E}_1$ is the mean over 100 runs (10 split seeds $\times$ 10 search seeds). Ratio denotes $100\bar{E}_1/E_1^{\mathrm{cap}}$.

Dataset	d	D_1	E_1^{cap}	$\bar{E}_1$	Ratio (%)
228_elusage	2	22	30,800	10,341	33.58
556_analcatdata_apnea2	3	27	37,800	14,633	38.71
529_pollen	4	32	44,800	18,763	41.88
210_cloud	5	37	51,800	22,742	43.90
706_sleuth_case1202	6	42	58,800	26,352	44.82
547_no2	7	47	65,800	29,777	45.25

For n-mutation, choosing j operator nodes and $n - j$ leaves and replacing their labels gives

$$D_n(\ell, d) := |\mathcal{N}_n(e)| = \sum_{j=\max(0,n-m-1)}^{\min(n,m)} \binom{m}{j}\binom{m+1}{n-j} 3^j d^{\,n-j}. \tag{10}$$

The corresponding evaluation bound is

$$E_n^{\mathrm{cap}}(\ell, d, s, R) = s|\mathcal{T}_\ell| R D_n(\ell, d). \tag{11}$$

Larger mutation neighborhoods substantially increase search cost. Across the six datasets, 2-mutation search uses 9.88-20.93 times as many recorded evaluations as single-mutation search at the same $s = 5$ and $R = 10$. On dataset 529_pollen ($d = 4$), $D_1 = 32$, $D_2 = 454$, and $D_3 = 3,748$. At $R = 10$, 2-mutation search with $s = 5$ has a bound of 317,800 evaluations and a recorded mean of 132,109.50 (41.57%). 3-mutation search with only $s = 2$ has a bound of 1,049,440 and a mean of 357,641.27 (34.08%). Thus, equal starts or round limits do not imply comparable evaluation budgets across mutation orders.

4.3 Random Search Baseline

To provide a reference for search efficiency, we consider uniform random sampling without replacement from N expressions. Let K denote the number of sampled expressions and let $R_{\min}$ be the best rank, where rank 1 is best. For any rank r,

$$\Pr(R_{\min} \geq r) = \frac{\binom{N-r+1}{K}}{\binom{N}{K}}, \qquad \mathbb{E}[R_{\min}] = \sum_{r=1}^{N-K+1} \Pr(R_{\min} \geq r) = \frac{N+1}{K+1}. \tag{12}$$

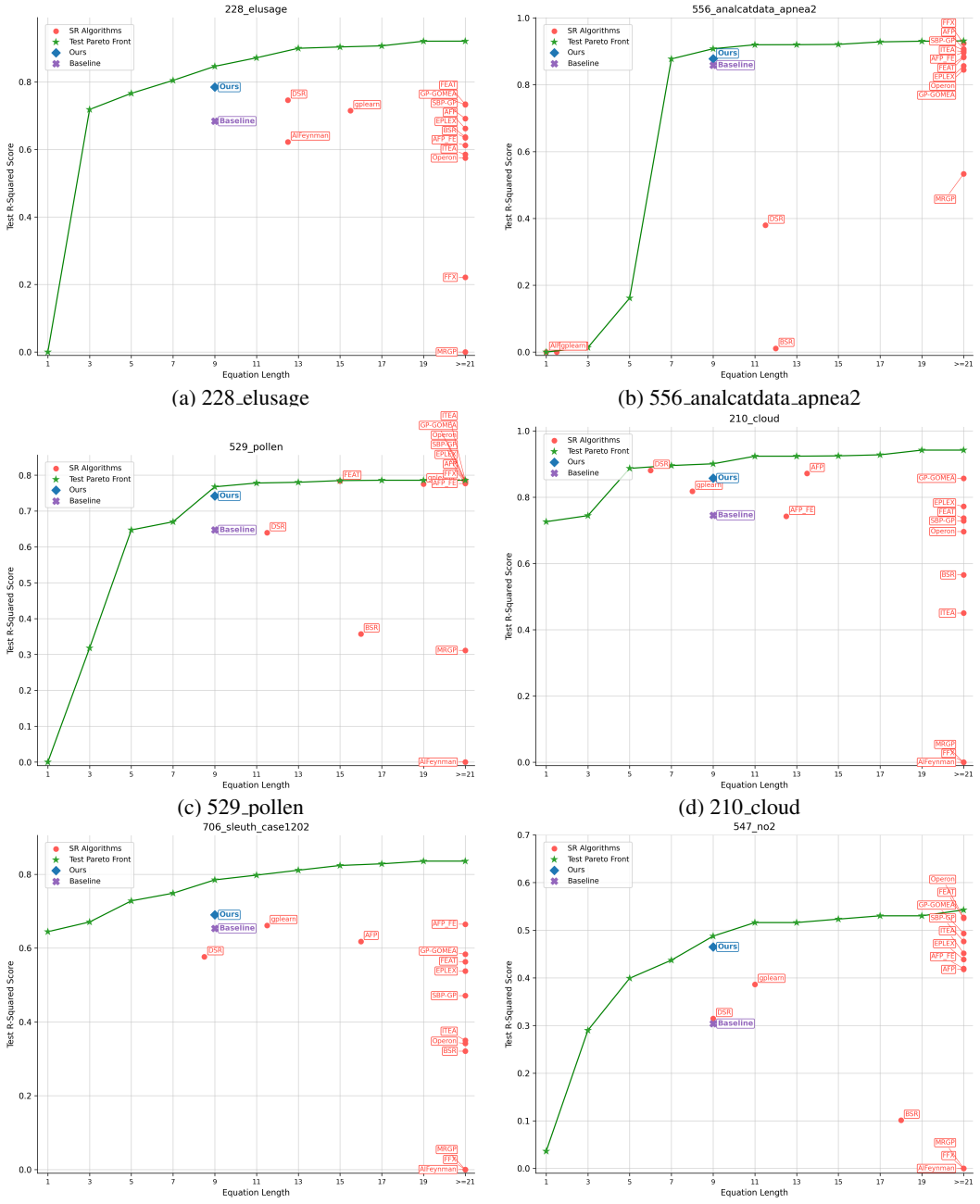
Thus, the expected rank percentile is

$$\mathbb{E}\left[100\frac{R_{\min}}{N}\right] = 100\frac{N+1}{N(K+1)} \approx \frac{100}{K+1}. \tag{13}$$

Note that directly interpreting this expectation as a test rank would be optimistic for random search, since it would effectively use test performance for model selection and provide access to a test oracle. A valid test-side comparison requires preserving the expression-wise correspondence between training and test results: selection based on training performance, with test performance used only to evaluate the selected expression.

4.4 Empirical Pareto Fronts and Search Performance

Figure 1 summarizes performance using medians. At each length limit ℓ, the green front reports the median over 10 splits of the best test R^2 among retained expressions of lengths at most ℓ. Red SRBench points report median test R^2 and model size. Blue *Ours* takes the median over 10 searches within each split, followed by the median over 10 splits.



(a) 228_elusage (b) 556_analcatdata_apnea2

(c) 529_pollen (d) 210_cloud

(e) 706_sleuth_case1202 (f) 547_no2

Figure 1: Test Pareto fronts for six representative datasets.

Extending the reference fronts. Extending the length limit from 9 to 21 yields a median gain of 0.0529 in reference test R^2 across the 32 datasets. The reference increases from 0.8462 to 0.9208 on `228_elusage` and from 0.4876 to 0.5426 on `547_no2`. At length limit 21, it exceeds the best archived median among the 14 SRBench SR methods on 21 of the 32 datasets. The extended fronts characterize accuracy-complexity tradeoffs and provide references for search at each length budget. Results for the remaining datasets appear in the appendix.

Effective search with a small budget. With $\ell = 9$, $s = 2$, and $R = 10$, single-mutation search exceeds the analytic random baseline on 25 of the 32 datasets, with a median test R^2 gain of 0.0362

Table 3: Mean model size and test R^2 differences across 32 datasets. Differences are computed as method minus Ours; negative values favor Ours.

Method	Ours	AFP	AFP_FE	AIFeynman	BSR	DSR	EPLEX	FEAT
Mean size	9	30.14	31.19	442.86	26.33	10.50	53.02	59.30
Mean ΔR^2	—	+0.0594	+0.0621	-9.3097	-0.2241	-0.0113	+0.0730	+0.1059
Method	Ours	FFX	GP-GOMEA	ITEA	MRGP	Operon	SBP-GP	gplearn
Mean size	9	934.95	27.55	81.19	8490.22	52.22	473.00	16.33
Mean ΔR^2	—	-3.5524	+0.0809	+0.0312	-0.5669	+0.0427	+0.0869	-0.0044

and deficits greater than 0.01 on only 4. Search also approaches the length-9 reference fronts. On `547_no2`, it reaches a test R^2 of 0.4649 against the reference value of 0.4876, reducing the gap from the baseline's 0.1836 to 0.0227. On `529_pollen`, the corresponding gap decreases from 0.1203 to 0.0253. These gains demonstrate substantial progress toward the best test scores in the length-9 libraries and highlight the value of explicit reference fronts for assessing search quality.

Comparison with SRBench. Table 3 reports mean model sizes and mean test R^2 differences from Ours across the 32 datasets. The length-9 search exceeds the test R^2 of SRBench methods in 220 of the 448 algorithm-dataset comparisons and surpasses the best archived SR result on 4 datasets, with gains up to 0.0943. On `228_elusage`, it reaches 0.7846 compared with the best archived value of 0.7463, using approximately 2,078 evaluations on average. 415 of the 448 archived entries report median model sizes above 9, and the median reported size across all entries is 46. These results show that structured search finds compact expressions that can compete with substantially larger models while using relatively few candidate evaluations. This combination is particularly valuable when predictive performance must be balanced with expression simplicity and search cost.

Evaluation efficiency. Search uses 3,963 evaluations per run on average, only equivalent to 0.793% of SRBench's 500k limit. Using the precomputed vault, all 3,200 searches completed in less than 88 minutes (excludes precomputation of fitness values). The low evaluation count and reuse of stored fitness values make repeated search experiments inexpensive, supporting systematic comparisons of search strategies and hyperparameters.

4.5 Search Quality versus Evaluation Budget

We examined how search quality changes with evaluation budget using 5 beam search groups of width 2, expression length $\ell = 9$, and maximum rounds $R \in \{2, 4, 6, 8, 10\}$. Each dataset budget setting comprised 10 searches on each of 10 train-test splits. Results cover all 32 datasets. For each run, we report the test rank of the expression selected by training fitness. Lower values indicate better ranks. Results are arithmetic means over 100 runs; we also examine medians to identify skewed outcomes. Evaluation counts measure recorded search effort.

Increasing the budget from 2 to 10 rounds reduced the mean test rank percentile on 20 of the 32 datasets. At 10 rounds, 13 datasets achieved a mean test rank percentile below 0.01%. However, 25 datasets showed worsening mean test rank between at least one pair of adjacent budgets, indicating that additional search does not consistently improve generalization.

Table 4: Mean recorded evaluations and test rank percentiles over 100 runs. Mean evaluation counts are rounded to integers; test ranks are reported in percent.

	557_analcatdata_apnea1		529_pollen		1027_ESL	
Rounds	Evaluations	Rank (%) ↓	Evaluations	Rank (%) ↓	Evaluations	Rank (%) ↓
2	7,235	0.018547	8,541	0.005540	8,549	0.008276
4	13,482	0.005660	16,038	0.001358	16,024	0.006018
6	19,542	0.002736	23,399	0.000389	23,349	0.006040
8	25,428	0.002587	30,562	0.000366	30,422	0.006051
10	31,187	0.000958	37,573	0.000364	37,308	0.006047

Table 4 reveals different returns from additional computation. On `557_analcatdata_apnea1`, mean test rank percentile decreased by 94.8%, while mean test R^2 increased from 0.8524 to 0.9094. `529_pollen` instead showed early saturation: 6 rounds attained approximately the same mean test R^2 as 10 rounds, using 37.7% fewer evaluations. `1027_ESL` required 57.0% fewer evaluations at 4 rounds than at 10, with slightly better mean test rank and test R^2.

4.6 Search Strategies and Evaluation Efficiency

We use training ranks to assess each strategy's ability to find expressions with high fitness $F(e)$, rather than generalization. Each configuration comprises 3,200 runs (32 datasets $\times$ 10 split seeds $\times$ 10 search seeds). Let $\widehat{p}_1$ be the fraction of runs attaining training rank 1 in the reference library and $\bar{E}$ the mean evaluation count. Recovery efficiency per 10^4 evaluations is

$$Y_1 = 10^4 \frac{\widehat{p}_1}{\bar{E}}. \tag{14}$$

Table 5 compares representative configurations.

Table 5: Training quality and efficiency ($\ell = 9$, $R = 3$). Evaluations are rounded means over 3,200 runs. Arrows indicate preferred directions; $\widehat{p}_1$ is displayed as a percentage.

Configuration	Evaluations ↓	Mean training rank ↓	$\widehat{p}_1$ (%) ↑	Y_1 ↑
Single, $s = 10$	12,516	18.29	18.50	0.1478
Single, $s = 15$	18,803	12.88	22.22	0.1182
Beam, $(G, w) = (5, 2)$	12,862	16.70	19.97	0.1553
Beam, $(G, w) = (5, 3)$	19,181	9.69	24.41	0.1272
2-mut, $s = 1$	19,831	13.25	27.81	0.1402
2-mut, $s = 2$	39,678	6.34	37.16	0.0936
3-mut, $s = 1$	187,347	3.16	54.75	0.0292

Beam search with (G,w)=(5,3) and single-mutation search with s=15 both use 15 initial expressions per topology. Beam search requires only 2.01% more evaluations on average, while increasing the recovery rate from 22.22% to 24.41% and improving Y_1 by 7.68%. Mean training rank also improves from 12.88 to 9.69. With $(G, w) = (5, 2)$, beam search improves Y_1 by 5.04% over single-mutation search with $s = 10$. These results show that retaining multiple candidates can improve the quality-cost tradeoff.

With s=1, 2-mutation search requires only 5.47% more evaluations on average than single-mutation search with $s = 15$, while increasing the recovery rate to 27.81% and improving Y_1 by 18.69%. Its benefit is thus a higher frequency of recovering a training optimum.

For 2-mutation search, increasing s from 1 to 2 raises the recovery rate from 27.81% to 37.16%, but approximately doubles evaluations, reducing Y_1 by 33.23%. At $s = 1$, increasing n from 2 to 3 raises the recovery rate to 54.75%, but requires 9.45 times as many evaluations and reduces Y_1 by 79.16%. Thus, excessive increases in the number of starts or mutation order can incur costs that outweigh gains in search quality.

5 Conclusion

In this work, we extend empirical accuracy-complexity Pareto fronts to expression lengths up to 21 on 32 PMLB datasets and develop a traversal search framework. The extended fronts provide broader references for the tradeoffs observed in the explored expression space. Building on pre-computed fitness values, we investigate multistart n-mutation search and beam search, and derive neighborhood sizes and evaluation bounds for n-mutation search.

Our experiments show that structured search finds compact expressions with competitive predictive performance using few evaluations. With suitable parameters, beam and n-mutation search can improve recovery of training optima with modest additional cost. Analyses of training-test alignment and evaluation budgets guide the balance between search quality and cost. These findings inform SR search design and demonstrate the value of empirical reference fronts for assessing progress toward known high quality expressions under limited evaluation budgets.

AI USE STATEMENT

In this work, we used generative AI tools to assist with translating author-written text, polishing the manuscript to improve its clarity and readability, and developing auxiliary code for non-core tasks, such as data aggregation, plotting, and formatting. The core research ideas were developed by the authors. Generative AI tools were not used for data collection, interpretation of experimental results, or drafting new manuscript sections. The authors reviewed the AI-assisted translations, edits, and code and take responsibility for the final content of this work.

REFERENCES

Deaglan J Bartlett, Harry Desmond, and Pedro G Ferreira. Exhaustive symbolic regression. *IEEE Transactions on Evolutionary Computation*, 28(4):950–964, 2024. doi: 10.1109/TEVC.2023.3280250.

Luca Biggio, Tommaso Bendinelli, Alexander Neitz, Aurelien Lucchi, and Giambattista Parascandolo. Neural symbolic regression that scales. In *International conference on machine learning*, pp. 936–945. Pmlr, 2021.

GF Bomarito, TS Townsend, KM Stewart, KV Esham, JM Emery, and JD Hochhalter. Development of interpretable, data-driven plasticity models with symbolic regression. *Computers & Structures*, 252:106557, 2021.

Bogdan Burlacu, Gabriel Kronberger, and Michael Kommenda. Operon c++ an efficient genetic programming framework for symbolic regression. In *Proceedings of the 2020 genetic and evolutionary computation conference companion*, pp. 1562–1570, 2020.

Niels Johan Christensen, Samuel Demharter, Meera Machado, Lykke Pedersen, Marco Salvatore, Valdemar Stentoft-Hansen, and Miquel Triana Iglesias. Identifying interactions in omics data for clinical biomarker discovery using symbolic regression. *Bioinformatics*, 38(15):3749–3758, 2022.

Miles Cranmer, Alvaro Sanchez Gonzalez, Peter Battaglia, Rui Xu, Kyle Cranmer, David Spergel, and Shirley Ho. Discovering symbolic models from deep learning with inductive biases. *Advances in neural information processing systems*, 33:17429–17442, 2020.

Candida Ferreira. Gene expression programming: a new adaptive algorithm for solving problems. *Complex Systems*, 13(2):87–129, 2001.

Kei Sen Fong and Mehul Motani. Symbolic regression for discovery of medical equations: A case study on glomerular filtration rate estimation equations. In *2024 IEEE Conference on Artificial Intelligence (CAI)*, pp. 1373–1379. IEEE, 2024.

Kei Sen Fong and Mehul Motani. Pareto-optimal fronts for benchmarking symbolic regression algorithms. In *ICML*, 2025a.

Kei Sen Fong and Mehul Motani. Pove: A preoptimized vault of expressions for symbolic regression research and benchmarking. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pp. 5436–5446, 2025b.

Jinwoo Im, Felipe PJ de Barros, Sami Masri, Muhammad Sahimi, and Robert M Ziff. Data-driven discovery of the governing equations for transport in heterogeneous media by symbolic regression and stochastic optimization. *Physical Review E*, 107(1):L013301, 2023.

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. In *Genetic programming theory and practice XVII*, pp. 79–99. Springer, 2020.

Michael Kommenda, Andreas Beham, Michael Affenzeller, and Gabriel Kronberger. Complexity measures for multi-objective symbolic regression. In *International Conference on Computer Aided Systems Theory*, pp. 409–416. Springer, 2015.

John R Koza. Genetic programming as a means for programming computers by natural selection. *Statistics and computing*, 4(2):87–112, 1994.

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

Jorge Martinez-Gil and Jose M Chaves-Gonzalez. A novel method based on symbolic regression for interpretable semantic similarity measurement. *Expert Systems with Applications*, 160:113663, 2020.

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.

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. Pmlb v1. 0: an open-source dataset collection for benchmarking machine learning methods. *Bioinformatics*, 38(3):878–880, 2022.

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

Parshin Shojaee, Kazem Meidani, Amir Barati Farimani, and Chandan Reddy. Transformer-based planning for symbolic regression. *Advances in Neural Information Processing Systems*, 36: 45907–45919, 2023.

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

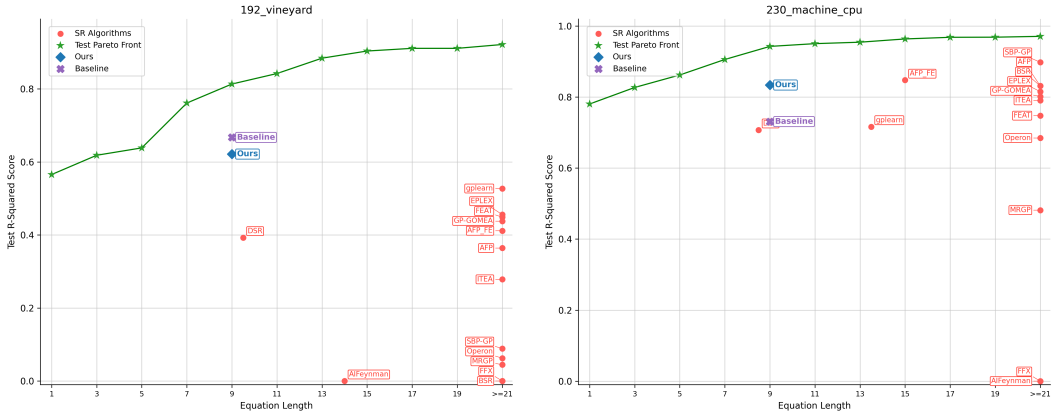
Ekaterina J Vladislavleva, Guido F Smits, and Dick Den Hertog. Order of nonlinearity as a complexity measure for models generated by symbolic regression via pareto genetic programming. *IEEE Transactions on Evolutionary Computation*, 13(2):333–349, 2008.

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

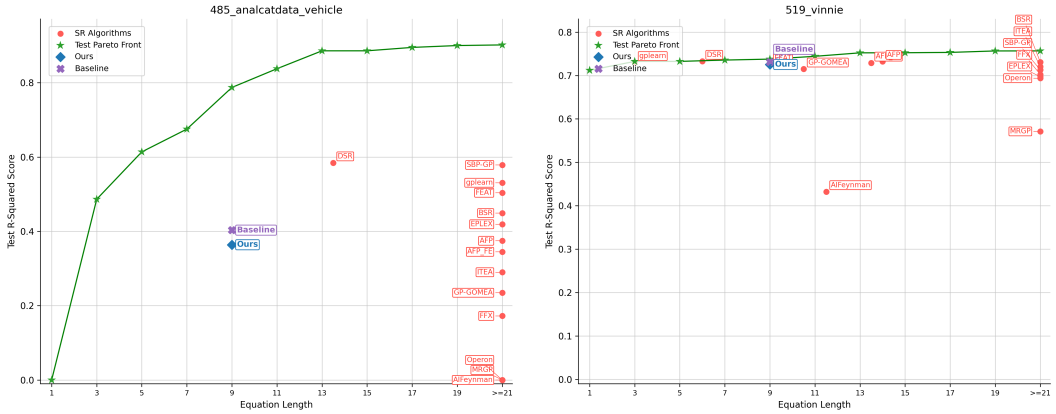
Hengzhe Zhang, Qi Chen, Bing Xue, Wolfgang Banzhaf, and Mengjie Zhang. Benchmarking zero-shot llm-generated parent selection in genetic programming for symbolic regression. In *International Conference on Parallel Problem Solving from Nature*, pp. 233–248. Springer, 2026.

A Additional Pareto Fronts and Search Results

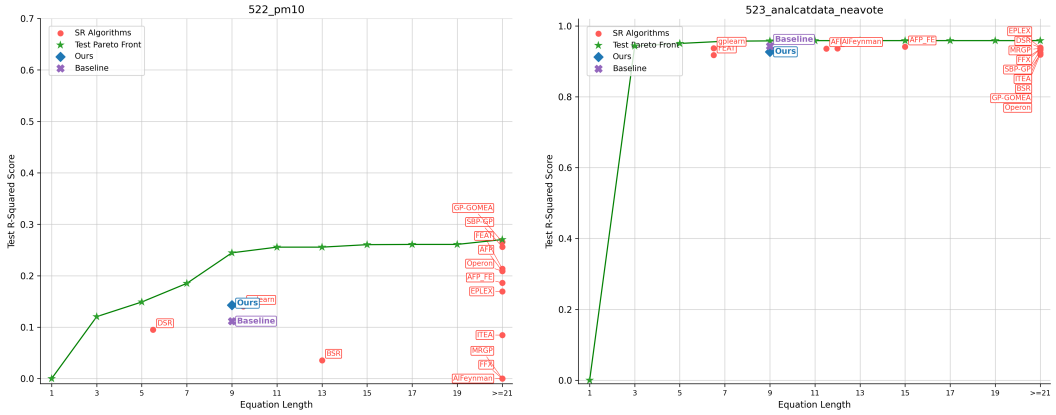
Figures 2–6 present the test Pareto fronts and search results for the remaining 26 datasets, complementing the six datasets shown in Figure 1.



(a) `192_vineyard` (b) `230_machine_cpu`

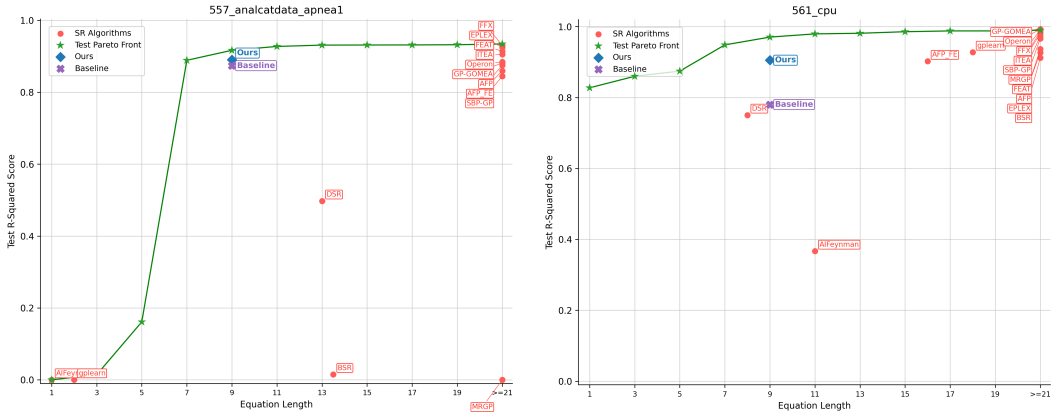


(c) `485_analcatdata_vehicle` (d) `519_vinnie`

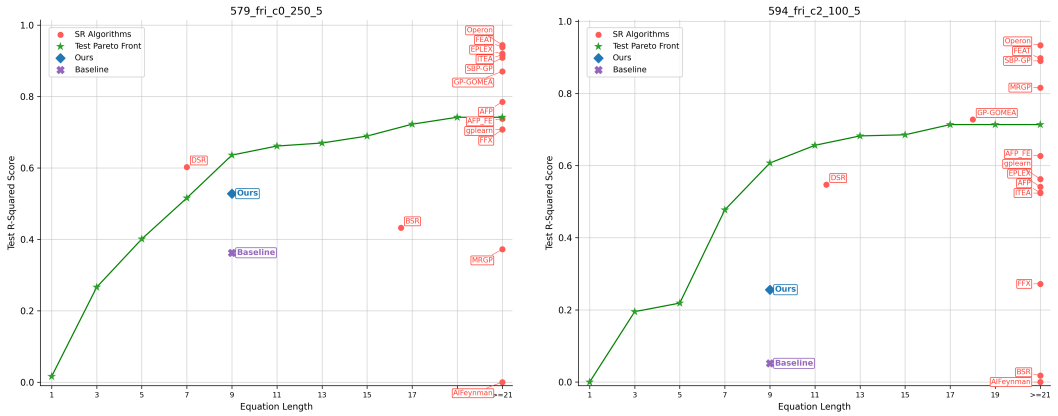


(e) `522_pm10` (f) `523_analcatdata_neavote`

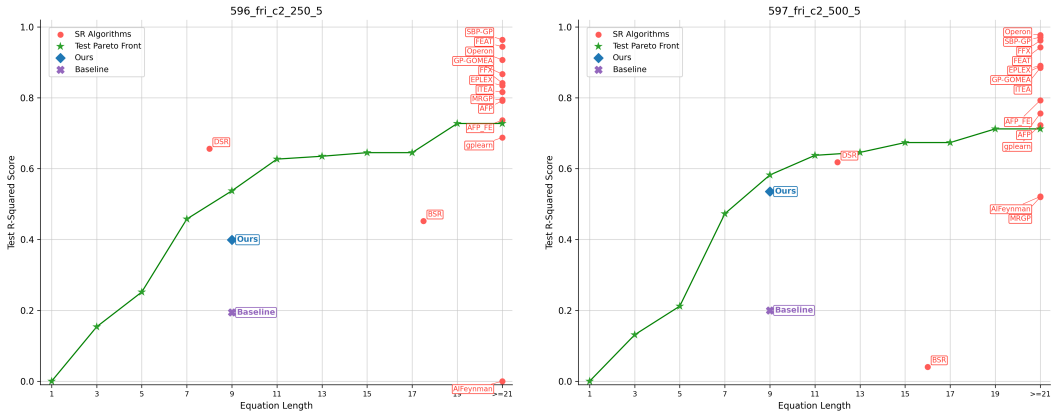
Figure 2: Test Pareto fronts and search results for additional datasets (1/5).



(a) 557_analcatdata_apnea1 (b) 561_cpu

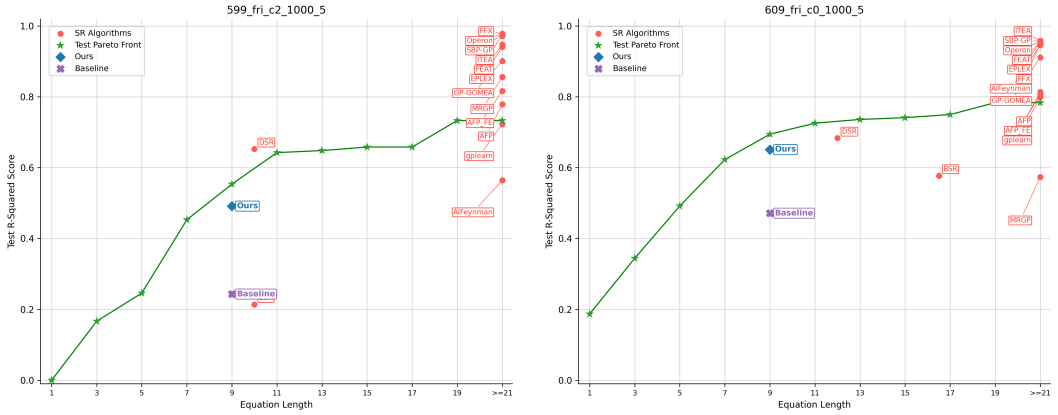


(c) 579_fri_c0_250_5 (d) 594_fri_c2_100_5

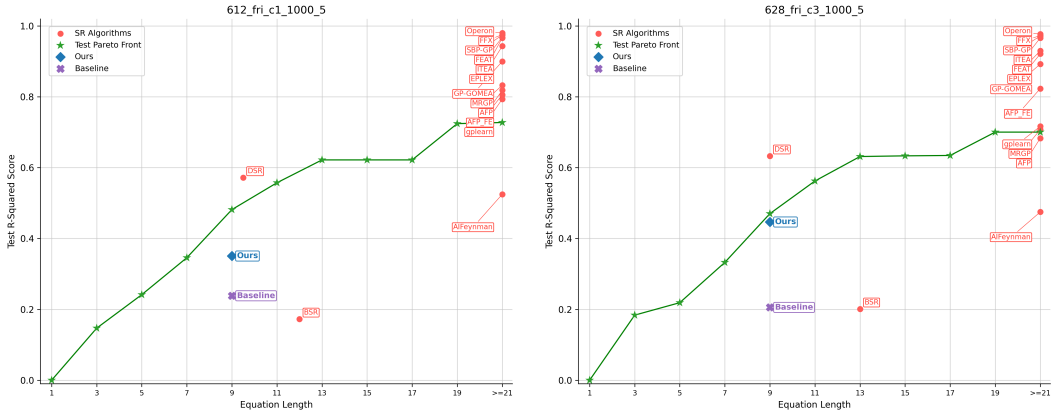


(e) 596_fri_c2_250_5 (f) 597_fri_c2_500_5

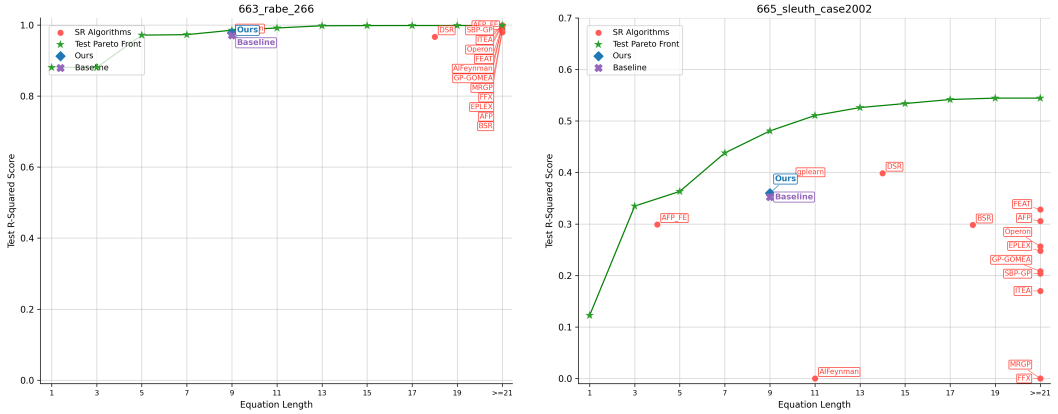
Figure 3: Test Pareto fronts and search results for additional datasets (2/5).



(a) `599_fri_c2_1000_5` (b) `609_fri_c0_1000_5`

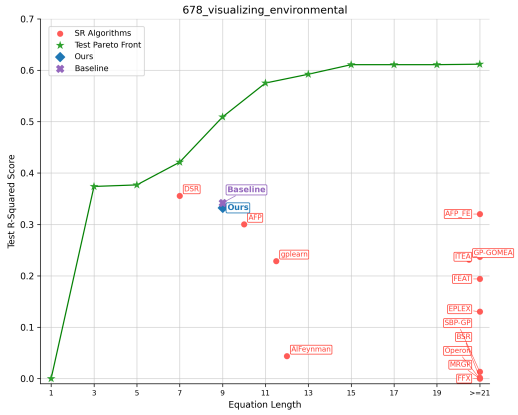


(c) `612_fri_c1_1000_5` (d) `628_fri_c3_1000_5`

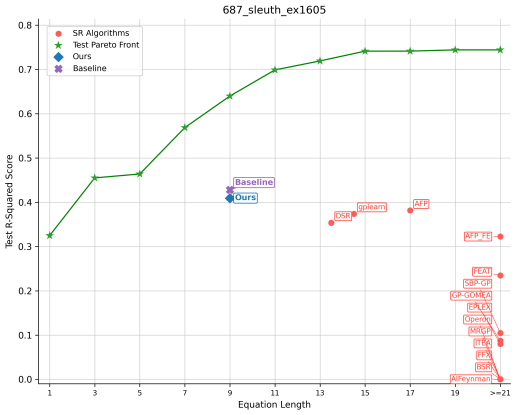


(e) `663_rabe_266` (f) `665_sleuth_case2002`

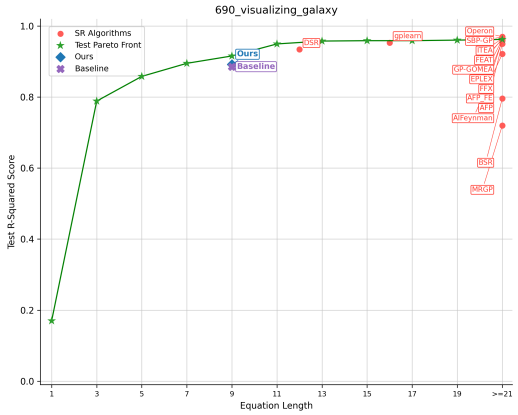
Figure 4: Test Pareto fronts and search results for additional datasets (3/5).



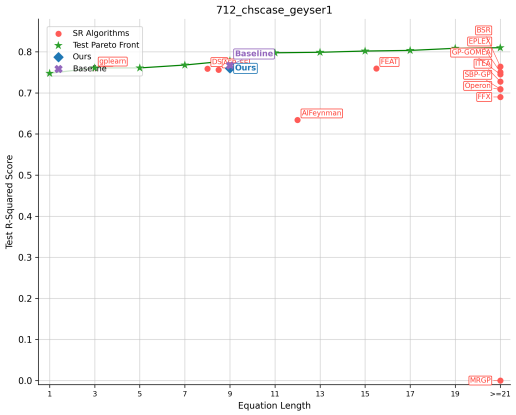
(a) `678_visualizing_environmental`



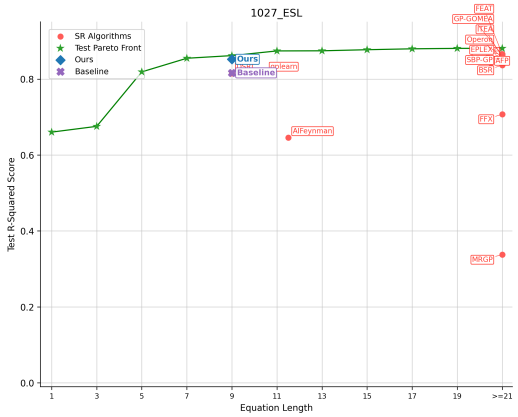
(b) `687_sleuth_ex1605`



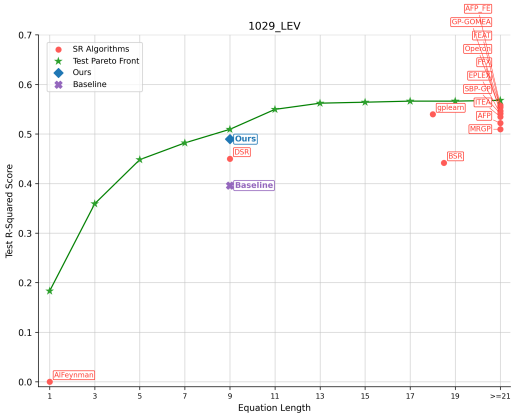
(c) `690_visualizing_galaxy`



(d) `712_chscase_geyser1`

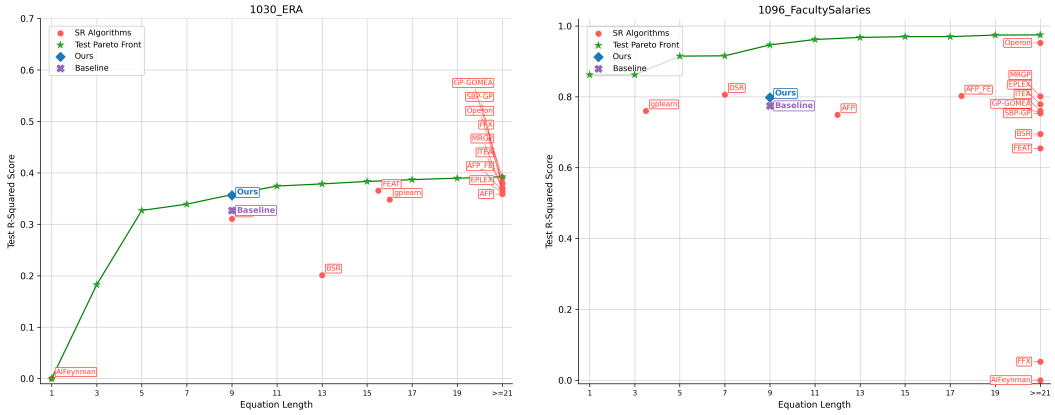


(e) `1027_ESL`



(f) `1029_LEV`

Figure 5: Test Pareto fronts and search results for additional datasets (4/5).



(a) `1030_ERA` (b) `1096_FacultySalaries`

Figure 6: Test Pareto fronts and search results for additional datasets (5/5).