BEHAVIOURAL PARTITIONS: EVOLVING PIECEWISE LAWS FROM DATA

Anonymous authors
Paper under double-blind review

ABSTRACT

Many empirical laws feature a symbolic formula that only holds within some region of the input space. Learning such laws from data is especially challenging, because different formulas apply in each region and the regions themselves are unknown. The challenge is to learn both at the same time: the regions and the formula that holds within each of them. We propose BEPART, a model that learns a symbolic representation of the regions and of the law governing each region. To this end, BEPART searches for a set of gates, one for each region, compared by an argmax, so that every input is routed to exactly one symbolic formula, the law of its region. In contrast to other mixture-of-experts approaches, BEPART finds a symbolic representation of both: the gates and the law of each region. BEPART evolves whole partitions by a genetic algorithm under Pareto selection on error and complexity. On a new suite of 38 regime problems with known partitions and laws, spanning synthetic recipes, piecewise physics and hybrid dynamical systems, BEPART recovers the partition (ARI 0.91) and the complete piecewise law in 53% of the runs, where model trees and piecewise-affine regressors find isolated affine laws but never the partition, mixtures of experts recover the partition only roughly and no laws, and the best single-expression system with conditional primitives reaches an ARI of 0.63 and 17% piecewise law recovery. With a single region, the task becomes classical symbolic regression. BEPART shows its strength in this case as well: under the SRBench evaluation cap it attains the best median test R^2 and mean rank against the 21 published methods of the SRBench black-box suite, and on the SRBench ground-truth problems it recovers more laws symbolically than any published method at noise levels up to 1%.

1 INTRODUCTION

Many empirical laws hold only within a regime. The drag on a sphere follows Stokes' law at low Reynolds number, an empirical correlation in the transition range and a constant coefficient in the turbulent regime. A transistor's current obeys different equations in its cut-off, triode and saturation regions, separated by an oblique boundary in the space of its two voltages. Discovering such laws from data means discovering two things at once: *where* the regimes are and *what* holds in each of them. Symbolic regression (SR) recovers closed-form laws from data, and its best systems now rival gradient-boosted trees on tabular benchmarks (La Cava et al., 2021). But an SR system searches for one expression. Asked for a piecewise law, it must encode the switching inside the formula, and the result is rarely simple, exact or both. Methods that do partition the input space have the opposite limitation. Model trees and piecewise-affine regressors recover a partition but fit only linear leaves, mostly with axis-aligned boundaries (Quinlan, 1992; Bemporad, 2023). Mixtures of experts blend their experts with a soft gate, so their partition is neither explicit nor symbolic (Jacobs et al., 1991). The piecewise SR methods take the partition from a decision tree rather than searching it with the experts (Zhang et al., 2022), or restrict the splits to single thresholds (Doquet, 2025).

We propose BEPART, a model in which *both* the partition and the laws are symbolic and are searched for jointly. A BEPART model is a set of pairs (c_j, f_j) of expression trees. The gates c_j are compared by an argmax that routes every input to exactly one expert f_j (Figure 1). The gates define a *behavioural partition* of the input space whose cells can have arbitrary boundaries, and each cell owns one closed-form law. We evolve whole partitions with a genetic algorithm under Pareto selection on error and complexity, and the paper is about what it takes to make this joint search work.

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.

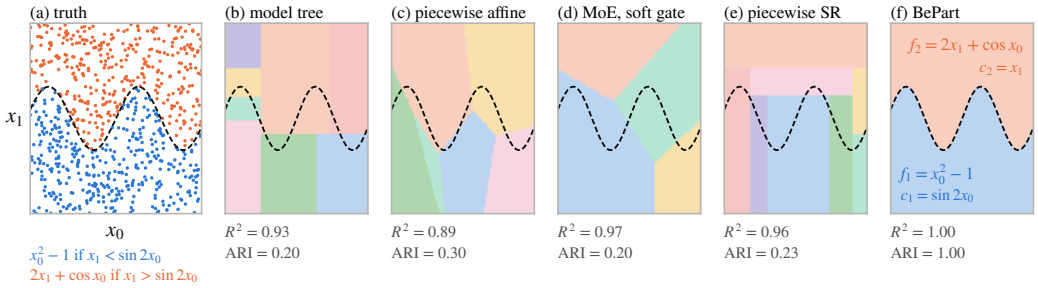

Figure 1: A law that switches along a sine. **(a)** The truth: one law below the sine curve, another above it. **(b–e)** The partitions the baselines find: no finite set of axis-aligned or affine splits reproduces the sine, so their cells cut across the true boundary. **(f)** The BEPART model: two symbolic gates (c_1, c_2) compared by an argmax recover the true partition and the true laws (f_1, f_2).

Contributions.

- A model class for regime regression, argmax-gated symbolic experts, and a search that evolves whole partitions with NSGA-II selection over error and complexity (Sections 2 and 3.1).
- Mechanisms that make the search practical: experts kept least-squares calibrated on their own cells and gate creation that builds new components from the data through fitted gates in an expectation–maximisation (EM) style step (Sections 3.2 and 3.4).
- A regime suite of 38 problems with known partitions and laws (synthetic recipes, piecewise physical laws, hybrid dynamical systems) with metrics for partition and per-regime law recovery (Section 5.1).
- Evidence. On the regime suite BEPART recovers the partition (ARI 0.91) and every law of a problem in 53% of the runs, against an ARI of 0.63 and 17% for the best baseline. As plain SR under the SRBench evaluation cap it has the best median test R^2 and mean rank against the 21 published methods on the SRBench black-box suite and recovers more laws symbolically than any of them on the ground-truth problems at noise levels up to 1% (Section 5).

2 BEHAVIOURAL PARTITIONS

Let $\{(x_i, y_i)\}_{i=1}^n \subset \mathbb{R}^d \times \mathbb{R}$ be a dataset whose target follows a different closed-form law in each of k regions of the input space, i.e. $y = g_j(x)$ for x in region j. We call the regions *regimes* and the g_j their *laws*. Nothing about them is known, not even k, and the goal is to recover all three: the number of regimes, a partition of the input space that matches them, and one expression per cell that is symbolically equivalent to the law of that regime. Classical symbolic regression is the case $k = 1$.

Model. A BEPART model, a *behavioural partition*, represents a partition and its laws with expression trees. It is a set of k *components* $\{(c_j, f_j)\}_{j=1}^k$, each a pair of trees over x: a *gate* c_j and an *expert* f_j. The gates are scores that compete: every input is routed to the component whose gate scores highest, the prediction is the expert of that component, and the *cell* $\mathcal{X}_j$ of component j is the set of inputs routed to it,

$$j^*(x) = \arg\max_j c_j(x), \qquad \hat{y}(x) = f_{j^*(x)}(x), \qquad \mathcal{X}_j = \{x \in \mathbb{R}^d : j^*(x) = j\}. \tag{1}$$

Both trees belong to the same space of symbolic expressions $c, f : \mathbb{R}^d \to \mathbb{R}$: internal nodes are the primitives $+, -, \times, \div, \sin, \cos, \exp, \log, \sqrt{\cdot}, |\cdot|$, a power and negation; leaves are input variables, constants and an *affine leaf* $w^\top x + b$, a single tree position that holds a linear form in all inputs. The affine leaf makes every linear gate or expert one node deep, which is what lets the search fit gates and experts by least squares and insert them as a whole (Section 3.2).

Objective. A model $M = \{(c_j, f_j)\}_{j=1}^{k}$ is scored by the mean loss of its routed predictions and by the size of its trees,

$$E(M) = \frac{1}{n}\sum_{i=1}^{n} \ell\big(\hat{y}(x_i), y_i\big), \qquad C(M) = \sum_{j=1}^{k} \big(|c_j| + |f_j|\big), \tag{2}$$

with the squared error as ℓ throughout and $|t|$ the number of nodes of tree t (Appendix A.2). We search for the Pareto front of (E, C) rather than a single model, and report one front member chosen on held-out data (Section 3.3). Hard argmax routing rather than a softmax blend is deliberate: every prediction is the value of exactly one formula, and the partition is an explicit object that can be compared with the true regimes.

3 Approach

BePart evolves whole partitions. The data are split once: a fraction of 0.2 is held out as the validation split and the search runs on the rest. A population of behavioural partitions is improved by a genetic algorithm with NSGA-II selection over (error, complexity) (Deb et al., 2002), so that the population approximates the whole error–complexity front, and the reported model is chosen from that front by its error on the validation split, which the search never sees (Section 3.3). The outer loop is standard evolution (Section 3.1). What makes the search tractable are two additions to it. Gate-creation proposals build new components from the data, since random variation is very unlikely to produce a useful gate-expert-pair in one shot (Section 3.2), and every child's experts are calibrated on their own cells by least squares at no search cost (Section 3.4). Hyperparameters in the following are named where they matter, with their default value in square brackets, and are shaded (μ, G, ...) so that they stand out from the other symbols. A full list of the mechanisms and their details can be found in Appendix A.

3.1 Evolving whole partitions

The population of μ [300] members starts from random partitions with a few components, their gates and experts random trees, plus one single-component model whose expert is the least-squares linear fit to all the data, so the search never starts below a linear model, and runs for G [50] generations. Children are produced by crossover and mutation on two levels: On the partition level, crossover swaps whole components between two parents, and mutation adds, removes or splits a component. A split clones a component and perturbs the clone's gate, so that the clone carves a part out of a proven expert's cell. On the expression level, gates and experts undergo the usual subtree crossover and subtree, point and hoist mutation within a depth cap. An expression edit modifies a gate with a probability that grows with the share of the error the gates cause, measured against the oracle error of Section 3.2 (a credit assignment, Appendix A.2). Every child is scored, and parents and children are truncated back to μ members by non-dominated sorting on (error, complexity) with crowding distance. The complexity objective is what keeps partitions honest: a component survives only if the error it removes pays for the nodes it adds. Besides children, the population receives *proposals*: candidates built from its best members by the mechanisms of Sections 3.2 and 3.4, injected every few generations and judged by the same selection as everything else.

3.2 Gate creation

Selection pressure on gates is weak, for three reasons. A gate score has no meaning on its own, only the argmax across the gates of a member does: an edit that reroutes no training point leaves the error exactly unchanged and is invisible to selection, and one that does reroute points changes the error by a jump. A new component lowers the error only if its gate carves out a cell *and* its expert fits that cell. Either without the other adds complexity for no gain and is dominated at the next selection. Random variation therefore finds a second regime occasionally and a fourth almost never. BePart builds gates from the data in three ways, all applied to the best members of the population, and all aimed at the same target, the *oracle assignment*, which routes every training point to the expert that fits it best, with the *oracle error* as its routed error:

$$o_i = \arg\min_j \ell\big(f_j(x_i), y_i\big), \qquad E_{\text{oracle}} = \frac{1}{n}\sum_i \ell\big(f_{o_i}(x_i), y_i\big) \;\le\; E. \tag{3}$$

Given the experts, it is the best partition the model could express, and E_{oracle} the error the model would have with perfect gates; the gap $E - E_{\text{oracle}}$ is the share of the loss caused by the partition.

The EM gate step. Freezing the experts of a member freezes its loss matrix $L_{ji} = \ell(f_j(x_i), y_i)$, and the routed error of *any* set of gates is then $\frac{1}{n}\sum_i L_{j^*(x_i),\,i}$: a candidate gate costs one gate expression evaluation and an argmax, no expert evaluation is needed. A short hill climb over the gates therefore acts as the maximisation step of an EM scheme, the gates being improved with the experts held fixed: a component j is drawn, a candidate for c_j is proposed, and it replaces c_j if the routed error drops. Candidate gates are of three kinds: local edits of c_j, gates borrowed from elsewhere in the population, and the *fitted gate*, which is aimed directly at the oracle assignment. Let $O_j = \{i : o_i = j\}$ be the training points the oracle assignment gives to component j, and $s_j(x_i) = \max_{l \neq j} c_l(x_i)$ the score of its strongest competitor at each point. A gate wins the argmax exactly on O_j if it exceeds s_j there and stays below it elsewhere. We want the candidate to clear its competitors by a margin of one on O_j and to lose by one everywhere else, i.e. it is fitted to the target

$$t_i = \begin{cases} s_j(x_i) + 1 & i \in O_j, \\ s_j(x_i) - 1 & \text{otherwise}, \end{cases} \tag{4}$$

by ordinary least squares over a quadratic score,

$$\min_{w,v,b} \sum_{i=1}^{n} \left(w^\top x_i + v^\top (x_i \circ x_i) + b - t_i \right)^2, \tag{5}$$

where $x \circ x$ is the element-wise square. The solution becomes the candidate gate. Of the coefficients $(v_1, \ldots, v_d)$ of the quadratic terms we keep only the q_{EM} [3] largest in absolute value so that the gate stays small, i.e. the resulting gate is $w^\top x + b + \sum_{m \in S} v_m x_m^2$ with S the indices of the q_{EM} largest $|v_m|$. Its linear part $w^\top x + b$ is a single affine leaf. Being quadratic in x, the gate can fit the boundaries such as the shape of a circular or elliptic region exactly.

The residual split. Mutation can add a component to any member (Section 3.1), but the gate and the expert it adds are random. The residual split is the aimed alternative, proposed for the best front members every few generations: we take the training points the member fits worst, choose an expert from those the population evaluated recently, with the lowest loss on exactly those points. An affine gate is aimed to route to those points with the target of Equation (4), the worst-fit points taking the role of O_j, so that the new component wins the argmax where it helps, and nowhere else. An EM step refines the gate before the proposal is scored and whether it survives is left to selection. For the best member the split is chained (split, EM, split, … up to the component cap) and the best link of the chain is proposed, because a problem with several regimes needs several components added together, which single proposals almost never achieve.

Merging. Merges are proposed as well: for a good member with k components, each of the k models obtained by removing one of its components is a proposal. Removing a component hands its cell to whichever remaining gates win there. When the expert that takes over predicts nearly the same values on that cell, the predictions barely change while the complexity strictly drops, and selection accepts the trade. In case a different law takes over, the error rises and the proposal is dominated.

3.3 Choosing k and the reported model

Every member of the final front is scored on the validation split, and the reported model is the least complex member whose validation error is indistinguishable from the best member's. A paired standard-error rule is applied on the per-point validation losses, together with a cap on the accuracy a simpler model may cost (Appendix A.5). This rule is what decides k: a model keeps its extra cells only if they lower the validation error by more than the noise of that measurement. The number of components is capped at K [8]. After selection, the numerical parameters of the chosen model's experts are refit on all data while the gates are untouched.

3.4 THE ENGINE IN BRIEF

Two further ingredients keep the experts accurate without search effort. (1) Every child's experts are calibrated on their own cells the moment the child is born: an expert that is a sum of terms gets all its coefficients refit by least squares, with the terms as the features of a linear regression as in multiple-regression genetic programming (Arnaldo et al., 2014), and every expert is linearly scaled, $af_j + b$, as in Keijzer (2003), but on its cell. The calibrated child replaces the raw one when its error drops clearly. (2) The experts of the best members also grow additively. Every few generations, for each expert f_j of a member, we take its residual $y_i - f_j(x_i)$ on the training points of its own cell and search for an expression that the run has recently evaluated and that has the largest absolute correlation with that residual. That expression g is added to the expert: $f_j + a\,g$. All coefficients of the grown expert are refit by least squares as in (1), and the grown model is proposed next to the original (Appendix A.3). Both mechanisms are cheap because every expression evaluation is cached: buffers hold the gate scores and the expert function values with their per-point losses, so scoring a member, its oracle, a refit and a candidate gate are all operations over cached rows (Appendix A.6). Budgets, restarts and the configurations are in Appendix A.7.

4 RELATED WORK

Symbolic regression. Genetic programming (GP) remains the dominant approach to SR (Koza, 1994), with Operon (Burlacu et al., 2020), GP-GOMEA (Virgolin et al., 2021) and PySR (Cranmer, 2023) as its strongest systems. Linear scaling (Keijzer, 2003) and least-squares fits of additive terms are standard ingredients that we apply per cell. Neural and pretrained approaches include deep symbolic regression (Petersen et al., 2021), transformers trained on synthetic formulas (Kamienny et al., 2022) and parametric families (Scholl et al., 2025). AI Feynman (Udrescu & Tegmark, 2020) recovers laws by recursive decomposition. SRBench (La Cava et al., 2021) fixed the evaluation protocol we follow, its successors added interpretability judging and runtime caps (de França et al., 2025; Imai Aldeia et al., 2025), and SRSD re-sampled the Feynman problems with realistic ranges (Matsubara et al., 2024). None of these benchmarks contains piecewise targets, which is why we built the regime suite.

Piecewise and regime symbolic regression. Ly & Lipson (2012) is the closest ancestor of our model: an EM procedure clusters the data softly, an SR engine fits one expression per cluster, and symbolic inequality guards are learned afterwards from the cluster memberships. Gates and experts are thus symbolic, but they are found in separate stages and the partition is not searched for jointly with the laws. CluBS (Zdankin et al., 2026) grows clusters one at a time around the points an SR model explains and assigns every training point to its best expert. The partition is defined by the experts and no gate is learned, so new points are routed by a post-hoc classifier, which its authors report to cost much of the oracle accuracy. PS-Tree (Zhang et al., 2022) partitions the input space with a CART tree and fits, in every leaf, a ridge regression over a shared set of GP-evolved features, so its boundaries are axis-aligned and its leaf models are not free-form expressions. UPSR (Doquet, 2025) adds Heaviside "piece" operators with evolved thresholds to a GP expression, which encodes exclusive or stacked pieces inside one formula but limits boundaries to thresholds on inputs. Mitrai et al. (2026) learn symbolic decision trees whose splits and leaves are sparse combinations of a fixed library of candidate terms by mixed-integer programming: oblique gates, but experts bound to that library. Hybrid-automaton learners recover modes from time series by segmentation and per-segment SR (Plambeck et al., 2024) or by sparse regression over such a library (Mangan et al., 2019; Liu et al., 2024; Jin et al., 2021). Our hybrid-system problems are sampled in state space so that modes become regimes of a static regression problem. Conditional primitives in ordinary GP are the cheap alternative and are a baseline.

Partition regressors and mixtures. Model trees (Quinlan, 1992), their evolutionary and oblique variants (Barros et al., 2011; Czajkowski & Kretowski, 2013) and piecewise-affine regression with polyhedral regions (Bemporad, 2023) recover partitions with linear leaves. Mixtures of experts (Jacobs et al., 1991; Jordan & Jacobs, 1994) learn a soft gate jointly with the experts. Cluster-wise regression (Späth, 1979; Park et al., 2017) assigns points to regressions without any gate, so routing new points needs a separate classifier. Switching dynamical systems discover regimes in time series with neural or linear dynamics per regime (Linderman et al., 2017; Dong et al., 2020).

5 EXPERIMENTS

5.1 SETUP

Suites. The *regime suite* (Appendix C) has 38 problems with known partitions and laws, so that both can be scored: 24 synthetic problems that vary the boundary type, the number of regimes and the dimension, 8 piecewise laws of engineering and 6 hybrid dynamical systems. Seven are taken or adapted from earlier work (Ly & Lipson, 2012; Mangan et al., 2019; Mitrai et al., 2026; Schmidt & Jordan, 2020), six built with the constructions of Doquet (2025). The training targets carry 0, 1% or 10% noise. The two *SRBench* tracks (La Cava et al., 2021) test the method as a plain symbolic regressor: the black-box track with 122 regression datasets and the ground-truth track with 130 physics formulas at four noise levels, under the SRBench protocol and budgets and against its published results (Appendix B). Every experiment uses 10 seeds. OpenML-CTR23 results are in Appendix D.

Baselines. On the regime suite we compare with every family of method that produces a partition and has runnable code (Appendix B): PS-Tree (Zhang et al., 2022) with its adaptive and with a plain CART partition, a linear model tree, the piecewise-affine regressor PARC (Bemporad, 2023), mixtures of linear and of MLP experts (Jacobs et al., 1991), a mixture of linear regressions fitted by EM with a logistic router, CluBS (Zdankin et al., 2026) with its PySR backend and MLP gate, and, as non-partitioning references, PySR (Cranmer, 2023) without and with conditional primitives (the latter scored on the partition its branch conditions define, Appendix B), Operon (Burlacu et al., 2020) and XGBoost (Chen & Guestrin, 2016). PySR and Operon get 10 minutes or 500,000 evaluations per fit; CluBS, which has no budget knob, and the others run to completion, all with their documented defaults, nothing is tuned per dataset.

Configurations. BEPART is run at $k =$ auto (up to K [8] components, selected as in Section 3.3) and at $k = 1$. Four configurations differ only in their budget (Table 4): `default` (one run, one to three seconds per fit), `racing` (eight racing starts and an end-merge, 5–10 s), `srbench-cap` (six restarts, the SRBench evaluation cap) and `max` (eight restarts, one hour, no evaluation cap; Appendix D only). The regime suite uses `default` and `srbench-cap`, the ground-truth suite the same two at $k = 1$ and `default` also at $k =$ auto; symbolic rates are reported at $k = 1$ only (Appendix B).

Metrics. As before, we call the true regions of a problem its regimes and the test points the fitted model routes to one component a cell (Section 2). On the regime suite we report three metrics. (1) The *adjusted Rand index* (ARI) compares cells with regimes: it is 1 when they coincide and 0 when they agree no more than by chance, as for a model with a single cell. (2) The *law-recovery rate* (LRR) is the share of the k true regimes whose law the model found. Every regime is first paired with one cell. The pairing is one to one and maximises the number of test points that regimes and their cells share (Hungarian matching), and a regime stays unpaired when no remaining cell shares a point with it. A paired regime counts as recovered when the expert of its cell equals its law up to an additive or a multiplicative constant, which is the SRBench check with constants rounded to three significant digits. LRR asks for the laws only: one shared point is enough for a pairing, so the boundaries may be wrong. (3) The *piecewise-exact rate* (PER) is the share of fits that get the whole model right: exactly k cells, all k laws recovered, and more than 99% of the test points in the cell paired with their regime. With 250 test points per problem, at most two may be routed wrongly.

On SRBench we report test R^2 and model size as SRBench defines them, the size being the node count of the exported expression, not the complexity C of Equation (2) (Appendix B). We also report the *mean rank*: on every dataset the methods are ranked by their median test R^2, rank 1 for the best, and a method's mean rank is the average of its ranks over the datasets, so that no single dataset with large differences dominates. On the ground-truth suite we report the symbolic solution rate and the accuracy solution rate ($R^2 > 0.999$). All fits ran on one pinned core of a Xeon E5-2650 v4 cluster node.

Table 1: Regime suite (38 problems, 10 seeds). ARI between predicted and true regimes, law-recovery rate (LRR) and piecewise-exact rate (PER), all means over problems and seeds, median test R^2 and median wall-clock per fit at noise 0. PySR (conditionals) is scored on the partition its branch conditions define; –: no partition (ARI) or no closed-form expert per cell to check (LRR, PER). A fit that produced no model within its cap counts as 0 on all three recovery metrics (18% of the CluBS fits, 2 PS-Tree fits).

	target noise 0				target noise 0.1				
Method	ARI	LRR	PER	R^2	ARI	LRR	PER	R^2	time (s)
BEPART default (k=auto)	0.79	0.46	0.28	1.000	0.75	0.05	0.00	0.999	2.6
BEPART srbench-cap (k=auto)	0.91	0.72	0.53	1.000	0.82	0.06	0.00	0.999	27
BEPART default (k=1)	0.00	0.00	0.00	0.978	0.00	0.00	0.00	0.973	1.6
PySR (conditionals)	0.63	0.29	0.17	0.999	0.56	0.16	0.02	0.992	603
MoE (linear)	0.46	0.00	0.00	0.976	0.45	0.00	0.00	0.976	6.5
Linear model tree	0.36	0.21	0.00	0.984	0.32	0.01	0.00	0.984	0.5
Mixture of linear regr. (EM)	0.35	0.13	0.05	0.901	0.28	0.00	0.00	0.852	0.1
MoE (MLP)	0.34	–	–	0.993	0.33	–	–	0.991	15
PARC	0.29	0.14	0.00	0.985	0.29	0.01	0.00	0.958	5.6
PS-Tree (CART)	0.29	–	–	0.999	0.29	–	–	0.989	13
PS-Tree	0.26	–	–	0.987	0.19	–	–	0.986	17
CluBS (PySR, MLP gate)	0.15	0.06	0.03	0.983	0.09	0.01	0.00	0.987	4,979
XGBoost	–	–	–	0.991	–	–	–	0.979	0.1
PySR	–	–	–	0.985	–	–	–	0.977	603
Operon	–	–	–	0.976	–	–	–	0.975	36

5.2 REGIME DISCOVERY

Table 1 summarises the regime suite. **Accuracy does not separate the methods, the partition does.** Every method fits the data well, with a median test R^2 above 0.97 for all but the EM mixture, yet few find the regimes behind the fit. BEPART at k = auto recovers the partition (ARI 0.91) and the complete piecewise law in 53% of the runs on noiseless data, and already its `default` configuration, at under three seconds per problem, is ahead of every baseline on all three recovery metrics. **One baseline comes close, and the others fail for structural reasons.** PySR with conditional primitives reaches an ARI of 0.63 and a PER of 17%: a single expression with an if-then-else primitive does find regimes when its search is given ten minutes, but fewer of them, with less agreement and at two hundred times the cost of `default`. The remaining baselines lack expressiveness, not budget: model trees and PARC place axis-aligned or polyhedral boundaries that fragment curved regimes, so their affine leaves find affine laws but never the true partition, PS-Tree splits to improve the fit of features that all its leaves share, not where the law changes, and mixtures of experts find a partition but no laws. **Noise separates the two halves of the task.** Under 10% target noise the partitions are still found, but exact law recovery almost vanishes for every method, because the constants no longer round to the truth; scored per regime by accuracy against the true law instead ($R^2 > 0.999$, Appendix D), `srbench-cap` still has 77% of the laws right at 1% noise and 41% at 10%. PySR with conditionals holds up best on exact single laws there, better than any configuration of ours. Per family (Appendix D), the synthetic and hybrid-system problems are recovered best and the physical laws least.

5.3 WHAT MAKES IT WORK

Figure 2 removes one mechanism at a time from the `default` k = auto configuration on the regime suite and on 14 SRBench black-box datasets (Appendix D). **Calibrated experts matter most.** Removing the expert refit is the largest loss on every metric, and on noiseless data it costs two fifths of the recovered laws. An expert whose coefficients are never calibrated on its own cell cannot win points from a neighbouring cell, so the partition cannot improve either. **Gates have to be built, not found.** The EM step, its fitted gate, additive growth and the residual split are the next largest contributions, in that order (the split's aimed gate is no better than a random one). Removing the

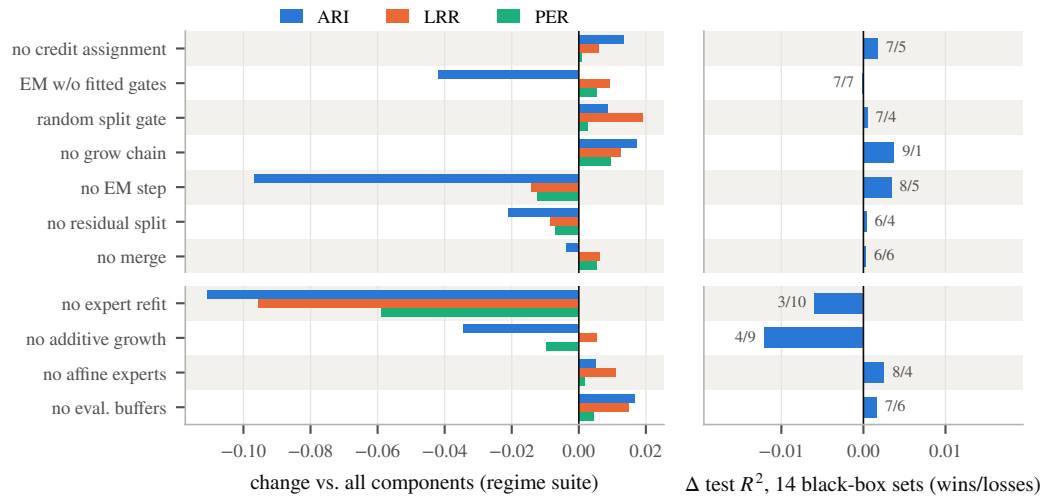

Figure 2: Ablation of the `default` k = auto configuration. **Left:** change in ARI, LRR and PER on the regime suite (38 problems, 10 seeds, mean over noise levels) when one mechanism is removed. **Right:** change in test R^2 on 14 SRBench black-box datasets (median of the paired per-dataset difference; wins/losses next to each bar). The upper seven variants remove a mechanism of the partition side, the lower four one of the expert side.

grow chain or the merge proposals does not hurt, and credit assignment has no measurable benefit. **The mechanisms that find regimes cost no accuracy on data without known regimes.** On the black-box datasets, removing any partition-side mechanism changes the median R^2 by at most 0.005. The only removals that cost accuracy there are additive growth and the expert refit.

5.4 Symbolic regression as the special case

With one component, BePart is a competitive SR system. Under the SRBench evaluation cap (Table 2) the `srbench-cap` configuration at $k = 1$ has the best median test R^2 and the best mean rank against the 21 published methods (4.6 against 4.9 for Operon). The margin is not significant: the gap to Operon is below the Nemenyi critical difference of 2.8, and fit for fit it wins 47 datasets against Operon and loses 61; we claim the best median and rank and nothing more. The cheaper configurations give up little (`racing`: 0.916 in five seconds per fit, `default`: about one second). Without the cap (`max`, one hour) the median is 0.946 at $k = 1$, a value that one minute already reaches with about the cap's number of evaluations (Tables 5 and 14). **Allowing several cells costs little.** The k = auto rows trail their $k = 1$ counterparts only slightly in median R^2, match or beat them in mean rank at the two cheaper budgets, and keep about three cells at every budget rather than collapsing to one expression (Table 2). **More laws are recovered than by any published method at noise levels up to 1%** (Figure 3). At $k = 1$ and the SRBench evaluation cap, BePart recovers 50.5% of the 130 laws symbolically on noiseless data against 49.5% for AI Feynman, the best published method, and stays ahead of every published method up to 1% noise; at 10% noise DSR and gplearn recover more (Table 15). Robustness to noise, sample size, irrelevant inputs and extrapolation is studied in Appendix E.

6 Conclusion

We introduced BePart, a model for regime regression whose partition and laws are both symbolic: gates compared by an argmax, each owning one expert. Evolving whole partitions works when the experts are kept calibrated on their cells and when new regimes are built from the data rather than left to chance. On 38 regime problems it recovers partitions and exact laws where model trees and mixtures of experts recover at most a rough partition and no complete law, and PySR

Table 2: SRBench black-box suite (122 datasets, 10 seeds): median R^2 overall and on the Friedman and non-Friedman subsets, mean rank (each configuration of ours ranked alone among the baselines, Appendix D), median model size, mean cells per fit and median training time per fit; baselines with a median R^2 above 0.85 (all in Table 14). †: published SRBench v2 run (Xeon E5-2690 v4, 2.6 GHz; ours: E5-2650 v4, 2.2 GHz). *: includes SRBench's cross-validated search over the method's settings (Appendix B); ours are single untuned fits. *at its defaults*: run by us without that search.

Method	R^2 (all)	R^2 (Fr.)	R^2 (non-Fr.)	rank	size	cells	time (s)
BEPART srbench-cap (k=1)	0.945	0.960	0.751	4.6	57	–	29
Operon†	0.934	0.956	0.695	4.9	50	–	636*
BEPART srbench-cap (k=auto)	0.918	0.951	0.736	5.0	126	3.2	41
BEPART racing (k=1)	0.916	0.940	0.741	5.7	46	–	4.6
BEPART racing (k=auto)	0.914	0.944	0.735	5.3	146	3.1	11
SBP-GP†	0.908	0.940	0.720	5.9	721	–	170,961*
FEAT†	0.895	0.918	0.725	7.5	75	–	6,251*
BEPART default (k=1)	0.890	0.911	0.752	6.3	44	–	1.1
BEPART default (k=auto)	0.888	0.929	0.726	6.1	150	2.9	2.9
EPLEX†	0.868	0.891	0.715	9.0	58	–	4,304*
XGBoost†	0.854	0.877	0.742	7.0	9,641	–	187*
at its defaults, no search	0.835	0.851	0.704	–	–	–	0.15

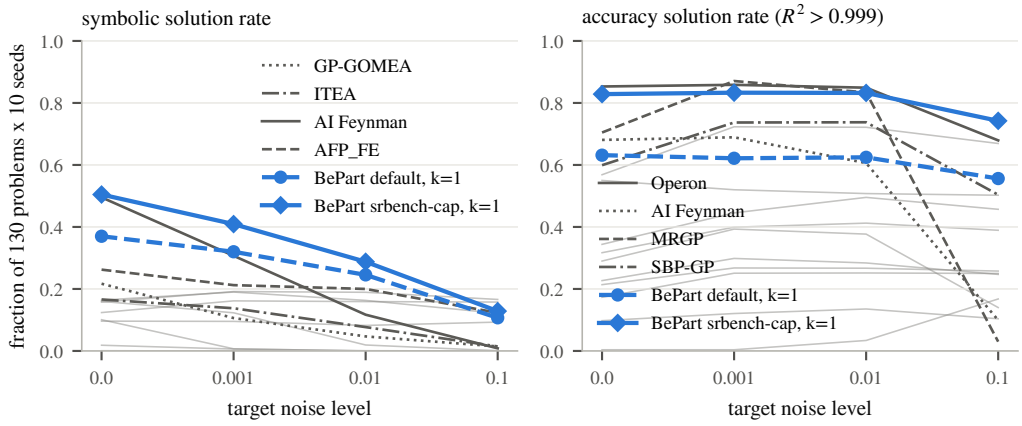

Figure 3: SRBench ground-truth track (130 problems, 10 seeds): share of runs that recover the law symbolically (left) and reach test $R^2 > 0.999$ (right) per noise level; BEPART at $k = 1$ against the 14 published methods (four best labelled).

with conditional primitives recovers fewer, at a far higher cost per fit; with one component it is a competitive SR system on SRBench.

Limitations. Boundaries are hard, which is wrong for smooth transitions. Exact recovery degrades quickly with target noise, as for every symbolic method, because the constants must round to the true ones. On single-law problems the `default` k = auto configuration selects a one-component model in only 41% of the runs: extra cells that fit equally well are not always removed, a weakness of model selection rather than of the search, and the first thing we intend to fix. On the search side, a merge only deletes a component, so two components with the same law but wrong gates are merged only if one already wins the other's cell; refitting the survivor's gate to both cells is the natural extension.

AI USE STATEMENT

AI coding assistants were used in this work for the implementation of the evaluation harness, the analysis scripts that produce the tables and figures, and parts of the library code. All generated code was reviewed and tested by the authors. AI writing assistance was used for drafting and editing the text. All claims, numbers and citations were checked by the authors, who take full responsibility for the content.

ETHICS STATEMENT

This work uses public benchmark data only (PMLB, OpenML and generated data). No human subjects or personal data are involved. The method produces interpretable models for scientific and engineering data and, to our knowledge, carries no risks beyond those of machine learning in general.

REPRODUCIBILITY STATEMENT

The library, the evaluation harness, the experiment definitions (one file per experiment listing suites, seeds, budgets and methods), the regime suite generator, the analysis scripts that produce every table and figure of this paper and a script that reproduces every experiment on one machine are provided in an anonymous repository (`https://anonymous.4open.science/r/bepart/`) together with the aggregated summaries and the per-fit result rows of all experiments, each row carrying the commit of the code that produced it; a snapshot of the code is also attached as supplementary material. Data snapshots are pinned to fixed versions, seeds control both the data splits and the search, and Appendix B lists every protocol detail and configuration knob.

REFERENCES

Ignacio Arnaldo, Krzysztof Krawiec, and Una-May O'Reilly. Multiple regression genetic programming. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pp. 879–886, 2014.

Rodrigo C. Barros, Duncan D. Ruiz, and Márcio P. Basgalupp. Evolutionary model trees for handling continuous classes in machine learning. *Information Sciences*, 181(5):954–971, 2011. doi: 10.1016/j.ins.2010.11.010.

Alberto Bemporad. A piecewise linear regression and classification algorithm with application to learning and model predictive control of hybrid systems. *IEEE Transactions on Automatic Control*, 68(6):3194–3209, 2023. doi: 10.1109/TAC.2022.3183036.

Hans-Georg Beyer and Hans-Paul Schwefel. Evolution strategies – a comprehensive introduction. *Natural Computing*, 1(1):3–52, 2002.

Leo Breiman, Jerome H. Friedman, Richard A. Olshen, and Charles J. Stone. *Classification and Regression Trees*. Wadsworth, 1984.

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. doi: 10.1145/3377929.3398099.

Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 785–794, 2016.

Miles Cranmer. Interpretable machine learning for science with PySR and SymbolicRegression.jl. *arXiv preprint arXiv:2305.01582*, 2023.

Marcin Czajkowski and Marek Kretowski. Global induction of oblique model trees: An evolutionary approach. In *Artificial Intelligence and Soft Computing (ICAISC 2013), Part II*, volume 7895 of *Lecture Notes in Computer Science*, pp. 1–11, 2013. doi: 10.1007/978-3-642-38610-7_1.

Fabrício Olivetti de França, Marco Virgolin, Michael Kommenda, Maimuna S. Majumder, Miles Cranmer, Guilherme Espada, Leon Ingelse, Alcides Fonseca, Mikel Landajuela, Brenden Petersen, Ruben Glatt, Nathan Mundhenk, Chak Shing Lee, Jacob D. Hochhalter, David L. Randall, Pierre-Alexandre Kamienny, Hengzhe Zhang, Grant Dick, Amanda Simon, Bogdan Burlacu, Jaan Kasak, Meera Machado, Casper Wilstrup, and William G. La Cava. SRBench++: Principled benchmarking of symbolic regression with domain-expert interpretation. *IEEE Transactions on Evolutionary Computation*, 29(4):1127–1137, 2025. doi: 10.1109/TEVC.2024.3423681.

Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

Zhe Dong, Bryan Seybold, Kevin Murphy, and Hung Bui. Collapsed amortized variational inference for switching nonlinear dynamical systems. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, volume 119 of *Proceedings of Machine Learning Research*, pp. 2638–2647, 2020.

Guillaume Doquet. Unified piecewise symbolic regression. In *Genetic Programming (EuroGP 2025)*, volume 15609 of *Lecture Notes in Computer Science*, pp. 190–206, 2025. doi: 10.1007/978-3-031-89991-1_12.

Sebastian Felix Fischer, Liana Harutyunyan, Matthias Feurer, and Bernd Bischl. OpenML-CTR23 – a curated tabular regression benchmarking suite. In *AutoML Conference 2023 (Workshop Track)*, 2023.

Guilherme Seidyo Imai Aldeia, Hengzhe Zhang, Geoffrey Bomarito, Miles Cranmer, Alcides Fonseca, Bogdan Burlacu, William G. La Cava, and Fabrício Olivetti de França. Call for action: Towards the next generation of symbolic regression benchmark. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion (GECCO '25 Companion)*, pp. 2529–2538, 2025. doi: 10.1145/3712255.3734309.

Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive mixtures of local experts. *Neural Computation*, 3(1):79–87, 1991.

Kevin Jamieson and Ameet Talwalkar. Non-stochastic best arm identification and hyperparameter optimization. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 51 of *Proceedings of Machine Learning Research*, pp. 240–248, 2016.

Xiangyu Jin, Jie An, Bohua Zhan, Naijun Zhan, and Miaomiao Zhang. Inferring switched nonlinear dynamical systems. *Formal Aspects of Computing*, 33(3):385–406, 2021. doi: 10.1007/s00165-021-00542-7.

Michael I. Jordan and Robert A. Jacobs. Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, 6(2):181–214, 1994.

Pierre-Alexandre Kamienny, Stéphane d'Ascoli, Guillaume Lample, and François Charton. End-to-end symbolic regression with transformers. In *Advances in Neural Information Processing Systems*, 2022.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. LightGBM: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems*, 2017.

Maarten Keijzer. Improving symbolic regression with interval arithmetic and linear scaling. In *Genetic Programming (EuroGP 2003)*, volume 2610 of *Lecture Notes in Computer Science*, pp. 70–82, 2003.

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

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, Patryk Orzechowski, Bogdan Burlacu, Fabrício Olivetti de França, Marco Virgolin, Ying Jin, Michael Kommenda, and Jason H. Moore. Contemporary symbolic regression methods and their relative performance. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks (NeurIPS Datasets and Benchmarks 2021)*, 2021. arXiv:2107.14351.

Scott W. Linderman, Matthew J. Johnson, Andrew C. Miller, Ryan P. Adams, David M. Blei, and Liam Paninski. Bayesian learning and inference in recurrent switching linear dynamical systems. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 54 of *Proceedings of Machine Learning Research*, pp. 914–922, 2017.

Yongtuo Liu, Sara Magliacane, Miltiadis Kofinas, and Efstratios Gavves. Amortized equation discovery in hybrid dynamical systems. In *International Conference on Machine Learning (ICML)*, volume 235 of *Proceedings of Machine Learning Research*, 2024. arXiv:2406.03818.

Daniel L. Ly and Hod Lipson. Learning symbolic representations of hybrid dynamical systems. *Journal of Machine Learning Research*, 13:3585–3618, 2012.

Niall M. Mangan, Travis Askham, Steven L. Brunton, J. Nathan Kutz, and Joshua L. Proctor. Model selection for hybrid dynamical systems via sparse regression. *Proceedings of the Royal Society A*, 475(2223):20180534, 2019.

Oded Maron and Andrew W. Moore. Hoeffding races: Accelerating model selection search for classification and function approximation. In *Advances in Neural Information Processing Systems 6*, pp. 59–66, 1994.

Yoshitomo Matsubara, Naoya Chiba, Ryo Igarashi, and Yoshitaka Ushiku. Rethinking symbolic regression datasets and benchmarks for scientific discovery. *Journal of Data-centric Machine Learning Research*, 1(3):1–38, 2024. arXiv:2206.10540.

Ilias Mitrai, Tongjia Liu, and Gabriel E. Sanoja. Learning regime-dependent governing equations: A symbolic decision tree approach. *AIChE Journal*, pp. e70632, 2026. doi: 10.1002/aic.70632. arXiv:2605.24275.

Randal S. Olson, William La Cava, Patryk Orzechowski, Ryan J. Urbanowicz, and Jason H. Moore. PMLB: A large benchmark suite for machine learning evaluation and comparison. *BioData Mining*, 10(36), 2017.

Young Woong Park, Yan Jiang, Diego Klabjan, and Loren Williams. Algorithms for generalized clusterwise linear regression. *INFORMS Journal on Computing*, 29(2):301–317, 2017. doi: 10.1287/ijoc.2016.0729.

Brenden K. Petersen, Mikel Landajuela, T. Nathan Mundhenk, Claudio P. Santiago, Soo K. Kim, and Joanne T. Kim. Deep symbolic regression: Recovering mathematical expressions from data via risk-seeking policy gradients. In *International Conference on Learning Representations*, 2021.

Swantje Plambeck, Maximilian Schmidt, Audine Subias, Louise Travé-Massuyès, and Görschwin Fey. Usability of symbolic regression for hybrid system identification – system classes and parameters (short paper). In *35th International Conference on Principles of Diagnosis and Resilient Systems (DX 2024)*, volume 125 of *Open Access Series in Informatics (OASIcs)*, pp. 30:1–30:14, 2024. doi: 10.4230/OASIcs.DX.2024.30.

J. Ross Quinlan. Learning with continuous classes. In *Proceedings of the 5th Australian Joint Conference on Artificial Intelligence*, pp. 343–348, 1992.

Ingo Rechenberg. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*. Frommann-Holzboog, Stuttgart, 1973.

Maximilian Schmidt and Jakob Jordan. hal-cgp: Cartesian genetic programming in pure Python, 2020. Zenodo, doi:10.5281/zenodo.3889163.

Michael Schmidt and Hod Lipson. Distilling free-form natural laws from experimental data. *Science*, 324(5923):81–85, 2009. doi: 10.1126/science.1165893.

Philipp Scholl, Katharina Bieker, Hillary Hauger, and Gitta Kutyniok. ParFam – (neural guided) symbolic regression via continuous global optimization. In *International Conference on Learning Representations (ICLR)*, 2025.

Helmuth Späth. Algorithm 39: Clusterwise linear regression. *Computing*, 22(4):367–373, 1979. doi: 10.1007/BF02265317.

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

Marco Virgolin, Tanja Alderliesten, Cees Witteveen, and Peter A. N. Bosman. Improving model-based genetic programming for symbolic regression of small expressions. *Evolutionary Computation*, 29(2):211–237, 2021.

Peter Zdankin, Arne Kummerow, and Torben Weis. CluBS: Clustering behavioural similarity. In *Proceedings of the 32nd ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2 (KDD '26)*, pp. 6385–6396, 2026. doi: 10.1145/3770855.3817875.

Hengzhe Zhang, Aimin Zhou, Hong Qian, and Hu Zhang. PS-Tree: A piecewise symbolic regression tree. *Swarm and Evolutionary Computation*, 71:101061, 2022.

A THE SEARCH ENGINE IN DETAIL

This appendix documents the parts of BEPART that Section 3 only names, and Algorithms 1 to 4 give the search in pseudo-code with the operator mix, probabilities, cadences and acceptance rules of the implementation. Implementation defaults are given in brackets and are the same in every experiment unless a configuration in Appendix A.7 says otherwise.

Hyperparameters. Table 3 lists the hyperparameters, grouped by mechanism, with their defaults, which are the same in every experiment. The configurations of Table 4 change only the number of runs, the generations and the budget. In the text of this appendix a hyperparameter is followed by its default in brackets. The pseudo-code and the equations carry the symbol alone.

Table 3: Hyperparameters of BEPART and their defaults, grouped by mechanism. Hyperparameter symbols are shaded throughout the paper.

Symbol	Meaning	Default
Population and budget		
μ	population size	300
G	generations per run	50
K	component cap	8 (k = auto), 1 ($k = 1$)
k_0	largest number of components of an initial partition (one to k_0)	4
Expression trees and variation (Appendix A.2)		
D	tree depth cap	8
d_0	depth of the initial trees, full or grow	4
d_{f}	depth cap of a subtree that variation creates (subtree replacement, appended term)	3
p_{x}, p_{m}, p_{+}	crossover, mutation and term-append probabilities	0.7, 0.5, 0.1
ω_{x}	probabilities of the four crossover variants	0.25, 0.10, 0.25, 0.40
ω_{m}	probabilities of the four mutation variants	0.60, 0.10, 0.10, 0.20
ω_{e}	probabilities of the three expression edits	0.50, 0.35, 0.15
$[\pi_0, \pi_1]$	range of the gate-mutation probability of Equation (6)	$[0.2, 0.8]$
Elite proposals (Algorithm 2)		
c	cadence of the elite proposals, in generations (affine-expert proposals and constant refits run every $2c$)	5
e_{ref}	best front members that are refined in place (EM step, constants) and offered for merging	10
e_{prop}	best front members that receive a constructive proposal (split, additive growth, affine experts)	5
e_{arch}	best front members scored on validation at a snapshot, and size of the validation archive	10
B	recently used expressions mined per proposal	256
Gate creation (Appendix A.4)		
T_{EM}	trials of one EM step	80
p_{local}	probability that an EM trial is a local mutation of the gate	0.6
p_{mined}	probability that it is a mined gate (otherwise it is the fitted gate)	0.2
q_{EM}	squared terms kept in the fitted gate of the EM step, whose linear part is kept in full	3
ρ	residual-split quantile: the split is aimed at the points whose loss lies above it	0.75
q_{split}	weights kept in the affine gate of a residual split	3
$n_{\min}$, $\phi_{\min}$	smallest set of points a split is aimed at: number of points, share of the training points	10, 0.02
Expert refinement (Appendix A.3)		
δ	relative error drop that accepts an offspring refit	0.1

Table 3, continued.

Symbol	Meaning	Default
δ_{e}	the same for the periodic linear scaling of the best front members	0.01
$m_{\max}$	largest number of terms the term-wise refit handles	8
n_{scale}	smallest cell whose expert is scaled	3
n_{int}	input variables whose pairwise products are the interaction terms of additive growth	6
$\nu_{\sin}$; $\nu_{\cos}$	frequencies at which such a product is offered inside a sine, a cosine	1, 2, 3; 1, 2
$r_{\min}$	smallest correlation with the residual at which a term is added	0.1
n_{add}	smallest cell in which additive growth looks for a term	10
n_{aff}	smallest cell that receives an affine expert	5
T_{ES}	trials of the $(1{+}1)$ evolution strategy on the constants	30
σ_0	its initial step size	0.5
γ_+, γ_-	step-size factor after a success, a failure	1.3, 0.9
κ_0	offset in the perturbation scale $\sigma\,(\lvert\kappa\rvert + \kappa_0)$ of a constant κ	0.1
Model selection and prediction (Appendix A.5)		
v	validation fraction	0.2
τ	selection tolerance of Equation (10)	0.01
ζ	predictions are clipped to the training target range, widened by this share of its width per side	0.1
Evaluation buffers (Appendix A.6)		
N_{gate}	entries of the gate buffer	8192
N_{expert}	entries of the expert buffer	32768
Numerical tolerances		
$\varepsilon_{\mathrm{ridge}}$	ridge factor of the term-wise refit (Equation (7))	10^{-10}
$\varepsilon_{\mathrm{term}}$	the term-wise refit drops a term that contributes less than this share of the target's spread on the cell	10^{-6}
$\varepsilon_{\mathrm{prune}}$	the final refit prunes a term whose removal raises the error by less than this share of the target variance	10^{-9}
$\varepsilon_{\mathrm{tie}}$	a member whose extra validation error is below this share of the target variance counts as tied with the best one in selection	10^{-9}

A.1 Pseudo-code

Names in small capitals are functions of the implementation. The ones not spelled out do what their name says. Shaded symbols are the hyperparameters of Table 3. $E(m)$ and $E_{\mathrm{o}}(m)$ are the routed and the oracle error of a member m, $k(m)$ its number of components, U a fresh uniform random number, $\textsc{Tournament}(P)$ a binary tournament on rank and crowding distance, $\textsc{Top}(F, e)$ the e members of a front F with the lowest error and F_1 the best of them, and $\textsc{RoutedError}(C, L) = \frac{1}{n}\sum_i L_{\arg\max_j C_{ji},\, i}$ the routed error of gate rows C under loss rows L. $C(m)$ and $L(m)$ are the gate-score and expert-loss matrices of a member m on the training points, $C_{ji} = c_j(x_i)$ and $L_{ji} = \ell(f_j(x_i), y_i)$ for its gates c_j and experts f_j. They are read from the evaluation buffers (Appendix A.6), not recomputed. A function returns $\perp$ when it has nothing to propose, and adding $\perp$ to a set adds nothing.

A.2 Expression trees, primitives and the affine leaf

Gates and experts are trees over the primitives $\{+, -, \times, \div, \sin, \cos, \exp, \log, \mathrm{neg}, \sqrt{\cdot}, |\cdot|, \mathrm{pow}\}$, variables $x_0, \ldots, x_{d-1}$ and constants, with a depth cap of D [8]. pow raises its first argument to its second; off its real domain, like log of a negative number, it yields NaN and the candidate is discarded. One further primitive, square, is never drawn by variation and enters a model only through the fitted gates of the EM step (Appendix A.4). A special leaf $\mathrm{lin}(w^\top x + b)$ holds an affine function whose weights are set by least squares and only perturbed by variation. It is charged

Algorithm 1 FIT(X, y): one BEPART fit.

$(X_t, y_t), (X_v, y_v) \leftarrow$ SPLIT(X, y, v)
$\mathcal{R} \leftarrow$ PLANRUNS(config) ▷ single run, restarts or racing, see Table 4
for all $run \in \mathcal{R}$ **do**
  $\mathcal{F}_{run} \leftarrow$ EVOLVE$(X_t, y_t, X_v, y_v,$ BUDGETSHARE$(run))$ ▷ Algorithm 2
if end-merge configured **then** ▷ Table 4
  $\mathcal{F}_{\text{merge}} \leftarrow$ EVOLVE(X_t, y_t, X_v, y_v) started from POOL$(\{\mathcal{F}_{run}\})$
$\mathcal{F} \leftarrow \bigcup_{run} \mathcal{F}_{run} \cup \mathcal{F}_{\text{merge}} \cup$ validation archives
$m^* \leftarrow \arg\min_{m \in \mathcal{F}} \text{err}_v(m)$
$\mathcal{E} \leftarrow \{m \in \mathcal{F} : \overline{\Delta}(m) \le \min(\text{SE}(\Delta(m)),\ \tau\, \text{err}_v(m^*))\}$ ▷ $\Delta_i(m) = \ell_i(m) - \ell_i(m^*)$
$c^* \leftarrow \min_{m \in \mathcal{E}} \text{complexity}(m)$ ▷ the least complex eligible members ...
$m \leftarrow \arg\min_{m \in \mathcal{E}:\, \text{complexity}(m) = c^*} \text{err}_v(m)$ ▷ ... and the most accurate of those
$m \leftarrow$ FINALREFIT(m, X, y) ▷ expert parameters on all data, gates unchanged
return m ▷ predictions clipped to $[\min y, \max y]$, widened by a share ζ per side

Algorithm 2 EVOLVE(X, y, X_v, y_v): one evolution run (symbols as in Section 3).

$P \leftarrow \{$RANDOMPARTITION$(1..k_0$ components$)\}^{\times \mu} \cup \{$LINEARMODEL$(X, y)\}$
SCORE(P)
$\mathcal{A} \leftarrow \emptyset$ ▷ validation archive
for $g \leftarrow 1$ **to** G **do**
  $Q \leftarrow \emptyset$
  for $i \leftarrow 1$ **to** μ **do**
    $m_1, m_2 \leftarrow$ TOURNAMENT(P), TOURNAMENT(P) ▷ the two parents
    $m' \leftarrow m_1$ ▷ the child starts as a copy of the first parent
    if $U < p_{\text{x}}$: $m' \leftarrow$ CROSSOVER(m_1, m_2) ▷ Appendix A.2
    if $U < p_{\text{m}}$ **or** $m' = m_1$ **then** ▷ a child is never a copy of its parent
      $p_{\text{gate}} \leftarrow \pi_0 + (\pi_1 - \pi_0)\big(E(m_1) - E_{\text{o}}(m_1)\big)/E(m_1)$ ▷ Equation (6)
      $m' \leftarrow$ MUTATE$(m',\ p_{\text{gate}})$
    if $U < p_{+}$: $m' \leftarrow$ ADDTERM(m') ▷ append a term to one expert
    $m' \leftarrow$ PRUNEEMPTY(SCORE$(m'))$ ▷ drop components that win no point
    $m' \leftarrow$ SCALE(TERMREFIT$(m'))$ ▷ per cell, each kept iff E drops by $> \delta$
    $Q \leftarrow Q \cup \{m'\}$
  $P \leftarrow$ NSGASELECT$(P \cup Q, \mu)$ ▷ duplicates removed, objectives (error, complexity)
  $F \leftarrow$ FRONT(P) ▷ rank 0, sorted by error
  if $g \bmod c = 0$ **then**
    for all $m \in$ TOP(F, e_{ref}) **do**
      $m \leftarrow$ EMSTEP(m) ▷ Algorithm 3
      $P \leftarrow P \cup$ MERGEVARIANTS(m) ▷ one proposal per component: m without it
    $P \leftarrow P \cup \{$GROWCHAIN$(F_1)\}$ ▷ Algorithm 4, a proposal
    $P \leftarrow P \cup \{$RESIDUALSPLIT$(m) : m \in$ TOP$(F, e_{\text{prop}}) \setminus \{F_1\}\}$ ▷ proposals
    $P \leftarrow P \cup \{$ADDITIVEGROWTH$(m) : m \in$ TOP$(F, e_{\text{prop}})\}$ ▷ add one term
    $\mathcal{A} \leftarrow$ UPDATEARCHIVE$(\mathcal{A},$ TOP$(F, e_{\text{arch}}), X_v, y_v)$ ▷ keeps the e_{arch} best
  if $g \bmod 2c = 0$ **then**
    $P \leftarrow P \cup \{$AFFINEEXPERTS$(m) : m \in$ TOP$(F, e_{\text{prop}})\}$ ▷ affine fit per cell
    for all $m \in$ TOP(F, e_{ref}) **do**
      $m \leftarrow$ LINEARREFIT(m) ▷ kept iff E drops by $> \delta_{\text{e}}$
      $m \leftarrow$ REFITCONSTANTS(m) ▷ T_{ES} ES trials
  if BUDGETEXHAUSTED(): **break** ▷ time or evaluation share
return SIMPLIFY(FRONT$(P))$, $\mathcal{A}$

Algorithm 3 EMSTEP: hill climb on the gates with frozen experts.

function EMSTEP(m) ▷ $c_1, \dots, c_k$ are the gates of m
  $L \leftarrow L(m)$ ▷ expert losses, frozen for the whole step
  $C \leftarrow C(m)$ ▷ gate scores, updated as gates are replaced
  $o_i \leftarrow \arg\min_j L_{ji}$
  $E \leftarrow \text{ROUTEDERROR}(C, L)$
  for $t \leftarrow 1$ **to** T_{EM} **do**
    $j \leftarrow \text{UNIFORM}(1..k)$
    $c' \leftarrow \begin{cases} \text{MUTATEEXPR}(c_j) & \text{w.p. } p_{\text{local}} \\ \text{RECENTGATE}() & \text{w.p. } p_{\text{mined}} \\ \text{FITTEDGATE}(j, o, C) & \text{otherwise, at most once per } j \end{cases}$
    $C' \leftarrow C$ with row j replaced by $c'(X)$
    $E' \leftarrow \text{ROUTEDERROR}(C', L)$
    if $E' < E$ **or** $\big(E' = E$ **and** $|c'| < |c_j|\big)$ **then**
      $c_j, C, E \leftarrow c', C', E'$
  return m ▷ with the accepted gates

function FITTEDGATE(j, o, C)
  $t_i \leftarrow \max_{l \neq j} C_{li} + (+1 \text{ if } o_i = j \text{ else } -1)$
  $(w, v, b) \leftarrow \text{LEASTSQUARES}([X, X^2, \mathbf{1}], t)$ ▷ weights of x, of $x \circ x$, and the bias
  $S \leftarrow \arg\text{top}_{q_{\text{EM}}} |v_m|$ ▷ the indices m of the q_{EM} largest $|v_m|$
  return $w^\top x + b + \sum_{m \in S} v_m x_m^2$

Algorithm 4 RESIDUALSPLIT and GROWCHAIN: creating components from the data.

function RESIDUALSPLIT(m)
  if $k(m) = K$: **return** $\perp$
  $L \leftarrow L(m)$
  $C \leftarrow C(m)$
  $\ell_i \leftarrow L_{j^*(x_i), i}$ ▷ loss of the expert point i is routed to (Equation (1))
  $M \leftarrow \{i : \ell_i > \text{quantile}_\rho(\ell)\}$ ▷ the worst-fit points, a quarter at $\rho = 0.75$
  if $|M| < \max(n_{\min},\ \phi_{\min} n)$: **return** $\perp$ ▷ most of the n losses tie, e.g. an exact fit
  $\mathcal{H} \leftarrow \text{RECENTEXPERTS}(B) \cup \{\bar{y}_M\}$ ▷ recently used experts and the mean of y on M
  $f_{\text{new}} \leftarrow \arg\min_{f \in \mathcal{H}} \frac{1}{|M|} \sum_{i \in M} \ell(f(x_i), y_i)$ ▷ cached losses
  $t_i \leftarrow \max_l C_{li} + (+1 \text{ if } i \in M \text{ else } -1)$ ▷ target of Equation (4) with M as O_j
  $(w, b) \leftarrow \text{LEASTSQUARES}([X, \mathbf{1}], t)$ ▷ weights of x and the bias
  $S \leftarrow \arg\text{top}_{q_{\text{split}}} |w_m|$ ▷ the indices m of the q_{split} largest $|w_m|$
  $c_{\text{new}} \leftarrow b + \sum_{m \in S} w_m x_m$
  $m \leftarrow m \cup \{(c_{\text{new}}, f_{\text{new}})\}$ ▷ a model is a set of (gate, expert) pairs
  $m \leftarrow \text{EMSTEP}(m)$ ▷ refine the aimed gate
  return m

function GROWCHAIN(m)
  $m_0 \leftarrow m$ ▷ the model the chain starts from
  best $\leftarrow \perp$
  while $k(m) < K$ **do**
    $m' \leftarrow \text{RESIDUALSPLIT}(m)$
    if $m' = \perp$: **break** ▷ nothing left to split off: the size guard above fired
    if $E(m') \geq E(m_0)$ **and** best $= \perp$: **break** ▷ a first link that does not help
    $m \leftarrow m'$ ▷ the next split starts from this link
    if best $= \perp$ **or** $E(m') < E(\text{best})$: best $\leftarrow m'$
  return best ▷ the best link, or $\perp$ when no link helped

one complexity node for its bias and two per non-zero weight, a product and a variable, which is about half of what the explicit tree would cost: its weights and the additions joining its terms are not charged, since least squares sets them and no search is spent on them. Point mutation of an affine leaf perturbs its existing weights and never introduces new terms. A single-component model carries a constant gate that is never consulted and is charged one node. Constant folding, identity and affine-collapsing rewrites are applied to the front at the end of a run, each rewrite verified against the training data before it is accepted, and the front then keeps one member per (error, complexity) point.

Initial trees. The gates and experts of the random initial partitions (RANDOMPARTITION in Algorithm 2) are drawn by Koza's half-and-half method (Koza, 1992). The trees of a member are all *full*, every branch reaching depth d_0 [4], or all *grow*, every node being a leaf with probability 0.3 before that depth, so that the trees are irregular and mostly smaller; the method is drawn once per member with probability $1/2$ each. Unlike Koza's ramped variant, the depth is the same for every member. The random gate of the "random split gate" ablation is drawn by the grow method to the same depth.

Variation operators. CROSSOVER of parents m_1 and m_2 picks one of four variants, with the probabilities ω_{x} [0.25, 0.10, 0.25, 0.40] in this order: a fair coin decides for every component of $m_1 \cup m_2$ whether the child takes it; one expert of m_1 is replaced by an expert of m_2; subtree crossover between a gate of m_1 and a gate of m_2; the same between two experts. Between two single-component parents the first and the third variant have nothing to swap and are replaced by the fourth. MUTATE picks one of four, with the probabilities ω_{m} [0.60, 0.10, 0.10, 0.20] in the order listed: an expression edit of one gate (with probability p_{gate}, Equation (6)) or one expert; a random component added; a component removed; a component cloned and the clone's gate edited. An expression edit is, with the probabilities ω_{e} [0.50, 0.35, 0.15] in the order listed, the replacement of a random subtree by a new one of depth at most d_{f} [3], a point change of an operator, leaf or constant, or a hoist, and is rejected when it exceeds the depth cap D [8]. The gate-edit probability is a credit assignment between the two halves of a member: with E its routed error and E_{o} its oracle error (Section 3.2),

$$p_{\mathrm{gate}} = \pi_0 + (\pi_1 - \pi_0)\,\frac{E - E_{\mathrm{o}}}{E}, \tag{6}$$

rising from π_0 [0.2] when all of the loss is expert error to π_1 [0.8] when all of it is partition error, so that a member whose experts are right but whose boundaries are wrong receives gate edits. The ablation (Figure 2) finds no measurable effect of it against a fixed $p_{\mathrm{gate}} = 0.5$.

A.3 EXPERT REFINEMENT FROM CACHED ROWS

Term-wise refit. Applied to every child, right after it has been scored and pruned and before it joins the offspring (TERMREFIT in Algorithm 2). The additive-growth proposal below ends with the same refit. An expert is flattened over its top-level $+/-$ operations into a list of terms $g_1, \ldots, g_m$, their constant multipliers and additive constants being dropped, so that $f_j = \sum_{l=1}^{m} a_l\, g_l + b$. For an expert of two to $m_{\max}$ [8] terms the weights a_l and the bias b are refit to minimise the squared error on the expert's cell, $R_j = \{i : j^*(x_i) = j\}$ being the training points routed to it. Let $A \in \mathbb{R}^{|R_j| \times (m+1)}$ be the design matrix whose row for point $i \in R_j$ is $[\, g_1(x_i), \ldots, g_m(x_i), 1\,]$, and y_{R_j} the targets of the cell. Then

$$(a_1, \ldots, a_m, b)^\top = \theta = (A^\top A + \lambda I)^{-1} A^\top y_{R_j}, \qquad \sum_{i \in R_j} \big(f_j(x_i) - y_i\big)^2 = y_{R_j}^\top y_{R_j} - \theta^\top A^\top y_{R_j}, \tag{7}$$

with a tiny ridge $\lambda = \varepsilon_{\mathrm{ridge}}\, \mathrm{tr}(A^\top A)/(m+1)$, $\varepsilon_{\mathrm{ridge}} = 10^{-10}$, that only matters when terms are collinear and is neglected in the error on the right. A cell needs at least $m+2$ points with finite term values for the refit. The entries of $A^\top A$ and $A^\top y_{R_j}$ are sums over R_j of products of term values, which are read from the buffer, so a refit costs about $|R_j|\, m^2$ multiplications and the solution of an $(m+1) \times (m+1)$ system. Terms whose contribution falls below a share $\varepsilon_{\mathrm{term}}$ [10^{-6}] of the target's spread on the cell are dropped and the rest refit again. The function values of the refit expert on the training points are the combination θ of the cached term rows, $A\theta$ on the cell, so it is never evaluated as a tree. They are written back into the buffers, and its individual terms become mineable

experts for the proposals below. *Acceptance rule.* A refit is not kept unconditionally. With m' the child and $\tilde{m}'$ the child with the refit expert, the refit replaces the expert only if it lowers the routed error of the whole model by more than a fraction δ [0.1],

$$E(\tilde{m}') \;<\; (1-\delta)\,E(m'), \tag{8}$$

and otherwise the child keeps its expert as it was. The bar is deliberately loose: a fragment calibrated on its own cell always buys a little training error, and a small gain should not be allowed to keep extra components alive. Linear scaling below uses the same rule.

Linear scaling. Applied at two points. Every child is scaled directly after its term-wise refit (SCALE in Algorithm 2). In addition, every $2c$ [10] generations the experts of the top e_{ref} [10] front members are scaled again (LINEARREFIT), just before the hill climb on their constants described below. There the scaled member is kept if its routed error drops by more than a fraction δ_{e} [0.01], a lower bar than the δ [0.1] of Equation (8), which applies to children. Every expert is scaled to $af_j + b$, with a and b minimising the squared error on the expert's cell R_j; a cell with fewer than n_{scale} [3] points is left unscaled. This is the term-wise refit with a single term, $m = 1$, where Equation (7) reduces to a few sums. Setting the two derivatives of $\sum_{i\in R_j}(af_j(x_i)+b-y_i)^2$ to zero gives

$$a = \frac{S_{fy}}{S_{ff}}, \qquad b = \bar{y} - a\bar{f}, \qquad \sum_{i\in R_j}\big(af_j(x_i)+b-y_i\big)^2 = S_{yy} - \frac{S_{fy}^2}{S_{ff}}, \tag{9}$$

where $\bar{f}$ and $\bar{y}$ are the means of $f_j(x_i)$ and y_i over R_j, and $S_{ff} = \sum(f_j(x_i)-\bar{f})^2$, $S_{fy} = \sum(f_j(x_i)-\bar{f})(y_i-\bar{y})$ and $S_{yy} = \sum(y_i-\bar{y})^2$ are sums over R_j. These few sums are all the fit needs, and the expert's values are already in the buffer, so a, b and the error of the scaled expert cost one pass over cached numbers. Scaling leaves the gates, and hence the cells, unchanged, so that error is exactly the cell's share of the scaled model's routed error: the acceptance rule of Equation (8) is checked on it first, and the scaled tree is built only when it passes. For a child, existing scale and offset constants are reused, so repeated scaling never grows its tree; the periodic scaling of the front members wraps the expert in $af_j + b$, and the simplification at the end of a run folds such wrappers. Scaling is applied to every expert of the child. It is the only calibration a single-term expert gets, and it leaves a freshly refit multi-term expert unchanged, since that expert is already optimal on its cell.

Additive growth. Two mechanisms let experts grow term by term. The first is an elite proposal, ADDITIVEGROWTH in Algorithm 2, made every c [5] generations for each of the top e_{prop} [5] front members. For every expert of such a member it computes the expert's residual on its own cell and looks for one expression to add; cells with fewer than n_{add} [10] points are skipped. The candidates come from two pools. One is the B [256] most recently used expert expressions. The other is a set of interaction terms built for this expert: the n_{int} [6] input variables most correlated with the residual are taken, all their pairwise products are formed, and each product is also offered inside a sine at the frequencies $\nu_{\sin}$ [1, 2, 3] and inside a cosine at the frequencies $\nu_{\cos}$ [1, 2]. These are structures such as $\sin(2\,x_ix_j)$ that random variation almost never assembles. The candidate whose values correlate best with the residual is chosen, and if that correlation exceeds $r_{\min}$ [0.1] it is appended to the expert as a new term and all coefficients are refit by the term-wise refit, here without the bar of Equation (8), since the new term has no coefficient yet. If the grown model has a lower error than the member it was built from, it is added to the population next to that member.

The second is a mutation, ADDTERM. With probability p_+ [0.1] a child, after crossover and mutation, has a random subtree of depth at most d_{f} [3] or, with equal probability, one of the 64 most recently used expert expressions appended to one of its experts, and the term-wise refit that follows sets the new term's coefficient.

Affine-expert proposals and constants. Two further steps run every $2c$ [10] generations. The first is a proposal, AFFINEEXPERTS in Algorithm 2, made for each of the top e_{prop} [5] front members. It keeps the member's gates, so the cells stay where they are, and replaces every expert by the affine function of the inputs that fits the expert's cell best in the least-squares sense, one affine leaf per cell; a cell with fewer than n_{aff} [5] points keeps its expert. This is what a model tree is, a partition with a linear model in every cell, and it offers selection a simple, well-calibrated version

of a partition whose evolved experts may be complicated and wrong. The proposal is added to the population next to the member it was built from.

The second tunes the numeric constants of the top e_{ref} [10] front members, and the tuned member takes the place of the original. Every expert is first linearly scaled as described above (LINEARREFIT). Then the constants inside the expert trees are improved by a $(1{+}1)$ evolution strategy (REFITCONSTANTS), a stochastic hill climb on the vector of all constants of the member's experts. In each of T_{ES} [30] trials every constant κ is perturbed with Gaussian noise of standard deviation $\sigma\,(|\kappa| + \kappa_0)$, where the offset κ_0 [0.1] keeps a constant near zero moving. The model is scored, and the perturbed constants are kept only if the error dropped. The step size σ starts at σ_0 [0.5], is multiplied by γ_+ [1.3] after a success and by γ_- [0.9] after a failure, never below 10^{-6}, so the search widens while it makes progress and narrows when it does not. This is a success-based step-size control in the spirit of Rechenberg's one-fifth rule (Rechenberg, 1973; Beyer & Schwefel, 2002). The hill climb reaches the constants that least squares cannot, those inside a nonlinearity such as the 3 in $\sin(3x)$. Gate constants are never fitted numerically, because a gate is only compared with other gates and has no target of its own.

A.4 GATE CREATION IN DETAIL

EM step. Run in two places: every c [5] generations on the top e_{ref} [10] front members, where the refined member takes the place of the original, and once on every residual-split proposal before it is scored (EMSTEP in Algorithms 2 and 4). The expert loss matrix L and the gate rows C are read from the buffers and the oracle assignment $\arg\min_j L_{ji}$ is computed once. For T_{EM} [80] iterations a component j is drawn at random and a candidate gate is proposed: with probability p_{local} [0.6] a local mutation of c_j, with probability p_{mined} [0.2] a gate mined from the B [256] most recently used gates, and otherwise the fitted gate, which is tried at most once per component. The fitted gate solves $\min_w \|Aw - t\|^2$ with $A = [X, X^2, \mathbf{1}]$ and target $t_i = \max_{l \neq j} C_{li} + 1$ on the component's oracle points and -1 elsewhere. The pseudo-inverse of A is computed once per dataset, so a fit is one matrix product. Its linear part becomes an affine leaf that is kept in full, and of its squared terms only the q_{EM} [3] with the largest coefficients in absolute value are kept, as square terms. A candidate replaces c_j when the routed error under the frozen L decreases, or stays equal with a smaller gate. Every trial that yields a new candidate gate counts as one fitness evaluation; a draw that repeats the fitted gate of a component, or returns the current gate, is skipped.

Residual split and grow chain. Proposed every c [5] generations: the chain for the best front member and single splits for the other members of the top e_{prop} [5], all added to the population next to the members they were built from (GROWCHAIN and RESIDUALSPLIT in Algorithm 2). Given a member with fewer than K [8] components, the per-point routed loss is computed from the cached rows. M is the set of points whose routed loss lies above the ρ [0.75] quantile of the routed losses, the worst-fit quarter (M must hold at least $n_{\min}$ [10] points and a share $\phi_{\min}$ [0.02] of the training points). The new expert is the expression with the lowest mean cached loss on M among the B [256] most recently used experts and the constant $\mathrm{mean}(y_M)$. The new gate is the least-squares affine fit on $[X, \mathbf{1}]$ to the target $\max_l C_{li} + 1$ on M and $\max_l C_{li} - 1$ off M, truncated to its bias and the q_{split} [3] weights that are largest in absolute value. Unlike the fitted gate of the EM step it has no squared terms, and it enters the model as an explicit tree of its bias and its weight-times-input terms rather than as an affine leaf, so its constants stay open to mutation. The proposal is passed through one EM step and scored. For the best front member the proposal is chained (split, EM, split, ...) until the component cap or until a step no longer helps, and the best link is injected. In the ablation "random split gate" the aimed gate is replaced by a random expression of depth at most d_0 [4].

Merge and pruning. Every c [5] generations, right after their EM step, each of the top e_{ref} [10] front members is offered in merged form (MERGEVARIANTS in Algorithm 2). A member with $k \geq 2$ components yields k proposals, the j-th being the member without its j-th (gate, expert) pair. The remaining gates then share out the cell of the removed component. All k are scored and added to the population, and selection decides which of them survive. A member with a single component yields none. Independently, whenever a model is scored, components that win no training point are removed (PRUNEEMPTY).

A.5 MODEL SELECTION

Selection happens once, at the end of a fit, after every run and the end-merge have finished (Algorithm 1). Every member of the pooled front $\mathcal{F}$ (the final front of every run, the end-merge front and the validation archives) is scored on the n_v held-out validation points. Let M^* be the member with the lowest validation error E^* and, for another member M, let $\Delta_i = \ell(\hat{y}_M(x_i), y_i) - \ell(\hat{y}_{M^*}(x_i), y_i)$ be its extra loss on validation point i, so that $\overline{\Delta} = E_M - E^*$ is its extra validation error and $\mathrm{SE}(\Delta) = \mathrm{sd}(\Delta)/\sqrt{n_v}$ the standard error of that mean. M is *eligible* if

$$E_M - E^* \;\le\; \min\big(\mathrm{SE}(\Delta),\ \tau\, E^*\big), \qquad \tau = 0.01, \tag{10}$$

and the reported model is the least complex eligible member, ties broken by validation error. M^* is eligible by definition, so the rule always returns a model, and it never returns one that is both larger and worse.

Why an eligible member rather than M^*. E^* is the minimum of the validation errors of a whole front, so it is an optimistic estimate of that member's accuracy: the member that comes out in front on n_v points is in part the one that was lucky on them. When $\overline{\Delta}$ is no larger than the noise of the comparison, the data therefore carry no evidence that M^* predicts better than M on fresh points, and accuracy cannot decide between the two. The simpler model is then preferred, because two cells that predict the same thing are one regime split in two: the model is longer to read and its partition no longer matches the true one, and it buys no accuracy for that. This is the one-standard-error rule of Breiman et al. (1984), applied to paired differences rather than to a cross-validation curve.

Why paired, and why the cap. Both members are scored on the same points, so the spread of the losses themselves, which the two share, cancels in Δ and $\mathrm{SE}(\Delta)$ measures only how consistently the two models disagree. A member that splits a cell in two and predicts almost the same thing on both halves has $\Delta \approx 0$ and is collapsed however widely the losses are spread. A component with a small but systematic gain has a $\overline{\Delta}$ that exceeds $\mathrm{SE}(\Delta)$ and survives, where an unpaired standard error, computed from the losses of M^* alone, would hide it. The cap τE^* guards the opposite failure: on a small validation split $\mathrm{SE}(\Delta)$ can be a sizeable fraction of E^*, and without the cap a simpler member could pass the test while costing accuracy that is real.

Tie floor. The tolerance on the right of Equation (10) is never taken smaller than $\varepsilon_{\mathrm{tie}}\,\mathrm{var}(y)$, where $\varepsilon_{\mathrm{tie}} = 10^{-9}$ and $\mathrm{var}(y)$ is the variance of the validation targets. A member is therefore also eligible when its extra validation error $E_M - E^*$ lies below that floor, which means that its validation R^2 is within $\varepsilon_{\mathrm{tie}}$ of that of M^*. The floor exists for noiseless data: a model that fits them exactly has $E^* \approx 0$, the cap τE^* collapses with it, and without the floor no equally exact but simpler member could ever tie.

A.6 EVALUATION BUFFERS AND THE C++ CORE

Expressions are immutable trees, hashed structurally. Three LRU memo tables map an expression to a row over the training data: the gate buffer (N_{gate} [8192] entries) stores the evaluation with NaN mapped to $-\infty$, the expert buffer (N_{expert} [32768] entries) the raw values, and the loss buffer the per-point losses against y with non-finite values mapped to $+\infty$. Scoring a k-component member is an argmax over k cached gate rows, a gather of the routed losses and a column-wise minimum for the oracle. The common $k = 2$ case has a stacking-free fast path and $k = 1$ is a single mean. Refits write their rows straight into the buffers (Appendix A.3), so a refit expert is never evaluated as a tree. A fourth table memoises whole models: it maps a model, hashed structurally like an expression, to its score (its routed and oracle error and the assignment of the points), so a child that is structurally identical to a model scored before is looked up instead of being scored again.

BEPART is implemented twice, as a pure NumPy reference engine and as a C++ core (pybind11) with identical proposal logic. The core stores every distinct subexpression exactly once: building an expression that already exists returns the existing node. Identical subtrees are therefore shared between all models, two expressions are equal exactly when they are the same node, and a buffer lookup costs one pointer hash instead of a walk over the tree. Expressions are evaluated vectorised over the data points. The store is rebuilt from the population and the validation archive once it exceeds two million nodes, which happens repeatedly in the one-hour runs; the buffers, the pools

of recently used expressions and the model memo start empty again and the next generation refills them. Expression evaluation, tree structure, Pareto sorting and model scoring of the two engines are checked against each other exactly by parity tests; the search is checked behaviourally, and the two engines draw different random streams, so runs with the same seed differ. All experiments use the C++ core.

A.7 Budgets, restarts, racing and the configurations

Counting evaluations. Under an evaluation cap we count every fitness value computed for a candidate model: every child, every re-scoring after a refit, every elite proposal, every EM trial and every constant-refit trial, whether or not the expression rows came from the buffers. Hits in the expression buffers therefore consume budget exactly like misses. Caching is visible only in wall-clock time. This is the unit SRBench uses (a candidate model that receives a fitness), and the conservative choice: counting only buffer misses would let us score roughly ten times more candidates under the same nominal cap. The one exception is an exact duplicate: a model that is structurally identical to one scored before takes its score from the model memo table (Appendix A.6) and is not counted, because no fitness is computed for it. We measured how much this leaves out with the reference engine on the 38 problems of the regime suite (`default` configuration): had the duplicates been counted, the evaluation counts would be higher by 9% at $k =$ auto (median, 5–19% across problems) and by 25% at $k = 1$ (20–34%). The wall-clock cap is checked once per generation and the evaluation cap also before every elite proposal step, so the overrun is at most one generation.

Restarts, racing and the end-merge. A fit may consist of several evolution runs on the same training and validation split. Selection (Appendix A.5) then chooses from the fronts of all of them. Table 4 lists the runs of each configuration. *Restarts* are independent runs with different seeds, each of G generations. *Racing* avoids spending full runs on unpromising seeds, as in Hoeffding races (Maron & Moore, 1994) and successive halving (Jamieson & Talwalkar, 2016). It starts independent runs in the same way but stops each of them after a quarter of the generations, $G/4$ rounded down and at least 10. These short runs, the probes, are ranked by the lowest validation error on their fronts. The better half of the probes is then continued for the remaining generations, each from the population its probe ended with (the probes' own fronts and buffers are not carried over). The other half is discarded and contributes nothing, so a race of eight probes ends with four full-length runs.

An *end-merge* can follow the restarts or the race. It is one more evolution run, a short one whose length is given in Table 4, and it does not start from a random population. Its starting population is pooled from the completed runs: every run contributes its final front and its lowest-error members, μ divided by the number of runs of them and at least 10. Duplicates are removed and the pool is cut to μ members in the order the runs contributed them. In this run the gates found by one run can recombine with the experts found by another, which separate runs never allow. Its front joins the fronts of the other runs for selection.

A whole-fit wall-clock budget is handed to the runs as deadline shares (remaining time divided by the runs still to go); an evaluation cap is split evenly over the runs, probes and end-merge included, and a run's unspent share is not handed on. Of this budget, 5% (0.5–60 s) is reserved for the final refit's constant polish, which stops at the deadline with the best constants seen.

Final refit. Once the model has been selected, the validation split is no longer needed for choosing. The numeric parameters of the selected model are therefore fitted once more on all the data, that is, on the training and the validation split together. Five steps run in this order. (1) Every affine-leaf expert is refit by least squares on its cell. (2) The term-wise refit and the linear scaling of Appendix A.3 are repeated, the term-wise refit and the child-style scaling kept whenever they lower the error and the elite-style scaling with its bar δ_{e} [0.01]; the bar δ [0.1] applies only during the search. (3) The constants inside the expert trees are polished, first by Levenberg–Marquardt and then by the $(1{+}1)$ evolution strategy of Appendix A.3. (4) Negligible terms are pruned: a term whose removal changes the error by less than a share $\varepsilon_{\mathrm{prune}}$ [10^{-9}] of the target variance is numerical debris, such as $10^{-16} \sin x_0$. (5) The model is simplified.

Only expert parameters move in these steps. The gates are only simplified by rewrites verified to leave their values on the data unchanged, so the partition is unchanged. Predictions are clipped to

the training target range, widened by a share ζ [0.1] of its width on each side, which bounds the damage of an expert with exp or $\div$ on a few out-of-envelope test points.

Configurations. The configurations used in this paper are listed in Table 4.

Table 4: The BEPART configurations. All share the defaults of Table 3 and differ only in the number of runs, the generations per run G, the end-merge and the budget. The last column gives the time per fit of the $k = 1$ and of the $k =$ auto variant on the SRBench black-box suite (one core, training sets of up to 10,000 rows). As in Tables 5 and 14, each time is the median over all datasets of the median over their 10 seeds.

configuration	runs	generations per run G	end-merge	budget	time per fit, $k = 1$ / $k =$ auto
`default`	1	50	–	none	1.1 s / 2.9 s
`racing`	8 probes of 12 generations, the best 4 continue to 50	50	15 generations	none	4.6 s / 11 s
`srbench-cap`	6 restarts	150	20 generations	500k evaluations (1M on ground truth), shared by all runs	29 s / 41 s
`max`	8 restarts	unbounded	30 generations	1 h wall-clock, shared by all runs	53 min / 53 min

A.8 EFFICIENCY SUMMARY

Table 5 summarises what our configurations cost on the SRBench black-box suite. Its last column shows what the buffers buy: most requests for the row of a gate or of an expert are answered from a buffer and need no expression evaluation. A detailed study of the engine, of what each mechanism costs and buys at equal wall-clock, and of its standing among fast symbolic-regression systems is the subject of ongoing research.

Table 5: Efficiency of the BEPART configurations on the SRBench black-box suite (122 datasets, 10 seeds, single-threaded on one core of an Intel Xeon E5-2650 v4 at 2.2 GHz). Every entry is the median over all datasets of the median over their 10 seeds. Time = wall-clock per fit, including model selection and the final refit; evals = counted evaluations (Appendix A.7); evals/s = evals divided by time; hit rate = share of the requests for a gate or expert row that a buffer answers. The `max` rows are runs of that configuration under a wall-clock budget of one and of five minutes in place of one hour.

Configuration	budget	R^2	time (s)	evals	evals/s	hit rate
BEPART default (k=1)	500k evals	0.890	1.1	20,272	18,546	0.82
BEPART default (k=auto)	500k evals	0.889	2.9	38,104	12,721	0.91
BEPART racing (k=1)	500k evals	0.916	4.6	66,140	14,200	0.82
BEPART racing (k=auto)	500k evals	0.914	11	126,147	11,517	0.92
BEPART srbench-cap (k=1)	500k evals	0.945	29	351,453	12,118	0.85
BEPART srbench-cap (k=auto)	500k evals	0.918	41	446,813	11,038	0.93
BEPART max (k=1)	1 min	0.947	53	572,690	10,577	0.85
BEPART max (k=auto)	1 min	0.932	55	625,321	11,229	0.93
BEPART max (k=1)	5 min	0.949	258	1,803,812	6,878	0.88
BEPART max (k=auto)	5 min	0.942	259	2,239,900	8,659	0.95

B PROTOCOL DETAILS

Hardware and timing. Every fit of every method ran single-threaded on one physical core of an Intel Xeon E5-2650 v4 (2.2 GHz): the 24 worker processes of a node are pinned to its 24 physical cores and the hyper-thread siblings stay idle (checked on the nodes with `cluster/cpu_check.sbatch`). One-time costs such as Julia precompilation are excluded from the timings by a warm-up fit. Our evaluation buffers take up to 5.5 GB per fit on the largest training sets (8 GB for the one-hour configuration). On the SRBench black-box suite a fit that fails or times out stays in the tables with a test R^2 of 0; on the regime suite and OpenML-CTR23 such fits are dropped, which concerns two PS-Tree fits.

SRBench protocol. The black-box track has the 122 PMLB (Olson et al., 2017) regression datasets of La Cava et al. (2021), the ground-truth track its 116 Feynman and 14 Strogatz problems at target noise 0, 0.001, 0.01 and 0.1. Every dataset is run with 10 seeds, and we compare with the published per-run results of SRBench's 14 symbolic-regression and 7 machine-learning methods. We follow the SRBench evaluation script: a 75/25 train/test split per seed, training sets capped at 10,000 rows, and Gaussian target noise, scaled by the RMS of the training targets, added to the training targets only. On the black-box track inputs and targets are standardised on the training split, as in the published runs. On the ground-truth track the data stay raw, so that recovered expressions can be compared with the true formulas. Budgets are the SRBench ones: at most 500,000 evaluations per fit on the black-box track and 1,000,000 on the ground-truth track, counted as in Appendix A.7.

The published methods were tuned in SRBench. Every run scores all settings of the method's grid by 5-fold cross-validation on a small sample of the training set, keeps the best third, triples the sample and repeats until the whole training set is in use, and refits the winner (scikit-learn's `HalvingGridSearchCV`). The published training time is that of this whole procedure (starred in Tables 2 and 14): 41 fits for a symbolic method with six settings, a few thousand for the tree ensembles with their grids of several hundred settings, and six for FFX, which has a single setting. BEPART uses one setting everywhere, so its times are single fits. For the two tree baselines we can say what the search costs, because we ran them at their library defaults under our protocol (the *at its defaults* lines):

	published, tuned		at its defaults	
Method	R^2	time (s)	R^2	time (s)
XGBoost	0.854	187	0.835	0.15
LightGBM	0.821	21	0.839	0.08

The search is thus almost all of the reported time and buys little or nothing. For LightGBM it even costs accuracy: the search starts on 10 to 30 samples, on which LightGBM cannot split a node, so it eliminates settings at random. The symbolic baselines cannot be treated the same way, because their runtime is set by a search budget. Published training times were measured on Xeon E5-2690 v4 processors (2.6 GHz) and ours on E5-2650 v4 (2.2 GHz), so our times are pessimistic.

Symbolic solution. A model counts as a symbolic solution by SRBench's rule: the model and the true formula are simplified with SymPy after rounding constants to three decimals, and their difference must be a constant or their ratio a non-zero constant, the model not being a constant itself. A check runs under a 30 s cap and is repeated under a 300 s cap if it times out. A model that still times out or raises a SymPy error counts as a non-solution. At $k = 1$ this concerns 15–24% of the checks, depending on the noise level and the configuration, so the reported rates are lower bounds. Of the models with several components 68–77% stay undecided, because SymPy rarely simplifies their piecewise expressions, which is why Section 5.4 reports symbolic rates for $k = 1$ only. Accuracy solutions are runs with test $R^2 > 0.999$.

Model size. Every size in the tables is SRBench's measure, the number of nodes of the model's SymPy expression, computed as for the published methods. An affine leaf is written out in that expression, so a leaf $1.5\,x_0 - 2\,x_2 + 0.7$ has size 8. This is not the complexity C of Equation (2),

which charges the same leaf 5 (Appendix A.2). C is the second objective of the search and is never reported. For a model with several components the exported expression repeats every gate in every branch condition, which would inflate the count several-fold, so we count the nodes of every gate and every expert once. With one component this is SRBench's size.

Regime metrics. The metrics are defined in Section 5.1. Four details complete them. In the law check constants are rounded to three significant digits rather than to SRBench's three decimals, which would truncate laws such as $2/1701 \cdot x^3$ to zero, and a coefficient below 10^{-4} of the largest one in the expression is dropped as numerical debris; a constant law is recovered by a constant expert within 10^{-3} of it. Every law check runs under a 20 s cap in the analysis process and, when it times out, once more in a fresh interpreter under a 120 s cap with a memory limit; a check that is still undecided counts as not recovered (2% of the checks of the `srbench-cap` k = auto models at noise 0). LRR and PER are undefined for a method that reports a partition but no closed-form expert per cell: PS-Tree, whose leaf models are not exported, and the mixture with MLP experts. PySR with conditional primitives reports a single expression, but its `cond` and `greater` nodes carry branch conditions, so it partitions the input implicitly: two test points fall in the same cell when every condition in the expression has the same truth value on both. We read that partition off the expression, take as the expert of a cell the expression with its conditions fixed to their values there, and score both like an explicit partition.

Baselines. Table 6 lists the baselines we ran and their settings. The partition, mixture and symbolic baselines and XGBoost run on the regime suite, the four black-box regressors on OpenML-CTR23, and XGBoost and LightGBM also on the SRBench black-box suite (the *at its defaults* rows). On the regime suite the search-based methods have 600 s and 500,000 evaluations per fit (PySR accepts only the time budget). The partition and mixture baselines run to completion. CluBS has no budget knob and runs to completion as its authors run it, under a 4 h safety cap per fit. With the PySR backend a fit makes 10–40 inner PySR calls of up to 30 min each (the authors' settings), so on one core 18% of its regime-suite fits and 28% of its black-box fits, among them every seed of 22 of the 25 datasets above 1,000 rows, did not finish within the cap; they count as failures (no model, no R^2; ARI, LRR and PER 0 on the regime suite). The authors' own PMLB runs use the proprietary QLattice backend, which fits in seconds per call and which we could not run. Three piecewise methods with symbolic gates could not be run: the implementation of UPSR is proprietary (Doquet, 2025), the clustered SR of Ly & Lipson (2012) ran on the Eureqa API (Schmidt & Lipson, 2009), which is no longer available, and Mitrai et al. (2026) solve their trees with the commercial solver Gurobi, for which our cluster has no licence. Their problems are in the regime suite instead (Appendix C).

OpenML-CTR23. The 35 datasets of OpenML suite 353 (Fischer et al., 2023). Every task comes with official train/test splits, a 10-fold cross-validation, and we use them in place of random splits: of our ten runs per dataset, run s tests on fold s and trains on the other nine. Numeric missing values are replaced by the column median. Categorical columns with at most 10 levels are one-hot encoded and larger ones ordinally encoded, with a missing category as its own level. Training folds are capped at 10,000 rows and standardised as on the black-box track.

Ablation variants. The eleven ablation variants of Section 5.3 each flip one group of switches of the `default` k = auto configuration (Table 4): credit assignment off (mutation target probability fixed at 0.5); the fitted gate removed from the EM step (replaced by a local mutation); the residual split's aimed gate replaced by a random expression; the grow chain replaced by single splits; the EM step switched off everywhere (the periodic step and the refinement of every split proposal, by giving it no trials); the residual split, the merge proposals, the additive-growth proposals and mutation, or the affine-expert proposals switched off; the term-wise refit *and* the offspring scaling switched off ("no expert refit"); and the evaluation buffers reduced to two entries, which also empties the pools that the split and additive-growth proposals mine.

C THE REGIME SUITE

The suite has 38 problems in three families, listed in Table 7. Every problem is a box in $\mathbb{R}^d$ with $d \in \{1, 2, 3, 5\}$, a regime map r into $k \in \{2, \ldots, 5\}$ regimes and one closed-form law per regime, stated in the form the law takes on its regime, which is what a method can

Table 6: The baselines we ran and their settings. K is the number of cells or experts of a baseline.

Baseline	Settings
Partition and mixture models	
PS-Tree 0.1.2 (Zhang et al., 2022)	GP population 25, 100 generations, tree height limit 6, size objective on; once with its adaptive partition (at most 4 leaves) and once with a plain CART partition of at most 8 leaves; its `python-glmnet` dependency no longer installs and is replaced by scikit-learn's lasso path with `python-glmnet`'s conventions (penalty grid, weighted standardisation, five-fold one-standard-error rule), checked against the original
Linear model tree (`linear-tree` 0.3.5)	depth 3 (at most 8 leaves), minimum leaf size 2% of the training set, ordinary least-squares leaves
PARC 2.0.4 (Bemporad, 2023)	$K = 8$ cells
Mixture of experts, linear or MLP experts	linear softmax gate, $K = 8$ experts (MLP: one hidden layer of 32 tanh units); gate and experts trained jointly by Adam for 3,000 full-batch epochs on the squared error of the soft mixture, on standardised data; predictions are the soft mixture, the cell of a point is the argmax of the gate
Mixture of linear regressions (EM)	$K = 8$, one noise variance; up to 100 EM iterations from a k-means initialisation; a logistic router fitted on the training assignments routes every prediction; the two-stage design of Ly & Lipson (2012) with linear experts and an affine guard
CluBS (Zdankin et al., 2026)	the authors' search settings with the PySR backend (their QLattice backend is proprietary and was not run); their MLP gate for routing (their oracle routing uses the test targets and is not a predictor); cluster-merge R^2 threshold 0.85, the value of the authors' other configurations (their PMLB configuration's 0.001 merges nearly every cluster); one core, seeded. No budget knob: every inner PySR call keeps the authors' 30 min cap, no phase is skipped, and a fit is stopped only by a 4 h safety timeout
Symbolic regression	
PySR 2.2.1 (Cranmer, 2023)	the primitive set of Appendix A.2 $(+, -, \times, \div, \sin, \cos, \exp, \log, \sqrt{\cdot}, \lvert\cdot\rvert$ and a power whose exponent is a leaf); serial and deterministic; run until its time budget; once plain and once with the conditional primitives `greater` and `cond`; library defaults otherwise
Operon 0.6.1 (Burlacu et al., 2020)	the same primitive set; MSE objective; its time and evaluation budgets; library defaults otherwise
Black-box regressors	
XGBoost 3.0 (Chen & Guestrin, 2016), LightGBM 4 (Ke et al., 2017)	library defaults, one thread
Random forest (scikit-learn)	100 trees, one thread
MLP (scikit-learn)	one hidden layer of 100 units, 500 iterations, standardised inputs

recover there (the two-tank laws lose their absolute value this way). The flight law of the bouncing ball keeps an absolute value of the velocity, which the primitive set contains. Inputs are sampled uniformly in the box, except in three problems of the physics family, whose inputs are sampled log-uniformly (`P-drag-sphere-Cd-Re`, `P-darcy-friction-factor`, `P-nusselt-pipe-flow`). Every problem has 1,000 points, drawn with a fixed seed, so that every method, seed and noise level sees the same design. The first 750 points form the training set and the remaining 250 the test set. The generators are noise-free. The harness adds Gaussian noise to the training targets only, with standard deviation equal to the noise level times the RMS of the training targets, exactly as SRBench does, and evaluates on clean test targets. Every method runs with 10 seeds at noise levels of 0, 1% and 10%. Boundary types are axis-aligned (23 problems), oblique (7), quadratic (6: circles, an ellipse, a parabola and rings) and nonlinear (2: a sine curve and the sign of a product).

Origin of the problems. To our knowledge no established benchmark provides both the true partition and the true law of every regime: piecewise symbolic regression has been evaluated on a handful of problems per paper, and partition methods on black-box data, where the regimes are unknown. We therefore assembled the suite from three sources, which Table 7 marks per problem. First, problems of earlier work. Taken as published are the circle and the polymer-viscosity case of Mitrai et al. (2026) and a piecewise example of the `hal-cgp` library (Schmidt & Jordan, 2020), the latter on a narrower input range. Adapted are the two-tank system of Mitrai et al. (2026) and the spring-mass hopper of Mangan et al. (2019), whose equations we keep with other constants, their SIR model with school terms, which we reduce to one switch per year with rates of our own and without births and deaths, and the non-linear three-mode system of Ly & Lipson (2012). Its modes switch by transitions that depend on the current mode, which no partition of the input space can express, so we keep its three laws, one of them shifted so that its pole lies outside its regime, and place them on a partition of our own. Six one-dimensional problems are built with the two constructions of Doquet (2025), exclusive and stacked pieces, because its benchmark is not released. Second, new synthetic problems that vary one property at a time, the boundary type, the number of regimes and the dimension, so that a failure can be traced to one property. Their laws come from the function classes of the SR benchmarks, polynomial, trigonometric, exponential and rational. Third, textbook laws of engineering and textbook hybrid systems. They are new as benchmark problems. We include them because their regimes have a physical meaning, so a recovered partition can be judged by a practitioner, and because their inputs span up to six orders of magnitude, which the synthetic problems do not.

Family S (synthetic, 24 problems) contains exclusive and stacked pieces in one dimension (the law of a stacked regime contains the law of the regime before it as a summand, which makes the pieces harder to separate), thresholds, quadrants and a five-cell grid in two dimensions, oblique boundaries in two and three dimensions, quadratic boundaries, two nonlinear boundaries, the adopted three-mode, circle and polymer-viscosity problems, and two five-dimensional problems in which three inputs are irrelevant.

Family P (piecewise physics, 8 problems) samples textbook laws over physical ranges: the sphere drag coefficient with its Stokes, Schiller–Naumann and Newton regimes, the Darcy friction factor (laminar law against the Haaland correlation with the relative roughness as a second input), the Nusselt number (laminar constant against the Dittus–Boelter correlation), Stribeck friction with a sign switch, the Shichman–Hodges MOSFET model with its cut-off, triode and saturation regions, the last two separated by an oblique boundary, a wind-turbine power curve whose zero law holds both below cut-in and above cut-out (a regime is a law, not a connected region), Hertz contact and a bilinear elastic–plastic law. For the friction-factor and Nusselt problems the transition band between the laminar and turbulent regimes carries the laminar law, so both are two-regime problems with a single threshold in the Reynolds number.

Family H (hybrid systems, 6 problems) samples hybrid dynamical systems in their state space, with the state as input, one component of the derivative as target and the discrete mode as regime label: a spring-mass hopper (a mass on a spring leg, which pushes against the ground while the spring is compressed and is in free flight otherwise), a bouncing ball with quadratic drag, a two-tank system whose mode switches on the oblique boundary $x_0 = x_1$, a Lotka–Volterra system with a

harvested regime, an impact oscillator and an SIR model with two transmission rates switching at day 200 of the year.

Table 7: The regime suite. Left: the problem, its input dimension d, number of regimes k and boundary type, and its input box. Right: every regime with its region and its law. Every problem has 750 training and 250 test points sampled uniformly (log-uniformly where marked) in the box with a fixed seed. A superscript marks a problem that is not new: [1] built with the exclusive or the stacked pieces of Doquet (2025); [2] an example of the `hal-cgp` library (Schmidt & Jordan, 2020); [3] the three laws of a system of Ly & Lipson (2012), on a partition of our own; [4] from Mitrai et al. (2026), the two-tank system with another tank area; [5] adapted from Mangan et al. (2019).

Problem d, k, boundary type inputs	Regimes: the region and the law of each
S: synthetic (24 problems)	
S-axis1d-k2-poly-sin[1] $d = 1$, $k = 2$, axis $x_0 \in [-2, 2]$	(1) $x_0 < 0$: $x_0^2 + 1$ (2) $x_0 \geq 0$: $2\sin(3x_0)$
S-axis1d-k2-halcgp[2] $d = 1$, $k = 2$, axis $x_0 \in [-2, 2]$	(1) $x_0 < 0$: $-x_0$ (2) $x_0 \geq 0$: $x_0^2 + 1$
S-axis1d-k3-lin-quad-exp[1] $d = 1$, $k = 3$, axis $x_0 \in [-2, 2]$	(1) $x_0 < -0.7$: $2x_0 + 3$ (2) $-0.7 \leq x_0 < 0.7$: $x_0^2 - 1$ (3) $x_0 \geq 0.7$: $e^{x_0} - 2$
S-axis1d-k4-mixed[1] $d = 1$, $k = 4$, axis $x_0 \in [-2, 2]$	(1) $x_0 < -1$: $\sin(4x_0)$ (2) $-1 \leq x_0 < 0$: x_0^3 (3) $0 \leq x_0 < 1$: $\cos(2x_0) + 1$ (4) $x_0 \geq 1$: $-0.5x_0 + 2$
S-axis1d-k2-xsinx-const[1] $d = 1$, $k = 2$, axis $x_0 \in [-2, 2]$	(1) $x_0 < 0.5$: $x_0 \sin(2x_0)$ (2) $x_0 \geq 0.5$: 1.5
S-stack1d-k2-sin-plus-quad[1] $d = 1$, $k = 2$, axis $x_0 \in [-2, 2]$	(1) $x_0 \leq 0.5$: $\sin(2x_0)$ (2) $x_0 > 0.5$: $\sin(2x_0) + x_0^2$ The second piece is stacked on the first.
S-stack1d-k3-lin-plus-sin-plus-exp[1] $d = 1$, $k = 3$, axis $x_0 \in [-2, 2]$	(1) $x_0 < -0.5$: x_0 (2) $-0.5 \leq x_0 < 1$: $x_0 + 2\sin(3x_0)$ (3) $x_0 \geq 1$: $x_0 + 2\sin(3x_0) + e^{x_0 - 1}$ Every piece is stacked on the one before.
S-axis2d-k2-threshold-x1 $d = 2$, $k = 2$, axis $x_0, x_1 \in [-2, 2]$	(1) $x_1 \leq 0.3$: $x_0 x_1 + x_0$ (2) $x_1 > 0.3$: $\sin(x_0) - x_1^2$
S-axis2d-k4-quadrants $d = 2$, $k = 4$, axis $x_0, x_1 \in [-2, 2]$	(1) $x_0 \leq 0$, $x_1 \leq 0$: $x_0 + x_1$ (2) $x_0 > 0$, $x_1 \leq 0$: $x_0^2 - x_1$ (3) $x_0 \leq 0$, $x_1 > 0$: $\sin(2x_0) + x_1$ (4) $x_0 > 0$, $x_1 > 0$: $x_0 x_1 + 1$
S-axis2d-k5-grid $d = 2$, $k = 5$, axis $x_0, x_1 \in [-2, 2]$	(1) $x_0 \leq -0.7$, $x_1 \leq 1$: $x_0^2 + x_1$ (2) $-0.7 < x_0 \leq 0.7$, $x_1 \leq 1$: $2x_0 - x_1$ (3) $x_0 > 0.7$, $x_1 \leq -1$: $e^{0.5x_1}$ (4) $x_0 > 0.7$, $-1 < x_1 \leq 1$: $\cos(x_0 x_1)$ (5) the upper band $x_1 > 1$: $x_0 - x_1^2$

Problem d, k, boundary type inputs	Regimes: the region and the law of each
S-oblique2d-k2-readme $d = 2$, $k = 2$, oblique $x_0, x_1 \in [-3, 3]$	(1) $x_0 + x_1 \leq 0$: $3\sin(x_0)$ (2) $x_0 + x_1 > 0$: $x_0 x_1$
S-oblique2d-k2-tilted $d = 2$, $k = 2$, oblique $x_0, x_1 \in [-2, 2]$	(1) $x_0 - 2x_1 + 0.5 \leq 0$: $x_0^2 + x_1^2$ (2) $x_0 - 2x_1 + 0.5 > 0$: $e^{x_0} - x_1$
S-oblique2d-k3-wedge $d = 2$, $k = 3$, oblique $x_0, x_1 \in [-2, 2]$	(1) $x_1 > x_0$ and $x_1 > -x_0$: $\sin(x_0) + \cos(x_1)$ (2) $x_1 > x_0$ and $x_1 \leq -x_0$: $x_0 x_1$ (3) $x_1 \leq x_0$: $x_0^2 - x_1$
S-oblique3d-k2-plane $d = 3$, $k = 2$, oblique $x_0, x_1, x_2 \in [-2, 2]$	(1) $x_0 + x_1 - x_2 \leq 0.5$: $x_0 x_1 + x_2$ (2) $x_0 + x_1 - x_2 > 0.5$: $\sin(x_0 + x_2) - x_1$
S-circle2d-k2-mitrai[4] $d = 2$, $k = 2$, quadratic $x_0, x_1 \in [-2, 2]$	(1) inside the circle, $x_0^2 + x_1^2 \leq 2.5$: $x_0^2 + x_1^2$ (2) outside it: $x_0^2 + x_1$
S-ellipse2d-k2 $d = 2$, $k = 2$, quadratic $x_0, x_1 \in [-2, 2]$	(1) inside the ellipse, $x_0^2/2 + 2x_1^2 \leq 1.5$: $\sin(3x_0)$ (2) outside it: $x_0 x_1 - 1$
S-parabola2d-k2 $d = 2$, $k = 2$, quadratic $x_0, x_1 \in [-2, 2]$	(1) $x_1 \leq x_0^2 - 1$: $x_0 + 2x_1$ (2) $x_1 > x_0^2 - 1$: $e^{-x_0^2} + x_1$
S-circle2d-k3-rings $d = 2$, $k = 3$, quadratic $x_0, x_1 \in [-2, 2]$	(1) $r^2 < 1$: $x_0 x_1$ (2) $1 \leq r^2 < 2.5$: $x_0^2 - x_1^2$ (3) $r^2 \geq 2.5$: $\sin(x_0) + \cos(x_1)$ $r^2 = x_0^2 + x_1^2$.
S-sine2d-k2 $d = 2$, $k = 2$, nonlinear $x_0, x_1 \in [-3, 3]$	(1) $x_1 \leq \sin 2x_0$: $x_0^2 - 1$ (2) $x_1 > \sin 2x_0$: $\cos(x_0) + 2x_1$
S-product2d-k2-xor $d = 2$, $k = 2$, nonlinear $x_0, x_1 \in [-2, 2]$	(1) $x_0 x_1 \leq 0$: $x_0 + x_1$ (2) $x_0 x_1 > 0$: $x_0^2 + x_1^2$ Each region is a diagonal pair of quadrants.
S-lylipson-k3-modes[3] $d = 2$, $k = 3$, quadratic $x_0, x_1 \in [-2, 2]$	(1) the unit disk, $x_0^2 + x_1^2 < 1$: $x_0 x_1$ (2) outside it with $x_0 > 0$: $6x_0/(6 + x_1)$ (3) outside it with $x_0 \leq 0$: $(x_0 + x_1)/(x_0 - x_1 - 4)$
S-polymer-viscosity-k2[4] $d = 1$, $k = 2$, axis $x_0 \in [3, 6]$	(1) $x_0 < \log_{10} 31200$: $x_0 - 0.49$ (2) $x_0 \geq \log_{10} 31200$: $3.4x_0 - 11.28$ $x_0 = \log_{10} M$ is the logarithm of the molar mass.
S-dummy5d-k2-oblique $d = 5$, $k = 2$, oblique $x_0, \ldots, x_4 \in [-2, 2]$	(1) $x_0 + x_1 \leq 0$: $3\sin(x_0)$ (2) $x_0 + x_1 > 0$: $x_0 x_1$ x_2, x_3 and x_4 are irrelevant.
S-dummy5d-k3-circle $d = 5$, $k = 3$, quadratic $x_0, \ldots, x_4 \in [-2, 2]$	(1) $r^2 < 1$: $x_0 x_1$ (2) $1 \leq r^2 < 2.5$: $x_0^2 - x_1^2$ (3) $r^2 \geq 2.5$: $\sin(x_0) + \cos(x_1)$ $r^2 = x_0^2 + x_1^2$; x_2, x_3 and x_4 are irrelevant.
P: piecewise physics (8 problems)	

Problem d, k, boundary type inputs	Regimes: the region and the law of each
P-drag-sphere-Cd-Re $d = 1$, $k = 3$, axis $x_0 \in [0.1, 10^5]$ (log)	(1) Stokes, $x_0 < 1$: $24/x_0$ (2) Schiller–Naumann, $1 \le x_0 \le 1000$: $24\left(1 + 0.15x_0^{0.687}\right)/x_0$ (3) Newton, $x_0 > 1000$: 0.44 Input: the Reynolds number; target: the drag coefficient of a sphere.
P-darcy-friction-factor $d = 2$, $k = 2$, axis $x_0 \in [500, 10^6]$ (log), $x_1 \in [10^{-5}, 0.01]$ (log)	(1) laminar law, $Re = x_0 < 4000$: $64/x_0$ (2) turbulent, $Re \ge 4000$: $1/\left(-1.8\log_{10}\left((x_1/3.7)^{1.11} + 6.9/x_0\right)\right)^2$ Inputs: the Reynolds number x_0 and the relative roughness $x_1 = \varepsilon/D$; target: the Darcy friction factor. Law 2 is the Haaland correlation; the transition band from 2300 to 4000 carries the laminar law.
P-nusselt-pipe-flow $d = 2$, $k = 2$, axis $x_0 \in [100, 10^5]$ (log), $x_1 \in [0.7, 100]$ (log)	(1) constant Nu of laminar flow, $Re = x_0 < 10^4$: 3.66 (2) Dittus–Boelter, $Re \ge 10^4$: $0.023x_0^{0.8}x_1^{0.4}$ Inputs: the Reynolds number x_0 and the Prandtl number $Pr = x_1$; target: the Nusselt number. The transition band carries the laminar law.
P-stribeck-friction $d = 1$, $k = 2$, axis $x_0 \in [-2, 2]$	(1) velocity $x_0 < 0$: $\left(-1 - 0.5e^{-(x_0/0.1)^2}\right) + 0.4x_0$ (2) $x_0 \ge 0$: $1 + 0.5e^{-(x_0/0.1)^2} + 0.4x_0$ Input: the sliding velocity; target: the friction force ($F_c = 1$, $F_s = 1.5$, $v_s = 0.1$, viscous coefficient 0.4).
P-mosfet-IV $d = 2$, $k = 3$, oblique $x_0, x_1 \in [0, 5]$	(1) cutoff, $V_{gs} = x_0 < V_{th} = 1$: 0 (2) triode, $V_{ds} = x_1 < V_{gs} - V_{th}$: $0.5\left((x_0 - 1)\,x_1 - x_1^2/2\right)$ (3) saturation: $0.25\,(x_0 - 1)^2\,(1 + 0.02x_1)$ Inputs: the gate and the drain voltage; target: the drain current ($k = 0.5$, $\lambda = 0.02$).
P-wind-power-curve $d = 1$, $k = 3$, axis $x_0 \in [0, 30]$	(1) no power, below the cut-in speed 3 or above the cut-out speed 25: 0 (2) cubic ramp, $3 \le x_0 < 12$: $2\,(x_0^3 - 3^3)/(12^3 - 3^3)$ (3) rated power of 2 MW, $12 \le x_0 \le 25$: 2 Input: the wind speed; target: the power in MW.
P-hertz-contact $d = 2$, $k = 2$, axis $x_0 \in [-1, 1]$, $x_1 \in [0.5, 2]$	(1) no contact, indentation $x_0 < 0$: 0 (2) Hertz contact, $x_0 \ge 0$: $4\sqrt{x_1}x_0^{1.5}/3$ Inputs: the indentation x_0 and the sphere radius x_1; target: the contact force ($E^* = 1$).
P-bilinear-plasticity $d = 1$, $k = 2$, axis $x_0 \in [0, 0.01]$	(1) elastic, strain $x_0 < 0.002$: $200x_0$ (2) plastic hardening, $x_0 \ge 0.002$: $0.4 + 20\,(x_0 - 0.002)$ Input: the strain; target: the stress ($E = 200$, $H = 20$).
H: hybrid systems (6 problems)	
H-hopper[5] $d = 2$, $k = 2$, axis $x_0 \in [0, 2]$, $x_1 \in [-3, 3]$	(1) compression, height $x_0 \le 1$: $-10\,(x_0 - 1) - 9.81$ (2) flight, $x_0 > 1$: -9.81 Inputs: the height x_0 and the velocity x_1, which is irrelevant; target: the acceleration (spring $k/m = 10$).
H-bouncing-ball-drag $d = 2$, $k = 2$, axis $x_0 \in [-0.5, 3]$, $x_1 \in [-5, 5]$	(1) contact, $x_0 < 0$ (stiff spring-damper): $-9.81 - 100x_0 - 2x_1$ (2) flight with quadratic drag, $x_0 \ge 0$: $-9.81 - 0.1x_1\,\lvert x_1\rvert$ Inputs: the height x_0 and the velocity x_1; target: the acceleration.

Problem d, k, boundary type inputs	Regimes: the region and the law of each
H-two-tank[4] $d = 3$, $k = 2$, oblique $x_0, x_1 \in [0, 5]$, $x_2 \in [0.5, 2]$	(1) inflow, levels $x_0 \leq x_1$: $(x_2 + 0.5\sqrt{x_1 - x_0})/2$ (2) outflow, $x_0 > x_1$: $(x_2 - 0.5\sqrt{x_0 - x_1})/2$ Inputs: the two levels and the inflow x_2; target: the rate of change of the first level ($C_v = 0.5$, $A_1 = 2$).
H-switched-lotka-volterra $d = 2$, $k = 2$, axis $x_0, x_1 \in [0, 4]$	(1) free growth, prey $x_0 \leq 2$: $1.0x_0 - 0.5x_0x_1$ (2) harvested growth, $x_0 > 2$: $0.4x_0 - 0.5x_0x_1$ Inputs: prey x_0 and predators x_1; target: the rate of change of the prey, whose growth rate drops from 1.0 to 0.4.
H-impact-oscillator $d = 2$, $k = 2$, axis $x_0 \in [-2, 2]$, $x_1 \in [-3, 3]$	(1) free oscillator, $x_0 \leq 1$: $-4x_0 - 0.2x_1$ (2) contact with a stiff wall, $x_0 > 1$: $-4x_0 - 0.2x_1 - 50\,(x_0 - 1)$ Inputs: the position x_0 and the velocity x_1; target: the acceleration ($k_c = 50$).
H-sir-school-terms[5] $d = 3$, $k = 2$, axis $x_0 \in [0, 1]$, $x_1 \in [0, 0.5]$, $x_2 \in [0, 365]$	(1) vacation, day of the year $x_2 \geq 200$ ($\beta = 0.4$): $0.4x_0x_1 - 0.1x_1$ (2) school term, $x_2 < 200$ ($\beta = 0.8$): $0.8x_0x_1 - 0.1x_1$ Inputs: the susceptible x_0, the infected x_1 and the day x_2; target: the rate of change of the infected.

D FULL RESULTS

D.1 REGIME SUITE

Table 8 gives the per-family results of the regime suite at the three noise levels, Table 10 the per-problem results at noise 0, and Table 9 sets the symbolic law-recovery rates next to accuracy-based counterparts: the *acc* columns accept a paired expert when it reproduces the true law with $R^2 > 0.999$ on the regime's test points (SRBench's accuracy-solution rule per regime) instead of matching it symbolically; pairing and the other conditions of PER are unchanged. The gap between the two is laws that are numerically right but whose constants no longer round to the truth, which is what target noise produces.

Table 8: Regime suite per family at the three levels of target noise (means over problems and 10 seeds): ARI, law-recovery rate (LRR) and piecewise-exact rate (PER). Plain PySR, Operon and XGBoost define no partition, and BEPART at $k = 1$ has a single cell and scores zero throughout, so they have no row here. Their accuracy is in Table 1. PySR with conditionals is scored on the partition its branch conditions define. Bold: the best value of a column at a noise level.

	S: synthetic (24)			P: physics (8)			H: hybrid (6)		
Method	ARI	LRR	PER	ARI	LRR	PER	ARI	LRR	PER
target noise 0									
BEPART default (k=auto)	0.78	0.41	0.24	0.75	0.39	0.15	0.90	0.77	0.63
BEPART srbench-cap (k=auto)	**0.92**	**0.73**	**0.55**	**0.84**	**0.57**	**0.24**	**0.98**	**0.88**	**0.82**
PySR (conditionals)	0.65	0.29	0.19	0.51	0.19	0.00	0.71	0.42	0.35
MoE (linear)	0.42	0.00	0.00	0.44	0.00	0.00	0.65	0.00	0.00
Linear model tree	0.32	0.15	0.00	0.47	0.23	0.00	0.35	0.42	0.00
Mixture of linear regr. (EM)	0.30	0.10	0.00	0.36	0.06	0.00	0.49	0.36	0.33
MoE (MLP)	0.30	–	–	0.40	–	–	0.38	–	–
PARC	0.30	0.12	0.00	0.32	0.20	0.00	0.24	0.15	0.00
PS-Tree (CART)	0.27	–	–	0.37	–	–	0.26	–	–
PS-Tree	0.23	–	–	0.42	–	–	0.20	–	–
CluBS (PySR, MLP gate)	0.19	0.09	0.05	0.11	0.00	0.00	0.01	0.00	0.00
target noise 0.01									
BEPART default (k=auto)	0.80	0.15	0.05	0.75	0.19	**0.04**	**0.90**	0.13	0.03
BEPART srbench-cap (k=auto)	**0.87**	0.23	0.06	**0.78**	**0.24**	0.00	**0.90**	0.16	0.03
PySR (conditionals)	0.68	**0.25**	**0.09**	0.48	0.20	0.01	0.50	**0.17**	**0.12**
MoE (linear)	0.42	0.00	0.00	0.44	0.00	0.00	0.65	0.00	0.00
Linear model tree	0.30	0.04	0.00	0.49	0.11	0.00	0.32	0.02	0.00
Mixture of linear regr. (EM)	0.30	0.04	0.00	0.36	0.01	0.00	0.45	0.01	0.00
MoE (MLP)	0.30	–	–	0.40	–	–	0.38	–	–
PARC	0.30	0.03	0.00	0.29	0.00	0.00	0.31	0.05	0.00
PS-Tree (CART)	0.27	–	–	0.38	–	–	0.26	–	–
PS-Tree	0.20	–	–	0.38	–	–	0.17	–	–
CluBS (PySR, MLP gate)	0.22	0.07	0.02	0.03	0.00	0.00	0.00	0.01	0.00
target noise 0.1									
BEPART default (k=auto)	0.75	0.04	0.00	0.68	0.10	0.00	**0.86**	**0.01**	0.00
BEPART srbench-cap (k=auto)	**0.85**	0.05	0.00	**0.72**	**0.14**	**0.01**	0.82	0.00	0.00
PySR (conditionals)	0.63	**0.20**	**0.03**	0.33	0.13	0.00	0.59	**0.01**	0.00
MoE (linear)	0.41	0.00	0.00	0.44	0.00	0.00	0.62	0.00	0.00
Linear model tree	0.30	0.01	0.00	0.39	0.05	0.00	0.29	0.00	0.00
Mixture of linear regr. (EM)	0.22	0.00	0.00	0.35	0.00	0.00	0.44	0.00	0.00
MoE (MLP)	0.30	–	–	0.40	–	–	0.34	–	–
PARC	0.27	0.01	0.00	0.33	0.00	0.00	0.33	**0.01**	0.00
PS-Tree (CART)	0.26	–	–	0.39	–	–	0.26	–	–
PS-Tree	0.14	–	–	0.34	–	–	0.18	–	–
CluBS (PySR, MLP gate)	0.11	0.01	0.00	0.06	0.00	0.00	0.07	0.00	0.00

Table 9: Regime suite: LRR and PER by the symbolic check (*exact*, as in Section 5.1) and by the accuracy check (*acc*: the paired expert reproduces the true law with $R^2 > 0.999$ on the regime's test points) per noise level, means over problems and 10 seeds, for the methods with a partition and closed-form experts. Bold: the best rate of a column.

	target noise 0				target noise 0.01				target noise 0.1			
	LRR		PER		LRR		PER		LRR		PER	
Method	exact	acc	exact	acc	exact	acc	exact	acc	exact	acc	exact	acc
BEPART default (k=auto)	0.46	0.66	0.28	0.38	0.16	0.64	0.05	0.34	0.05	0.35	0.00	0.10
BEPART srbench-cap (k=auto)	**0.72**	**0.86**	**0.53**	**0.66**	0.22	**0.77**	0.04	**0.40**	0.06	**0.41**	0.00	**0.12**
BEPART default (k=1)	0.00	0.06	0.00	0.00	0.00	0.06	0.00	0.00	0.00	0.02	0.00	0.00
PySR (conditionals)	0.29	0.44	0.17	0.31	**0.23**	0.43	**0.08**	0.29	**0.16**	0.25	**0.02**	0.10
MoE (linear)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Linear model tree	0.21	0.23	0.00	0.00	0.05	0.17	0.00	0.00	0.01	0.04	0.00	0.00
Mixture of linear regr. (EM)	0.13	0.15	0.05	0.05	0.03	0.08	0.00	0.00	0.00	0.00	0.00	0.00
MoE (MLP)	–	–	–	–	–	–	–	–	–	–	–	–
PARC	0.14	0.17	0.00	0.00	0.03	0.09	0.00	0.00	0.01	0.04	0.00	0.00
PS-Tree (CART)	–	–	–	–	–	–	–	–	–	–	–	–
PS-Tree	–	–	–	–	–	–	–	–	–	–	–	–
CluBS (PySR, MLP gate)	0.06	0.11	0.03	0.04	0.04	0.11	0.01	0.03	0.01	0.03	0.00	0.00

Table 10: Regime suite per problem at target noise 0 (means over 10 seeds): BEPART k = auto at the two budgets (ARI, LRR, PER) and the best baseline by ARI (PySR-cond = PySR with conditionals, EM mixture = mixture of linear regressions fitted by EM, model tree = linear model tree). Bold: the best ARI of a row.

		BEPART						best baseline	
		default			srbench-cap				
Problem	k	ARI	LRR	PER	ARI	LRR	PER	method	ARI
H-bouncing-ball-drag	2	0.68	0.75	0.50	0.99	1.00	1.00	PySR-cond	**1.00**
H-hopper	2	**1.00**	1.00	1.00	**1.00**	1.00	1.00	EM mixture	0.97
H-impact-oscillator	2	**1.00**	1.00	1.00	**1.00**	1.00	1.00	EM mixture	0.98
H-sir-school-terms	2	0.95	0.75	0.40	**0.99**	1.00	1.00	MoE (linear)	0.40
H-switched-lotka-volterra	2	0.97	1.00	0.90	**1.00**	0.95	0.90	PySR-cond	**1.00**
H-two-tank	2	0.80	0.10	0.00	0.87	0.30	0.00	MoE (linear)	**0.91**
P-bilinear-plasticity	2	**0.99**	1.00	1.00	**0.99**	1.00	1.00	MoE (linear)	0.76
P-darcy-friction-factor	2	0.65	0.20	0.00	0.59	0.50	0.00	PySR-cond	**0.76**
P-drag-sphere-Cd-Re	3	0.88	0.63	0.10	**0.98**	0.80	0.30	PySR-cond	0.59
P-hertz-contact	2	0.69	0.30	0.00	0.90	0.60	0.20	PySR-cond	**1.00**
P-mosfet-IV	3	0.49	0.03	0.00	**0.73**	0.30	0.00	PS-Tree	0.37
P-nusselt-pipe-flow	2	0.77	0.40	0.00	0.90	0.50	0.00	PySR-cond	**0.92**
P-stribeck-friction	2	0.73	0.00	0.00	0.66	0.00	0.00	EM mixture	**0.99**
P-wind-power-curve	3	0.81	0.57	0.10	**0.96**	0.83	0.40	model tree	0.70
S-axis1d-k2-halcgp	2	**1.00**	0.60	0.30	**1.00**	0.80	0.60	PySR-cond	**1.00**
S-axis1d-k2-poly-sin	2	0.87	0.65	0.40	**1.00**	1.00	1.00	PySR-cond	0.86
S-axis1d-k2-xsinx-const	2	0.60	0.50	0.00	**1.00**	1.00	1.00	PySR-cond	0.87
S-axis1d-k3-lin-quad-exp	3	0.95	0.87	0.60	**1.00**	0.97	0.90	PySR-cond	0.84
S-axis1d-k4-mixed	4	0.96	0.30	0.00	**1.00**	0.70	0.20	PySR-cond	0.77
S-axis2d-k2-threshold-x1	2	0.93	0.70	0.60	**1.00**	1.00	1.00	PySR-cond	0.88
S-axis2d-k4-quadrants	4	0.82	0.35	0.10	**0.98**	0.88	0.50	model tree	0.65
S-axis2d-k5-grid	5	0.78	0.14	0.00	**0.92**	0.24	0.00	model tree	0.73
S-circle2d-k2-mitrai	2	0.75	0.05	0.00	**0.88**	0.45	0.00	PS-Tree (CART)	0.34
S-circle2d-k3-rings	3	0.37	0.03	0.00	**0.66**	0.17	0.00	PySR-cond	0.50
S-dummy5d-k2-oblique	2	0.90	0.70	0.60	**1.00**	1.00	1.00	PySR-cond	**1.00**
S-dummy5d-k3-circle	3	0.38	0.00	0.00	**0.62**	0.00	0.00	PS-Tree (CART)	0.29
S-ellipse2d-k2	2	0.68	0.05	0.00	**0.81**	0.70	0.10	PySR-cond	0.59
S-lylipson-k3-modes	3	0.66	0.00	0.00	**0.74**	0.23	0.00	PySR-cond	0.54
S-oblique2d-k2-readme	2	**1.00**	1.00	1.00	**1.00**	1.00	1.00	PySR-cond	**1.00**
S-oblique2d-k2-tilted	2	0.77	0.15	0.00	**0.98**	0.90	0.80	MoE (linear)	0.38
S-oblique2d-k3-wedge	3	0.76	0.40	0.00	**0.95**	0.97	0.70	CluBS	0.56
S-oblique3d-k2-plane	2	0.73	0.15	0.10	0.87	0.60	0.40	CluBS	**0.92**
S-parabola2d-k2	2	0.74	0.25	0.00	**0.99**	0.60	0.20	PySR-cond	0.92
S-polymer-viscosity-k2	2	**1.00**	1.00	1.00	**1.00**	1.00	1.00	EM mixture	0.95
S-product2d-k2-xor	2	0.83	0.70	0.40	**1.00**	1.00	1.00	PySR-cond	0.77
S-sine2d-k2	2	0.54	0.05	0.00	**0.87**	0.80	0.70	CluBS	0.73
S-stack1d-k2-sin-plus-quad	2	0.94	0.80	0.70	**1.00**	0.95	0.90	PySR-cond	**1.00**
S-stack1d-k3-lin-plus-sin-plus-exp	3	0.76	0.33	0.00	0.92	0.57	0.20	PySR-cond	**0.96**

D.2 Ablation tables

Tables 11 and 13 are the tables behind Figure 2, and Table 12 gives the mean paired difference of every variant to the reference with its standard error over the 114 (problem, noise level) cells, so that the differences of Figure 2 can be read against their uncertainty. The black-box ablation runs on 14 of the 122 SRBench datasets. Twelve are the black-box subset of SRBench 2025 (Imai Aldeia et al., 2025), which its authors selected by clustering the 122 datasets by size and number of features, with at most a quarter Friedman sets. We adopt it so that the choice of datasets is not ours. Its real-world datasets have at most 13 features, so we add two large, high-dimensional ones, `201_pol` (10,000 training rows, 48 features) and `4544_GeographicalOriginalofMusic` (117 features, the second most of the suite). Together the 14 span the suite's range of 2 to 124 features and of 35 to 10,000 training rows. The twelve are `1028_SWD`, `1089_USCrime`, `1193_BNG_lowbwt`, `1199_BNG_echoMonths`, `192_vineyard`, `210_cloud`, `522_pm10`, `557_analcatdata_apnea1`, `579_fri_c0_250_5`, `606_fri_c2_1000_10`, `650_fri_c0_500_50` and `678_visualizing_environmental`.

Table 11: Ablation on the regime suite (38 problems, 10 seeds): ARI, LRR and PER per noise level. Reference = the `default` k = auto configuration with all mechanisms.

	noise 0			noise 0.01			noise 0.1		
Variant	ARI	LRR	PER	ARI	LRR	PER	ARI	LRR	PER
BePart default (k=auto), all components	0.79	0.46	0.28	0.81	0.16	0.05	0.75	0.05	0.00
partition side: credit assignment and gate creation (Appendix A.2 and Section 3.2)									
no credit assignment (uniform target)	0.82	0.47	0.28	0.79	0.16	0.05	0.78	0.05	0.00
EM step without fitted gates	0.77	0.47	0.29	0.74	0.17	0.06	0.72	0.05	0.00
residual split with a random gate	0.82	0.48	0.29	0.79	0.18	0.05	0.76	0.06	0.00
no grow chain (single split proposals)	0.83	0.48	0.29	0.80	0.17	0.07	0.78	0.05	0.00
no EM step	0.70	0.42	0.23	0.69	0.17	0.06	0.67	0.04	0.00
no residual split	0.78	0.44	0.25	0.76	0.15	0.05	0.75	0.05	0.01
no merge proposals	0.80	0.48	0.29	0.80	0.16	0.05	0.75	0.05	0.01
expert side: refit, growth and affine experts, and the evaluation buffers (Section 3.4)									
no expert refit (term refit and scaling)	0.68	0.26	0.14	0.72	0.09	0.02	0.62	0.04	0.00
no additive growth	0.76	0.44	0.26	0.75	0.19	0.04	0.73	0.06	0.00
no affine experts	0.81	0.49	0.28	0.80	0.16	0.06	0.76	0.05	0.00
no evaluation buffers	0.81	0.47	0.29	0.80	0.18	0.05	0.79	0.06	0.01

Table 12: Ablation on the regime suite: mean and standard error of the paired difference to the reference (variant minus reference, per problem and noise level, each the mean over 10 seeds).

Variant	ΔARI	ΔLRR	ΔPER
partition side: credit assignment and gate creation (Appendix A.2 and Section 3.2)			
no credit assignment (uniform target)	+0.014 ± 0.007	+0.006 ± 0.007	+0.001 ± 0.008
EM step without fitted gates	-0.042 ± 0.013	+0.009 ± 0.008	+0.005 ± 0.007
residual split with a random gate	+0.009 ± 0.007	+0.019 ± 0.007	+0.003 ± 0.007
no grow chain (single split proposals)	+0.017 ± 0.007	+0.012 ± 0.006	+0.010 ± 0.008
no EM step	-0.097 ± 0.017	-0.014 ± 0.010	-0.012 ± 0.010
no residual split	-0.021 ± 0.014	-0.008 ± 0.010	-0.007 ± 0.011
no merge proposals	-0.004 ± 0.007	+0.006 ± 0.007	+0.005 ± 0.008
expert side: refit, growth and affine experts, and the evaluation buffers (Section 3.4)			
no expert refit (term refit and scaling)	-0.111 ± 0.013	-0.095 ± 0.016	-0.059 ± 0.015
no additive growth	-0.034 ± 0.010	+0.005 ± 0.009	-0.010 ± 0.010
no affine experts	+0.005 ± 0.007	+0.011 ± 0.007	+0.002 ± 0.007
no evaluation buffers	+0.017 ± 0.008	+0.015 ± 0.007	+0.004 ± 0.007

Table 13: Ablation on 14 SRBench black-box datasets (10 seeds, standardised inputs). R^2 median and mean: over the 14 datasets, of the per-dataset median test R^2. ΔR^2: on every dataset the variant's R^2 minus the reference's, and of these 14 differences the median. Wins/losses: the datasets on which that difference is above $+0.001$ or below -0.001. Time and evaluations: medians per fit.

Variant	R^2 median	R^2 mean	ΔR^2	wins/losses	time (s)	evaluations
BEPART default (k=auto), all components	0.665	0.639	–	–	3.2	35,936
partition side: credit assignment and gate creation (Appendix A.2 and Section 3.2)						
no credit assignment (uniform target)	0.610	0.635	+0.002	7/5	3.2	36,156
EM step without fitted gates	0.656	0.645	0.000	7/7	3.0	37,104
residual split with a random gate	0.708	0.650	+0.001	7/4	3.3	35,721
no grow chain (single split proposals)	0.705	0.655	+0.004	9/1	2.9	34,880
no EM step	0.702	0.646	+0.003	8/5	2.3	26,398
no residual split	0.708	0.651	0.000	6/4	1.9	31,174
no merge proposals	0.678	0.625	0.000	6/6	3.1	35,472
expert side: refit, growth and affine experts, and the evaluation buffers (Section 3.4)						
no expert refit (term refit and scaling)	0.660	0.614	-0.006	3/10	1.8	26,300
no additive growth	0.681	0.610	-0.012	4/9	1.2	34,264
no affine experts	0.662	0.648	+0.003	8/4	3.0	35,623
no evaluation buffers	0.717	0.642	+0.002	7/6	4.1	34,676

D.3 SRBENCH BLACK-BOX: ALL CONFIGURATIONS

Table 14 extends Table 2 to all 21 published methods and adds the `max` configuration, which runs without the SRBench evaluation cap: a median of 10 M counted evaluations per fit at $k = 1$ and 27 M at $k =$ auto against the cap of 500k, so its rows are not comparable with the published runs on budget. Figure 4 sets accuracy against model size. SRBench's own analysis ranks the methods per dataset as well, reporting median ranks (La Cava et al., 2021); we report mean ranks, on which the Nemenyi test below is defined. Ranking all our configurations and all baselines in one pool would push every baseline down by the number of our rows, so every configuration of ours is ranked alone among the baselines: the best configuration by the mean rank among our own is ranked together with all baselines, which fixes their ranks, and each other configuration is ranked in a pool of itself and the baselines. A difference between two mean ranks may be chance. The Nemenyi test (Demšar, 2006), which SRBench itself does not apply, gives the smallest gap between two mean ranks that is significant at the 5% level, the *critical difference*. It is 2.8 ranks here, computed over the 21 methods of the first pool (the baselines and BEPART srbench-cap (k=1)) with results on all 122 datasets (AI Feynman has none on 15 of them). CluBS finished within its cap on 99 datasets (Appendix B); its median is over those, on which `srbench-cap` $k = 1$ reaches 0.949, and it gets no rank.

Table 14: SRBench black-box suite, all published methods and all BEPART configurations. Columns as in Table 2, mean rank recomputed over all rows of this table.

Method	R^2 (all)	R^2 (Fr.)	R^2 (non-Fr.)	rank	size	cells	time (s)
BEPART max (k=auto)	0.948	0.967	0.739	4.6	83	3.3	3,156
BEPART max (k=1)	0.946	0.971	0.783	3.7	80	–	3,163
BEPART srbench-cap (k=1)	0.945	0.960	0.751	4.6	57	–	29
Operon†	0.934	0.956	0.695	5.2	50	–	636*
BEPART srbench-cap (k=auto)	0.918	0.951	0.736	5.0	126	3.2	41
BEPART racing (k=1)	0.916	0.940	0.741	5.7	46	–	4.6
BEPART racing (k=auto)	0.914	0.944	0.735	5.3	146	3.1	11
SBP-GP†	0.908	0.940	0.720	6.1	721	–	170,961*
FEAT†	0.895	0.918	0.725	7.6	75	–	6,251*
BEPART default (k=1)	0.890	0.911	0.752	6.3	44	–	1.1
BEPART default (k=auto)	0.888	0.929	0.726	6.1	150	2.9	2.9
EPLEX†	0.868	0.891	0.715	9.0	58	–	4,304*
CluBS (PySR, MLP gate)	0.858	0.869	0.594	–	26	1.7	6,442
XGBoost†	0.854	0.877	0.742	7.0	9,641	–	187*
at its defaults, no search	0.835	0.851	0.704	–	–	–	0.15
LightGBM†	0.821	0.880	0.607	10.2	5,168	–	21*
at its defaults, no search	0.839	0.881	0.668	–	–	–	0.05
GP-GOMEA†	0.816	0.844	0.737	9.0	26	–	6,670*
AdaBoost†	0.757	0.771	0.702	11.3	14,590	–	48*
Random forest†	0.748	0.765	0.698	10.8	129,511	–	84*
ITEA†	0.713	0.733	0.682	11.3	111	–	3,589*
AFP_FE†	0.706	0.709	0.680	12.9	38	–	1,837*
AFP†	0.692	0.699	0.647	13.2	37	–	1,793*
FFX†	0.688	0.828	0.500	12.4	1,470	–	144*
Kernel ridge†	0.686	0.593	0.762	10.5	375	–	1.2*
DSR†	0.590	0.590	0.589	15.4	8	–	32,660*
MRGP†	0.588	0.679	0.408	14.6	9,220	–	200,749*
gplearn†	0.582	0.569	0.596	15.4	15	–	23,196*
MLP†	0.576	0.298	0.694	12.4	2,402	–	13*
Linear regression†	0.380	0.255	0.689	15.1	10	–	0.2*
BSR†	0.110	0.039	0.424	19.4	16	–	7,200*
AI Feynman†	-0.437	-0.716	-0.077	20.3	742	–	29,244*

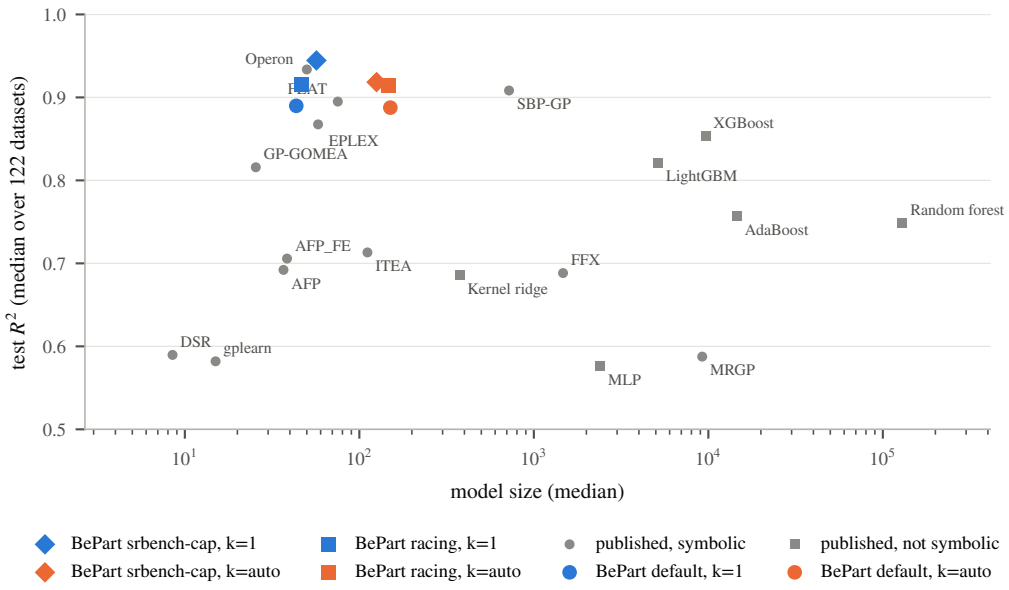

Figure 4: SRBench black-box suite: median test R^2 against median model size. Published methods in gray (dots for symbolic regressors, squares for the other regressors), BEPART configurations in colour. The two `max` markers almost coincide (80 and 83 nodes) and are drawn 12% apart so that both are visible.

D.4 GROUND-TRUTH RATES

Table 15 gives the symbolic and accuracy solution rates of the SRBench ground-truth track.

Table 15: SRBench ground-truth track: symbolic and accuracy solution rates (%) per target-noise level over 130 problems and 10 seeds. The baseline rows are the published SRBench runs. Their missing runs count as unsolved. Bold: the best rate of a column.

	symbolic solution (%)				accuracy solution (%)			
Method	0.0	0.001	0.01	0.1	0.0	0.001	0.01	0.1
BEPART srbench-cap (k=1)	**50.5**	**40.9**	**28.8**	12.8	82.8	83.3	83.2	**74.2**
AI Feynman	49.5	30.7	11.7	0.8	68.1	68.9	60.6	9.9
BEPART default (k=1)	37.0	32.0	24.5	10.7	63.2	62.2	62.5	55.6
AFP_FE	26.2	21.2	20.0	12.3	55.0	52.0	50.8	50.2
GP-GOMEA	21.7	10.6	4.7	1.5	56.8	72.3	72.2	66.9
ITEA	16.6	13.8	7.7	1.5	21.4	26.8	26.7	25.8
AFP	16.4	19.0	16.3	12.2	31.7	39.9	41.2	38.9
Operon	16.0	12.3	1.9	0.1	**85.3**	85.8	**84.9**	67.9
DSR	15.8	19.2	18.9	**16.6**	17.3	25.1	25.2	25.0
gplearn	12.4	16.2	15.9	15.8	22.7	29.8	28.4	24.7
SBP-GP	10.1	0.7	0.0	0.0	59.9	73.7	73.8	50.4
EPLEX	9.6	9.5	8.2	9.3	34.4	44.5	49.5	45.7
BSR	1.8	0.6	0.1	0.0	9.8	12.1	13.5	10.4
FEAT	0.1	0.0	0.0	0.0	29.0	39.3	37.7	14.0
FFX	0.0	0.0	0.0	0.1	0.4	0.4	3.4	16.8
MRGP	0.0	0.0	0.0	0.0	70.5	**87.1**	83.4	3.0

D.5 OpenML-CTR23

As a second black-box suite we ran the 35 curated regression datasets of OpenML-CTR23 (Fischer et al., 2023) under their official 10-fold splits (seed = fold, training sets capped at 10,000 rows, inputs standardised), against gradient boosting, a random forest and an MLP run by us with their library defaults (Table 16). Three datasets (California housing, CPU activity and the geographical origin of music) also occur in PMLB.

Table 16: OpenML-CTR23 (35 datasets, 10 folds): median, mean (clipped at 0) and quartiles of the per-dataset median test R^2, median model size and median wall-clock per fit. The size of a tree ensemble or an MLP is not comparable to the node count of an expression, so it is not reported here (Table 14 lists SRBench's own measure for its published runs). Bold: the best value of an R^2 column.

Method	R^2 median	mean	Q1	Q3	size	time (s)
XGBoost	**0.879**	0.687	0.513	**0.949**	–	0.2
LightGBM	0.841	0.682	0.502	0.942	–	0.2
BePart max (k=auto)	0.825	0.668	0.454	0.947	325	3,182
BePart srbench-cap (k=auto)	0.821	0.686	0.459	0.947	340	248
BePart racing (k=auto)	0.820	0.684	0.462	0.948	274	82
Random forest	0.805	0.688	**0.560**	0.942	–	5.0
BePart default (k=auto)	0.805	0.677	0.461	0.943	240	16
BePart max (k=1)	0.798	**0.695**	0.456	**0.949**	158	3,175
BePart srbench-cap (k=1)	0.776	0.682	0.481	0.928	88	181
MLP	0.772	0.633	0.393	0.943	–	2.1
BePart racing (k=1)	0.769	0.669	0.478	0.910	66	37
BePart default (k=1)	0.754	0.658	0.467	0.897	66	7.3

E Robustness

Figure 5 shows the robustness sweeps summarised in Section 5.4. They use ten problems of family S that cover all boundary types (`S-axis2d-k2-threshold-x1`, `S-axis2d-k5-grid`, `S-oblique2d-k2-readme`, `S-oblique2d-k3-wedge`, `S-oblique3d-k2-plane`, `S-circle2d-k2-mitrai`, `S-circle2d-k3-rings`, `S-parabola2d-k2`, `S-sine2d-k2`, `S-lylipson-k3-modes`) and perturb one thing at a time. The first sweep varies the target noise (0, 1, 3, 10 and 30%) at 750 training points. The other three run at 1% noise and vary the sample size (100, 300, 3,000 and 10,000 points, 75/25 split, redrawn with the same seed), the number of irrelevant inputs appended (one, three or five, uniform on $[-2, 2]$) and the test split (an extrapolation split that trains on the points with x_0 below its 75th percentile and tests above it).

Three things stand out. First, the partition is robust: the ARI of our configurations stays high on every axis, with only a moderate loss at 30% noise, at the smallest training sets and under extrapolation; only at the smallest training set does the `default` configuration fall below PySR with conditionals, the strongest baseline, and the two larger configurations stay above it on every axis. PS-Tree and the mixture of experts stay far below all of them. Second, exact law recovery is what noise destroys: without noise the two larger configurations recover two fifths to three fifths of the regime laws (`default` under a third), and already 1% noise cuts this to a fraction, while the accuracy of the recovered models stays high. More training points help: at 7,500 points the two larger configurations recover about one and a half times the share of laws they recover at 750. Third, PySR with conditionals behaves differently: it recovers a fifth to a quarter of the laws at every sample size and at noise levels up to 10%, which is less than we recover on clean data and more than we recover at 3% noise and beyond, at a far lower agreement of its partition with the truth.

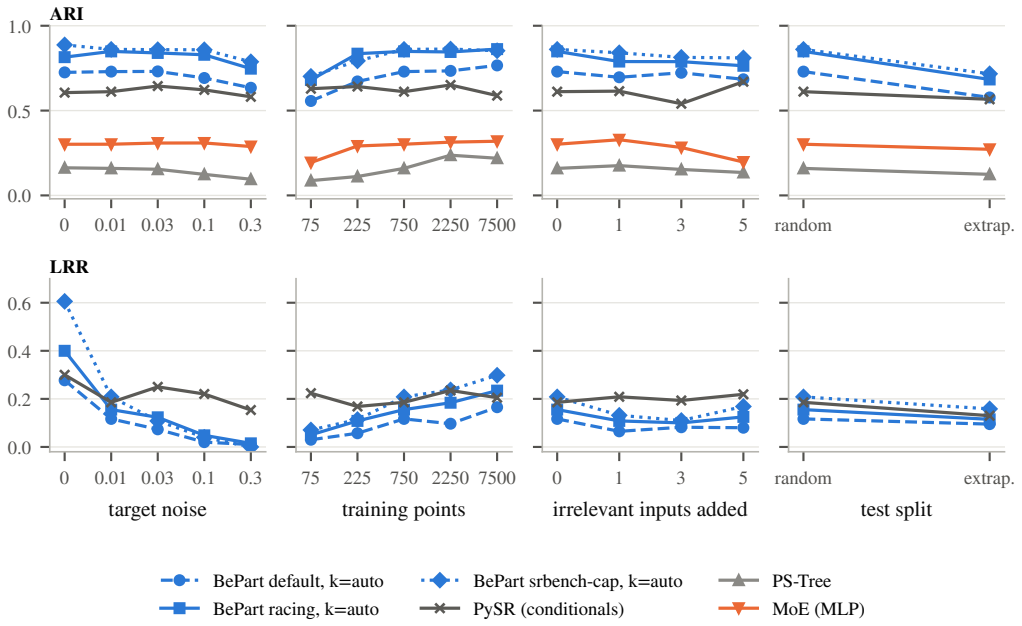

Figure 5: Robustness on ten regime problems (10 seeds): ARI (top) and LRR (bottom) against target noise, training-set size, number of irrelevant inputs and the test split.