Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Evolutionary Minimalism and Phase-Space Brittleness: A Comparative Analysis of the Universal EML Operator versus Heterogeneous Operators in Genetic Programming

Dimitar Georgiev Kazakov

dimitar.kazakov@gmail.com

Technical University of Varna

Research Article

Keywords: Genetic programming, Symbolic regression, EML operator, Phase-space brittleness, K-complexity, Crossover ablation, Tree bloat, Deceptive fitness landscape

Posted Date: June 23rd, 2026

DOI: https://doi.org/10.21203/rs.3.rs-10006050/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Evolutionary Minimalism and Phase-Space Brittleness: A Comparative Analysis of the Universal EML Operator versus Heterogeneous Operators in Genetic Programming

Dimitar Georgiev Kazakov[1]

[1] Technical University of Varna, 1 Studentska Str, 9010 Varna, Bulgaria

dimitar.kazakov@gmail.com

ORCID: 0009-0006-5618-9174

Abstract Traditional Genetic Programming (GP) for symbolic regression relies on heterogeneous operator sets (e.g., +, −, ×, ÷, exp, ln, sin) to explore continuous function spaces. The recent publication of a paper describing the Exp-Minus-Log (EML) universal operator, eml(x,y) = exp(x) − ln(y), proposes that all elementary continuous mathematics can in principle be generated from a single primitive and the constant 1. This offers an interesting area for exploration, does grammatical homogeneity benefit evolutionary search. We present a controlled empirical comparison of EML-GP against standard heterogeneous GP across twelve symbolic regression targets organised in four phases of increasing complexity. Since the EML analogues of primitives require long rigid cascades we isolated the role of crossover, the experiment is conducted under two parallel regimes: Regime A (crossover + mutation) and Regime B (mutation only). To account for parametric influence we implemented a 3×3 grid of generation and population budgets, Each cell performed 30 deterministic runs, yielding in total 12,960 trials. EML-GP is able to recover its native axioms (exp at K=3, ln at K=7) but fails systematically outside this domain. Two structurally distinct mechanisms produce that failure. A process we termed, Phase-space brittleness produces a single-node collapse on targets requiring rigid intermediate-K cascades. The second process is uncontrolled bloat on targets requiring longer emulation paths. The regime comparison reveals that crossover's effects on EML-GP are not uniformly destructive, but rather K-complexity dependent. At lower K-complexity (K=7) crossover has a constructive effect, enabling exact convergence that mutation alone fails to achieve. At intermediate complexity ($19 \leq K \leq 100$) structural failure reaches its maximum. At high complexity ($K \geq 300$) both operators are exhausted by the search landscape. Semantic similarity-based crossover likewise fails to mitigate the pattern, confirming that the failure is grammar-intrinsic rather than operator-specific. Semantic similarity-based crossover likewise fails to mitigate the pattern, confirming that the failure is grammar-intrinsic rather than operator-specific. More broadly, the results demonstrate that representational universality and evolutionary accessibility are orthogonal properties. A grammar that can express any continuous function does not become searchable by gradient free methods. Code and full grid-search results are available at: https://doi.org/10.5281/zenodo.20366631 and https://github.com/swogger/EML-GP.

Keywords: Genetic programming; Symbolic regression; EML operator; Phase-space brittleness; K-complexity; Crossover ablation; Tree bloat; Deceptive fitness landscape

1. Introduction

Genetic programming is a popular method for explicit discovery of analytical models of data through symbolic regression [1, 2, 8]. The selection of the primitive functional set plays a major role in the success or failure of this process [3, 7, 11]. If the set is too restricted or the selection of functions is suboptimal the model struggles to approximate the function and often results in a sharp growth in the functional nodes (tree bloat) [5, 6, 20]. On the other end of the spectrum, overloading the set with an excess of unnecessary functions results in the exponential expansion of the search space and heterogeneous nodes create disjointed crossover dynamics. Balancing reduction of the elements and completeness of the set are two diametrically opposed factors. At present there is no definitive universal methodology that can offer optimal set selection for the discovery of unknown functions.

The tension between representational simplicity and search efficiency is not limited to GP only, it can be found within the field of soft computing in various forms. Other major domains such as Neural Networks face the same challenge when optimising between expressive heterogeneous activation functions and simpler unified primitives such as ReLU. Similarly, the fields of Fuzzy Systems and Neuro-Fuzzy hybrids face similar considerations when dealing with membership functions and rule structures. Recent developments in Cartesian Genetic Programming have been focussed on minimal primitive sets for evolutionary circuit design [21]. An intuitive hypothesis evolves from investigating those challenges, if a mathematically universal primitive for continuous functions exists, does

it benefit gradient-free search and how? GP offers a clean environment for testing this question. Since tree-based representations make the trade-off explicit, the cost of homogeneity is paid in tree depth, which is easily quantifiable.

In his paper published in March 2026, Odrzywołek [9] proposed a new universal operator in the field of continuous mathematics. It was presented as an analogue of the NAND gate in Boolean digital logic.
The operator is defined as:

$$eml(x, y) = exp(x) - ln(y)$$

When the constant 1 is introduced the operator can natively reconstruct arithmetic, algebraic, and transcendental functions over the complex plane.

If empirically viable, EML could substantially simplify operator-set design in evolutionary symbolic regression. A genetic algorithm utilising only EML operates on a uniform grammar: $S \to 1 \mid V \mid eml(S, S)$, where V represents input variables. This has the potential to remove one of the major challenges in this field, a goal shared by recent work in compact symbolic regression [19]. The implementation of the EML operator, however, has some intrinsic issues: some basic functions, like algebraic addition, require deep nested tree architectures.

This paper addresses: *Does the syntactic simplicity and uniformity of the EML search space outweigh the penalty of requiring deeper computational trees in an evolutionary context?*

The remainder of this paper is organised as follows:

- **Section 2** - relevant literature review of genetic programming and the EML formalism
- **Section 3** - details the experimental methodology and the K-complexity-scaled patience mechanism used to ensure fair comparison
- **Section 4** - presents the four-phase test matrix and execution grid
- **Section 5** - reports the empirical results: tracing the EML-GP's arc from success on its native axioms; through algebraic collapse and deceptive-landscape failure; to bloat-driven timeouts at scale .
- **Section 6** - introduces phase-space brittleness as a destructive mechanism and its K-complexity-dependent refinement based on the regime comparison.
- and **Section 7** - offers a conclusion with notes on operator design and three directions for future work.

2. Background and Related Work

2.1 Genetic Programming (GP)

Classical GP represents mathematical functions using a mathematical tree structure where each internal node represents a function and each leaf node contains a value (variables and constants). Both functions and values are usually predefined in sets, functions/operators inF_{std} , and terminal set T_{std} for variables and ephemeral random constants respectively (ERCs) [1, 10, 13].

An initial population of randomly generated trees of predefined size is created. This population is then iterated upon. In each iteration, each tree's fitness is assessed against a fitness function. At the end of each iteration assessment a subset of the entire population that has shown the best results is selected, subtrees are recombined via crossover and a new population set is created for the next iteration. This process continues until a predefined limit of iterations is reached or one or more trees achieve 0 error in their fitness testing.

Due to the unpredictable nature of the crossover and in many cases the distinctly different mathematical features of the functions within nodes (e.g., the periodicity of $sin(x)$ versus the unbounded growth of $exp(x)$), crossover often produces non-viable offspring. This phenomenon is known as the "building block disruption problem" [12]. A substantial body of follow-on work has sought to mitigate this issue through semantic-aware variation operators that constrain offspring to remain close to their parents in the space of fitness behaviour [17, 18], but these approaches have not yet been evaluated against universal-operator grammars such as EML.

2.2 Bloat: Causes, Theories, and Control

"Tree bloat" defined as the uncontrollable growth of the tree size without notable corresponding improvement in fitness is a well studied phenomenon [5, 6, 22]. Several theories aiming at explaining the root cause of this phenomenon have been proposed. Langdon and Poli [6, 25] advance the hypothesis that selection pressure towards

fitness has a positive bias towards larger programs. The mechanics behind this hypothesis are explained by the existence of introns (semantically inactive code) in GP trees which shield functional subtrees during crossover. Another theory advanced by Soule and Foster [23] identified removal bias as a driver. Because smaller subtrees are more likely to be selected for crossover the functional code migrates towards the root of the tree while inert code drifts towards the branches. Soule and Foster [34] show that when evolutionary dynamics interfere with parsimony pressure, size penalties can overwhelm the fitness score. This creates "failed" populations that halt productive search. A comprehensive study by Silva, Dignum, and Vanneschi [24] attempts to unify these theories. The authors propose their own technique: operator equalisation. It targets the issue by directly controlling the distribution of program lengths in the population. This results in an essentially bloat-free evolution on standard benchmarks. Another comparative study by Luke and Panait [22] investigated nine bloat-control methods and found that depth limiting combined with a punitive size measure generally outperformed either approach alone. The current study uses a fixed maximum tree-size budget of 2,000 nodes as a hard ceiling. It doesn't apply parsimony pressure or operator equalisation. This decision was made to allow the observation of EML-GP's natural bloat dynamics rather than to suppress them.

2.3 Crossover Disruption and Semantic Approaches

The process of standard subtree crossover is defined as: selecting random crossover points in two parent trees and exchanging those subtrees. Each respective subtree is rooted at the crossover root point in their new host tree. Angeline [26] demonstrated in his experiment of the "headless chicken crossover" that replacing a parent tree for a completely randomly generated new parent tree yields the same results, essentially acting as a mutation rather than a recombination. This calls into question the constructive role of crossover as originally intended. D'haeseleer [27] proposes a more sophisticated version of crossover: context-preserving crossover. Instead of fully relying on randomness as a selecting factor for crossover, his approach restricts the crossover points to corresponding depths and positions within the parent trees. Nordin, Francone, and Banzhaf [33] showed that explicitly defined introns have a protective effect against a crossover disruption but the cost is an increase in the tree size and compounding bloat. Poli and McPhee [35] formalised a general schema theory for GP with subtree-swapping crossover. They demonstrated that the probability of transmitting a complete schema to offspring decays rapidly with schema size. This finding is directly applicable to the phase-space brittleness we observe with EML's large, rigid K-complexity cascades. Recent developments such as semantically-guided crossover operators have emerged as promising alternatives. Beadle and Johnson [29] enforced semantic change at every crossover event by rejecting offspring equivalent to their parents, while Krawiec and Pawlak [30] developed locally geometric semantic crossover, approximating Moraglio et al.'s [18] theoretical framework at the subtree level. Galván-López et al. [31] and Daida et al. [32] analyzed how fitness landscape structure impacts problem difficulty in GP. They showed that landscapes lacking smooth gradients toward the optimum are systematically harder for crossover-based search. This matches the deceptive landscape structure we document for EML-GP in Section 5.2. These findings collectively explore the destructive potential of subtree crossover in standard GP. The present study demonstrates that these destructive effects are K-complexity-dependent: crossover assists short EML cascades but renders intermediate-to-high-K targets non-viable.

2.4. The Universal EML Operator

Odrzywołek developed a new operator eml(x, y) [9] that unifies the distinct function classes of continuous mathematics. He provided examples in his paper of the derivation of: exponential, logarithmic, algebraic, and trigonometric, all under a single primitive. For example:

$$exp(x) = eml(x, 1)$$

$$ln(x) = eml(1, eml(eml(1, x), 1))$$

He argues that each mathematical function can be fully represented through self composition of the EML operator. In some cases such as $sin(x)$ and constants like π, the EML the implementation requires transitioning through negative and imaginary spaces.

This necessitates EML to evaluate strictly over the complex domain $\mathbb{C}$ in delivering real number outputs for real number inputs. Despite this cost of increased complexity, the overall advantage of utilising a universal, unifying EML operator for a GP algorithm in place of a standard functional set is considerable. The utilisation of one universal operator can enable, in principle, the evolution of any primitive required and reduce the grammatical constraints.

3. Methodology and Theoretical Foundations

3.1. Representation and Grammar

The experiment was designed as a comparative analysis of two otherwise identical GP implementations, where the first utilises a standard set of mathematical functions, while the second uses exclusively the EML operator. A further complementary test was performed to measure the effects of semantic-aware crossover.

- **Std-GP (Control):** Operates with a standard heterogenous set of standard primitives (e.g. $+, -, \times, \div, exp, log, sin, sinh, arctan$).Standard operators possess an evolutionary structural cost of exactly 1 node.
- **EML-GP (Experimental):** Uses exclusively the EML operator $eml(x, y) = exp(x) - ln(y)$. It evaluates over the complex plane using IEEE-754 floating-point precision, imposed by our Python implementation.
- **SC-GP (Semantic baseline):** This test was introduced later in the research to investigate the effects of semantic-aware crossover on the brittleness. Semantic Similarity-based Crossover (SSC) following Uy et al. [28]. For each crossover event, ten candidate points are sampled from each parent. The algorithm selects the pair with the most similar subtree outputs, measured by the mean squared difference across 20 random fitness cases. This setup is tested at the reference cell using six representative benchmarks across Phases 1–3.

3.2. Structural Cost and the Patience Multiplier

The inherent design of the Odrzywołek's EML operator requires deep, nested trees of specific structure. The complexity of the tree is measured as K-complexity.

Definition 1 (K-complexity). For a target function f, the K-complexity K(f) is the minimum number of nodes in an EML-only tree that exactly computes f over the evaluation domain.

$$K(f) = min\{|T|: T \in L(G_{EML}) \wedge \forall x \in D, [\![T]\!](x) = f(x)\}$$

where

$$G_{EML}: S \to 1 \mid V \mid eml(S, S).$$

$|T|$is the number of nodes in tree$T$,
$L(G_{EML})$ represents the language generated by the EML grammar,
D is the evaluation domain,
and $[\![T]\!]$ denotes the semantics of T under IEEE-754 complex arithmetic.
By construction, $K(f)$ is unique, since it is defined as the minimum over a well-ordered set.
Where $K(f)$ cannot be derived in closed form from Odrzywołek's axioms [9], we report shortest-known upper-bound estimates, denoted by $\leq$ in Tables 10 and 11.

An example of a simple EML analogue is the $exp(x)$, which requires exactly K=3 nodes $(eml(x, 1))$. A simple addition $(x + y)$demands K=19. Synthesising $sin(x)$requires K=351 nodes.

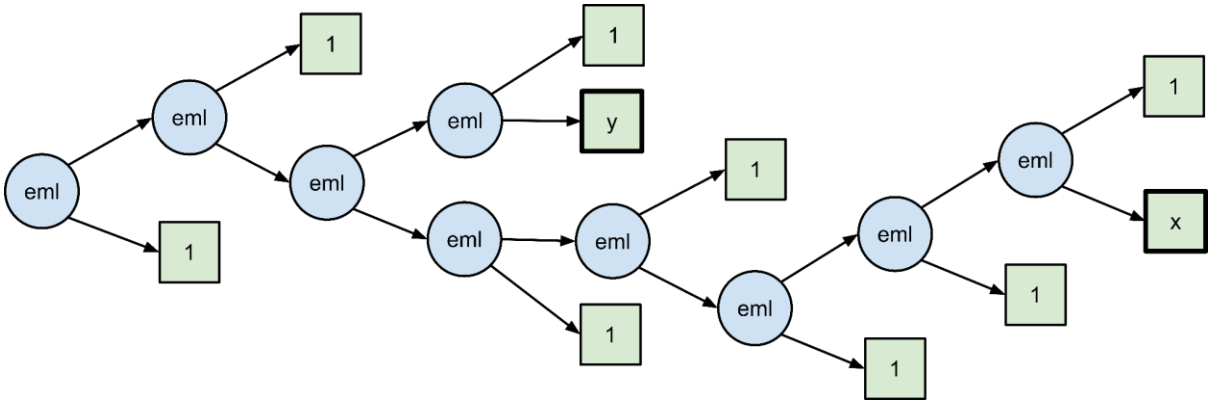


Fig. 1. Tree representation of the EML synthesis of algebraic addition $x_1 + x_2$, requiring exactly K=19 nodes. Each internal node is the EML operator; leaves are either the input variables x_1, x_2 or the constant 1. The structural rigidity of this configuration illustrates the phase-space brittleness exploited by random subtree crossover.

As a result of EML's inherently deeper trees, a flat early-stopping patience window would unfairly terminate EML runs before they could construct these larger architectures. To account for this, we scaled EML-GP's early-stopping patience by the mean K-complexity of the Std-GP operator set on each test.

4. Experimental Setup and Test Matrix

The experiment consists of two parallel operator regimes. Each regime has a four-phase matrix of 12 tests of increasing complexity. These tests are executed under a 3×3 computational budget grid. The phased approach borrows from modern symbolic regression benchmarks [4].

Evaluation Environment: The EML operator requires negative and imaginary logarithms to generate periodic functions and specific constants. Therefore, all nodes in both algorithms are evaluated over the complex domain ($\mathbb{C}$) using Python's cmath library (IEEE-754 floating-point arithmetic). Division by zero and logarithmic singularities return a complex infinite penalty. This penalty successfully culls non-viable branches.

4.1 Experimental Regimes

During the experimental design phase we identified an expected theoretical limitation of the EML model. Because EML requires long cascades to implement analogous functions to the primitives in the competing standard GP implementation, crossover is likely to affect this variant's fitness adversely: subtree recombination across non-aligned cascade points may destroy semantic structure that mutation alone would preserve. This yields a specific, falsifiable hypothesis: crossover causes the performance gap between EML and Std-GP. Our experimental design directly tests this claim.

Multiple strategies to mitigate subtree brittleness are outlined in the related work section; however, iterating through each of them, or sweeping crossover rates across a continuous range, would multiply the experimental volume several-fold and are therefore reserved for dedicated future work. For the purposes of exploring the boundaries of crossover as a factor within the present study, we implemented a binary ablation in the form of two test regimes that share the Phase structure and execution grid described above.

Regime A configures the experiment with a complete classic GP setup, where tree evolution is driven by both mutation and crossover. Measurements of EML performance against Standard GP in this regime allow direct forecasting of the impact of EML if transplanted into existing commercial and academic platforms, where crossover is standard practice.

Regime B removes crossover as a factor completely, leaving mutation as the only evolutionary force. This regime is proposed as a controlled measurement that isolates EML's intrinsic search behaviour from the operator most likely to exploit its structural fragility.

The two regimes are designed to isolate the effects of crossover. The comparison is informative under either outcome. Substantial improvement by EML-GP in Regime B supports the long-cascade hypothesis, motivating targeted mitigation work. Alternatively, if EML-GP shows little change between regimes, crossover is ruled out as the dominant factor. Thus strengthening the parsimony observations in Regime A against that confound.

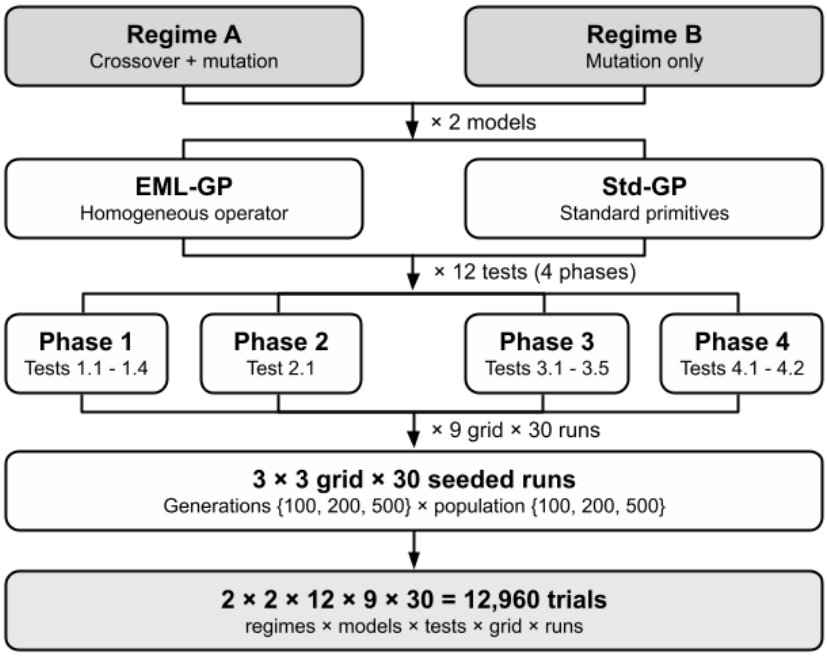


Fig. 2. Organisational structure of the experimental design. Both operator regimes: crossover with mutation (Regime A) and mutation only (Regime B), are applied to both GP models across twelve tests. Those twelve tests are organized into four phases of increasing complexity. Phase 5 is a separate investigation outside this hierarchy. Each test is evaluated using a 3×3 grid of generation and population sizes. With 30 independent runs per grid cell, the main experimental setup comprises 12,960 total trials.

Detailed grid comparisons are quantified in Appendix A (Table A2), which logs where EML-GP's bloat exhausts the computational budget. For non-native targets, results at larger scales (generation ≥ 200, population ≥ 200) are dominated by timeout rather than evolutionary outcomes.

4.2 Execution Grid

In order to detect any parameter dependent effects we structured the test by introducing a simple 3×3 grid search. Combination of two fundamental GP parameters were considered: generation count and population size. Each parameter takes a predefined value from an incremental set: [100,200,500]. And each combination is executed for 30 deterministic runs (pre-selected seed was used to aid reproducibility). Two practical measures to mitigate non-constructive tree bloat were introduced: A practical hard limit of 2,000 nodes per tree; and a CPU time timeout of 600 seconds for each individual trial. Timeouts are recorded as FAIL in the raw results data.

Analysis of the timeout data, summarised in Table 1, reveals that this limit produces a structured computational boundary rather than random censoring across the grid. In comparison, for the four targets where EML-GP trivially converges or collapses (exp(x), ln(x), addition, sinh(x)) to a single node, no timeouts are observed at any grid cell. For the remaining eight targets where EML can potentially exhibit tree bloat, the timeout rate is jointly determined by generation count and population size. All eight targets complete reliably at (generations ≤ 200, population = 100) and (generations = 100, population ≤ 200), where total computational work remains at or below approximately 20,000 individual-generations. The timeout rate rises to 75% at (generations = 200, population = 200) and reaches 100% for all bloat-prone targets at (generations = 200, population = 500) and at every cell with generations ≥ 500.

Notably, generation count contributes disproportionately to this boundary relative to population size: because EML-GP trees bloat over successive generations, the per-individual evaluation cost rises non-linearly with generation count in a way that a one-time population multiplier does not.

Table 1. EML-GP timeout rates across the 3×3 grid for the eight bloat-prone targets. Each dash in Table A2 (Appendix A) counts as one timeout. Rates are computed over eight tests per cell.

	Population: 100	Population: 200	Population: 500
Generations: 100	0%	0%	50%
Generations: 200	0%	75%	100%
Generations: 500	100%	100%	100%

Cells within the timeout zone are reported separately in Appendix A and interpreted as evidence of computational infeasibility rather than algorithmic performance. The 600-second limit should therefore be understood as a conservative lower bound on the true computational asymmetry between the two algorithms: EML-GP trials that reached the timeout before completing would, if permitted to run to termination, have produced further bloated trees, deepening rather than narrowing the gap against Std-GP.

4.3 Experimental Progression and Phase Objectives

Each phase targets a specific mathematical capability:

- **Phase 1: Native Axioms (Tests 1.1 – 1.4)**
  Goal: Establishing the baseline by testing EML-GP's ability to evolve its own theoretical minimal axioms (exp and ln). Test synthesis of $(x_1 + x_2)$without native arithmetic functions.

- **Phase 2: Periodic Space (Test 2.1)**
  Goal: Evaluate complex-plane phase bridging. Because EML uses Euler's formula to generate trigonometric functions, this phase targets how the algorithm composes such structures and how they respond to subtree crossover across both experimental regimes.

$$e^{ix} = cos(x) + i \times sin(x)$$

- **Phase 3: Complex Limits (Tests 3.1 – 3.5)**
  Goal: Test for deep topological nesting. The algorithm is expected to evolve hyperbolic functions, non-linear inverse mappings, sigmoids and polynomials. We pay particular attention to the K-complexity required by the EML to construct these advanced domains.

- **Phase 4: Rugged Space (Tests 4.1 – 4.2)**
  Goal: This phase defines the most complex challenge, testing the relative limits of EML's algorithmic resilience. It examines the synthesis of multidomain composites and how they respond to subtree crossover across both experimental regimes.

- **Phase 5: Standard Benchmarks (Tests 5.1–5.4)**
  Goal: Validate findings against established symbolic regression benchmark suites [3, 4] to confirm that the failure modes identified in Phases 1–4 generalise to targets not designed for this study.

Table 2. Experimental test matrix. This table aligns the target function f(x) the functional subset for Std-GP (F_{std}), and the evaluation domain. In the case of EML-GP, it is restricted to the universal operator $eml(x, y)$ and the terminal constant 1 across all tests. Operator set abbreviations: **Arith** = {+, −, ×, ÷}; **Exp** = {exp, ln}; **Trig** = {sin, cos, tan}; **HTrig** = {sinh, cosh, tanh}; **Inv** = {arctan}; **Sqrt** = {√x}. Phase 5 benchmarks are evaluated at the reference cell (Generations: 200, Population: 100) only, under both regimes, to validate findings against established suites without replicating the full 3×3 grid.

ID	Target f(x)	Std-GP Set (Fstd)	Std-GP Range	Objective / Competency
1.1	$exp(x)$	Arith, Exp	$[-3.0, 3.0]$	Axiom 2 (K=3). Baseline sanity check.
1.2	$ln(x)$	Arith, Exp	[0.1,5.0]	Axiom 3 (K=7). Proves native discovery.

1.3	$x_1 + x_2$	Arith	[−5.0,5.0]	Algebraic baseline synthesis (K=19).
1.4	$exp(x)exp(-x^2)$	Arith, Exp	[−2.0,2.0]	Native composite. Tests smooth gradients.
2.1	$sin(x) + cos(x)$	Arith, Trig	[−π,π]	Euler's formula bridging (K≤35l).
3.1	$sinh(x)$	Arith, Exp, HTrig	[−3.0,3.0]	Deep exponential phase compositions.
3.2	$arctan(x)$	Arith, Exp, Trig, Inv	[−10.0,10.0]	Complex non-linear inverse mappings.
3.3	x^x	Arith, Exp, Sqrt	[0.1,3.0]	Dynamic arbitrary exponentiation.
3.4	$(1 + exp(-x))^{-1}$	Arith, Exp	[−5.0,5.0]	Sigmoid activation; evaluates asymptotic bounds.
3.5	$x^3 - 2x^2 + x$	Arith	[−5.0,5.0]	High-stress integer composition and polynomial roots.
4.1	$exp(sin(x)) + ln(x^2 + 1)$	Arith, Exp, Trig	[−3.0,3.0]	Multi-domain composite disruption.
4.2	$sin(x_1 exp(x_2)) - cos(ln(x_1^2 + x_2^2 + 0.1))$	Arith, Exp, Trig	[−2.0,2.0]	Absolute limits of symbolic crossover dynamics.
5.1	$x^3 + x^2 + x$	Arith, Exp	[−1.0,1.0]	Nguyen-1 [4]. Standard polynomial benchmark.
5.2	$ln(x + 1) + ln(x^2 + 1)$	Arith, Exp	[0.0,2.0]	Nguyen-7 [4]. EML-native composite with addition barrier.
5.3	$\Sigma(1/i, i = 1..x)$	Arith, Exp	[1,50]	Keijzer-6 [3]. Harmonic series; iterative structure.
5.4	$10/(5 + \Sigma(x_i - 3)^2)$	Arith, Exp	[0.05,6.05][5]	Vladislavleva-4 [3]. Multivariate rational function.

5. Results and Critical Analysis

A number of important observations are worth noting in advance of a more detailed phase-by-phase result analysis. For each population size, convergence rates were essentially invariant to the number of generations. Runs that succeeded at the same rate across the each (100,200,500) generations limit. The same uniformity in outcomes was observed in failed runs. We propose two processes that explain this stability:

1. **The early stopping mechanism** which employs a K-complexity-scaled patience window for EML-GP. It terminates execution once fitness improvement plateaus. Increasing the maximum generation budget rarely alters the effective run length.
2. **The deceptive structure of the EML search landscape** (analysed in Section 5.2), once a population settles to a local maximum, it fails to reliably escape it within any practical generation budget. We propose that the failure of EML-GP is structural rather than compute-bound. This pattern is replicated in Regime B, where the same generation-invariant structure persists under mutation-only evolution. This confirms that the cause is the topology of the EML search landscape rather than any property of the crossover operator.

To formalise the following comparisons, we apply two non-parametric tests at each target function. Convergence rates are compared using Fisher's exact test on the binary success/failure outcomes across 30 runs per algorithm. Final tree sizes are compared using the Mann-Whitney U test, with rank-biserial correlation reported as the effect size (negligible: $|r| < 0.1$; small: < 0.3; medium: < 0.5; large: ≥ 0.5). All p-values are adjusted using the Holm-Bonferroni procedure ($\alpha = 0.05$) to control the family-wise error rate across the twelve target functions. Tables 6 and 7 report results for the (Generations: 200, Population: 100) grid cell. It serves as the primary reference within the "safe zone" identified in Table 1 and corresponds to Figure 6. Meanwhile, Table 3 summarizes significance counts across all nine grid cells. The complete per-cell tables are provided in Supplementary Material S1.

Table 3. Cross-cell summary of statistical significance counts. For each grid cell, the table displays the number of target functions (out of 12) with statistically significant differences between Std-GP and EML-GP after Holm-Bonferroni correction. The third column indicates the number of tests where a Mann-Whitney comparison was feasible. At higher computational budgets, EML-GP timeouts exclude paired data, thereby reducing testable comparisons. Finally, convergence-rate counts remain identical across generation budgets within each population size, corroborating the early-determination finding.

Grid cell (Gens × Pop)	Convergence: significant of 12	Tree size: significant of 12	Tree size: tests with EML-GP data
100 × 100	6	6	12
100 × 200	7	7	12
100 × 500	8	4	4
200 × 100	6	8	12
200 × 200	7	4	4
200 × 500	8	4	4
500 × 100	6	4	4
500 × 200	7	4	4
500 × 500	8	4	4

5.1 Validation of Theoretical Minima (The Native Domain)

EML-GP performs exceptionally well when the target topology is native to its core axioms.

- **Exponential (Test 1.1):** EML-GP achieved a 100% convergence rate across all grid sizes with median tree size of exactly 3.0 nodes. This perfectly matches Axiom 2: eml(x,1).
- **Logarithmic (Test 1.2):** EML-GP achieved a 76% to 90% convergence rate. The rate settles consistently on the expected theoretical minimum of 7.0 nodes: $eml(1, eml(eml(1, x), 1))$.

The convergence trajectories for these native targets are shown in Figs. 3 and 4. Both algorithms locate the target solution within the first few generations. EML-GP's slightly slower descent on Test 1.2 reflects the extra structural cost of the K=7 logarithm cascade.

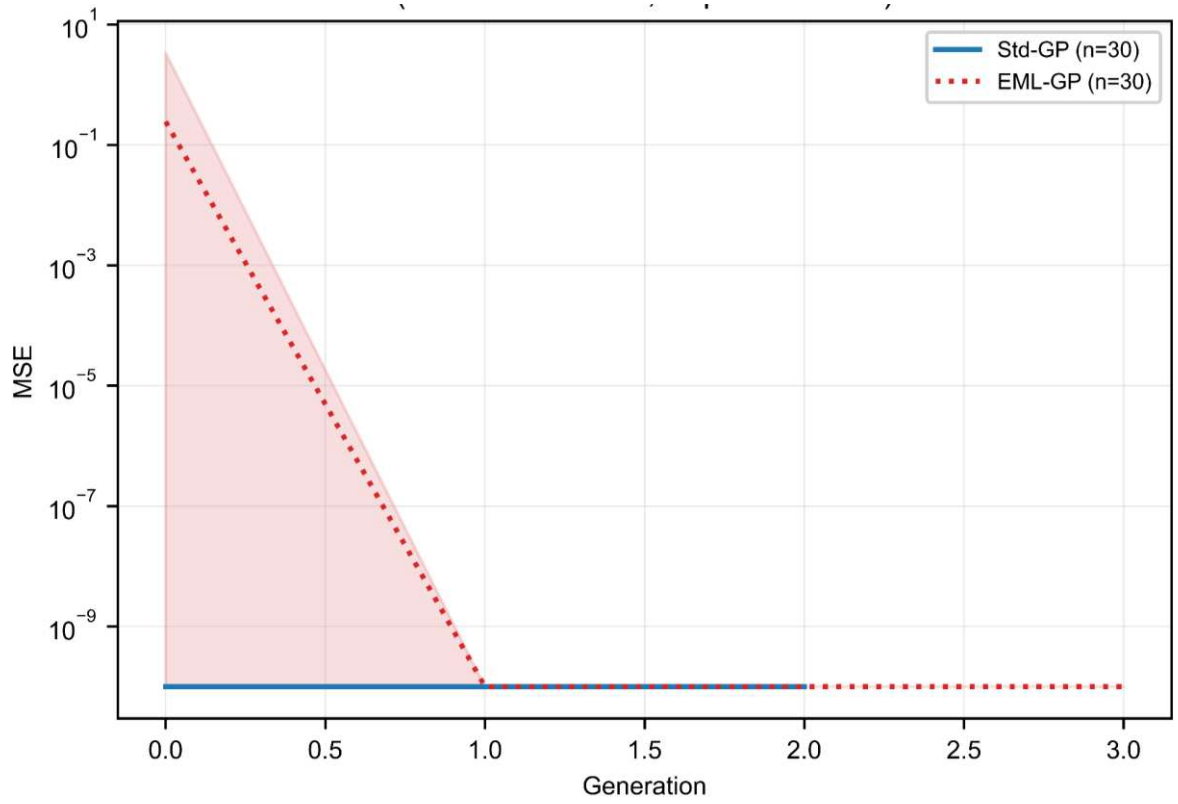
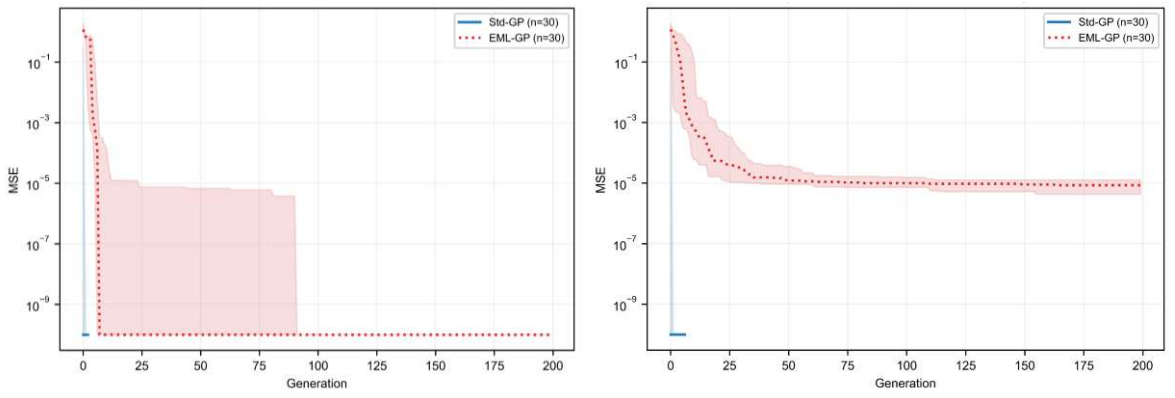


Fig. 3. Convergence trajectory for Test 1.1, $exp(x)$(Generations: 200, Population: 100; n = 30 runs per algorithm). Solid lines show median best-fitness, shaded bands show interquartile range across runs. Both Std-GP and EML-GP discover the target essentially in the initial population, Std-GP through the exp primitive in F_{std} and EML-GP through the K=3 axiom $eml(x, 1)$. The trajectory is identical under Regime B (not shown).



(a) Regime A : crossover + mutation (b) Regime B : mutation only

Fig. 4. Convergence trajectories for Test 1.2, $ln(x)$, under (a) Regime A: crossover + mutation, and (b) Regime B: mutation only at (Generations: 200, Population: 100; n = 30 runs per algorithm). In Regime A, EML-GP discovers the K=7 axiom structure within 5 to 10 generations and the median trajectory reaches exact convergence by generation 7. In Regime B, the median trajectory descends more slowly and stalls at approximately 8.5×10^{-6}. It remains flat through the full 200-generation budget. Std-GP converges in the initial generation in both regimes.

The regime comparison on Test 1.2 reveals the first significant finding from the crossover ablation. In Regime A, crossover enables EML-GP to reach exact convergence on the K=7 cascade. The median fitness drops to 0.0 by generation 7 and remains there. In Regime B, mutation alone drives steady improvement but is unable to close the final gap: the median stalls at approximately 8.5×10^{-6} across all nine grid cells without exception and does not improve with additional generations or larger populations. The K=7 cascade is short enough that subtree recombination can assemble correct sub-structures faster than point mutation can discover them, consistent with Poli and McPhee's schema theory in which small schemas transmit reliably under crossover [35]. This result

establishes that crossover's effect on EML-GP is K-dependent rather than uniformly destructive, a mechanistic distinction developed further in Section 6.1.

Table 4. Convergence rates of Std-GP across the 3×3 (Generations × Population) grid for each target function. Each cell reports the proportion of 30 deterministic runs achieving fitness convergence (within tolerance) before the budget exhaustion or the 600-second CPU time timeout. Std-GP reaches near-perfect convergence on native and polynomial targets but degrades on multi-domain composites (Nested I, Nested II).

	Generations: 100			**Generations: 200**			**Generations: 500**		
Test Case	**Pop: 100**	**Pop: 200**	**Pop: 500**	**Pop: 100**	**Pop: 200**	**Pop: 500**	**Pop: 100**	**Pop: 200**	**Pop: 500**
Exp(x)	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Ln(x)	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Addition	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Exp Comp.	0.47	0.60	0.77	0.47	0.60	0.77	0.47	0.60	0.77
Sin+Cos	0.20	0.43	0.60	0.20	0.43	0.60	0.20	0.43	0.60
Sinh(x)	1.00	1.00	0.97	1.00	1.00	0.97	1.00	1.00	0.97
Arctan(x)	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
x^x	0.33	0.37	0.50	0.33	0.37	0.50	0.33	0.37	0.50
Sigmoid	0.17	0.13	0.27	0.17	0.13	0.27	0.17	0.13	0.27
Polynomial	0.93	1.00	1.00	0.93	1.00	1.00	0.93	1.00	1.00
Nested I	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Nested II	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Table 5. Convergence rates of EML-GP across the 3×3 grid. Convergence is restricted to the operator's native axioms (exp, ln) with marginal partial success in addition. EML-GP fails to converge (rate = 0.00) on every target outside the native exponential domain, regardless of computational budget, consistent with the phase-space brittleness mechanism described in Section 6. Regime A only; Regime B convergence data are available in the supplementary material.

	Generations: 100			**Generations: 200**			**Generations: 500**		
Test Case	**Pop: 100**	**Pop: 200**	**Pop: 500**	**Pop: 100**	**Pop: 200**	**Pop: 500**	**Pop: 100**	**Pop: 200**	**Pop: 500**
Exp(x)	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Ln(x)	0.77	0.80	0.90	0.77	0.80	0.90	0.77	0.80	0.90
Addition	0.00	0.03	0.03	0.00	0.03	0.03	0.00	0.03	0.03
Exp Comp.	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Sin+Cos	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Sinh(x)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Arctan(x)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
x^x	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

	Generations: 100			Generations: 200			Generations: 500		
Test Case	**Pop: 100**	**Pop: 200**	**Pop: 500**	**Pop: 100**	**Pop: 200**	**Pop: 500**	**Pop: 100**	**Pop: 200**	**Pop: 500**
Exp(x)	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Ln(x)	0.77	0.80	0.90	0.77	0.80	0.90	0.77	0.80	0.90
Sigmoid	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Polynomial	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Nested I	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Nested II	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Table 6. Convergence-rate comparison between Std-GP and EML-GP at the (Generations: 200, Population: 100) grid cell. p-values from Fisher's exact test on success/failure counts; Holm-corrected across the 12 target functions at $\alpha = 0.05$. Δ is the difference in convergence proportion (Std-GP − EML-GP). The six target functions marked significant at this cell are significant in all nine grid cells (Table 3; Supplementary Material S1).

ID	Test Case	Std-GP rate	EML-GP rate	Δ	p (raw)	p (Holm)	Significant
1.1	**Exp(x)**	1.00	1.00	+0.00	1.000	1.000	No
1.2	**Ln(x)**	1.00	0.77	+0.23	0.011	0.063	No
1.3	**Addition**	1.00	0.00	+1.00	<0.001	<0.001	Yes
1.4	**Exp Comp.**	0.47	0.00	+0.47	<0.001	<0.001	Yes
2.1	**Sin+Cos**	0.20	0.00	+0.20	0.024	0.119	No
3.1	**Sinh(x)**	1.00	0.00	+1.00	<0.001	<0.001	Yes
3.2	**Arctan(x)**	1.00	0.00	+1.00	<0.001	<0.001	Yes
3.3	**x^x**	0.33	0.00	+0.33	<0.001	0.006	Yes
3.4	**Sigmoid**	0.17	0.00	+0.17	0.052	0.209	No
3.5	**Polynomial**	0.93	0.00	+0.93	<0.001	<0.001	Yes
4.1	**Nested I**	0.00	0.00	+0.00	1.000	1.000	No
4.2	**Nested II**	0.00	0.00	+0.00	1.000	1.000	No

Table 7. Tree-size comparison between Std-GP and EML-GP at the (Generations: 200, Population: 100) grid cell. Median final tree size across runs; p-values from Mann-Whitney U test on the distributions; effect size as rank-biserial correlation r. Sign convention: positive r means EML-GP produces larger trees; negative r means Std-GP produces larger trees. This grid cell is reported as the primary reference cell within the timeout-free safe zone (Table 1); at higher computational budgets, EML-GP timeouts remove tree-size observations from the comparison (Table 3).

ID	Test	Std-GP median	EML-GP median	p (raw)	p (Holm)	Effect size	Significant
1.1	**Exp(x)**	2.0	3.0	0.004	0.022	+0.400 (medium)	Yes

1.2	**Ln(x)**	2.0	7.0	<0.001	<0.001	+0.844 (large)	Yes
1.3	**Addition**	5.0	1.0	<0.001	<0.001	−0.933 (large)	Yes
1.4	**Exp Comp.**	13.5	273.0	0.008	0.040	+0.583 (large)	Yes
2.1	**Sin+Cos**	15.0	295.0	0.041	0.165	+0.624 (large)	No
3.1	**Sinh(x)**	2.0	1.0	<0.001	<0.001	−0.923 (large)	Yes
3.2	**Arctan(x)**	2.0	461.0	<0.001	<0.001	+1.000 (large)	Yes
3.3	**x^x**	13.0	399.0	<0.001	<0.001	+0.904 (large)	Yes
3.4	**Sigmoid**	136.0	648.0	0.115	0.345	+0.528 (large)	No
3.5	**Polynomial**	43.0	485.0	<0.001	<0.001	+0.953 (large)	Yes
4.1	**Nested I**	321.0	443.0	0.377	0.377	+0.259 (small)	No
4.2	**Nested II**	228.5	419.0	0.184	0.369	+0.313 (medium)	No

5.2 Algebraic Collapse and Deceptive Landscapes

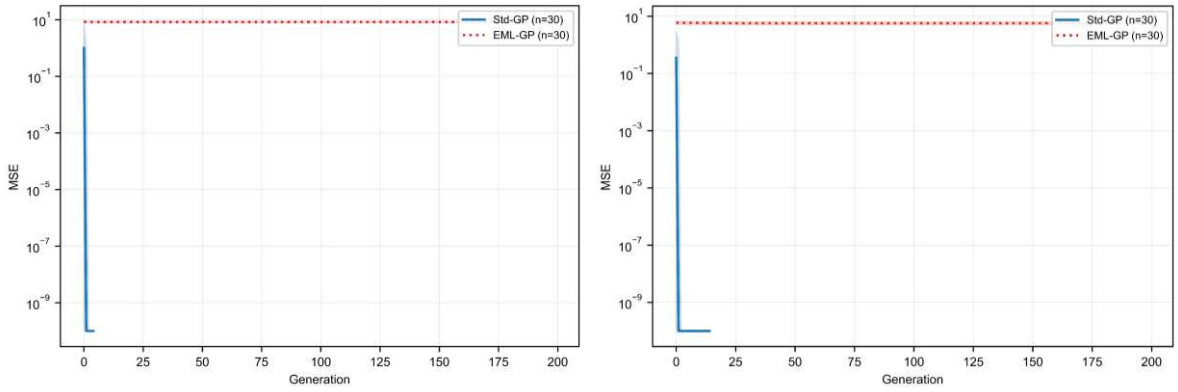
The EML performance, however, is exclusively limited to its native domain. Outside of its bounds the EML search space proves highly deceptive and brittle. Vanneschi and Tomassini [14] previously defined fitness landscapes in GP as "deceptive" when partial building blocks do not offer smooth gradients toward the global optimum. EML exhibits this pattern strongly.

Addition (Test 1.3): For $f(x1, x2) = x1 + x2$, Std-GP converged 100% of the time with 3 nodes. EML-GP exhibited a near 0% convergence rate, stalling at large error margins. Crucially, EML-GP repeatedly collapsed to a median tree size of exactly 1 node.

To synthesize algebraic addition EML requires a very specific 19-node tree structure. Fig. 1 demonstrates the required cascade to achieve simple addition (K=19). Evolving this specific branch results in random, unproductive fitness gradients right to the moment the exact configuration is hit. The GP routinely abandons the search space, defaulting to a single leaf node as a safer local optimum.

The collapse pattern is visible directly in the convergence trajectory of Fig. 5. On both Tests 1.3 (Addition) and 3.1 (sinh), EML-GP's median fitness remains pinned at fitness ≈ 10^1 for the entire 200-generation budget, a horizontal line indistinguishable across runs, while Std-GP plunges to near-zero error in a single generation. The flat trajectory is the visual signature of the topology-rigid collapse: no partial cascade offers an improving gradient, so selection abandons the search at generation 0.

The Regime B data reveals that crossover contributes to this collapse but is not its primary cause. Under mutation-only evolution, EML-GP's final median MSE on Addition improves modestly from ~8.3 to ~7.0 (1.2×) and on sinh more substantially from ~5.7 to ~2.9 (1.9×), with the sinh improvement consistent across all nine grid cells. However, neither target approaches convergence in either regime. Both remain at MSE levels orders of magnitude above Std-GP. The improvement confirms that crossover accelerates the abandonment of partial cascades at intermediate K-complexity (Section 5.1), but the dominant barrier is the deceptive landscape itself: no improving gradient exists regardless of whether subtree recombination is present.



(a) Test 1.3 - Addition (b) Test 3.1 - sinh

Fig. 5. Convergence trajectories for the two collapse cases (Generations: 200, Population: 100; n = 30 runs per algorithm). Panel (a): Test 1.3, $x_1 + x_2$ (Addition). Std-GP converges in a single generation. EML-GP's median trajectory remains perfectly flat at fitness $\approx 10^1$ for the full 200-generation budget, with a vanishingly narrow IQR. Every run collapses to the trivial single-node tree at generation 0 and remains there. The identical collapse pattern across both panels confirms that the failure mode is structural: no partial EML cascade offers an improving gradient on either target. Panel (b): Test 3.1, sinh(x). EML-GP again retreats to a single-node local optimum, producing a flat trajectory at fitness $\approx 10^1$ despite sinh being expressible through a compact exponential difference. Std-GP converges within the first few generations using its native sinh primitive.

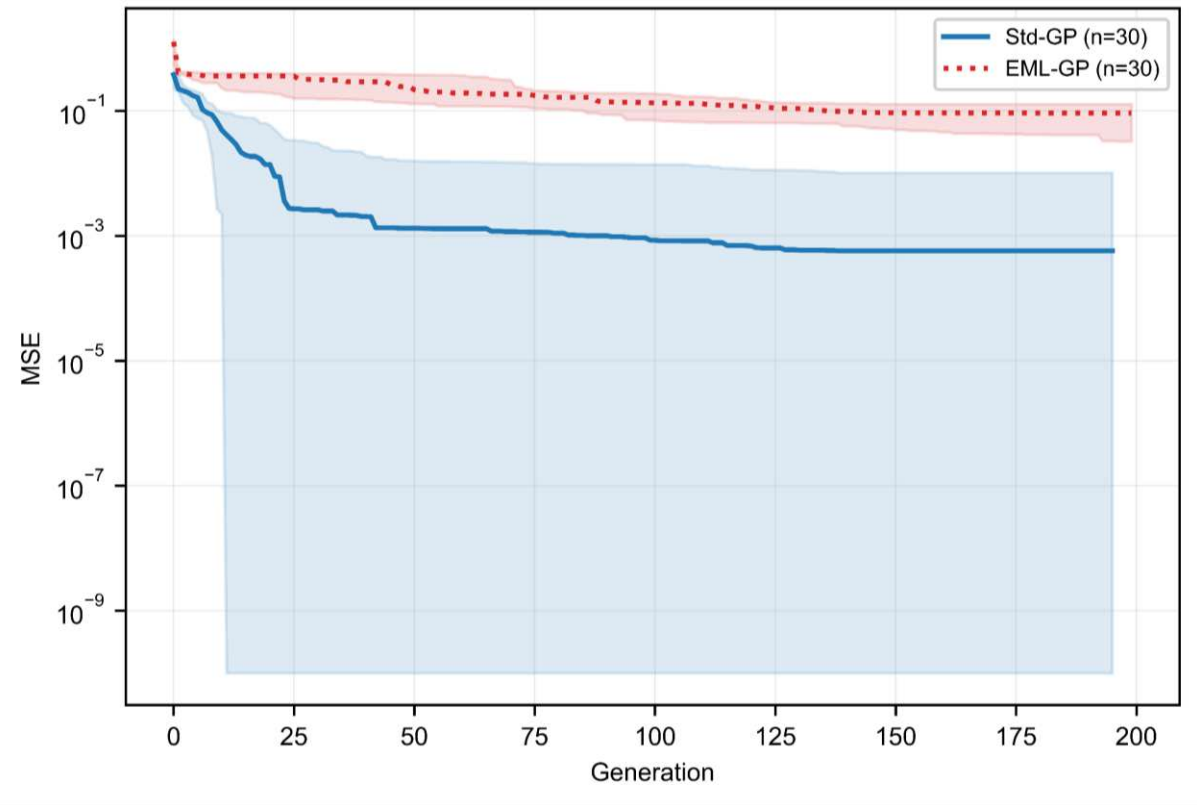


Fig. 6. Convergence trajectory for Test 1.4, *exp(x)·exp(−x²)*, the exp composite borderline case (Generations: 200, Population: 100; n = 30 runs per algorithm). Both algorithms make genuine progress. Std-GP descends to $\sim 10^{-2}$ within 25 generations and continues improving slowly. EML-GP descends from $\sim 10^{0}$ to $\sim 10^{-1}$ over the full budget, maintaining an order-of-magnitude gap with Std-GP throughout. The smooth descent of both trajectories distinguishes this case from the collapse panels: EML-GP is engaging with the search but cannot match Std-GP's compact native composition.

5.3 Parsimony Breakdown and Computational Timeouts (Phases 3 & 4)

Beyond the algebraic collapse documented in Section 5.2, EML-GP exhibits a second failure mode on the more difficult targets of Phases 3 and 4. This mode involves uncontrolled tree bloat that exhausts the compute budget

and causes systematic timeouts. Fig. 7 contrasts the median tree sizes of Std-GP and EML-GP across the test matrix, illustrating both this bloat and the remaining single-node collapses.

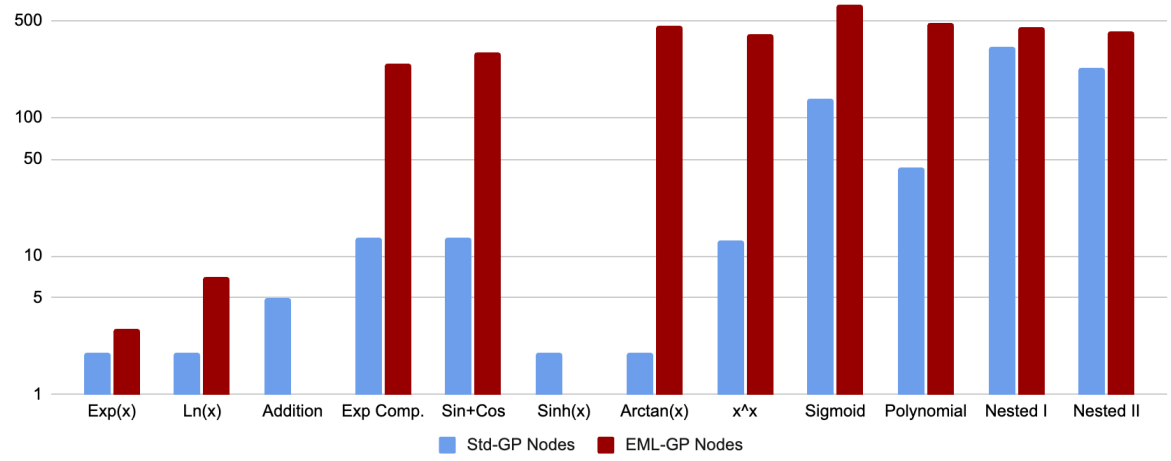


Fig. 7. Median tree size by target function for Std-GP and EML-GP at the (Generations: 200, Population: 100) grid cell. Std-GP converges to compact solutions of 2–3 nodes for native targets like sinh and arctan, where the relevant operators exist in F_{std}. Conversely, EML-GP produces median sizes of 400 to over 600 nodes on most non-native targets. However, for addition and sinh, it collapses to a median size of 1.

This bloat-without-convergence regime is evident in Figs. 8 and 9. On Tests 2.1 (sin+cos) and 3.2 (arctan), EML-GP's median best-fitness drifts slowly downward but never closes the order-of-magnitude gap with Std-GP. This stagnation occurs even as tree sizes balloon to several hundred nodes (Tables 7 and A.2).

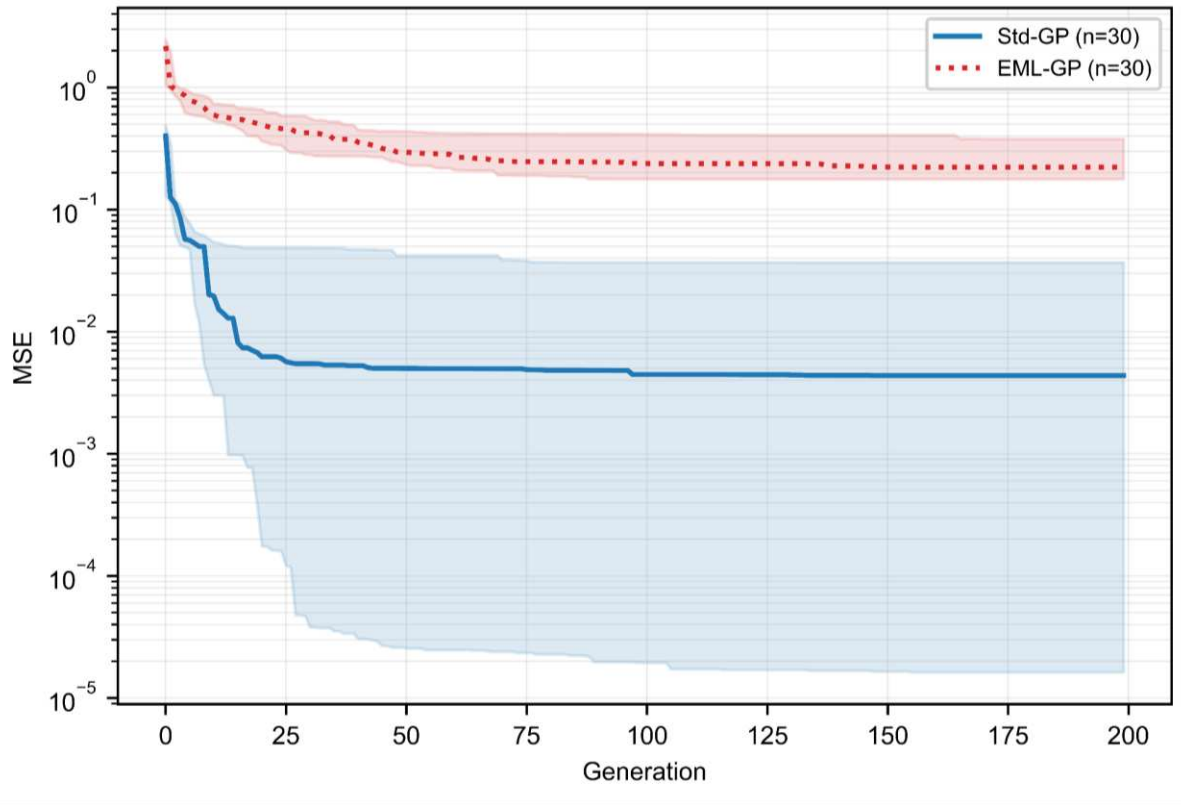


Fig. 8. Convergence trajectory for Test 2.1,$sin(x) + cos(x)$, Euler's-formula bridging (Generations: 200, Population: 100; n = 30 runs per algorithm). Std-GP descends rapidly to ~10^{-3} using its native Trig primitives. In contrast, EML-GP must construct these periodic functions through complex-plane cascades anchored on Euler's identity. As a result, EML-GP plateaus at ~$2 \cdot 10^{-1}$, which is two orders of magnitude worse than Std-GP. However, it shows a real downward trend rather than a total

collapse. This persistent gap illustrates the fragility of crossover-disrupted phase cancellations rather than complete search failure.

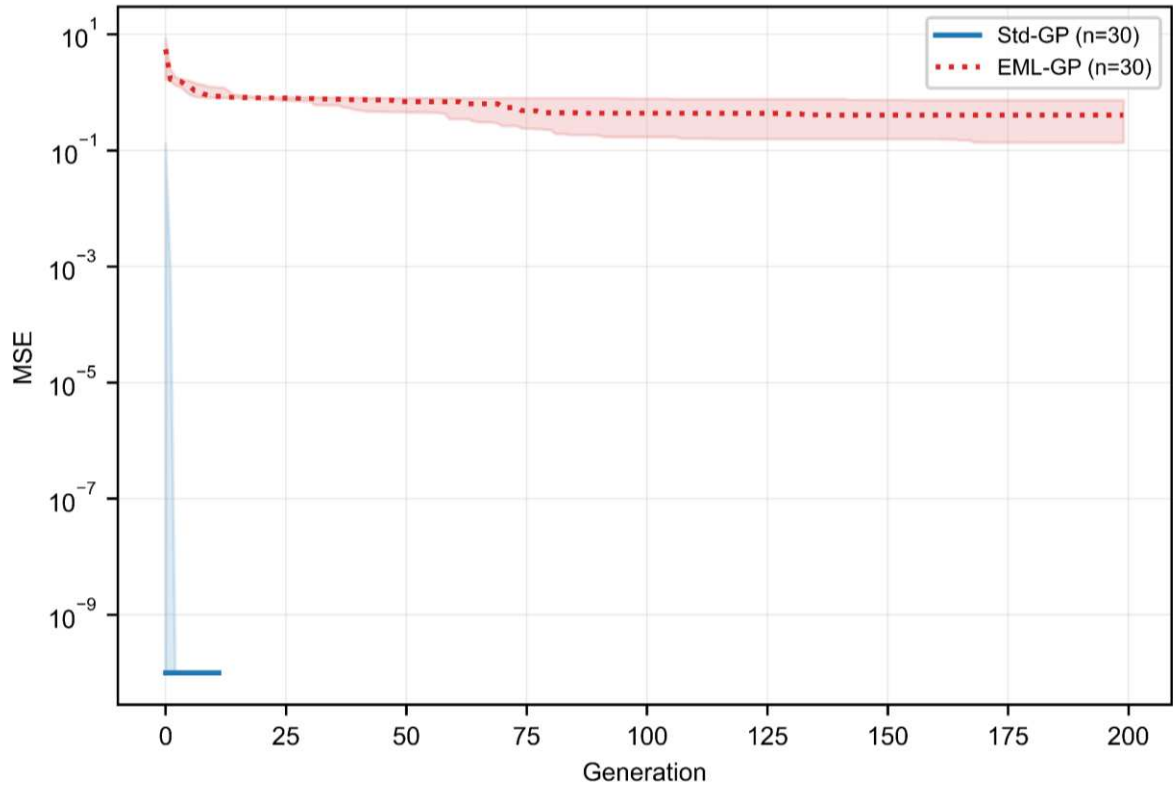


Fig. 9. Convergence trajectory for Test 3.2, $arctan(x)$ (Generations: 200, Population: 100; n = 30 runs per algorithm). Std-GP converges to near-zero error within the first few generations using its native $\arctan$ primitive. EML-GP descends steadily from ~10^{0} to ~$3 \cdot 10^{-1}$ over the full 200 generations. However, it never closes the order-of-magnitude gap. Meanwhile, its median tree size inflates to 461 nodes (Table 7). This trajectory illustrates the bloat-without-convergence failure mode. Unlike the collapse cases in Fig. 5, EML-GP actively searches and improves. Even so, the structural cost of emulating $arctan(x)$ through deep EML cascades prevents convergence within any practical budget.

5.3.1 Computational Frontiers

The computational feasibility frontier documented in Table 1 (Section 4.2) provides direct empirical evidence of this asymmetry. Std-GP completes every run within the 600-second CPU time limit across all grid cells and targets. Conversely, EML-GP exhausts the timeout on most non-native targets beyond the safe zone.

Two compounding factors produce this disparity:

- **Intrinsic Computational Expense:** IEEE-754 complex-domain arithmetic is inherently more expensive per node than real-domain arithmetic. Every evaluation propagates two floating-point components and traverses additional branch-handling paths through cmath for negative and imaginary logarithms.
- **Tree Size Explosion:** EML-GP evaluates trees that are typically one to two orders of magnitude larger than their Std-GP counterparts (Tables 7 and A.2).

The product of higher per-node cost and higher node count yields the observed timeout pattern. As noted in Section 4.2, this represents a conservative lower bound on the true computational asymmetry. If permitted to complete, timed-out runs would have produced even larger trees, further widening the gap against Std-GP.

5.3.2 Refutation of Budgetary Defenses

Considered together, the per-cell analyses refute the defense that EML-GP requires more computation. Across all nine grid cells, no parameter combination reverses or even attenuates EML-GP's disadvantage (Table 3). The six significant convergence-rate differences at population 100 remain significant at every higher budget. Two

additional functions reach significance at larger populations, but none lose it. Furthermore, when tree-size comparisons lose statistical resolution at larger budgets, the cause is EML-GP timeouts removing data from paired comparisons, not algorithmic improvement. The limitations of EML-GP are structural rather than budgetary.

5.3.3 General Crossover Disruption

One finding from the regime comparison extends beyond EML-GP. On Test 3.3 (x^x), Std-GP itself is substantially harmed by crossover. At the reference cell (Generations: 200, Population: 100), mutation-only Std-GP achieves a final median MSE of 0.004 compared to 0.067 with crossover, an 18-fold improvement. This effect is even stronger at (Generations: 100, Population: 100), where the ratio reaches 23-fold. This penalty persists across all cells at population ≤ 200 before disappearing at population 500, where both regimes converge.

A similar pattern appears on Test 2.1 ($sin + cos$) and Test 4.1 (Nested I). On these targets, mutation-only Std-GP outperforms the crossover regime by 2 to 3 times at small populations. These results demonstrate that crossover disruption on structurally complex targets is not an EML-specific pathology. Instead, it is a general phenomenon modulated by population size, a point developed further in Section 6.1.

5.3.4 Semantic Similarity-Based Crossover (SSC)

To test whether phase-space brittleness is an artifact of standard subtree crossover or an intrinsic property of the EML grammar, we evaluated Semantic Similarity-based Crossover (SSC) on six representative tests (Table 8). SSC does not improve EML-GP on any test.

- **Test 1.2 ($ln, K = 7$):** SSC prevents the exact convergence achieved by standard crossover. EML-GP+SSC stalls at 8.5×10^{-6}, matching the mutation-only regime. This indicates that the semantic similarity constraint suppresses the constructive recombinations needed to assemble the K=7 cascade.
- **Test 3.2 ($arctan$) and test 3.3 (x^x):** SSC nearly doubles EML-GP's final error, raising it from 0.406 to 0.799 and 1.741 to 2.477, respectively.
- **Test 2.1 ($sin + cos$):** SSC degrades both algorithms. Std-GP worsens from 0.004 to 0.024, and EML-GP degrades from 0.222 to 0.334.

Notably, SSC also harms Std-GP on Test 3.3 (x^x). It increases the median MSE from 0.067 to 0.177 (a 2.6-fold degradation). This matches the general crossover sensitivity on structurally complex targets documented in Section 5.3. These results confirm that phase-space brittleness stems from the search landscape of the EML grammar itself, rather than the specific crossover operator used.

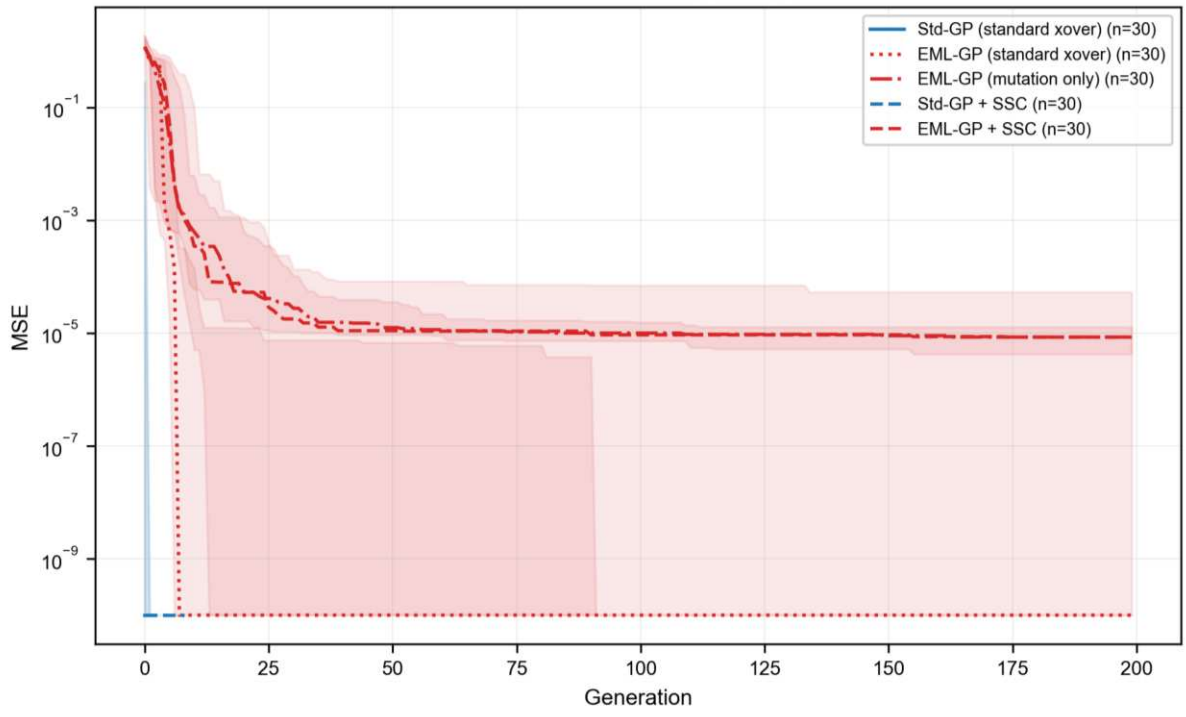


Fig. 11. Convergence trajectories for Test 1.2, ln(x), comparing five configurations at the reference cell (n = 30). EML-GP with standard crossover (dotted red) reaches exact convergence. EML-GP with SSC (dashed red) stalls at 8.5×10^{-6}, identical to mutation-only (dash-dot red), confirming that semantic crossover suppresses the constructive recombinations required to assemble the K=7 cascade.

Table 8. Effect of Semantic Similarity-based Crossover (SSC) on median final MSE at the reference cell (Generations: 200, Population: 100; n = 30). Std, crossover = standard subtree crossover (Regime A); Mut only = mutation only (Regime B).

ID	Test	Std-GP Std. crossover	Std-GP SSC	EML-GP Std. crossover	EML-GP SSC	EML-GP Mutation only	SSC effect on EML
1.1	**exp**	0.0	0.0	0.0	0.0	0.0	No change
1.2	**ln**	0.0	0.0	0.0	8.5×10^{-6}	8.5×10^{-6}	Worse
1.3	**Addition**	0.0	0.0	8.304	8.304	7.047	No change
2.1	**sin+cos**	0.004	0.024	0.222	0.334	0.288	Worse
3.2	**arctan**	0.0	0.0	0.406	0.799	0.688	Worse
3.3	**x^x**	0.067	0.177	1.741	2.477	1.998	Worse

The failure patterns documented in Phases 1–4 replicate on standard symbolic regression benchmarks evaluated at the reference cell under both regimes (Table 9). EML-GP achieves 0% convergence on all four benchmarks. On Nguyen-1 (x^3+x^2+x), EML-GP produces a median MSE of 0.015, an order of magnitude above Std-GP (5×10^{-4}), with no regime effect. Nguyen-7 ($\ln(x+1)+\ln(x^2+1)$) is particularly informative: the target is composed entirely of EML-native operations (ln at K=7 and addition at K=19), yet EML-GP cannot synthesise their combination, reaching a median MSE of 0.19 in Regime A 215× worse than Std-GP. Regime B improves EML-GP to 0.06 (3× better), consistent with the intermediate-K destructive crossover pattern. On Keijzer-6 and Vladislavleva-4, crossover helps EML-GP modestly, and the MSE gap narrows to 1.8–2× on the multivariate target, but convergence remains at 0% throughout.

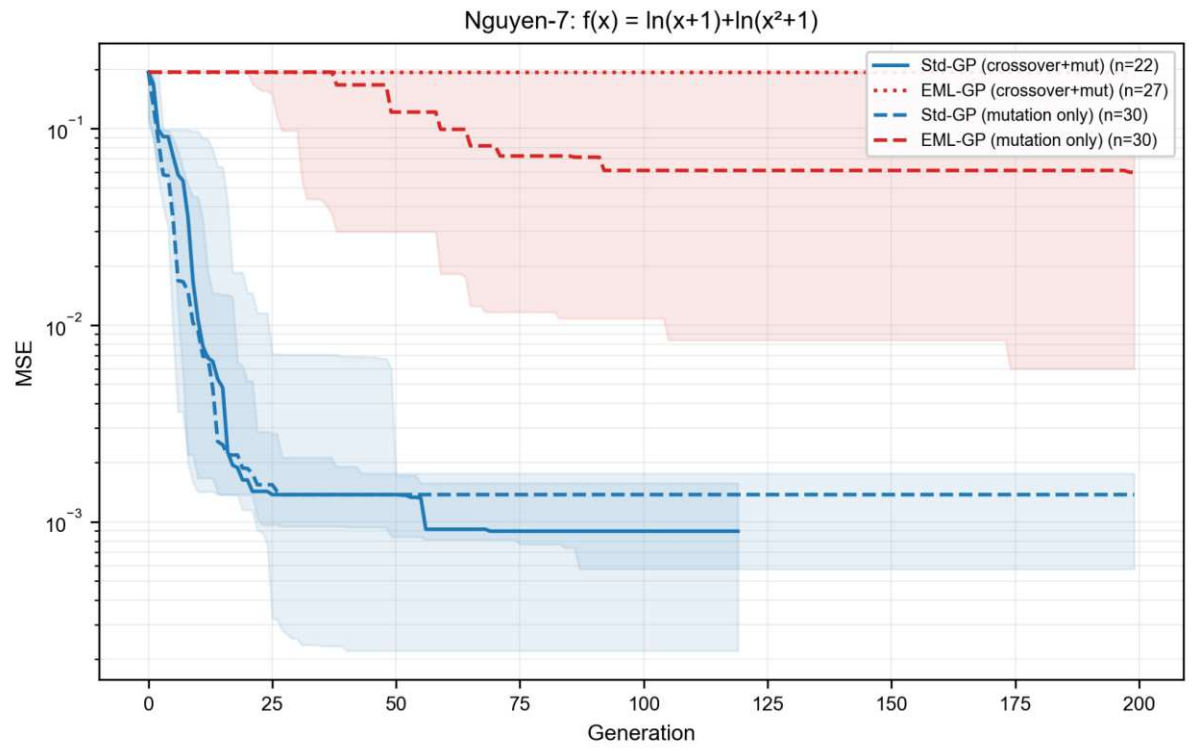


Fig. 12. Convergence trajectories for Nguyen-7, ln(x+1)+ln(x²+1), at the reference cell. Despite the target being composed entirely of EML-native operations (ln at K=7, addition at K=19), EML-GP cannot synthesise their combination. Regime B (dashed red) outperforms Regime A (dotted red) by 3×, consistent with the intermediate-K destructive crossover pattern.

Table 9. Standard benchmark results at the reference cell (Generations: 200, Population: 100; 30 seeded runs). Convergence rates and median final MSE under both regimes. n values below 30 indicate runs that exceeded the 600-second timeout.

Benchmark	Std-GP A conv	EML-GP A conv	Std-GP A MSE	EML-GP A MSE	EML-GP B MSE	Regime
Nguyen-1	0.41 (n=27)	0.00 (n=3)	4.9×10^{-4}	0.015	0.015	≈ Equivalent
Nguyen-7	0.00 (n=22)	0.00 (n=27)	9.0×10^{-4}	0.194	0.060	B better (3.2×)
Keijzer-6	0.00 (n=24)	0.00 (n=18)	1.1×10^{-3}	0.078	0.140	A better (1.8×)
Vladislavleva-4	0.00 (n=14)	0.00 (n=27)	0.039	0.069	0.153	A better (2.2×)

Table 10. EML-GP final median MSE under Regime A (crossover + mutation) and Regime B (mutation only) at the (Generations: 200, Population: 100) reference cell. Direction indicates which regime produced lower final error. Exact K values are known for exp (K=3), ln (K=7), addition (K=19), and sin+cos (K≤351) from Odrzywołek [9]. Remaining values are estimated from minimal EML tree depth and marked with ≤. The non-monotonic pattern is interpreted in Section 6.1, Table 11.

ID	Test	K	EML-GP Regime A	EML-GP Regime B	Direction
1.1	**exp(x)**	3	0.000	0.000	Equivalent
1.2	**ln(x)**	7	0.000	8.5×10^{-6}	A better
1.3	**Addition**	19	8.300	7.050	B better (1.2×)
1.4	**Exp Comp**	≤50	0.092	0.052	B better (1.8×)
2.1	**sin+cos**	351	0.222	0.288	A better (1.3×)
3.1	**sinh(x)**	≤ 20	5.700	2.920	B better (1.9×)
3.2	**arctan(x)**	≤ 300	0.406	0.688	A better (1.7×)
3.3	**x^x**	≤ 50	1.740	2.000	≈ Equivalent
3.4	**Sigmoid**	≤ 100	0.019	0.027	A better (1.4×)
3.5	**Polynomial**	≤ 50	333.500	326.800	≈ Equivalent
4.1	**Nested I**	1000+	0.436	0.482	≈ Equivalent
4.2	**Nested II**	1000+	0.361	0.366	≈ Equivalent

6. Discussion: Re-evaluating Grammatical Homogeneity

The fundamental hypothesis promoting the adoption of EML operator in GP is that the homogenous grammar of the operator set allows for the self selection of all the required operators naturally evolving from the universal operator. The empirical data however clearly shows that such expected advantages are negated by the building block disruption" caused by sub-tree crossover.

6.1 Phase-Space Brittleness

In standard GP swapping primitives have a destructive but localised effect. In EML-GP this destructive effect is significantly amplified. This is caused by the long specific cascades required to construct analogues of primitive mathematical functions out of the homogeneous EML operator. Random sub-tree crossover within these massive 350-node structures acts in a similar way as a cryptographic hash, where a small change alters the delicate imaginary phase cancellation, sending the output wildly across the complex plane. An important finding in this research is that the degree to which this effect acts upon the GP tree depends on the K-complexity. The regime comparison documented in Section 5 reveals that this mechanism is not uniform across K-complexity. At K=7

(ln, Test 1.2), crossover is constructive: it assembles the short cascade faster than mutation alone can discover it (Section 5.1). At K=19–100 (addition, sinh, exp composite), crossover becomes net-destructive, accelerating collapse or amplifying bloat. At K=300+ (arctan, sin+cos, nested composites), both operators fail and the regime effect is negligible. Crossover in EML is therefore not uniformly more destructive than in standard GP. Its effect is K-dependent, with the destructive regime concentrated at intermediate K-complexity where cascades are long enough to be fragile but short enough that partial states mislead the search. Table 11 summarises this relationship across the test matrix.

Definition 2 (Phase-space brittleness). A GP grammar Gexhibits phase-space brittleness on a target function$f$ if the following three conditions hold simultaneously:

(i) **Representability**:f is expressible in$G$, i.e., there exists $T \in L(G)$such that $[\![T]\!] = f$.

(ii) **Structural rigidity**: The minimal representation requires that a fraction $\alpha > \alpha_0$ of the tree's internal nodes occupy specific positions. Random perturbation of any single internal node produces a semantically catastrophic change (fitness degradation exceeding one order of magnitude).

(iii) **Gradient absence**: No proper subtree of the minimal representation T* produces a fitness value intermediate between the trivial solution (single leaf node) and T* itself. The fitness landscape between the two is flat or deceptive.

In this study, conditions (ii) and (iii) are assessed empirically: condition (ii) is evidenced by the bimodal tree-size distributions (Fig. 10), and condition (iii) by the flat convergence trajectories (Fig. 5). A grammar satisfying all three conditions on a target f will exhibit either single-node collapse or uncontrolled bloat, but not convergence.

Table 11. Crossover effect by K-complexity zone, derived from the regime comparison (Regime A vs Regime B) across the full test matrix. "A better" indicates crossover improves EML-GP performance; "B better" indicates mutation-only is superior.

K range	Representative tests	EML-GP: A vs B	Mechanism
K=3	exp(x)	Equivalent	Cascade trivially discoverable by either operator
K=7	ln(x)	A better	Short schema transmits reliably under crossover
K=19–100	Addition, sinh, exp comp, sigmoid	B better or equivalent	Crossover destroys intermediate cascade structures
K=300+	arctan, sin+cos, nested I/II	Equivalent or A slightly better	Both operators exhausted; landscape dominates

The collapse on tightly constrained short cascades, bloat on loosely defined long ones documented in Sections 5.2 and 5.3, follows directly from Definition 2. The discriminating factor is the topology of the EML emulation path rather than the difficulty of the target itself. Fig. 10 visualises the distributional contrast, confirming that the two outcomes are structurally distinct.

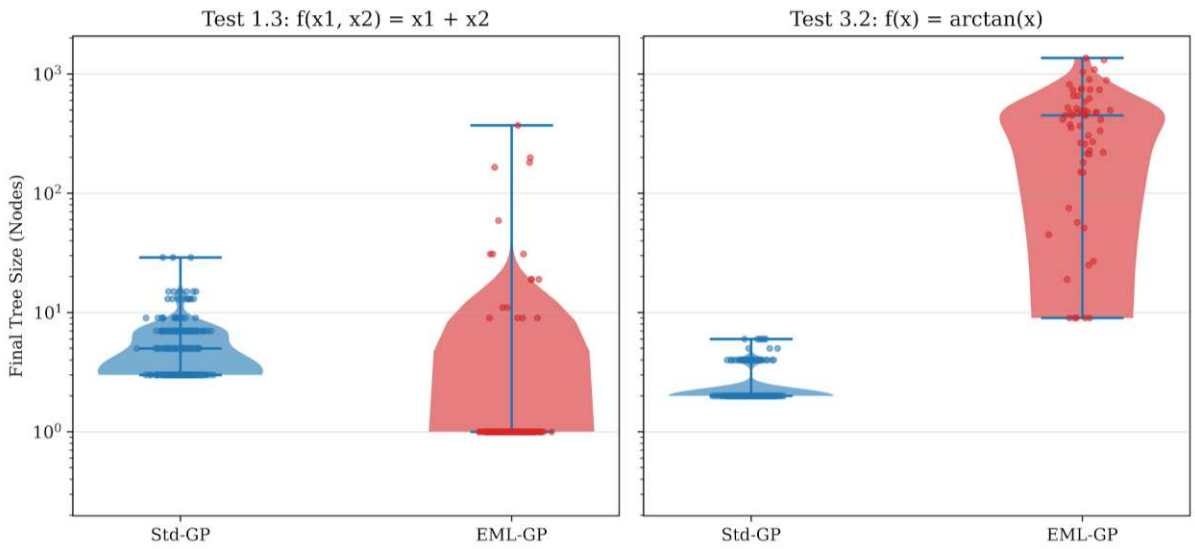


Fig. 10. Distribution of final tree sizes for Std-GP (blue) and EML-GP (red) on two representative targets illustrating the bimodal failure pattern (Generations: 200, Population: 100; n = 30 runs per algorithm). Left panel: Test 1.3 (Addition), where EML-GP collapses, the distribution is concentrated at exactly 1 node, with a sparse tail of runs that bloated before early stopping intervened. Right panel: Test 3.2 (arctan), where EML-GP bloats, the distribution spans 10 to 1,500 nodes with a median near 461. In both panels, Std-GP produces compact solutions (median 5 and 2 nodes respectively).

The negative effects of K-dependent crossover are not exclusive to EML-GP. As previously documented in Section 5.3, Std-GP suffers significantly in Test 3.3 (xx), where mutation-only evolution achieves 18–23× better final MSE at population 100. The effects are no longer noticeable at population 500. It can be reasoned that the broader initial diversity provides enough coverage to compensate for crossover's destructive recombination of partial sub-expressions. EML-GP in contrast is unaffected by the increase in population, this is because the main factor inhibiting EML-GP performance is not diversity but cascale length. This distinction identifies the specific structural property that separates EML-GP's failure from standard crossover disruption.

It is not the inherent nature of crossover that is destructive, but the EML's representational overhead that makes the search space non-modular. Essentially crossover transitions from constructive force that is combining and recombining functional elements into a recompositional force breaking apart existing long cascade functional blocks.The convergence trajectories in Section 5 suggest a deeper characterisation of this non-modularity. On bloat-prone targets, EML-GP's fitness improves gradually and asymptotically under both regimes, resembling iterative numerical fitting rather than the discrete structural discoveries characteristic of successful symbolic regression. The algorithm can partially approximate target functions but cannot compress these approximations into compact symbolic structures. EML-GP behaves, in effect, as a continuous numerical approximator rather than a symbolic constructor. A distinction with implications for operator design discussed in Section 6.3.

6.2 The Penalty of Complex Evaluation

Mathematically, EML relies on exact symbolic cancellations over the extended reals. Computationally, deep nestings of exp andln compound floating-point errors and produce numerical overflow $(complex(inf, inf))$ that eliminates viable genetic lineages.

6.3 Implications for Soft-Computing Operator Design

The findings generalise beyond GP to other soft-computing approaches that consider unified or universal primitives. Four design principles emerge.

First, representational universality is necessary but not sufficient: a primitive capable of encoding any target function may still be unsuitable for gradient-free search if discovering those foundational structures exceeds the reliability of selection mechanisms.

Second, homogeneous primitives shift the burden of expressivity from the operator set to structural composition. This exposes the search to disruption proportional to the depth of the required hierarchies. However, as shown in

the regime comparison (Table 11), this disruption is non-uniform, yielding a narrow K-complexity window where the EML grammar remains both reachable and evolvable. A similar pattern should be expected in any system where the choice of primitives necessitates deep cascades to express common target functions.

Third, the computational cost becomes a significant consideration when emulating an operator carries an order-of-magnitude cost in evaluation time (exp/ln). The practical search budget is exhausted well before the theoretical. This point has important implications in cross-field neural architecture search and learned-representation systems. The promise of simplicity through minimal primitive sets must be weighed against per-operation cost.

Fourth, crossover disruption on structurally complex targets is not unique to universal-operator grammars. The regime comparison reveals that standard Genetic Programming (Std-GP) suffers an 18–23× performance drop when using crossover on xx at small population sizes (Section 5.3). An effect that disappears at a population of 500. This demonstrates that the building block disruption problem documented by Angeline [26] and formalized by Poli and McPhee [35] remains practically significant for standard heterogeneous GP on targets requiring distinct structural sub-expressions. This finding is highly relevant to any system deploying GP with modest population budgets, which is typical in production symbolic regression. Furthermore, replacing standard subtree crossover with semantic similarity-based crossover (Table 10) fails to mitigate EML-GP's poor performance and actively worsens results on four of six tests. This confirms that the underlying issue stems from the grammar's search landscape topology rather than the specific recombination operator.

These limitations should be balanced against EML-GP's genuine strengths within its native domain. As Table 11 documents, EML-GP excels within its native K range from 3 to 7 and serves as an effective continuous approximator at high K. Its failure zone is concentrated entirely at intermediate complexity K between 19 and 100. This non-monotonic relationship implies that EML remains viable for hybrid grammars. The operator retains value at low and high complexities, provided complementary operators handle the intermediate algebraic domain where phase-space brittleness is most acute.As characterized in Section 6.1, EML-GP functions as a continuous numerical approximator rather than a symbolic constructor. This highlights a clear dissociation between representational completeness and search accessibility. This gap likely generalizes to any primitive set that achieves universality through deep composition. The hybrid approach suggested in Section 7 reflects this balance. Universality should be reserved for cases where the primitive's structural overhead is justified by access to genuinely novel function classes.

6.4 Limitations

Several methodological limitations require attention:

The scope of the study is restricted to a classical single tree-based GP with standard subtree crossover. Other niche frameworks such as strongly typed GP, grammatical evolution, gene expression programming are not in the scope of the research and may interact with EML's cascade structure in a different manner.

The crossover ablation tests only the binary presence or absence of subtree crossover. It does not include sweeping crossover rates, operator-specific strategies such as context-preserving crossover [27], or structurally protective mechanisms such as explicitly defined introns [33].

The SSC baseline (Table 8) addresses one semantic crossover variant. It does not include all semantic GP operators surveyed by Vanneschi et al. [17]. The standard benchmark validation (Table 9).

The standard benchmark validation (Table 9) covers four functions from three established suites evaluated at a single grid cell. A full-grid evaluation against comprehensive benchmark suites [4] remains for future work.

All experiments are conducted within a single Python implementation using IEEE-754 complex arithmetic. There has been no investigation into alternative computational platforms which theoretically may shift the computational feasibility frontier documented in Table 1 without altering the structural failure modes.

As noted in Section 7.1, this scope is deliberate for an early empirical assessment, and broader cross-framework evaluation is planned as follow-up work, therefore no comparison is made against modern symbolic regression systems. PySR [8], Operon, or Cartesian Genetic Programming [15] use fundamentally different representations and variation operators and may display different results.

7. Conclusion and Future Work

This research provides a comparative performance assessment of the universal mathematical operator EML tree-based Genetic Programming implementation. The findings are clear: while the EML operator offers exact analogues to most mathematical operators (with claim for complete universality), the pure EML-GP realisation is inferior to classic heterogeneous GP for practical symbolic regression. A more fundamental finding is that the results demonstrate that representational universality and evolutionary accessibility are orthogonal properties. A grammar that can express any continuous function doesn't become searchable by gradient-free methods, by default. Additional observation is that the gap between expressibility and discoverability grows with the structural depth required to encode common targets.

By design, emulating standard algebraic and trigonometric operators requires high K-complexity structure. This produces deceptive search landscapes, prohibitive computational overhead, and K-dependent sensitivity to crossover in the complex plane.Emulating standard algebraic and trigonometric operations requires structures with high K-complexity. This produces deceptive search landscapes, prohibitive computational overhead, and K-dependent sensitivity to crossover in the complex plane. The regime comparison tests reveal that constructive crossover effects have a non-monolithic dependency to K-complexity. At low complexity of K=7 cross over has a constructive effect; at intermediate K between 19 and 100 the effects are highly destructive; at K above 300 the effects are approximately neutral.

7.1 Future Work

The comparative analysis in this paper is limited to standard tree-based GP implementation with crossover introduced in specific regimes. The adherence to classic architecture is a deliberate decision. As EML was proposed only in March 2006 the goal of the present study is to provide an early empirical assessment of the viability of EML in the domain of GP and establish the baseline performance of the operator as a foundation for a more detailed future research utilising state-of-the art symbolic regression frameworks such as PySR, Operon, grammatical evolution, and semantic GP variants. The scope of the study focussed on twelve tests testing for representative competencies rather than establishing a comprehensive benchmark suite.

It is an interesting direction for research to test whether the the phase-space brittleness phenomenon produced by the underlying mechanism (high-K-complexity emulation cascades disrupted by random recombination) can also be observed in other evolutionary symbolic-regression frameworks such as strongly typed GP, grammatical evolution, gene expression programming, or modern symbolic regression methods (PySR).

Finally, the K-complexity-scaled patience mechanism was designed to give EML-GP a fair allocation relative to its representational cost. However, this mechanism does not exhaust the space of possible compensatory measures. Tailored mutation operators, structural priors, or curriculum learning over progressively harder targets are all plausible mitigations not tested here. The sharp boundary between convergence at K=7 and categorical failure at K=19 suggests a phase-transition-like threshold in evolvability as a function of K-complexity. Formalising this relationship through a larger sample of K values is a natural direction for future work.

The theoretical elegance of the EML operator should be further examined in contexts where its limitations can be minimised or mitigated. We suggest three primary avenues for future work:

1. **Cartesian Genetic Programming (CGP) [15]:** Graph-based representations would allow EML algorithms to calculate sub-expressions once and route them to multiple nodes. This can directly address the exponential tree-bloat problem.
2. **Gradient-Based Tuning:** Embedding EML nodes inside a neural architecture (similar to Kolmogorov-Arnold Networks [16]) where topologies are fixed but connection weights are updated via gradient descent could bypass the destructiveness of discrete genetic crossover.
3. **Hybrid Symbolics:** Utilising EML not as a replacement for algebra, but as a replacement for standard non-linear functions. A grammar consisting of {+, ×, eml} would retain the non-deceptive linear gradients of standard GP while allowing EML to synthesise novel periodic or hyperbolic functions natively. The ln(x) finding from Section 5.1 provides specific motivation for this direction: EML is most effective within its native K=3–7 domain, and within that domain crossover remains constructive, suggesting that hybrid grammars should restrict EML's role to short non-linear synthesis while retaining standard operators for algebraic composition. Combining this with semantic-aware variation operators [17, 18] could further mitigate the disruption phenomenon documented in Section 6.1.

CRediT authorship contribution statement

Dimitar Georgiev Kazakov: Conceptualisation, Methodology, Software, Validation, Formal analysis, Investigation, Data curation, Writing - original draft, Writing - review & editing, Visualisation.

Declaration of competing interests

The author declares that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Funding

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Data and Code Availability

The Python implementation of the EML-GP and Std-GP environments, the full evaluation matrix, and the raw experimental grid-search data utilized in this research are open-source and publicly accessible via GitHub at: https://github.com/swogger/EML-GP and https://doi.org/10.5281/zenodo.20366631

Supplementary Material

Supplementary Material S1 contains per-cell statistical comparison tables for all nine grid cells, Regime B convergence rate tables, and full convergence trajectories across both regimes. All data and code are available at the GitHub repository https://github.com/swogger/EML-GP and https://doi.org/10.5281/zenodo.20366631.

References

1. J. R. Koza, Genetic Programming: On the Programming of Computers by Means of Natural Selection. Cambridge, MA, USA: MIT Press, 1992.
2. M. Schmidt and H. Lipson, "Distilling free-form natural laws from experimental data," Science, vol. 324, no. 5923, pp. 81–85, 2009.
3. E. J. Vladislavleva, G. F. Smits, and D. Den Hertog, "Order of nonlinearity as a complexity measure for models generated by symbolic regression via pareto genetic programming," IEEE Transactions on Evolutionary Computation, vol. 13, no. 2, pp. 333–349, 2009.
4. W. La Cava et al., "Contemporary symbolic regression methods and their relative performance," in Proc. Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track, 2021.
5. S. Silva and E. Costa, "Dynamic limits for bloat control in genetic programming and a review of past and present bloat theories," Genetic Programming and Evolvable Machines, vol. 10, no. 2, pp. 141–179, 2009.
6. W. B. Langdon and R. Poli, "Fitness causes bloat: Mutation," in Soft Computing in Engineering Design and Manufacturing, London, UK: Springer, 1998, pp. 13–22.
7. T. Bäck, Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms. Oxford, UK: Oxford University Press, 1996.
8. M. Cranmer, "Interpretable Machine Learning for Science with PySR and SymbolicRegression.jl," arXiv preprint arXiv:2305.01582, 2023.
9. A. Odrzywołek, "All elementary functions from a single operator," arXiv preprint arXiv:2603.21852, 2026.
10. R. Poli, W. B. Langdon, N. F. McPhee, and J. R. Koza, A Field Guide to Genetic Programming. Lulu.com, 2008.
11. W. Banzhaf, P. Nordin, R. E. Keller, and F. D. Francone, Genetic Programming: An Introduction. San Francisco, CA, USA: Morgan Kaufmann, 1998.
12. U.-M. O'Reilly and F. Oppacher, "The troubling aspects of a building block hypothesis for genetic programming," in Foundations of Genetic Algorithms 3, L. D. Whitley and M. D. Vose, Eds. San Mateo, CA, USA: Morgan Kaufmann, 1995, pp. 73–88.
13. S. Luke, "Two fast tree-creation algorithms for genetic programming," IEEE Transactions on Evolutionary Computation, vol. 4, no. 3, pp. 274–283, 2000.
14. L. Vanneschi and M. Tomassini, "Introduction to fitness landscapes and difficulty in genetic programming," in Genetic Programming Theory and Practice II, U. M. O'Reilly et al., Eds. New York, NY, USA: Springer, 2005, pp. 241–258.
15. J. F. Miller, Cartesian Genetic Programming. Berlin, Germany: Springer, 2011.
16. Z. Liu et al., "KAN: Kolmogorov-Arnold Networks," arXiv preprint arXiv:2404.19756, 2024.
17. L. Vanneschi, M. Castelli, and S. Silva, "A survey of semantic methods in genetic programming," *Genetic Programming and Evolvable Machines*, vol. 15, no. 2, pp. 195–214, 2014.
18. A. Moraglio, K. Krawiec, and C. G. Johnson, "Geometric semantic genetic programming," in *Parallel Problem Solving from Nature – PPSN XII*, C. A. C. Coello et al., Eds. Berlin, Germany: Springer, 2012, pp. 21–31.

19. M. Virgolin, T. Alderliesten, C. Witteveen, and P. A. N. Bosman, "Improving model-based genetic programming for symbolic regression of small expressions," *Evolutionary Computation*, vol. 29, no. 2, pp. 211–237, 2021.
20. R. Poli and N. F. McPhee, "Parsimony pressure made easy: Solving the problem of bloat in GP," in *Theory and Principled Methods for the Design of Metaheuristics*, Y. Borenstein and A. Moraglio, Eds. Berlin, Germany: Springer, 2014, pp. 181–204.
21. D. Hodan, V. Mrazek, and L. Sekanina, "Semantically-oriented mutation operator in Cartesian genetic programming for evolutionary circuit design," *Genetic Programming and Evolvable Machines*, vol. 22, no. 4, pp. 539–572, 2021.
22. S. Luke and L. Panait, "A comparison of bloat control methods for genetic programming," *Evolutionary Computation*, vol. 14, no. 3, pp. 309–344, 2006.
23. T. Soule and J. A. Foster, "Effects of code growth and parsimony pressure on populations in genetic programming," *Evolutionary Computation*, vol. 6, no. 4, pp. 293–309, 1998.
24. S. Silva, S. Dignum, and L. Vanneschi, "Operator equalisation for bloat free genetic programming and a survey of bloat control methods," *Genetic Programming and Evolvable Machines*, vol. 13, no. 2, pp. 197–238, 2012.
25. W. B. Langdon and R. Poli, *Foundations of Genetic Programming*. Berlin, Germany: Springer, 2002.
26. P. J. Angeline, "Subtree crossover: Building block engine or macromutation?" in *Genetic Programming 1997: Proceedings of the Second Annual Conference*, J. R. Koza et al., Eds. San Francisco, CA, USA: Morgan Kaufmann, 1997, pp. 9–17.
27. P. D'haeseleer, "Context preserving crossover in genetic programming," in *Proceedings of the 1994 IEEE World Congress on Computational Intelligence*, 1994, pp. 256–261.
28. N. Q. Uy, N. X. Hoai, M. O'Neill, R. I. McKay, and E. Galván-López, "Semantically-based crossover in genetic programming: Application to real-valued symbolic regression," *Genetic Programming and Evolvable Machines*, vol. 12, no. 2, pp. 91–119, 2011.
29. L. Beadle and C. G. Johnson, "Semantically driven crossover in genetic programming," in *Proceedings of the IEEE World Congress on Computational Intelligence*, 2008, pp. 111–116.
30. K. Krawiec and T. Pawlak, "Locally geometric semantic crossover: A study on the roles of semantics and homology in recombination operators," *Genetic Programming and Evolvable Machines*, vol. 14, no. 1, pp. 31–63, 2013.
31. E. Galván-López, J. McDermott, M. O'Neill, and A. Brabazon, "Defining locality as a problem difficulty measure in genetic programming," *Genetic Programming and Evolvable Machines*, vol. 12, no. 4, pp. 365–401, 2011.
32. J. M. Daida, A. M. Hilss, D. J. Ward, and S. L. Long, "What makes a problem GP-hard? Analysis of a tunably difficult problem in genetic programming," *Genetic Programming and Evolvable Machines*, vol. 4, no. 2, pp. 165–191, 2003.
33. P. Nordin, F. Francone, and W. Banzhaf, "Explicitly defined introns and destructive crossover in genetic programming," in *Advances in Genetic Programming*, vol. 2, P. J. Angeline and K. E. Kinnear, Eds. Cambridge, MA, USA: MIT Press, 1996, pp. 111–134.
34. T. Soule and J. A. Foster, "Removal bias: A new cause of code growth in tree based evolutionary programming," in *Proceedings of the 1998 IEEE International Conference on Evolutionary Computation*, Anchorage, AK, USA, 1998, pp. 781–786.
35. R. Poli and N. F. McPhee, "General schema theory for genetic programming with subtree-swapping crossover: Part I," *Evolutionary Computation*, vol. 11, no. 1, pp. 53–66, 2003.

Appendix A: Full Grid Search Results for Median Tree Size

Table A.1. Median tree size produced by Std-GP at convergence (or terminal generation) across the grid. Compact solutions dominate for native targets (Exp, Ln, Sinh, Arctan: median = 2 nodes). Larger budgets generally reduce median size on harder targets through stronger selection pressure on parsimony.

	Generations: 100			**Generations: 200**			**Generations: 500**		
Test Case	**Pop: 100**	**Pop: 200**	**Pop: 500**	**Pop: 100**	**Pop: 200**	**Pop: 500**	**Pop: 100**	**Pop: 200**	**Pop: 500**
Exp(x)	2	2	2	2	2	2	2	2	2
Ln(x)	2	2	2	2	2	2	2	2	2
Addition	5	3	3	5	3	3	5	3	3
Exp Comp.	35	19	14	14	14	14	14	14	14
Sin+Cos	175	26	7	14	10	7	10	10	7
Sinh(x)	2	2	2	2	2	2	2	2	2
Arctan(x)	2	2	2	2	2	2	2	2	2
x^x	59	135	9	13	12	9	13	12	9

Sigmoid	286	391	20	136	33	20	108	38	20
Polynomial	43	33	24	43	33	24	43	33	24
Nested I	472	529	78	321	309	91	84	309	91
Nested II	328	262	5	229	23	5	111	23	5

Table A.2. Median tree size produced by EML-GP at convergence or termination. Bloat dominates whenever EML must emulate non-native operators (median 200–600+ nodes). Collapse to a single node (median = 1) occurs systematically for Addition and Sinh(x), reflecting abandonment of the deceptive search landscape. Dashes ("-") indicate runs that exceeded the 600-second CPU time timeout before producing a converged solution.

	Generations: 100			**Generations: 200**			**Generations: 500**		
Test Case	**Pop: 100**	**Pop: 200**	**Pop: 500**	**Pop: 100**	**Pop: 200**	**Pop: 500**	**Pop: 100**	**Pop: 200**	**Pop: 500**
Exp(x)	3	3	3	3	3	3	3	3	3
Ln(x)	7	7	7	7	7	7	7	7	7
Addition	1	1	1	1	1	1	1	1	1
Exp Comp.	207	299	419	243	47	-	-	-	-
Sin+Cos	386	459	-	295	-	-	-	-	-
Sinh(x)	1	1	1	1	1	1	1	1	1
Arctan(x)	415	476	305	461	-	-	-	-	-
x^x	177	271	199	399	-	-	-	-	-
Sigmoid	369	425	-	648	-	-	-	-	-
Polynomial	305	303	-	485	-	-	-	-	-
Nested I	365	382	299	443	133	-	-	-	-
Nested II	260	322	-	419	-	-	-	-	-

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- SupplementaryMaterialS1PerCellStatisticalAnalyses.docx