A Grammatical and Semantic Genetic Programming Technique for Symbolic Regression

João Marcos de Freitas[1], **Heder Soares Bernardino**[1], **Itamar Leite de Oliveira**[1]

[1]Universidade Federal de Juiz de Fora (UFJF)
Juiz de Fora – MG – Brazil

{joao.marcos, heder.bernardino, itamar.oliveira}@ufjf.br

Abstract. *Symbolic Regression (SR) aims to automatically discover the underlying structure of mathematical models from data, a task frequently addressed using Genetic Programming (GP). While Grammatical Genetic Programming (GGP) successfully introduces necessary syntactic constraints to these models, its standard crossover operators often suffer from search inefficiency by strictly adhering only to derivation rules. Conversely, Semantic Genetic Programming (SGP) leverages semantic evaluations to effectively guide the evolutionary search. To bridge this gap, this paper proposes Grammatical and Semantic Genetic Programming (GSGP), an approach that integrates context-free grammars with semantic evaluations to guarantee both syntactic validity and behavioral relevance. A key contribution of this work is the Roulette Semantic Crossover (RSC), a recombination operator that applies proportional roulette wheel selection based on the inverse semantic distance between candidate subtrees. On a suite of ten classical symbolic-regression functions, RSC attained the largest area under the Performance Profile curves and the highest overall success count among the operators compared, indicating robustness on this set rather than uniform dominance of the final error.*

1. Introduction

The accurate modeling of complex relationships is essential to describe and predict phenomena across various scientific domains, providing tools to represent abstract processes in a simplified manner and enabling predictions under different conditions. However, classical regression methodologies require the prior definition of a rigid functional structure (linear, polynomial, exponential), severely limiting the exploration of hidden patterns [Cheng et al. 2019, Huang and He 2023]. Symbolic Regression (SR) breaks this paradigm by heuristically searching for both the model's structure and its corresponding numerical parameters simultaneously, ensuring high flexibility and interpretability [Schmidt and Lipson 2009].

Genetic Programming (GP) has consolidated itself as the dominant technique in SR, evolving populations of mathematical expressions represented as trees through Darwinian mechanisms [Koza 1992]. A highly successful variant is Grammatical Genetic Programming (GGP), which uses Context-Free Grammars (CFG) to govern model generation. This imposes restrictions that ensure the syntactic validity of the individuals and introduces a directional bias based on prior domain knowledge [de Freitas et al. 2015].

Despite this notable success, GP-based algorithms face a critical operational bottleneck: the recombination (crossover) operation is essentially "blind" regarding the

mathematical behavior of the solutions. Studies show that up to 75% of genetic exchanges generate offspring identical to their parents (semantic clones) or produce destructive variations, consuming processing power without actively exploring new regions of the search space [McPhee et al. 2008].

To mitigate this problem, the integration of semantics (the model's output behavior) has been applied to genetic operators to control search locality and increase diversity [Uy et al. 2010]. However, existing semantic operators frequently depend on strict cutoff parameters, imposing binary discards that can stagnate evolution [Nguyen et al. 2009]. This work proposes the *Roulette Semantic Crossover* (RSC), a continuous and probabilistic operator that inverts the utility of semantics: instead of using it as a punitive filter, it transforms it into a dynamic and stochastic selection criterion.

The main contributions of this paper are: (i) the proposal of the probabilistic RSC operator, which introduces flexibility to semantic exploration; (ii) the formal adaptation of literature operators (SAC and SSC) to the restrictive GGP environment; (iii) the comparative empirical analysis of using Absolute Distance versus Cosine Similarity; and (iv) an empirical comparison, under a shared grammar and budget, that separates profile robustness from final accuracy.

2. Theoretical Background

2.1. Symbolic Regression

While classical regression fits numerical coefficients inside a predefined functional form (for instance, linear, polynomial, and exponential) [Huang and He 2023], Symbolic Regression (SR) searches jointly for the expression's structure and its parameters [Orzechowski et al. 2018, Schmidt and Lipson 2009]. The search involves operator sets, variable interactions, constant synthesis, and the cost of evaluating candidates on data, which must be balanced with the growth of the equations [Bongard and Lipson 2007]. Domain knowledge can further restrict when a grammar is available, by limiting which topologies are valid and biasing the search toward interpretable models rather than arbitrary algebraic strings.

2.2. Genetic Programming

Formalized by Koza [Koza 1992], GP is an evolutionary branch focused on computational synthesis. Using tree-based genotypic structures instead of fixed chromosomes, the leaves (terminals) contain variables (x_1, x_2) and numerals, while the internal nodes act as logical or arithmetic operators ($+, -, \times, \div$).

Once the representation of the candidate solutions is defined, GP operates similarly to other evolutionary algorithms. First, an initial population is generated; it is common to adopt specific initialization methods (such as ramped half-and-half) to ensure structural diversity and variability. When solving symbolic regression problems, the search is guided by a fitness function based on the model's error against a dataset. Until a stopping criterion is met, the evolutionary search is driven by genetic operators, recombination (subtree crossover) and mutation (random subtree replacement), to generate new candidate solutions, followed by population replacement. Also, it is common practice to limit candidate trees to a maximum depth to control bloat, and to apply elitism by preserving the best candidate solutions from the previous generation into the next.

2.3. Semantic Genetic Programming

Semantic Genetic Programming (SGP) was developed to bridge the gap between syntactic representation and behavioral performance. The syntax of a tree is the expression embedded within it, while its semantics is the resulting vector of its outputs over the training data. Distinct trees can have different syntax and yet be semantic clones if they share the same results (e.g. $x \times x$ and x^2). Even though a branch of a tree in GP has its own semantic, still up to 75% of successful GP crossovers yield clones or negligible change [McPhee et al. 2008]. Semantics are exploited in SGP trying to improve the exploration of the search space avoiding abrupt changes and the propagation of clones. Given the semantic vectors P and Q, two metrics can be adopted to quantify the Semantic Distance (SD) between them:

1. **Absolute Distance (ABS):** Computes the average point-wise Euclidean deviation ($SD_{ABS} = \frac{\sum |p_i - q_i|}{N}$) [Uy et al. 2010].
2. **Cosine Distance (COS):** Measures the dimensional angle ($SD_{COS} = 1 - \frac{P \cdot Q}{||P|| ||Q||}$), showing whether surfaces point to the same gradient orientations, which is fundamental in information retrieval [Juvekar and Purwar 2024, Mukherjee and Sonal 2023].

2.4. Grammatical Genetic Programming

Unrestricted GP trees can generate pathological algebraic expressions. Grammatical Genetic Programming (GGP) avoids such situations by representing each individual as a derivation tree of a context-free grammar $G = (N, \Sigma, S, P)$ [McKay et al. 2010]. Derivation starts at S (typically $\langle \text{expr} \rangle$); each internal node is expanded by a production rule until only terminals remain (Figure 1). Genetic operators apply exclusively to non-terminal nodes. For crossover between parents A and B, a non-terminal nt is selected in A, and the subtree rooted at nt is swapped with a subtree in B whose root shares the same non-terminal type (subject to a maximum depth constraint).

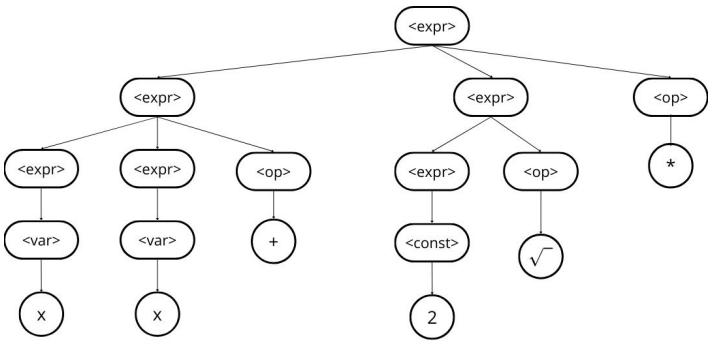


Figure 1. Generation of an individual (model) using successive syntactic derivations from a CFG.

3. Related Work in Semantic Crossover

In an attempt to abolish sterile recombinations, the literature has developed rigorous heuristics. The *Semantic Aware Crossover* (SAC) imposes a posterior veto: if the newly crossed offspring have identical semantics (zero SD) to the parents, the operation is annulled, and the processing is redone [Nguyen et al. 2009].

Improving this mechanics, the *Semantic Sensitive Crossover* (SSC) acts preventively, stipulating that the distance between the subtrees must fall strictly within a restricted zone (lower limit α and upper limit β) for the swap to be authorized [Uy et al. 2010]. Meanwhile, within the scope of formal derivations, the *Semantic Crossover for Program Synthesis* (SCPS) [Forstenlechner 2019] maps all subtrees and selects the one that is as similar as possible among those showing at least some semantic fluctuation.

However, the weakness of these constructs is their massive dependence on parameterization and abrupt thresholds (Max_Tries). The lack of fit neighbors in rigidly demarcated zones leads algorithms to stall or forces them to revert indiscriminately to conventional GP [Uy et al. 2010].

4. The Proposed Strategy: Roulette Semantic Crossover

Aiming to provide semantic plasticity without sacrificing random escapes from local minima, we propose the Roulette Semantic Crossover (RSC). The RSC replaces rigid acceptance cut-offs with a proportional environment.

When evaluating a target p in the first individual, the operator maps all syntactically compatible subtrees (q_i) in the second parent are mapped. Instead of preemptively discarding candidates based on hard distances, a roulette wheel is constructed where the probability (p_i) of selection is inversely proportional to the Semantic Distance $SD(p, q_i)$. The procedural details are shown in Algorithm 1.

Algorithm 1: Roulette Semantic Crossover (RSC)

1. Select a random subtree p in parent P;
2. Map candidate subtrees q_i in parent Q with the same Non-Terminal as the root of p;
3. Calculate the global semantic distance $SD(p, q_i)$ for the candidates;
4. Discard identical subtrees ($SD(p, q_i) == 0$);
5. Create the roulette wheel by setting probabilities $p_i = \frac{1}{SD(p,q_i)}$;
6. Normalize the roulette weights ($p_i = \frac{p_i}{\sum p_i}$);
7. Stochastically draw $s \in [0, 1]$;
8. Select the corresponding subtree and perform the recombination;

This architecture eliminates clone generation at the root (zero probability), massively rewards short-range refinement, and reserves a subtle (yet vital) probability for coupling completely distinct chains, providing the structural macro-mutations needed to overcome convergence plateaus.

4.1. Detailed Representation and Stochastic Calculation of RSC

Figure 2 and the subsequent tables illustrate the RSC orchestration. Starting from the dotted subtree $\langle expr \rangle$ in individual A, the operator isolates the four homologous candidates in individual B ($\langle expr \rangle^1$ to $\langle expr \rangle^4$).

Table 1 simulates the vector generated by evaluating these subtrees against the training data (x and y). Immediately, the deviation calculation in Table 2 quantifies their Semantic Distance (SD) relative to the target.

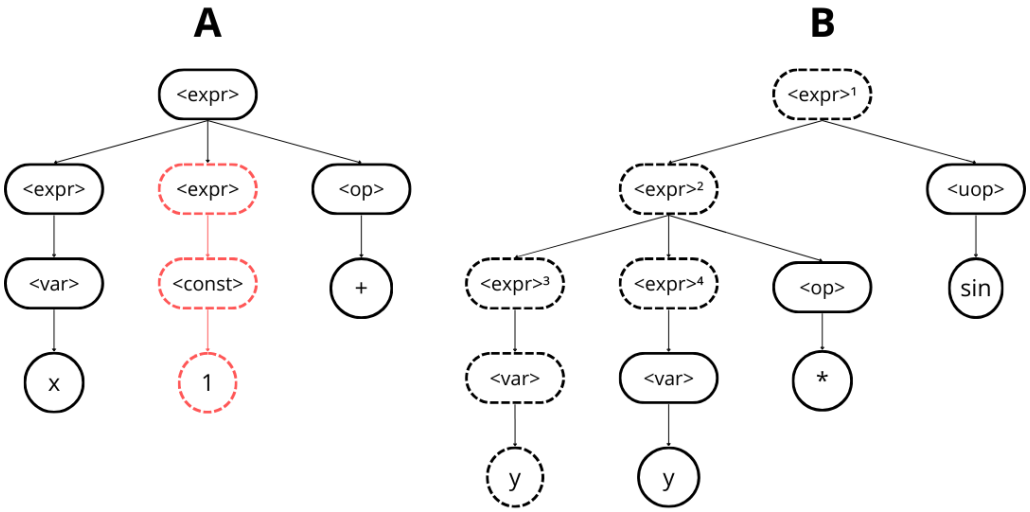


Figure 2. Example of the RSC selection matrix and mapping in derivation trees.

Table 1. Phenotypic semantic outputs extracted from the example subtrees.

x	= [	0.0,	1.0,	2.0,	3.0,	4.0,	5.0,	6.0,	7.0,	8.0,	9.0]
y	= [	0.0,	0.5,	1.0,	1.5,	2.0,	2.5,	3.0,	3.5,	4.0,	4.5]
$\langle expr \rangle$	= [	1.0,	1.0,	1.0,	1.0,	1.0,	1.0,	1.0,	1.0,	1.0,	1.0]
$\langle expr \rangle^1$	= [	0.0,	0.2,	0.8,	0.8,	-0.8,	0.0,	0.4,	-0.3,	-0.3,	1.0]
$\langle expr \rangle^2$	= [	0.0,	0.2,	1.0,	2.2,	4.0,	6.2,	9.0,	12.2,	16.0,	20.2]
$\langle expr \rangle^3$	= [	0.0,	0.5,	1.0,	1.5,	2.0,	2.5,	3.0,	3.5,	4.0,	4.5]
$\langle expr \rangle^4$	= [	0.0,	0.5,	1.0,	1.5,	2.0,	2.5,	3.0,	3.5,	4.0,	4.5]

The architecture of the distances (ABS or COS) relative to the boundaries (α, β) and the final weighted slicing of the roulette wheel are represented in Figure 3. Note that identical topologies $\langle expr \rangle^3$ and $\langle expr \rangle^4$ receive the exact same percentage slice. Conversely, $\langle expr \rangle^2$, holding the smallest delta ($\Delta = 0.15$), dominates the stochastic disk with the highest swap probability.

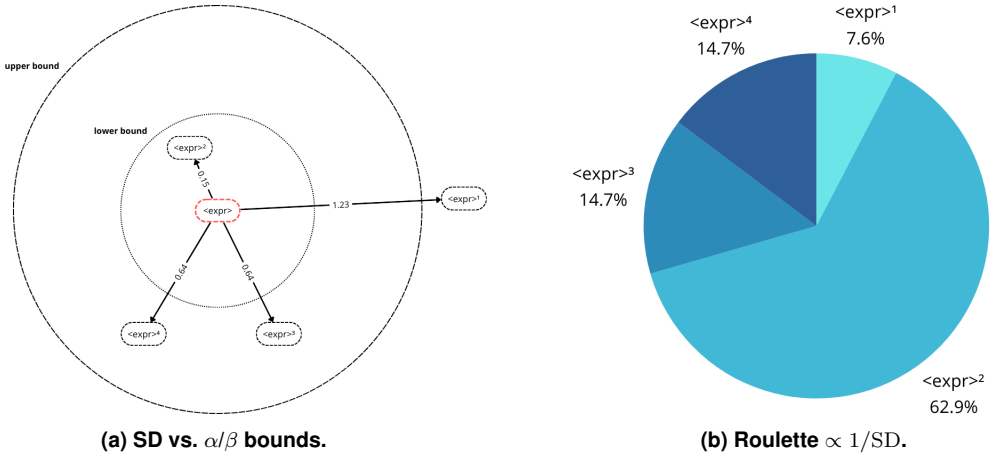
5. Computational Experiments

The performance of the RSC was evaluated against conventional GGP (DEF) and the grammatical versions of restricted semantic methods: SAC, SSC, and SCPS. The predictive function ecosystem, described in Table 3, follows the semantic-crossover suite of Uy et al. [Uy et al. 2010] with eight univariate polynomials or elementary nonlinearities (20 noise-free points) and two bivariate trigonometric surfaces (100 points). The protocol is designed to compare recombination operators under a shared grammar and a shared budget.

The postfix Grammar that generated the predictions excluded strict mathematical constants, handing the scalar synthesis burden entirely over to the GP stochastic search

Table 2. Calculation of the Semantic Distance between the target subtree and B candidates.

$SD(\langle expr \rangle, \langle expr \rangle^1) = 1.23$	$SD(\langle expr \rangle, \langle expr \rangle^2) = 0.15$
$SD(\langle expr \rangle, \langle expr \rangle^3) = 0.64$	$SD(\langle expr \rangle, \langle expr \rangle^4) = 0.64$



(a) SD vs. α/β bounds. **(b) Roulette $\propto 1/\mathrm{SD}$.**

Figure 3. Semantic distances mapped onto roulette slices.

engine:

$$
\begin{aligned}
\langle expr \rangle &::= \langle expr \rangle \langle expr \rangle \langle op \rangle \mid \langle expr \rangle \langle uop \rangle \mid \langle t \rangle \\
\langle op \rangle &::= + \mid - \mid * \mid / \\
\langle uop \rangle &::= sin \mid cos \mid exp \mid log \\
\langle t \rangle^1 &::= x_1 \mid 1 \quad \text{(when solving 1D problems)} \\
\langle t \rangle^2 &::= x_1 \mid x_2 \quad \text{(when solving 2D problems)}
\end{aligned}
$$

We adopted a *5-fold Cross Validation* to mitigate overfitting issues typical of unbounded approaches [Langford 2005]. At the end, all the best results are evaluated on a set of 1000 points for comparison between the algorithms. An allowance of 15×10^6 fitness cases was adopted to balance CPU cost fairness, given that semantic operators compute multiple internal prospects.

Table 4 outlines the experimental parameters. To fairly compare methods, computational cost is equalized by counting semantic evaluations ($DS(p, q)$) as fitness cases. Recombination operators may require distinct costs: SAC evaluates each offspring; SSC and SCPS allow up to a maximum number of attempts; and RSC varies based on subtree size and non-terminal type. All permutations of lower and upper bounds are evaluated where applicable. The semantic distance parameter defines the metric (absolute or cosine distance). Operator rates (for crossover and mutation) dictate execution probabilities, allowing for potential parent cloning or neutral mutations. Finally, elite preserves the best individuals across generations, while the maximum tree depth restricts tree height to prevent bloat and resource exhaustion.

Here, we used Performance Profiles (PPs) [Dolan and Moré 2002], a tool for comparing different algorithms on the same set of problems. PPs evaluate the relative perfor-

Table 3. Symbolic regression benchmark functions evaluated in the experiment.

Target Function	# Variables	Operators	# Points	Domain
$F1 = x + x^2 + x^3$	1	+, pow	20	[-1, 1]
$F2 = x + x^2 + x^3 + x^4$	1	+, pow	20	[-1, 1]
$F3 = x + x^2 + x^3 + x^4 + x^5$	1	+, pow	20	[-1, 1]
$F4 = x + x^2 + x^3 + x^4 + x^5 + x^6$	1	+, pow	20	[-1, 1]
$F5 = sin(x^2)cos(x) - 1$	1	pow, sin, cos	20	[-1, 1]
$F6 = sin(x) + sin(x + x^2)$	1	+, pow, sin	20	[-1, 1]
$F7 = log(x + 1) + log(x^2 + 1)$	1	+, pow, log	20	[0, 2]
$F8 = \sqrt{x}$	1	$\sqrt{}$	20	[0, 4]
$F9 = sin(x_1) + sin(x_2^2)$	2	+, pow, sin	100	[-1, 1]
$F10 = 2sin(x_1)cos(x_2)$	2	*, sin, cos	100	[-1, 1]

Table 4. Standardized evolutionary parameters throughout the benchmark.

Variable	Value / Domain
Similarity Limits (Lower / Upper)	$[0, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1]$
Semantic Distance Calculation	ABS, COS
Population Size (N)	500 individuals
Computational Budget Limit	15×10^6 fitness cases
Crossover / Mutation Rate	90% / 10%
Maximum Derivation Depth	10
Elite Size	5

mance of various algorithms across a set of problems. Given a set S of algorithms s_i for $i \in \{1, \ldots, n_s\}$ and a set P of problems p_j for $j \in \{1, \ldots, n_p\}$, $t_{p,s}$ represents the performance measure of algorithm s on problem p. The performance ratio $r_{p,s} = \frac{t_{p,s}}{\min\{t_{p,s}:s\in S\}}$ indicates the relative performance of method s. The probability that the performance ratio $r_{p,s}$ for method s is within a factor $\tau > 0$ of the best value observed in S is given by $\rho_s(\tau) = \frac{1}{n_p}|\{p \in P : r_{p,s} \leq \tau\}|$, where $\rho_s(\tau)$ defines the PP curves. According to PPs [Barbosa et al. 2010], it is possible to identify: (i) the approach that obtains the best results for the majority of the problems (highest $\rho(1)$), (ii) the most reliable approach (smallest τ such that $\rho(\tau) = 1$), and (iii) the approach with the best overall performance, indicated by the largest area under the PP curves (AUC). PPs are used here to measure (i) the performance of the best parameters for each algorithm, and (ii) the performance among the best algorithms of each type.

5.1. Preliminary Experiments

To understand the behavior of the proposed algorithms and identify their optimal configurations, a preliminary analysis was conducted for each method. Each configuration was executed 100 times, and the results were evaluated to establish the most effective setup for solving the targeted problems before comparing the best versions of each algorithm against one another.

The Roulette Semantic Crossover (RSC) algorithm was evaluated across all pos-

sible permutations of its parameters: the distance measure (Absolute, ABS, and Cosine, COS), the lower bound, and the upper bound. Performance profiles indicated that almost all configurations successfully solved the problems within a performance ratio of less than 10, though varying parameters significantly impacted efficiency. The highest overall performance area was achieved using the ABS distance with a lower bound of 10^{-5} and an upper bound of 10^{-2}. The variant that first reached $\rho(\tau) = 1$ (the smallest τ value such that $\rho(\tau) = 1$) used ABS with bounds 10^{-4} and 10^{-2}. While both distance metrics proved effective, ABS showed a slight advantage. The analysis also revealed that smaller lower bounds and moderate upper bounds yielded the best outcomes; removing the lower bound entirely caused the roulette selection to be dominated by overly similar subtrees, whereas excessively high upper bounds degraded performance.

The Semantic Crossover for Program Synthesis (SCPS) algorithm was tested using only two configurations, corresponding to the ABS and COS distance metrics. SCPS utilizes a two-step subtree selection process: first, it checks for subtrees with semantic differences and, if found, replaces the target with the most similar one. If this fails, the second step selects a subtree possessing at least one similarity and one difference. The choice of distance metric directly influences this first phase when identifying the most similar subtree among those with differences. The experimental results clearly demonstrated that the ABS distance metric is superior to the COS metric for the SCPS approach.

The Semantic-sensitive Crossover (SSC) method controls subtree exchanges by requiring candidate subtrees to fall within a specific similarity region defined by lower and upper bounds. The performance analysis revealed a strong dominance of the configuration using the COS distance, a lower bound of 10^{-5}, and an upper bound of 10^{-1}, closely followed by the variant using a lower bound of 10^{-4} with the same metric and upper bound. In general, the most successful outcomes were heavily associated with higher upper bound values. The optimal configurations consistently clustered around a lower bound of 10^{-5} and an upper bound of 10^{-1}, displaying a distinct performance advantage when paired with the COS distance metric.

Finally, the Semantic Aware Crossover (SAC) algorithm relies on a single boundary parameter, treated in this context as an upper bound. The experimental data highlighted the evident dominance of the configuration utilizing the COS distance metric with bounds set between 0 and 10^{-1}. Tracking the parameter adjustments showed that higher upper bound values consistently generated better results across both distance metrics. Overall, the COS distance metric proved to be highly effective for the SAC algorithm, whereas the ABS metric only yielded competitive results in one specific configuration.

5.2. Overall Performance

We perform here an overall evaluation of the techniques using PPs. The $\rho(\tau)$ curves are presented in Figure 4.

In the initial curve approximations (low τ values), imposing operators like SCPS and even the blind GGP search (DEF) shoot up rapidly. However, around the intercept of $\tau \sim 1.3$, the malleability of the roulette operator prevails: the RSC escapes the inertia of local valleys, breaks the lag, and establishes the largest overall Area Under the Curve, the most robust profile on this suite. The SAC systematically collapsed in general viability metrics, confirming that blindly discarding individuals imposes a punitive operational

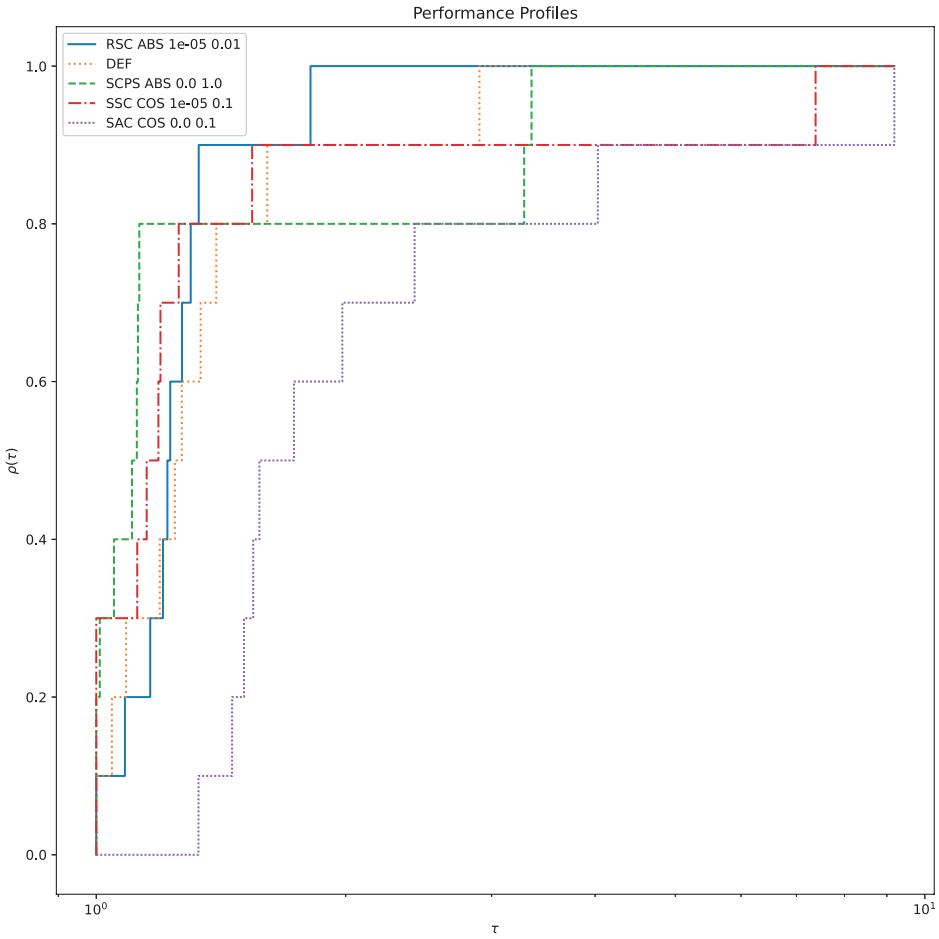


Figure 4. Performance Profiles (PP) compiling the semantic strands in GGP.

burden.

It is worth noting the exceptional resilience of the classic blind algorithm (DEF), occupying the macro runner-up position. This diverges from the reference literature base [Uy et al. 2010] because our experiments embedded severe *5-fold CV* techniques with data volumes up to five times larger. Under extensive validation training, the random search gains a very high structural stabilization margin.

5.3. Quantitative Evaluation of Success and Statistical Inference

Here, a run is a success if the absolute error on the 1000-point test set is smaller than or equal to 10^{-2} [Uy et al. 2010]. Table 5 presents the number of successes per function. RSC leads in volume (135), ahead of DEF (131), SCPS (126), SSC (109), and SAC (77). DEF, however, is the unique or joint maximum on the largest number of functions (4), versus 3 for RSC and SCPS.

F9 and F10 concentrate hits (219 and 153). Despite being bivariate and trigonometric, they use 100 training points rather than 20, so semantic distance separates good and poor subtrees more reliably. F7 (sum of two logarithms) yields a single success (SCPS). On F10 the success maximum is SSC (44), followed by RSC (33); RSC still trails DEF in the Kruskal-Wallis test on that target, consistent with a grammar that must

synthesize the constant 2 from arithmetic combinations.

Table 5. Number of success runs by crossover operator and function.

	RSC	DEF	SCPS	SSC	SAC	Total
F1	16	**25**	**25**	17	9	92
F2	6	6	**11**	5	4	32
F3	**3**	**3**	2	2	0	10
F4	0	**2**	0	1	1	4
F5	6	**9**	3	1	0	19
F6	**10**	5	6	2	4	27
F7	0	0	1	0	0	1
F8	6	5	**7**	1	2	21
F9	**55**	51	44	36	33	219
F10	33	25	27	**44**	24	153
Total	**135**	131	126	109	77	578
Best on #F_i	3	**4**	3	1	0	

Mechanically, the 2D non-linear expressions (F9 and F10) aggregated the largest constellation of hits. Counterintuitively to their higher dimensionality, the robust grid of data imputed to training (100 points in these matrices versus 20 points in 1D tests) offered a good mesh, facilitating rigorous semantic filtering against corrupted approximations. Conversely, the purely logarithmic mesh F7 suffocated virtually all generations, allowing a single unique victory by the SCPS operator.

Table 6. Kruskal-Wallis Validation: Sum of mathematical submissions in which the algorithm assumed either the isolated peak or the level of statistical equivalence (Maximum of 10).

	RSC	DEF	SCPS	SSC	SAC
Tie/Lead Count	**9**	**9**	**9**	6	0

The Kruskal-Wallis Test with a significance of $p = 0.05$ corroborated the mapped analytical scenario. The statistics confirmed the approaches RSC, SCPS, and DEF covering 90% of the resolutions. The only technical imperfection of the RSC occurred in the trigonometric inference F10 ($2\sin(x_1)\cos(x_2)$), where the grammar omitting scalar constants forced the process to synthesize the value "2" through organic combinatorial means. In this case, the stochastic precision of the RSC faltered in the face of blind randomness (DEF).

6. Conclusions and Future Work

This work addressed the algorithmic blindness inherent to Classic Genetic Programming, introducing the use of semantics in strict derivations (GGP). The classic mechanism of binary cut-off or costly candidate discard (enshrined in the literature by variants like SSC and SAC) was supplanted by the continuous construct of the *Roulette Semantic Crossover* (RSC).

RSC experimentally proved the fundamental premise: semantic proximity should not be the gene's condemnation, but the dictate of its probability slice. By weighting

crossovers on the inverse of the Euclidean phenotypic variation, the operator protected the micro-structural local search asset (allowing refined polishing in the neighborhoods) without prohibiting the radical passages that save the network from false attraction basins. Among the best operators, RSC shows the greatest robustness, even though Kruskal-Wallis tests mark DEF and SCPS as similar to it on most functions and DEF attains the per-function success maximum more often. That reading is reinforced by 5-fold CV, under which standard GGP crossover is more competitive than the literature considered here. Difficulty is heterogeneous: F7 is essentially unsolved, whereas F9 and F10 concentrate successes because their denser samples make semantic distance more discriminative. These ten low-dimensional, noise-free targets therefore support a comparison of grammatical semantic crossovers, not a general ranking of SR methods.

As future guidelines, replacement models for the roulette emerge, such as exploring stochastic weighting via exponential decay function of ranking, as do harder SR collections with more variables, noise, and contemporary test suites. Also, transposing this crossover to other problems, particularly to Gene Regulatory Network challenges, where incisive tree changes intrinsically affect multiple reactive elements of the matrix, requiring sensitive crossovers like the one promoted by the proposed technique, is a good research avenue.

Acknowledgments

The authors thank the financial support provided by FAPEMIG, CNPq, CAPES, and UFJF.

References

Barbosa, H. J. C., Bernardino, H. S., and Barreto, A. M. S. (2010). Using performance profiles to analyze the results of the 2006 CEC constrained optimization competition. In *Proc. of the Congress on Evolutionary Computation (CEC)*, pages 1–8. IEEE.

Bongard, J. and Lipson, H. (2007). Automated reverse engineering of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences*, 104(24):9943–9948.

Cheng, X., Khomtchouk, B., Matloff, N., and Mohanty, P. (2019). Polynomial regression as an alternative to neural nets.

de Freitas, J., Bernardino, H., Guerreiro, J., and Barbosa, H. (2015). Aplicação de uma programação genética gramatical multiobjetivo na inferência da máxima deformação longitudinal de dutos com amassamento. In *Proc. of the Iberian Latin American Congress on Computational Methods in Engineering (CILAMCE)*.

Dolan, E. D. and Moré, J. J. (2002). Benchmarking optimization software with performance profiles. *Math. Program.*, 91(2):201–213.

Forstenlechner, S. (2019). *Program Synthesis with Grammars and Semantics in Genetic Programming*. PhD thesis, University College Dublin, Ireland.

Huang, H.-H. and He, Q. (2023). *Nonlinear regression analysis*, page 558–567. Elsevier.

Juvekar, K. and Purwar, A. (2024). Cos-mix: Cosine similarity and distance fusion for improved information retrieval.

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

Langford, J. (2005). *The Cross Validation Problem*, page 687–688. Springer Berlin Heidelberg.

McKay, R. I., Hoai, N. X., Whigham, P. A., Shan, Y., and O'Neill, M. (2010). Grammar-based genetic programming: a survey. *Genetic Programming and Evolvable Machines*, 11(3–4):365–396.

McPhee, N. F., Ohs, B., and Hutchison, T. (2008). *Semantic Building Blocks in Genetic Programming*, page 134–145. Springer Berlin Heidelberg.

Mukherjee, S. and Sonal, R. (2023). A reconciliation between cosine similarity and euclidean distance in individual decision-making problems. *Indian Economic Review*, 58(2):427–431.

Nguyen, Q. U., Nguyen, X. H., and O'Neill, M. (2009). *Semantic Aware Crossover for Genetic Programming: The Case for Real-Valued Function Regression*, page 292–302. Springer Berlin Heidelberg.

Orzechowski, P., La Cava, W., and Moore, J. H. (2018). Where are we now?: a large benchmark study of recent symbolic regression methods. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '18. ACM.

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

Uy, N. Q., Hoai, N. X., O'Neill, M., McKay, R. I., and Galván-López, E. (2010). Semantically-based crossover in genetic programming: application to real-valued symbolic regression. *Genetic Programming and Evolvable Machines*, 12(2):91–119.