Simultaneous Model-Based Evolution of Constants and Expression Structure in GP-GOMEA for Symbolic Regression

Johannes Koch[1,2(✉)], Tanja Alderliesten[3], and Peter A.N. Bosman[1,2]

[1] Centrum Wiskunde & Informatica, Amsterdam, The Netherlands
{Johannes,Peter.Bosman}@cwi.nl
[2] Delft University of Technology, Delft, The Netherlands
[3] Leiden University Medical Center, Leiden, The Netherlands
T.Alderliesten@lumc.nl

Abstract. Genetic programming (GP) approaches are among the state-of-the-art for symbolic regression, the task of constructing symbolic expressions that fit well with data. To find highly accurate symbolic expressions, both the expression structure and any contained real-valued constants, are important. GP-GOMEA, a modern model-based evolutionary algorithm, is one of the leading algorithms for finding accurate, yet compact expressions. Yet, GP-GOMEA does not perform dedicated constant optimization, but rather uses ephemeral random constants. Hence, the accuracy of GP-GOMEA may well still be improved upon by the incorporation of a constant optimization mechanism. Existing research into mixed discrete-continuous optimization with EAs has shown that a simultaneous and well-integrated approach to optimizing both discrete and continuous parts, leads to the best results on a variety of problems, especially when there are interactions between these parts. In this paper, we therefore propose a novel approach where constants in expressions are optimized at the same time as the expression structure by merging the real-valued variant of GOMEA with GP-GOMEA. The proposed approach is compared to other forms of handling constants in GP-GOMEA, and in the context of other commonly used techniques such as linear scaling, restarts, and constant tuning after GP optimization. Our results indicate that our novel approach generally performs best and confirms the importance of simultaneous constant optimization *during* evolution.

Keywords: Genetic programming · Constant optimization · Symbolic regression · Model-based evolutionary algorithms.

1 Introduction

In recent years, the field of eXplainable AI (XAI) has received increased attention, especially for use cases where AI models can affect lives and livelihoods. Given a dataset and a library of atomic functions such as $\{+, -, \times, \div, \sin\}$, symbolic regression (SR) is the task of finding an interpretable expression that best

describes the relation between one (output) variable and other (input) variables [15]. Compact SR models (i.e., expressions) are interesting from the perspective of XAI and interpretable ML (IML), as they are readable and therefore have the potential to be humanly understandable [24,31]. GP-GOMEA is a genetic programming (GP) based SR method that is amongst the state-of-the-art for finding compact, yet accurate expressions and part of the non-dominated front on the recent SR benchmark SRBench [17,29].

GP approaches generally primarily optimize the expression structure through a process of iteratively recombining individuals in a population of (initially random) expressions. The strength of GP-GOMEA in particular is finding expression structures by dynamically learning and exploiting a linkage model during optimization that captures key dependencies between parts of an expression template. Still, in general, for an SR expression to be highly accurate, not only must the right expression structure be found, also the real-valued coefficients must be optimized. Typically, ephemeral random constants (ERCs) are used [15], which are random constants that are sampled during initialization and then not modified any further. Current approaches refine these by performing gradient-based search or randomly mutating constant values. While the expression structure is optimized by performing variation on the whole population, constant optimization typically is performed on individual solutions. In the context of XAI, where SR expressions need to be compact, finding better constants is expected to increase expression accuracy while keeping expressions compact.

In this paper, we present and evaluate GP-RV-GOMEA, a new approach to constant optimization in GP-GOMEA that takes inspiration from GAMBIT [25], a fully integrated model-based evolutionary algorithm (MBEA) approach to mixed discrete and continuous optimization. To make constant optimization a first-class citizen in GP-GOMEA, SR is considered to be a mixed discrete and real-valued problem, where GP-GOMEA is used to optimize the structure of expressions and the real-valued GOMEA (RV-GOMEA) is used to simultaneously optimize the constants of all expressions. Compared to random coefficient mutations, the design and use of a dedicated MBEA is expected to lead to a more directed and effective search that is better positioned to overcome potentially ill-conditioned gradients that non-linear-least-squares methods can encounter [16].

The remainder of this paper is organized as follows. Related works are first discussed in the following Section. In Section 3 we describe the new method GP-RV-GOMEA. In Section 4, we perform experiments to assess the performance of GP-RV-GOMEA as well as other GP-GOMEA variants. We discuss our main findings in Section 5 and draw our final conclusions in Section 6.

2 Related Work

Since the introduction of ERCs, constant optimization in GP has become a well-studied subject and various approaches have been suggested, including co-

efficient mutations [30], gradient-based search [9,14,23], and meta-heuristic optimization [5,10,20,1,26].

In [1] and [20], real-valued EAs are nested within a GP algorithm to separately optimize the constants of either the best or all individuals, respectively. In [26], simulated annealing is used in the same manner. Coefficient mutations, gradient-based search, and the approaches presented in [20,26,1], all optimize the constants in expressions separately per expression, either in a nested loop inside GP, or after GP terminates. In this paper, we consider *simultaneous* evolutionary optimization. The motivation for joint optimization is that as the GP population converges over time, similar constant values are likely needed across the population, and thus constant optimization is not viewed as an independent problem for each individual in this work. Moreover, in related work with mixed discrete and continuous variables, their joint optimization has been shown to be advantageous over independent nested, or sequential optimization [25].

In [5], differential evolution is used to optimize the expression structure and constants at once. However, GP is modeled as a fully real-valued optimization problem by using a fixed number of decision variables to encode the expression structure, combined with a mapping from continuous values to GP operators. The remaining decision variables are the constant values available to an individual. Instead of ERCs, constant references are added that are substituted with the corresponding constant value during evaluation. Our method differs from this approach by interleaving optimization of the structure and constants, and using separate algorithms to do so. A similar approach has been presented in [10], however, significant progress both in GP and real-valued optimization has been made since 1995. Moreover, our method further includes an optimization to avoid unnecessary fitness evaluations, not present in any of the previous approaches. To the best of our knowledge, this also is the first work comparing such a form of constant optimization with other approaches.

An approach to mixed discrete and real-valued optimization using GOMEA has been presented in [25]. Our new method takes direct inspiration from that work but uses GP-GOMEA as a specialized GP algorithm for the discrete part and RV-GOMEA for the constants. Regarding constants in GP-GOMEA, ERCs and coefficient mutation have been explored in [29] and [30] respectively. Both approaches are used as a baseline for the new method we introduce here.

3 Model-based Evolutionary Constant Optimization in GP-GOMEA

In this section, GP-RV-GOMEA, a combination of GP-GOMEA and RV-GOMEA based on GAMBIT is presented. First, the family of gene-pool optimal mixing evolutionary algorithms (GOMEAs) and the individual algorithms used are shortly described before the combination is introduced.

3.1 Gene-pool Optimal Mixing Evolutionary Algorithms

Similar to other population-based algorithms, GOMEAs work by iteratively refining an initially random set of candidate solutions. Each individual in a GOMEA is typically represented as a fixed-length list of decision variables. To achieve effective optimization, the aim in a GOMEA is to model and exploit dependencies between linked variables [6].

These dependencies can be set a priori or learned in every generation during optimization. This linkage information is modeled using a so-called family of subsets (FOS) structure, a set that contains subsets of all decision variable indices. Each subset in the FOS then corresponds to a group of linked variables and is used as a crossover mask during variation. To learn the FOS during optimization, hierarchical clustering using UPGMA [8] is often used on the decision variables. As a similarity measure, typically the mutual information between the decision variables is used [6]. Starting from all single variable subsets, a subset is added for every pair of joined subsets during hierarchical clustering. This results in a so-called Linkage Tree FOS.

Variation then is performed for each individual subset in the FOS, where all variables in the given subset are varied together. For discrete variables, the new values are inherited from a donor randomly picked from the population, or sampled from a previously estimated multivariate normal distribution in the real-valued case. The changed solution is then evaluated, and reverted in case the fitness is worse compared to before the modification. This variation procedure is called Gene-pool Optimal Mixing (GOM) and is performed for all FOS subsets and individuals in a single generation. For discrete decision variables, the donors are a copy of the population that is not modified, to match the distribution of values at the time of learning the linkage model. The real-valued distribution used corresponds to a potentially adapted (see Section 3.3) maximum-likelihood estimate of the top 35% solutions in the population for every FOS subset.

Further, forced improvements, a mechanism which forces solutions that did not improve within the last $1 + \log_{10}(\text{population size})$ generations, are used [6]. This mechanism subjects such solutions to an additional round of GOM steps until the solution has improved, where the donor is the elite of the current population. If no improvement can be found after processing all FOS subsets, the solution is replaced with the elite solution.

3.2 GP-GOMEA

GP-GOMEA [29] is the GP variant of GOMEA, where a fixed, tree-based symbolic expression template is mapped to a string representation. This inherently limits the maximum size of the learned expressions. Typically, full n-ary trees are used, where n is the largest arity in the function set used. Using a fixed template introduces syntactic introns for expressions smaller than the template size, i.e., all subtrees of terminal nodes such as input features or constants have no impact on the symbolic expression encoded by a solution. This is exploited during GOM when no actively used node is changed. The inherited modifications are simply accepted as the fitness remains unchanged.

3.3 RV-GOMEA

RV-GOMEA [4] is the real-valued version of GOMEA for continuous search spaces.When the linkage is not set a priori, then the similarity metric used during clustering typically is the Pearson product-moment correlation coefficient. However, in this paper, only the full FOS, i.e., a single crossover mask containing all variables, is used as all constants used in a solution can affect each other.

In addition to sampling new values from a learned distribution, additional mechanisms detailed in [4] influence the variation compared to other GOMEA variants. These are the Anticipated Mean Shift (AMS), which shifts a part of the population in the direction of the mean shift between the previous and current generation, and Adaptive Variance Scaling (AVS) to adapt the step size in case solutions are found more than a standard deviation away from the distribution mean. Forced improvements in the real-valued case are performed by bisecting the values of the solution and the elite for each FOS subset.

3.4 GP-RV-GOMEA

Solution Representation To make constant optimization a first-class citizen, the GP-GOMEA genotype consisting of discrete decision variables is extended with a fixed number of real-valued constants that will be optimized using RV-GOMEA. Instead of special constant nodes, the GP terminal set is extended with constant references for all added constants. This representation is shown in Figure 1 and during evaluation, constant references are substituted with the corresponding value from the real-valued decision variables. Note that introns can be both discrete or real-valued.

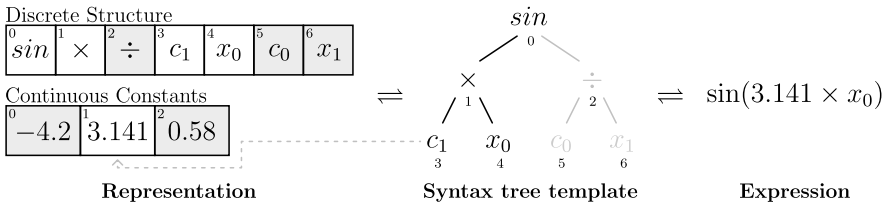

Fig. 1. The genotype (left) of a single individual in GP-RV-GOMEA and how it relates to the encoded expression (right). Shaded values are introns that do not affect the semantic meaning of the expression.

Interleaving Scheme Using this mixed representation, optimization then follows the approach described in [25] with additional modifications specific to GP. The approach is outlined in Algorithm 1. After initialization, in every generation, first, the discrete linkage model used by GP-GOMEA is learned from the current population. This is followed by a mixed variation procedure that interleaves performing discrete GOM using GP-GOMEA and real-valued variation steps using RV-GOMEA until all subsets in the discrete FOS have been processed. In [25], this interleaving is done by shuffling the order of all variation steps, both discrete and continuous, to balance the computational effort spent. However,

as GP-GOMEA only performs evaluations if actively used nodes are modified, this quickly leads to an imbalanced distribution of computational effort between the discrete and real-valued optimization. Hence, a different approach directly based on the number of evaluations performed for each variable type is used. To balance the computational effort spent, the ratio of real-valued to discrete evaluations ($\in [0, \infty]$) is transformed into a probability of performing a real-valued step and then the next step is sampled from the uniform distribution (lines 6-7 in Algorithm 1). Note that while GP steps amount to performing GOM for all solutions and a single FOS subset, the real-valued steps amount to performing a full RV-GOMEA generation. Thus, the real-valued steps consist of computing the maximum-likelihood estimate, performing GOM, and finally AMS.

Algorithm 1: GP-RV-GOMEA

```
1  P ⟵ InitializeAndEvaluatePopulation()
2  while ¬TerminationCriteriaSatisfied do
3      O ⟵ P
4      F ⟵ LearnLinkageModel(P)
5      while F not empty do
6          p_RV ⟵ 1 − 0.5 · #RV evaluations / #GP evaluations
7          if U(0,1) < p_RV then
8              O ⟵ RVStep(O)
9          else
10             O ⟵ GPStep(O, P, TakeRandom(F))
11     O ⟵ ApplyForcedImprovements(O)
12     P ⟵ O
```

Forced Improvements After the main variation steps have been performed, forced improvements are performed as described in [27,6]. The real-valued RV-GOMEA steps also perform forced improvements, however, as opposed to the procedure described in [4], the real-valued forced improvements in the proposed method are modified to take the discrete structure into account. This is done by ensuring that the donor used makes use of constant values and then interleaving the discrete and real-valued forced improvements steps, as is done for the main variation shown in Algorithm 1. Since improvements can be both of structural and real-valued nature, the number of generations without improvements needed before real-valued forced improvements are applied was decreased from the default of 100 in RV-GOMEA to 20 generations.

Real-valued Intron Handling Similar to how GP-GOMEA can have introns, it is possible for real-valued constants to not be used in the encoded expression. To avoid unnecessary evaluations, changes to unused constants thus are not evaluated and are simply accepted in line with the intron handling of GP-GOMEA. Note that as we use the full FOS for RV-GOMEA, any change to the constants of a solution that actively uses at least one constant still has to be evaluated.

In RV-GOMEA, the constant values used for the maximum-likelihood estimation are selected based on the solution fitness. However, with the presence of introns, some of the values used for this estimate possibly do not contribute to

the fitness of an individual. To avoid introducing noise through these inactive values, we make RV-GOMEA "intron aware" by filtering out these intron values in all steps where constant values are used to guide optimization. When selection is performed, for each constant index, the top 35% of the *active values* are selected. Similarly, AMS is performed only on individuals with active constants and AVS only considers active values when updating the variance scaling factors.

4 Experiments and Results

In this section, we perform two types of experiments. The first experiment is performed using noise-free synthetic problems to isolate constant optimization and validate the effectiveness of the proposed approach. The second experiment uses real-world data to confirm the practical usefulness of the approach.

4.1 Experimental Setup

We compare the proposed approach without and with intron-aware (IA) model updates to GP-GOMEA with ERCs ([29]) and GP-GOMEA with coefficient mutation ([30]), hereafter abbreviated as RV, RV+IA, ERCs, and ERCs+CM, respectively. To isolate the effect of constant optimization from other techniques commonly used to increase performance such as linear scaling [13] (hereafter LS), constant tuning after optimization, and restarts, all combinations are tested. The other parameters used are detailed in Table 1. The function set was chosen based on [21], where it was shown that this function set tends to generalize well to unseen data. In all experiments, cross-validation with 5 folds and 7 repeats per fold using different seeds is used, corresponding to 35 runs per problem, method, restart, and LS configuration.

We compare based on a fixed evaluation budget, as the different methods do not use the same number of fitness evaluations per generation. In addition, a run without restarts is stopped when it converges. A run is considered as converged when either all individuals encode the same structure or no evaluations

Table 1. The parameter settings used in the experiments.

	Method			
Parameter	ERCs	ERCs+CM	RV	RV+IA
Objectives	Mean squared error (MSE)			
Tree height	5 (31 Nodes)			
Operators	$+, -, \times, \div, \sin$			
Constant initialization	$\mathcal{U}(\min\{y_{train}\}, \max\{y_{train}\})$			
Termination	10^7 evaluations or convergence			
Population size	1000			
Constant probability	50%			
Number of constants			10	10
Intron Awareness			No	Yes

were spent during a full generation. For the synthetic problems, a target mean squared error (MSE) value of 10^{-8} was used as an additional termination criterion in the first experiment. With restarts, the previous convergence conditions or no improvement to the elitist solution of the current restart within 10 generations trigger a full restart. Note that two elitist solutions are maintained, one to maintain the best solution across all restarts and one for the best solution of the current restart. A budget of 10^7 evaluations is used to ensure that all methods can converge within the computational budget. Hence, without restarts the fitness after convergence is compared, albeit this is not a fair comparison based on the actual number of evaluations spent. With restarts, the comparison based on evaluations is fair, however, the number of restarts or generations performed is not. Constant optimization after GP is performed using the L-BFGS implementation from PyTorch[22] with a limit of 500 iterations, after which the resulting model is simplified using SymPy[19].

4.2 Synthetic Problems: Does RV within GP Work as Desired?

In the first experiment, we use the following synthetic problems with 1000 instances sampled with $x_i \sim \mathcal{U}(-10, 10)$:

- $-4.2 \cdot x_0 + \sqrt{2}x_1 + e \cdot x_2$
- $0.1 \cdot x_0 + 0.2 \cdot x_1 + 2.4 \cdot x_2$
- $\sin(\pi \cdot x_0)/(\pi \cdot x_0)$
- $\sin(1.772 \cdot x_0) + \sin(2.035 \cdot x_2)$
- $\sin(\frac{\pi}{2} \cdot x_0 + \frac{\pi}{3} \cdot x_1)$
- $\sin(\pi \cdot x + e)$

These problems were selected with specific criteria in mind. First, discovering the correct structure should not be overly challenging, to compare the constant optimization capabilities of various methods. Second, the chosen problems were designed to feature nested or non-linear combinations of constants, thereby ensuring that LS does not fully mitigate the need for constant optimization. Lastly, the synthetic problems were crafted to exhibit multi-modal constant optimization landscapes, motivating the use of gradient-free techniques. The landscape near the optimal constant values for $\sin(\pi \cdot x + e)$ is shown in Figure 2.

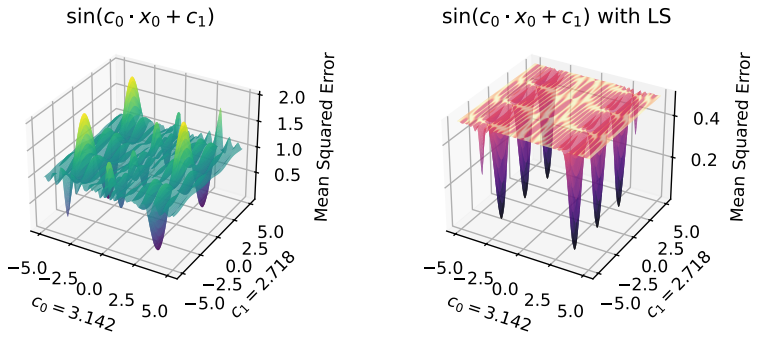

Fig. 2. The constant optimization landscape can be multi-modal, both with and without linear scaling (LS). The error for every constant combination was computed using (the same) 500 instances sampled from $\mathcal{U}(-10, 10)$.

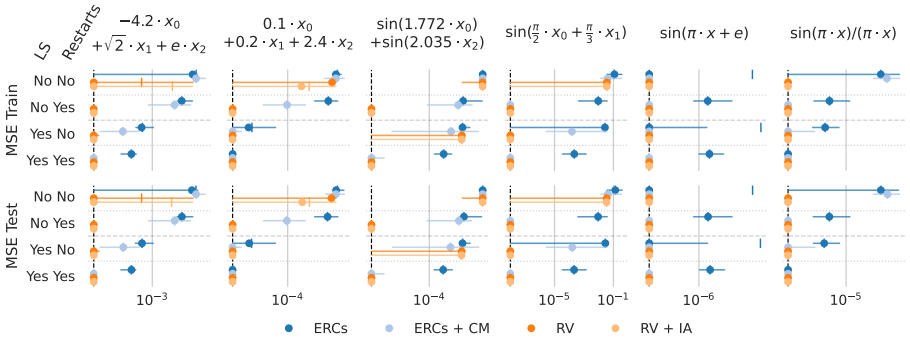

Fig. 3. The training and testing MSE scores for the synthetic problems on a logarithmic scale. The bar corresponds to the median MSE before tuning, while the circle and horizontal line correspond to the median and interquartile range (IQR) after post-processing. Note that the MSE was capped at the target value of $1e-8$, which is highlighted with a vertical line.

The results are shown in Figure 3 and Table 2. Note that this is an inherently biased comparison, as ERCs do not have an inherent ability to change the value of constants other than recombining several constants to represent new values not sampled initially. Likewise, coefficient mutation is random by nature, and thus the likelihood of improving constants decreases as the constant values get closer to the closest local optima. Nonetheless, we can confirm that our approach (i.e., RV and RV + IA) effectively optimizes constants, and clearly is better at reaching the MSE target of 10^{-8} compared to the other constant optimization types considered (Figure 3). Table 2 shows that with intron awareness the proportion of runs reaching the MSE target increases, indicating that the real-valued distributions estimated have a better fit.

Both LS and restarts increase the average performance of all methods and decrease variance. With restarts, both RV variants reliably reach the target value in almost all runs. The constant optimization performed during post-processing is most noticeable for ERCs without LS or restarts, where it can lead to noticeable MSE improvements. However, compared to optimizing constants during optimization, the overall effect of tuning the best model after GP is small. Interestingly, tuning after and restarts have an unintuitive interaction when considering ERCs. Without restarts, the re-occurring subexpression $\sin(\pi \cdot x)$ is often modeled with a constant that can be improved during post-processing. With restarts, however, many runs find $\sin(x + x + x)$ instead, as $3 \cdot x \approx \pi \cdot x$. Since constant tuning was only applied before simplification, the lack of an explicit constant explains the decreased performance of ERCs with restarts.

In terms of the expressions found, in Figures 4 and 5 we can see that apart from the simplification during post-processing, LS has a clear effect on both the number of constants used and the expression size, partly explained by the added scaling terms which add 2 constants and 4 nodes. The effect of simplification for the problems used tends to be considerable, for some problems the size after simplification is halved. The type of constant optimization used, however, has little impact on the expression size.

Table 2. Proportion of runs that reach the 10^{-8} MSE target. The best values per setting are highlighted in bold and the percentage in parentheses corresponds to the contribution of constant optimization after GP. The colored triangles indicate statistically significantly better methods ($p < 0.05$) as per Fisher's exact test [7].

LS	Restarts	ERCs ▼	ERCs + CM ▼	RV ▼	RV + IA ▼
$-4.2 \cdot x_0 + \sqrt{2} \cdot x_1 + e \cdot x_2$					
No	No	37.1% (+37.1%)	2.9 % (+0.0%)	**60.0**% (+25.7%)	**60.0**% (+22.9%)
	Yes	0.0% (+0.0%)	5.7 % (+0.0%)	88.6 % (+0.0%)	**97.1**% (+0.0%)
Yes	No	5.7% (+5.7%)	20.0 % (+0.0%)	74.3 % (+8.6%)	**94.3**% (+0.0%)
	Yes	0.0% (+0.0%)	74.3 % (+0.0%)	**97.1**% (+0.0%)	94.3 % (+0.0%)
$0.1 \cdot x_0 + 0.2 \cdot x_1 + 2.4 \cdot x_2$					
No	No	14.3% (+14.3%)	5.7 % (+0.0%)	40.0 % (+5.7%)	**42.9**% (+5.8%)
	Yes	0.0% (+0.0%)	14.3 % (+0.0%)	94.3 % (+0.0%)	**100.0**% (+0.0%)
Yes	No	34.3% (+2.9%)	71.4 % (+0.0%)	97.1 % (+0.0%)	**100.0**% (+0.0%)
	Yes	97.1% (+0.0%)	**100.0**% (+0.0%)	**100.0**% (+0.0%)	**100.0**% (+0.0%)
$sin(1.772 \cdot x_0) + sin(2.035 \cdot x_2)$					
No	No	2.9% (+2.9%)	5.7 % (+0.0%)	**14.3**% (+2.9%)	**14.3**% (+0.0%)
	Yes	0.0% (+0.0%)	11.4 % (+0.0%)	82.9 % (+0.0%)	**100.0**% (+0.0%)
Yes	No	5.7% (+5.7%)	25.7 % (+0.0%)	40.0 % (+2.9%)	**45.7**% (+0.0%)
	Yes	0.0% (+0.0%)	57.1 % (+0.0%)	97.1 % (+0.0%)	**100.0**% (+0.0%)
$sin(\frac{\pi}{2} \cdot x_0 + \frac{\pi}{3} \cdot x_1)$					
No	No	14.3% (+14.3%)	0.0 % (+0.0%)	34.3 % (+0.0%)	**45.7**% (+0.0%)
	Yes	2.9% (+0.0%)	80.0 % (+0.0%)	**100.0**% (+0.0%)	**100.0**% (+0.0%)
Yes	No	28.6% (+28.6%)	20.0 % (+0.0%)	**85.7**% (+0.0%)	80.0 % (+0.0%)
	Yes	0.0% (+0.0%)	94.3 % (+0.0%)	**100.0**% (+0.0%)	**100.0**% (+0.0%)
$sin(\pi \cdot x + e)$					
No	No	85.7% (+85.7%)	74.3 % (+0.0%)	97.1 % (+0.0%)	**100.0**% (+0.0%)
	Yes	0.0% (+0.0%)	**100.0**% (+0.0%)	**100.0**% (+0.0%)	**100.0**% (+0.0%)
Yes	No	74.3% (+74.3%)	74.3 % (+0.0%)	97.1 % (+0.0%)	**100.0**% (+0.0%)
	Yes	0.0% (+0.0%)	**100.0**% (+0.0%)	**100.0**% (+0.0%)	**100.0**% (+0.0%)
$sin(\pi \cdot x)/(\pi \cdot x)$					
No	No	40.0% (+40.0%)	2.9 % (+0.0%)	**88.6**% (+0.0%)	80.0 % (+0.0%)
	Yes	5.7% (+0.0%)	**100.0**% (+0.0%)	**100.0**% (+0.0%)	**100.0**% (+0.0%)
Yes	No	8.6% (+0.0%)	57.1 % (+0.0%)	**100.0**% (+0.0%)	**100.0**% (+0.0%)
	Yes	97.1% (+0.0%)	**100.0**% (+0.0%)	**100.0**% (+0.0%)	**100.0**% (+0.0%)

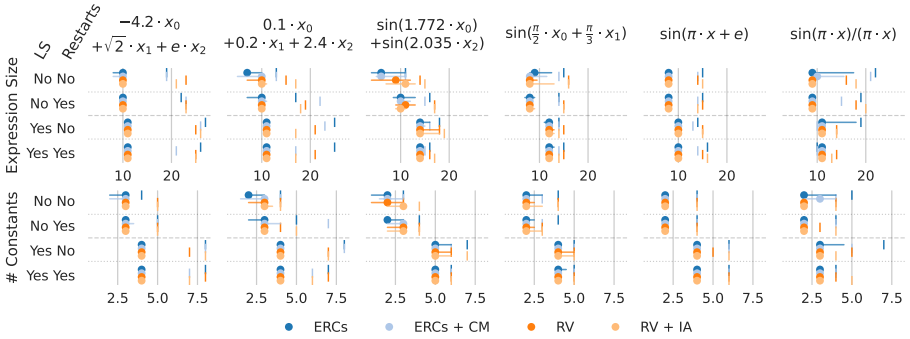

Fig. 4. The number of constants used and expression sizes for the synthetic problems. The bar corresponds to the median before post-processing, while the circle and horizontal line correspond to the median and IQR after post-processing.

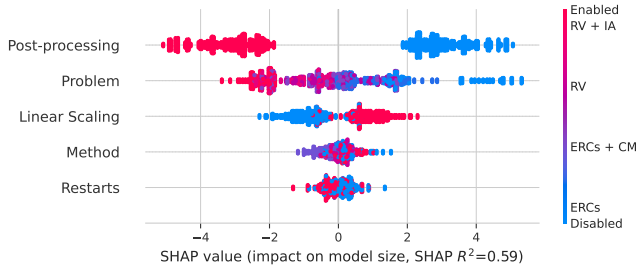

Fig. 5. The SHAP[18] values show how different aspects influence expression size for the synthetic problems. The different methods are highlighted on the color map, for the other binary aspects the color indicates whether it was enabled or not.

4.3 Real-world Problems: GP-RV-GOMEA vs ERCs and Coefficient Mutation

In this experiment, we consider the problems listed in Table 3 and compare the obtained expressions in terms of the coefficient of determination R^2 score, expression size, and the number of constants used. Compared to the previous result, this experiment aims to provide a practically relevant comparison to the other forms of constant optimization.

Table 3. The real-world problems used in the second experiment.

Problem	# Instances	# Features
Airfoil Self-noise [28]	1503	5
Concrete Compressive Strength [11]	1030	8
Energy Cooling [2]	768	8
Energy Heating [2]	768	8
Yacht Hydrodynamics [12]	308	6

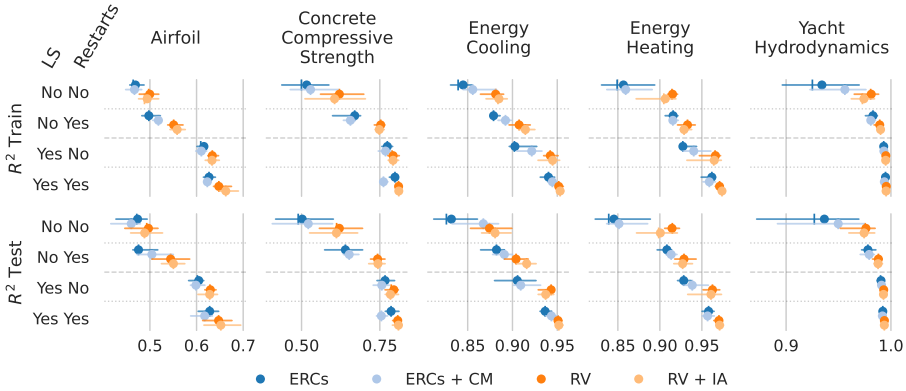

Fig. 6. The training and testing R^2 scores for the real-world problems. The bar corresponds to the median R^2 before tuning, while the circle and horizontal line correspond to the median and IQR after post-processing.

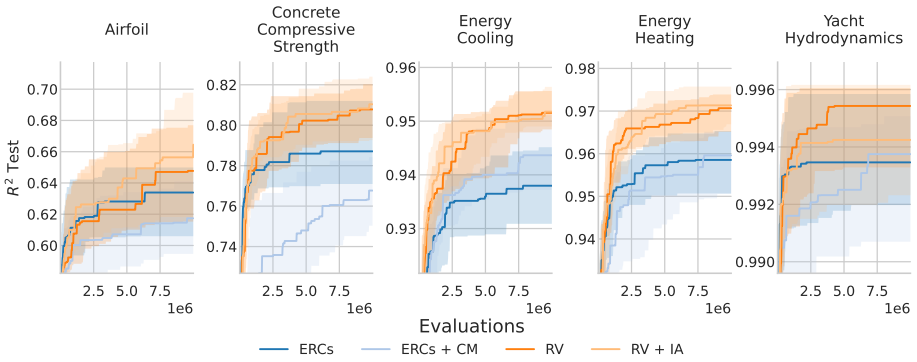

Fig. 7. The median and IQR for the test R^2 scores over evaluations spent with linear scaling and restarts enabled.

The results for the R^2 scores displayed in Figure 6 confirm that the proposed approach performs competitively on real-world problems, again outperforming both ERCs and coefficient mutation in terms of solution accuracy. Without LS or restarts, the RV version with intron-aware Gaussian model updates performs slightly worse compared to when intron awareness is not used. This could be explained by the noisy Gaussian distribution updates being more robust to changes in the expression structure between real-valued steps, as updating the distribution takes longer in contrast to the intron-aware version. With restarts or LS, however, the use of intron-awareness tends to perform a little better.

Figure 7 shows that while ERCs have an initial advantage as all computational effort is spent on finding better structures, the importance of constant optimization becomes apparent as the evolution progresses. Improvements are still found close to reaching the computational budget, indicating that an increased budget could lead to improved results.

Both LS and restarts improve accuracy and decrease variance for all methods tested, with LS having a bigger effect. In line with the synthetic problems in the

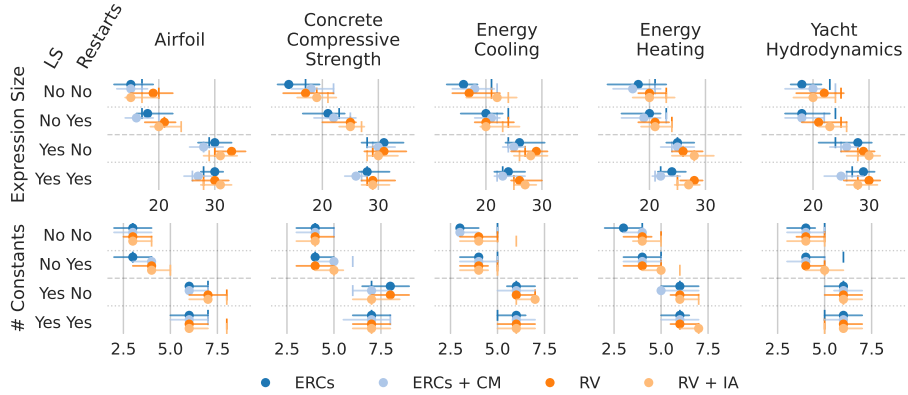

Fig. 8. The number of constants used and expression sizes for the real-world problems. The bar corresponds to the median before post-processing, while the circle and horizontal line correspond to the median and IQR after post-processing.

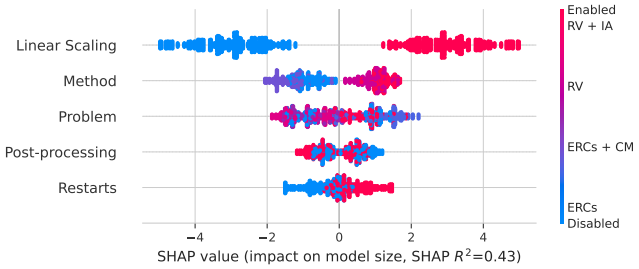

Fig. 9. The SHAP[18] values show how different aspects influence expression size for the real-world problems. The different methods are highlighted on the color map, for the other binary aspects the color indicates whether it was enabled or not.

previous experiment, the effect of tuning constants after GP is most noticeable for ERCs and in the absence of LS and restarts, but small compared to the effect of constant optimization during optimization.

In contrast to the synthetic problems, Figure 9 indicates that next to LS, the RV constant optimization methods also lead to larger expressions while the effect of simplification decreased. This motivates a multi-objective approach.

Statistical testing results following the approach recommended by [3] are shown in Table 4, indicating that GP-RV-GOMEA is highly likely to lead to similar or noticeably better R^2 scores compared to ERCs and coefficient mutation.

Table 4. The pair-wise probabilities of how likely M_1 is to perform better, approximately equal (within $0.01R^2$), or worse in terms of test R^2 compared to M_2 with LS and restarts, using a Bayesian hierarchical correlated t-test [3].

M_1	M_2	$P(R^2_{M_1} > R^2_{M_2})$	$P(R^2_{M_1} \approx R^2_{M_2})$	$P(R^2_{M_1} < R^2_{M_2})$
ERCs	ERCs + CM	0.172	0.749	0.079
ERCs	RV	0.019	0.400	0.581
ERCs	RV + IA	0.025	0.205	0.771
ERCs + CM	RV	0.059	0.114	0.827
ERCs + CM	RV + IA	0.056	0.026	0.918
RV	RV + IA	0.009	0.971	0.020

5 Discussion

We proposed GP-RV-GOMEA, a new form of constant optimization in GP-GOMEA based on GOMEA-based mixed discrete and continuous optimization. Our experiments confirmed that simultaenous constant optimization across the whole population as opposed to per individual indeed works well for GP-GOMEA, and that our proposed approach clearly outperforms previous forms of constant optimization both with and without intron-aware real-valued Gaussian model updates. This holds for all combinations of linear scaling, restarts, and constant tuning after GP. Furthermore, on the problems considered, constant optimization during evolution has a positive effect on solution quality. However,

on real-world problems, solution sizes tend to increase as well. While LS generally has the largest impact on accuracy, clearly, constant optimization too can have a noticeable impact.

The proposed approach introduces new parameters, such as the size of the constant pool available and the parameters of RV-GOMEA. These have not been explored extensively yet, as the goal of this work was to determine if such an approach is feasible and effective, revitalizing the research by [10]. Notably, not all mechanisms introduced in GAMBIT [25] were considered in this work, possibly increasing the effectiveness of this approach further.

Furthermore, the method has not yet been compared to on-line gradient-based constant optimization, which is a commonly used form of constant optimization in GP. Possibly a hybrid approach akin to basin hopping could prove to be more effective than only using one form of constant optimization.

This work introduced a novel, intron-aware approach to updating the real-valued optimizer, showing increased performance on synthetic and real-world problems when combined with linear scaling or restarts. Notably, intron-awareness affects how fast the real-valued model can adapt to the changes in the real-valued fitness landscape introduced by structural changes. Compared to the intron-aware version, the decreased adaptivity can be seen as an implicit form of regularization, although no overfitting was found with the settings used.

While primarily caused by LS, the observed increases in expression size with constant optimization suggest that a multi-objective approach is needed to ensure small and accurate models. Concerning constant tuning after GP, the observed interaction with restarts suggests that fine-tuning should be applied both before and after simplification to obtain better results.

6 Conclusion

Over the years, several approaches to constant optimization in GP have been proposed, however, they come with drawbacks such as the need for gradients or large numbers of evaluations due to blind search. With this in mind, we proposed a novel, model-based, way of optimizing constants in GP-GOMEA, which we evaluated in the context of linear scaling, restarts, and optimization after GP on both synthetic and real-world data.

Our experiments confirm that optimizing constants across GP individuals can be effective and that simultaneous (evolutionary) constant tuning during GP can be required for increased performance. Compared to ERCs and coefficient mutation with the same underlying GP algorithm, we find that the proposed method improves overall expression accuracy in all settings considered, while achieving similar expression size.

7 Acknowledgements

This research was funded by the European Commission within the HORIZON Programme (TRUST AI Project, Contract No.: 952060).

References

1. Alonso, C.L., Montaña, J.L., Borges, C.E.: Evolution strategies for constants optimization in genetic programming. In: 2009 21st IEEE International Conference on Tools with Artificial Intelligence. pp. 703–707 (2009). https://doi.org/10.1109/ICTAI.2009.35
2. Athanasios Tsanas, A.X.: Energy efficiency (2012). https://doi.org/10.24432/C51307, https://archive.ics.uci.edu/dataset/242
3. Benavoli, A., Corani, G., Demšar, J., Zaffalon, M.: Time for a change: a tutorial for comparing multiple classifiers through bayesian analysis. J. Mach. Learn. Res. **18**(1), 2653–2688 (jan 2017)
4. Bouter, A., Alderliesten, T., Witteveen, C., Bosman, P.A.N.: Exploiting linkage information in real-valued optimization with the real-valued gene-pool optimal mixing evolutionary algorithm. In: Proceedings of the Genetic and Evolutionary Computation Conference. p. 705–712. GECCO '17, Association for Computing Machinery, New York, NY, USA (2017). https://doi.org/10.1145/3071178.3071272, https://doi.org/10.1145/3071178.3071272
5. Cerny, B.M., Nelson, P.C., Zhou, C.: Using differential evolution for symbolic regression and numerical constant creation. In: Proceedings of the 10th annual conference on Genetic and evolutionary computation. GECCO08, ACM (Jul 2008). https://doi.org/10.1145/1389095.1389331, http://dx.doi.org/10.1145/1389095.1389331
6. Dushatskiy, A., Virgolin, M., Bouter, A., Thierens, D., Bosman, P.A.N.: Parameterless gene-pool optimal mixing evolutionary algorithms (2021). https://doi.org/10.48550/ARXIV.2109.05259, https://arxiv.org/abs/2109.05259
7. Fisher, R.A.: On the interpretation of $\mathcal{X}^2$ from contingency tables, and the calculation of p. Journal of the Royal Statistical Society **85**(1), 87 (Jan 1922). https://doi.org/10.2307/2340521, http://dx.doi.org/10.2307/2340521
8. Gronau, I., Moran, S.: Optimal implementations of upgma and other common clustering algorithms. Information Processing Letters **104**(6), 205–210 (Dec 2007). https://doi.org/10.1016/j.ipl.2007.07.002, http://dx.doi.org/10.1016/j.ipl.2007.07.002
9. Harrison, J., Virgolin, M., Alderliesten, T., Bosman, P.: Mini-batching, gradient-clipping, first- versus second-order: What works in gradient-based coefficient optimisation for symbolic regression? In: Proceedings of the Genetic and Evolutionary Computation Conference. p. 1127–1136. GECCO '23, Association for Computing Machinery, New York, NY, USA (2023). https://doi.org/10.1145/3583131.3590368, https://doi.org/10.1145/3583131.3590368
10. Howard, L., D'Angelo, D.: The ga-p: a genetic algorithm and genetic programming hybrid. IEEE Expert **10**(3), 11–15 (Jun 1995). https://doi.org/10.1109/64.393137, http://dx.doi.org/10.1109/64.393137
11. I-Cheng Yeh: Concrete compressive strength (1998). https://doi.org/10.24432/C5PK67, https://archive.ics.uci.edu/dataset/165
12. J. Gerritsma, R.O.: Yacht hydrodynamics (1981). https://doi.org/10.24432/C5XG7R, https://archive.ics.uci.edu/dataset/243
13. Keijzer, M.: Scaled symbolic regression. Genetic Programming and Evolvable Machines **5**(3), 259–269 (Sep 2004). https://doi.org/10.1023/b:genp.0000030195.77571.f9, http://dx.doi.org/10.1023/B:GENP.0000030195.77571.f9

14. Kommenda, M., Burlacu, B., Kronberger, G., Affenzeller, M.: Parameter identification for symbolic regression using nonlinear least squares. Genetic Programming and Evolvable Machines **21**(3), 471–501 (Dec 2019). https://doi.org/10.1007/s10710-019-09371-3, http://dx.doi.org/10.1007/s10710-019-09371-3
15. Koza, J.: Genetic programming as a means for programming computers by natural selection. Statistics and Computing **4**(2) (Jun 1994). https://doi.org/10.1007/bf00175355, http://dx.doi.org/10.1007/BF00175355
16. Kronberger, G.: Local optimization often is ill-conditioned in genetic programming for symbolic regression. In: 2022 24th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC). pp. 304–310 (2022). https://doi.org/10.1109/SYNASC57785.2022.00055
17. La Cava, W., Orzechowski, P., Burlacu, B., de Franca, F., Virgolin, M., Jin, Y., Kommenda, M., Moore, J.: Contemporary symbolic regression methods and their relative performance. In: Vanschoren, J., Yeung, S. (eds.) Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks. vol. 1. Curran (2021), https://datasets-benchmarks-proceedings.neurips.cc/paper_files/paper/2021/file/c0c7c76d30bd3dcaefc96f40275bdc0a-Paper-round1.pdf
18. Lundberg, S.M., Lee, S.I.: A unified approach to interpreting model predictions. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) Advances in Neural Information Processing Systems 30, pp. 4765–4774. Curran Associates, Inc. (2017), http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf
19. Meurer, A., Smith, C.P., Paprocki, M., Čertík, O., Kirpichev, S.B., Rocklin, M., Kumar, A., Ivanov, S., Moore, J.K., Singh, S., Rathnayake, T., Vig, S., Granger, B.E., Muller, R.P., Bonazzi, F., Gupta, H., Vats, S., Johansson, F., Pedregosa, F., Curry, M.J., Terrel, A.R., Roučka, v., Saboo, A., Fernando, I., Kulal, S., Cimrman, R., Scopatz, A.: Sympy: symbolic computing in python. PeerJ Computer Science **3**, e103 (Jan 2017). https://doi.org/10.7717/peerj-cs.103, https://doi.org/10.7717/peerj-cs.103
20. Mukherjee, S., Eppstein, M.J.: Differential evolution of constants in genetic programming improves efficacy and bloat. In: Proceedings of the 14th Annual Conference Companion on Genetic and Evolutionary Computation. p. 625–626. GECCO '12, Association for Computing Machinery, New York, NY, USA (2012). https://doi.org/10.1145/2330784.2330891, https://doi.org/10.1145/2330784.2330891
21. Nicolau, M., Agapitos, A.: Choosing function sets with better generalisation performance for symbolic regression models. Genetic Programming and Evolvable Machines **22**(1), 73–100 (May 2020). https://doi.org/10.1007/s10710-020-09391-4, http://dx.doi.org/10.1007/s10710-020-09391-4
22. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: PyTorch: an imperative style, high-performance deep learning library. Curran Associates Inc., Red Hook, NY, USA (2019)
23. Rockett, P.: Constant optimization and feature standardization in multiobjective genetic programming. Genetic Programming and Evolvable Machines **23**(1), 37–69 (Aug 2021). https://doi.org/10.1007/s10710-021-09410-y, http://dx.doi.org/10.1007/s10710-021-09410-y

24. Rudin, C.: Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. Nature Machine Intelligence **1**(5), 206–215 (May 2019). https://doi.org/10.1038/s42256-019-0048-x, http://dx.doi.org/10.1038/s42256-019-0048-x
25. Sadowski, K.L., Thierens, D., Bosman, P.A.: Gambit: A parameterless model-based evolutionary algorithm for mixed-integer problems. Evolutionary Computation **26**(1), 117–143 (2018). https://doi.org/10.1162/evco_a_00206
26. Sharman, K.: Evolving signal processing algorithms by genetic programming. In: 1st International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA). IEE (1995). https://doi.org/10.1049/cp:19951094, http://dx.doi.org/10.1049/cp:19951094
27. Sijben, E.M.C., Alderliesten, T., Bosman, P.A.N.: Multi-modal multi-objective model-based genetic programming to find multiple diverse high-quality models. In: Proceedings of the Genetic and Evolutionary Computation Conference. p. 440–448. GECCO '22, Association for Computing Machinery, New York, NY, USA (2022). https://doi.org/10.1145/3512290.3528850, https://doi.org/10.1145/3512290.3528850
28. Thomas Brooks, D.P.: Airfoil self-noise (1989). https://doi.org/10.24432/C5VW2C, https://archive.ics.uci.edu/dataset/291
29. Virgolin, M., Alderliesten, T., Witteveen, C., Bosman, P.A.N.: Improving model-based genetic programming for symbolic regression of small expressions. Evolutionary Computation **29**(2), 211–237 (2021)
30. Virgolin, M., Bosman, P.A.N.: Coefficient mutation in the gene-pool optimal mixing evolutionary algorithm for symbolic regression. In: Proceedings of the Genetic and Evolutionary Computation Conference Companion. p. 2289–2297. GECCO '22, Association for Computing Machinery, New York, NY, USA (2022). https://doi.org/10.1145/3520304.3534036, https://doi.org/10.1145/3520304.3534036
31. Virgolin, M., Medvet, E., Alderliesten, T., Bosman, P.A.N.: Less is more: A call to focus on simpler models in genetic programming for interpretable machine learning (2022). https://doi.org/10.48550/ARXIV.2204.02046, https://arxiv.org/abs/2204.02046