Bias-Variance Decomposition: An Effective Tool to Improve Generalization of Genetic Programming-based Evolutionary Feature Construction for Regression

Hengzhe Zhang
hengzhe.zhang@ecs.vuw.ac.nz
Victoria University of Wellington
Wellington, New Zealand

Qi Chen
qi.chen@ecs.vuw.ac.nz
Victoria University of Wellington
Wellington, New Zealand

Bing Xue
bing.xue@ecs.vuw.ac.nz
Victoria University of Wellington
Wellington, New Zealand

Wolfgang Banzhaf
banzhafw@msu.edu
Michigan State University
East Lansing, MI, USA

Mengjie Zhang
mengjie.zhang@ecs.vuw.ac.nz
Victoria University of Wellington
Wellington, New Zealand

ABSTRACT

Evolutionary feature construction is a technique that has been widely studied in the domain of automated machine learning. A key challenge that needs to be addressed in feature construction is its tendency to overfit the training data. Instead of the traditional approach to control overfitting by reducing model complexity, this paper proposes to control overfitting based on bias-variance decomposition. Specifically, this paper proposes reducing the variance of a model, i.e., reducing the variance of predictions when exposed to data with injected noise, to improve its generalization performance within a multi-objective optimization framework. Experiments conducted on 42 datasets demonstrate that the proposed method effectively controls overfitting and outperforms six model complexity measures for overfitting control. Moreover, further analysis reveals that controlling overfitting adhering to bias-variance decomposition outperforms several plausible variants, highlighting the importance of controlling overfitting based on solid machine learning theory.

CCS CONCEPTS

• **Computing methodologies** → **Genetic programming**.

KEYWORDS

Genetic programming, bias-variance decompostion, automated machine learning, evolutionary feature construction

ACM Reference Format:
Hengzhe Zhang, Qi Chen, Bing Xue, Wolfgang Banzhaf, and Mengjie Zhang. 2024. Bias-Variance Decomposition: An Effective Tool to Improve Generalization of Genetic Programming-based Evolutionary Feature Construction for Regression. In *Proceedings of The Genetic and Evolutionary Computation Conference 2024 (GECCO '24).* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3638529.3654075

GECCO '24, July 14–18, 2024, Melbourne, VIC, Australia

ACM ISBN 979-8-4007-0494-9/24/07…$15.00
https://doi.org/10.1145/3638529.3654075

1 INTRODUCTION

Automated feature construction is an important topic in the field of automated machine learning, attracting considerable attention in recent years [38]. Formally, given a dataset (X, Y), the goal of automated feature construction is to construct a set of features $\Phi(X)$ to improve the predictive performance of a specific machine learning algorithm $\mathcal{A}$. Representative examples of automated feature construction techniques include neural networks [22], kernel methods [34] and genetic programming [19].

Despite the remarkable success of neural network-based feature construction methods, genetic programming (GP)-based evolutionary feature construction has been gaining increasing attention [1]. Compared to deep learning methods, GP has advantages such as an interpretable representation [17] and a gradient-free and global search mechanism, making it well-suited for automated feature construction, especially for constructing features for non-differentiable loss functions and base learners.

In current GP feature construction algorithms, a significant challenge is that the constructed features may overfit the training data, especially when sample size is limited or the dataset contains noise [31]. In GP-based learning, a common strategy to enhance generalization is controlling the size of the GP model [10], thereby reducing the risk of overfitting. However, other studies indicate that model size alone is insufficient to ensure good generalization [37]. In contrast, the semantics of GP trees significantly impact generalization performance [37]. Therefore, complexity measures such as VC dimension [7], Rademacher complexity [5], and model smoothness [36] are considered for optimization to improve the generalization of GP.

In the context of feature construction, it has been observed that large feature construction layers in deep neural networks can generalize effectively [42]. However, for a ReLu neural network with L layers and W parameters, the VC dimension has a lower bound of $\Omega(WL\log(W/L))$ [2], often exceeding the dataset size in modern neural networks. This observation raises questions about the practicality of avoiding models with high VC dimension and Rademacher

complexity in mitigating overfitting. In contrast, bias-variance analysis successfully demonstrates that large models can generalize well because of variance reduction introduced into large regularized neural networks, such as L2-regularized networks [40].

Given the success of the bias-variance decomposition framework in analyzing overfitting phenomena, the key objective of this paper is to propose a variance reduction technique for enhancing the generalization performance of GP-based feature construction within the context of bias-variance decomposition. More specifically, when using mean square error as the loss function, the test error on unseen data can be decomposed as [18]:

$$\mathbb{E}_{\mathcal{D}}\left[\{f(\mathbf{x_{test}};\mathcal{D})-y_{test}\}^2\right]=\underbrace{\{\mathbb{E}_{\mathcal{D}}[f(\mathbf{x_{test}};\mathcal{D})]-y_{test}\}^2}_{\text{Bias}} + \underbrace{\mathbb{E}_{\mathcal{D}}\left[\{f(\mathbf{x_{test}};\mathcal{D})-\mathbb{E}_{\mathcal{D}}[f(\mathbf{x_{test}};\mathcal{D})]\}^2\right]}_{\text{Variance}}. \tag{1}$$

Here, $\mathcal{D}$ represents the data distribution, and $\mathbb{E}_{\mathcal{D}}$ represents the expectation over the data sampling process. The first objective O_1 represents the squared error between the average prediction over models trained by bootstrapped training data and the ground truth, while O_2 represents the variance among different predictions over models trained by bootstrapped training data. The bias-variance decomposition is a general framework applicable to all learning algorithms. Commonly, in the GP domain, the bias-variance decomposition is used to analyze the behavior of a GP algorithm [29]. This means that the learning algorithm $\mathcal{A}$ will produce different models f through sampling from the training data X, and the bias-variance decomposition is performed on the test data $\mathbf{x_{test}}$ based on these different trained models f. The entire process of classical bias-variance decomposition [18] is illustrated in Figure 1. In contrast, as depicted in Figure 2, this paper focuses on the variance of a specific model f on the training data $\mathbf{x_{train}}$. Our aim is to obtain a robust model f that yields stable predictions with minor variations on the training data X, rather than developing a robust algorithm $\mathcal{A}$. The objective is defined in Equation (2):

$$\text{minimize}\begin{cases} O_1=\underbrace{\{\mathbb{E}_{\mathcal{D}}[f(\mathbf{x_{train}};\mathcal{D})]-y_{train}\}^2}_{\text{Bias}}, \\ O_2=\underbrace{\mathbb{E}_{\mathcal{D}}\left[\{f(\mathbf{x_{train}};\mathcal{D})-\mathbb{E}_{\mathcal{D}}[f(\mathbf{x_{train}};\mathcal{D})]\}^2\right]}_{\text{Variance}}. \end{cases} \tag{2}$$

Here, the first objective O_1 represents the squared error between the average prediction over data with noise and the ground truth, whereas O_2 represents the variance among different predictions over data with noise.

1.1 Goals

To simultaneously optimize bias and variance in Equation (2), we introduce a multi-objective optimization framework for evolutionary feature construction in regression[1]. While multi-objective optimization is not mandatory, it aligns with the common approach in existing literature to control overfitting [5, 7]. The main goals of this paper are summarized as follows:

- To address the problem of overfitting, we propose a multi-objective feature construction framework that optimizes both bias and variance of a regression model with GP constructed features, based on the bias-variance decomposition framework.
- To align with the objective function derived from bias variance decomposition, we propose an empirical method to estimate the variance for a fixed learning model.
- To demonstrate the effectiveness of the proposed method, we compare it with six popular overfitting control techniques in GP, as well as standard GP, on a regression benchmark with 42 datasets.

2 RELATED WORK

2.1 Evolutionary Feature Construction

Evolutionary feature construction is an automated machine learning technique that has received widespread attention. It can be categorized into filter-based, wrapper-based, and embedded feature construction methods based on the evaluation approach used [33]. Filter-based methods often utilize information-theory-based or other statistical and fuzzy metrics, such as impurity [24], to evaluate feature quality. The advantage of filter-based methods is their speed and generalizability to different learning algorithms, but they often provide sub-optimal learning performance. In contrast, wrapper-based methods evaluate feature quality using specific learning algorithms, such as decision trees [46] or linear regression [43]. In wrapper-based methods, multi-tree GP is widely employed as a representation for constructing multiple features, with notable examples including M3GP [23] and M4GP [21]. Wrapper-based methods are typically slower than filter-based methods, but they can achieve superior learning performance. Finally, embedded feature construction methods integrate feature construction into the learning process, with GP-based symbolic regression methods being a typical example [6]. Embedded methods often provide a compromise on learning performance and training speed between filter-based methods and wrapper-based methods. In this paper, our main focus is on wrapper-based feature construction methods because of their superior performance.

2.2 Overfitting Control in GP

Regarding overfitting control techniques in GP, it can be broadly categorized into three categories. The first category is optimizing from a statistical machine learning theory perspective, including techniques like Tikhonov regularization [26], VC dimension [7], Rademacher complexity [5], Bayesian model selection [3], and also includes implicit model size reduction methods like equality saturation [11] and hoist mutation [44]. Although these methods are underpinned by strong theoretical foundations, the success of modern deep learning techniques, particularly in large computer vision and language models, challenges the applicability of these measures. For instance, large computer vision model architectures can fit datasets with both correct and shuffled labels [42], yet they demonstrate robust generalization capabilities. This contradicts the theory of Rademacher complexity.

The second category is based on bias-variance decomposition theory, with a notable example being ensemble GP. Recent studies

[1]Souce Code: https://github.com/hengzhe-zhang/EvolutionaryForest/blob/master/experiment/methods/VR_GP.py

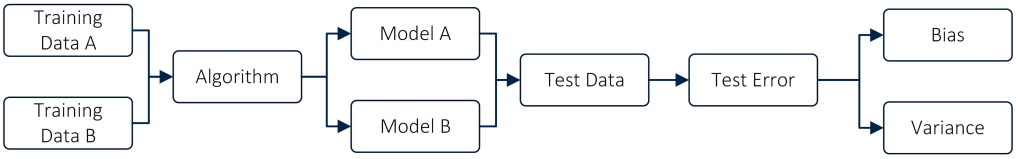

Figure 1: The traditional workflow of bias-variance decomposition for decomposing the test error into bias and variance. The variance arises from various models induced by a learning algorithm trained on different datasets [18] and random seeds [29].

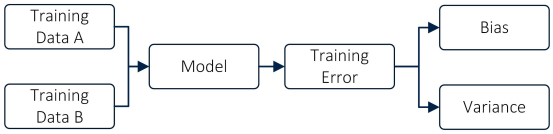

Figure 2: The workflow of bias-variance decomposition for overfitting control without involving the test data. The model refers to GP-constructed features with a base regression model. The variance arises from a model that makes different predictions for data with different minor variations.

demonstrate that an ensemble of GP-constructed features, using either homogeneous [46] or heterogeneous base learners [45], can yield impressive generalization performance. However, in many scenarios where interpretability is paramount, a single model is often preferred.

The third category is applying commonly used strategies in machine learning, such as early stopping [35], random sampling [15], semi-supervised learning [32], feature selection [39], and soft targets [36]. However, these methods generally lack a solid theoretical foundation, making their applicability to various scenarios less certain.

Given the success of bias-variance decomposition in explaining the generalization of deep learning techniques [40], as well as GP techniques [30], this paper takes a further step to optimize the variance of a model to control overfitting.

3 VARIANCE REDUCTION GP

3.1 Individual Representation

In this paper, a multi-tree GP representation is used for feature construction. Here, the number of trees m can be dynamically changed by the mutation operator, as illustrated in Figure 3. All GP trees in a GP individual, denoted as $\Phi = \{\phi_1, \ldots, \phi_m\}$, are employed to construct m features from input data X. These features are then input into a linear regression model LM to generate predictions $\hat{Y}$.

3.2 Algorithm Framework

The algorithm framework follows the traditional GP framework. However, to consider optimizing variance, we employ a multi-objective framework to evolve GP trees. The framework consists of the following five components:

- Population Initialization: In the initialization stage, a set of N GP individuals is randomly initialized, each with a single GP tree. However, these GP individuals are extendable, meaning that the number of trees in each individual can be extended through mutation operations.
- Parent Selection: In the selection phase, the parent selection operator selects several parents for crossover and mutation. In this paper, the lexicase selection operator is used because of its superiority in preserving population diversity [16]. To select one individual from a population, lexicase selection randomly selects one training instance k to eliminate individuals based on a threshold $\mathcal{L}_k(p) < \min_{p' \in P} \mathcal{L}_k(p') + \epsilon_k$, where $min_{p' \in P} \mathcal{L}_k(p')$ represents the minimum squared error achieved by the best individual, and ϵ_k represents the mean absolute deviation [20]. The elimination process repeats until only one individual remains, which is then selected as the parent.
- Offspring Generation: Once parents are selected, random subtree crossover and mutation operators are applied to generate offspring. Moreover, random subtree addition and random subtree deletion [23] are invoked according to a probability to add or delete one tree unless the upper or lower limit is reached.
- Objective Evaluations: In this stage, each GP individual Φ constructs features $\Phi(X)$ using the training data X. To ensure that these features generalize well on unseen data, they are evaluated with ridge regression using a leave-one-out cross-validation strategy. In addition to the cross-validation loss, we evaluate the variance of the constructed features on a linear model LM using the method proposed in Section 3.3. The cross-validation loss and variance compose the two objectives.
- Survival Selection: After objective evaluations, survival selection is performed on the combined pool of parents and offspring. For a population of $2N$ individuals, non-dominated sorting with crowding distance [12] is used to rank individuals and reduce the population to N.
- Archive Maintenance: An archive is maintained, and the individual with the smallest sum of the two objectives, $O_1(\Phi) + O_2(\Phi)$, is considered the historically best individual to be stored and preserved for making final predictions.

The processes of parent selection, offspring generation, objective evaluations and environmental selection are repeated until the maximum generation is reached.

3.3 Variance Estimation

3.3.1 Challenges in Variance Estimation. Currently, bias-variance decomposition frameworks mainly rely on a bootstrapping strategy to estimate variance [14]. This approach is feasible for estimating

Figure 3: GP individual with a variable number of trees.

the variance of a learning algorithm but is not suitable for estimating the variance for a specific model. When estimating the variance of a learning algorithm, different sets of bootstrapping sampling data $X_{bootstrap}$ produce different models $f_{bootstrap}$, resulting in variance on test data X_{test}. However, for the variance of a specific model f_{train} on training data, there is no clear definition of the variance term when using the bootstrapping method. One way to circumvent this issue is to optimize surrogate variance, as shown in Equation (3), and then use bootstrapping [14]:

$$\mathrm{Var}_{\mathcal{D}}\left[\left\{f(\mathbf{x_{bootstrap}};\mathcal{D})-Y\right\}^2\right] \tag{3}$$

The intuition behind this equation is to treat the variance of squared errors $\left\{f(\mathbf{x_{bootstrap}};\mathcal{D})-Y\right\}^2$ as the variance. Clearly, there is a discrepancy between optimizing Equation (3) and the actual variance in Equation (2). Thus, in this section, we propose a novel way to empirically estimate the variance of a model.

3.3.2 Empirical Estimation. To empirically estimate the variance of a GP individual, we assume that the unseen data is sampled from the same distribution as the training data but with some different values. Formally, we consider the test data to be $X+\epsilon$, where ϵ is random noise sampled from a Gaussian distribution $\mathcal{N}(0,\sigma)$. The empirical bias and variance of Equation (1) can be formulated as O_1 and O_2:

$$\text{minimize}\begin{cases} O_1(\Phi)=\frac{1}{|X|}\sum_{x\in X}\left(\frac{1}{K}LM(\Phi(x+\epsilon))-Y\right)^2, \\ O_2(\Phi)=\frac{1}{|X|}\sum_{x\in X}\left(LM(\Phi(x+\epsilon))-\frac{1}{K}LM(\Phi(x+\epsilon))\right)^2. \end{cases} \tag{4}$$

where K is the number of iterations to sample noise from a Gaussian distribution for reliable variance estimation. The goal of this paper is not to construct an ensemble model in which the average prediction, $\frac{1}{K}LM(\Phi(x+\epsilon))$, approximates the targets Y. Thus, in this paper, we replace the expectation of model outputs on noisy data, $\frac{1}{K}LM(\Phi(x+\epsilon))$, with predictions on clean data $LM(\Phi(x))$ to implicitly encourage the expectation of model outputs on noisy data $\mathbb{E}_{\mathcal{D}}[f(\mathbf{x_{train}};\mathcal{D})]$ to optimize predictions on clean data $LM(\Phi(x))$, resulting in the following objectives:

$$\text{minimize}\begin{cases} O_1(\Phi)=\frac{1}{|X|}\sum_{x\in X}\left(LM(\Phi(x))-Y\right)^2, \\ O_2(\Phi)=\frac{1}{|X|}\sum_{x\in X}\left(LM(\Phi(x+\epsilon))-LM(\Phi(x))\right)^2. \end{cases} \tag{5}$$

Based on Equation (5), the algorithm framework is outlined in Algorithm 1.

- Model Training (Lines 2-3): Initially, features $\Phi(X)$ are constructed from the original features X and GP trees Φ. A linear model LM is then fitted to the constructed features $\Phi(X)$ to make predictions $\hat{Y}$. After this stage, both GP trees and the linear model will be fixed, and only the data will change.

Algorithm 1 Variance Estimation

Require: GP Tree Φ, Inputs X, Target Outputs Y, Linear Model LM, Gaussian Noise Standard Deviation σ, Number of Iterations K

1: Initialize variance $V \leftarrow 0$
2: $\Phi(X) \leftarrow$ Feature Construction (X, Φ)
3: $LM \leftarrow$ Linear Regression$(\Phi(X))$
4: $\hat{Y} \leftarrow$ Prediction$(LM, \Phi(X))$
5: **for** $k = 1, \ldots, K$ **do**
6: **for** $i = 1, \ldots, N$ **do**
7: $\epsilon \leftarrow$ Sample Noise $(\mathcal{N}(0, \sigma))$
8: $\tilde{X}_i \leftarrow X_i + \epsilon$
9: $\tilde{\Phi}(X_i) \leftarrow$ Feature Construction$(\tilde{X}_i, \Phi)$
10: $\tilde{Y}_i \leftarrow$ Prediction$(LM, \tilde{\Phi}(X_i))$
11: $V_i \leftarrow V_i + (\tilde{Y}_i - \hat{Y}_i)^2$
return Variance $\sum_{i=1}^{N} V_i/(N * K)$

- Model Prediction (Lines 7-10): In each round of variance estimation, noise ϵ is first sampled and then added to the training data X_i, forming the noisy data $X_i+\epsilon$. Subsequently, features $\tilde{\Phi}(X_i)$ are constructed from the noisy data, and the linear model generates predictions $\tilde{Y}_i$ based on these newly constructed features $\tilde{\Phi}(X_i)$.
- Variance Estimation (Line 11): Using the predictions on the original training data $\hat{Y}_i$ and the predictions on the noisy data $\tilde{Y}_i$, the estimated variance based on Equation (5) is obtained.

As mentioned, to ensure reliability, the variance estimation process is repeated K iterations, and the average variance across these K iterations is used as the final variance.

4 EXPERIMENTAL SETTINGS

4.1 Experimental Datasets

The experiments consider only real-world datasets from the PENN machine learning benchmark (PMLB) [28]. PMLB contains 120 regression datasets. After excluding datasets generated by Friedman and those with fewer than 5 features, 42 datasets are selected for the experiments.

4.2 Parameter Settings

The parameter settings are presented in Table 1. Following the convention in GP, a high crossover rate is used for combining building blocks, while a low mutation rate is used to avoid disruption. The sum of the tree addition and tree deletion rates is set to a high value to encourage exploration [23]. To prevent zero-division errors, the analytical quotient (AQ) [25], defined as $AQ(x,y)=\frac{x}{\sqrt{1+y^2}}$, replaces the traditional division operator. Additionally, an analytical log operator, defined as $ALog(x)=\log(\sqrt{x^2+1})$, is used.

4.3 Benchmark Methods

The algorithm proposed in this paper is referred to as the variance reduction (VR) method because it optimizes both the cross-validation score and variance simultaneously. The benchmark methods used in this paper include traditional GP, as well as state-of-the-art techniques for controlling overfitting in GP, which are:

Table 1: Parameter Settings for GP.

Parameter	Value
Maximal Population Size	200
Number of Generations	200
Crossover and Mutation Rates	0.9 and 0.1
Tree Addition Rate	0.5
Tree Deletion Rate	0.5
Initial Tree Depth	0-3
Maximum Tree Depth	10
Initial Number of Trees	1
Maximum Number of Trees	20
Elitism (Number of Individuals)	1
Standard Deviation of Noise	0.5
Iterations of Variance Estimation	5
Functions	+, -, *, AQ, Square, Log, Sqrt, Max, Min, Sin, Cos, Abs, Negative

- Standard GP without regularization: Standard GP relies solely on leave-one-out cross-validation loss as the optimization objective.
- Parsimonious Pressure (PP) [41]: For parsimonious pressure, complexity is determined by the size of all GP trees. In this case, a smaller model size is preferred to enhance generalization.
- Tikhonov Regularization (TK) [26]: This paper employs zero-order Tikhonov regularization. For a function f, it is defined as $|f(x)|$, which regularizes extremely large values.
- Grand Complexity (GC) [26]: Grand Complexity incorporates both zero-order Tikhonov regularization and model size. The dominance rank of these two factors is utilized as the objective value.
- Rademacher Complexity [5]: Rademacher Complexity is a data-dependent metric that measures the capability of a model to fit a given dataset with arbitrary labels. Formally, Rademacher Complexity is defined as:

$$\mathrm{R}_n(\mathcal{L}) = \mathbb{E}\left[\sup_{l\in\mathcal{L}} \frac{1}{n}\sum_{i=1}^{n} \sigma_i l\left(x_i, y_i\right)\right], \tag{6}$$

  where σ_i is the Rademacher variable. A lower Rademacher Complexity indicates a simpler model.
- Weighted MIC between Residuals and Variables (WCRV) [4]: WCRV aims to reduce the correlation between residuals and input features. For those important features, i.e., $\mathrm{MIC}_{x^k,Y} \geq mv$, WCRV minimizes their correlation with residuals. For less important features, i.e., $\mathrm{MIC}_{x^k,Y} < mv$, WCRV minimizes their selection frequency.

$$\begin{aligned}\mathrm{WCRV}(\Phi) = &\sum_{\mathrm{MIC}_{x^k,Y}\geq mv} \mathrm{MIC}_{x^k,Y} \times \mathrm{MIC}_{x^k,R} \\ &+ \sum_{\mathrm{MIC}_{x^k,Y}<mv} \left(1 - \mathrm{MIC}_{x^k,Y}\right).\end{aligned} \tag{7}$$

- Correlation between Input and Output Distances (IODC) [36]: IODC first calculates the pairwise distance between input features and the pairwise distance between output predictions, resulting in two matrices, I and O. The Pearson correlation of these matrices is computed as the IODC value:

$$\mathrm{IODC}(\Phi) = \frac{\mathrm{Cov}(\mathrm{I}, \mathrm{O})}{\sigma_{\mathrm{I}}\sigma_{\mathrm{O}}} \tag{8}$$

  Intuitively, a higher correlation indicates better smoothness between input and output. Therefore, to achieve better generalization, IODC should be maximized.

All baseline methods, except standard GP without regularization, are optimized using the same multi-objective optimization framework. To balance accuracy and complexity, we use the minimum Manhattan distance (MMD)-based knee-point selection method [9] for the final model selection for these baseline methods. In brief, objectives are normalized to $\mathbf{O}_1(\Phi)$ and $\mathbf{O}_2(\Phi)$ since the baseline complexities have different scales compared to mean square error. Then, the Manhattan distance between each point and the extreme points is computed as $\mathbf{O}_1(\Phi) + \mathbf{O}_2(\Phi)$, and the model with the minimum Manhattan distance is selected as the final model. For the standard GP without regularization, the model with the best cross-validation loss is selected as the final model. For a fair comparison, all methods use the same parameter settings as shown in Table 1.

4.4 Evaluation Protocol

To ensure reliable results, experiments are independently conducted using 30 different random seeds on each dataset. For each seed, 100 samples are randomly chosen from the dataset for training, and the remaining samples are used for testing [27]. Before training, all datasets are standardized [30], and categorical variables are encoded using one-hot encoding. When making final predictions, the predictions are clipped according to the maximum Y_{Max} and minimum Y_{Min} targets to prevent extreme predictions. The evaluation metric used is the R^2 score, a normalized metric where 1 represents the optimal result. Formally, it is defined as $1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y})^2}$, where y_i is the predicted value, $\hat{y}_i$ is the true value, and $\bar{y}$ is the average of the true values. To assess statistical significance, the Wilcoxon signed-rank test with a significance level of 0.05 [13] is employed for statistical comparisons.

5 EXPERIMENTAL RESULTS

In this section, we present the experimental results for the proposed method and the benchmark methods on the 42 regression datasets. First, we show the test R^2 scores, followed by the training R^2 scores, to compare the effectiveness of various overfitting control methods. Additionally, we provide a comparison between the tree sizes of the final model and the training time.

5.1 Test Performance

5.1.1 General Analysis. The experimental results for test R^2 scores are presented in Table 2. These results show that GP with the proposed VR technique significantly outperforms other overfitting control techniques. For example, when compared to standard GP, VR significantly improves test R^2 scores on 35 datasets while leading to worse results on only 5 datasets, indicating the effectiveness of

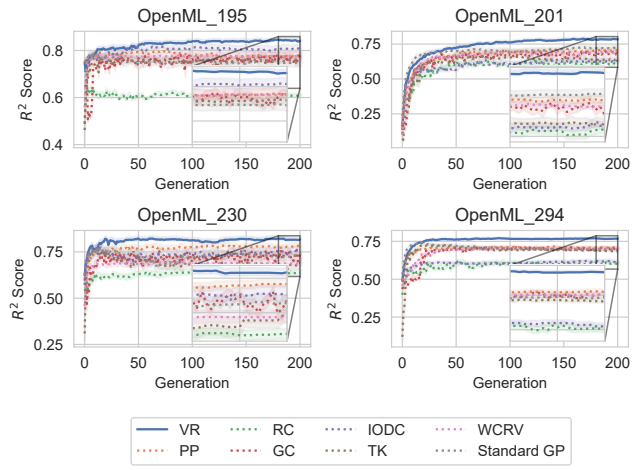

Figure 4: Evolutionary plots of the *test* R^2 *scores* for different complexity measures.

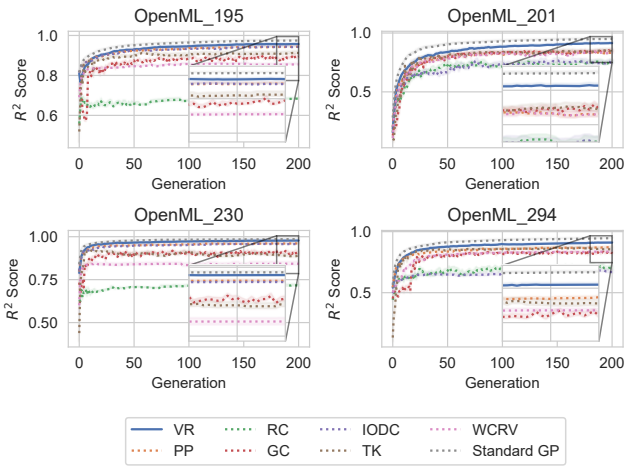

Figure 5: Evolutionary plots of the *training* R^2 *scores* for various complexity measures.

improving test R^2 scores using VR. Furthermore, evolutionary plots of test R^2 scores are provided in Figure 4. From Figure 4, it is evident that on some datasets like "OpenML_294," standard GP overfits after several generations of the evolutionary process, whereas the VR method successfully curbs overfitting and ultimately leads to better generalization performance.

5.1.2 Estimated Variance vs. Model Size. While model size has been shown to be ineffective in controlling overfitting in single-tree-based symbolic regression, for multi-tree-based evolutionary feature construction, model size remains a competitive complexity measure for overfitting control, as shown in Table 2. The reason is that with the help of the base learner, even a set of small features can perform well as long as they are relevant to the target and complementary [44]. A comparison of test R^2 scores between VR and PP in Table 2 shows that VR outperforms PP on 23 datasets and performs significantly worse on 9 datasets. These results indicate that VR is better in more cases despite PP being an effective way to control overfitting in many cases. Evolutionary plots of test R^2 scores in Table 2 provide a deeper understanding of the difference between PP and VR. These plots show that PP is a pessimistic metric for avoiding overfitting, and it restricts GP search to the region of simple models with low R^2 values to prevent overfitting. In comparison, VR is less restrictive and effectively controls overfitting while avoiding excessive restriction.

5.2 Training Performance

5.2.1 General Analysis. To further confirm the occurrence of overfitting, we present the training R^2 scores in Table 3. The results indicate that optimizing solely cross-validation loss can yield better training performance compared to optimizing both cross-validation loss and variance. However, as shown in Figure 4 and Figure 5, optimizing only cross-validation loss tends to lead to overfitted models, whereas optimizing variance typically results in better generalization performance. To further confirm that using VR as an additional objective effectively controls overfitting, i.e., there is no significant decrease in test R^2 scores with the increase of training R^2 scores, the Pearson correlation between training R^2 and test R^2 when using VR is presented in Figure 6. The results show that GP with VR establishes a high correlation between performance on training data and test data, indicating the effectiveness of VR in controlling overfitting.

5.2.2 Comparisons on R^2 Reduction. In order to improve generalization ability, different complexity measures add different inductive biases to select simpler models. Due to these different inductive biases, the reductions in training R^2 vary. As depicted in Figure 5, all complexity measures reduce training performance below the level achieved by standard GP. However, when checking these results together with those in Figure 4, it becomes apparent that the inductive biases added by the complexity measures, other than VR, seem too strong in favoring underfitting models, resulting in both training R^2 and test R^2 stay at low levels. In comparison, VR is a more effective strategy for controlling overfitting in GP compared to traditional complexity measures, as it only introduces moderate regularization in training accuracy to avoid underfitting while still effectively managing overfitting.

5.3 Tree Size

In the GP domain, several studies have already recognized that controlling overfitting involves more than merely controlling tree size [37]; it also requires considering semantic complexity. The results regarding tree size, as presented in Figure 7, further confirm this. In this paper, we define tree size as the number of nodes in an individual. As illustrated in the figure, VR does not significantly reduce tree size compared to standard GP. In fact, the median tree sizes of VR and standard GP are 81 and 86, respectively, suggesting that the tree sizes induced by VR and standard GP are not much different. Combining these results with those shown in Table 2, we confirm that reducing tree size is not always necessary to mitigate overfitting. Instead, semantic complexity plays an important role in controlling overfitting, especially the variance of semantics under noise as shown in this paper.

Table 2: Statistical comparison of *test* R^2 *scores* when optimizing various model complexity measures. ("+","~", and "-" indicate that using the method in a row is better than, similar to, or worse than using the method in a column, respectively.)

	PP	RC	GC	IODC	TK	WCRV	Standard GP
VR	23(+)/10(~)/9(-)	32(+)/10(~)/0(-)	23(+)/15(~)/4(-)	31(+)/10(~)/1(-)	35(+)/5(~)/2(-)	22(+)/14(~)/6(-)	35(+)/2(~)/5(-)
PP	—	24(+)/13(~)/5(-)	17(+)/17(~)/8(-)	18(+)/18(~)/6(-)	18(+)/24(~)/0(-)	17(+)/18(~)/7(-)	28(+)/10(~)/4(-)
RC	—	—	6(+)/11(~)/25(-)	11(+)/15(~)/16(-)	9(+)/13(~)/20(-)	5(+)/12(~)/25(-)	17(+)/5(~)/20(-)
GC	—	—	—	22(+)/17(~)/3(-)	20(+)/20(~)/2(-)	14(+)/23(~)/5(-)	20(+)/15(~)/7(-)
IODC	—	—	—	—	11(+)/20(~)/11(-)	12(+)/14(~)/16(-)	19(+)/10(~)/13(-)
TK	—	—	—	—	—	6(+)/23(~)/13(-)	16(+)/16(~)/10(-)
WCRV	—	—	—	—	—	—	17(+)/15(~)/10(-)

Table 3: Statistical comparison of *training* R^2 *scores* when optimizing various complexity measures.

	PP	RC	GC	IODC	TK	WCRV	Standard GP
VR	28(+)/8(~)/6(-)	42(+)/0(~)/0(-)	32(+)/7(~)/3(-)	25(+)/14(~)/3(-)	25(+)/10(~)/7(-)	35(+)/6(~)/1(-)	0(+)/0(~)/42(-)
PP	—	40(+)/2(~)/0(-)	30(+)/10(~)/2(-)	20(+)/16(~)/6(-)	20(+)/11(~)/11(-)	35(+)/6(~)/1(-)	0(+)/0(~)/42(-)
RC	—	—	0(+)/3(~)/39(-)	1(+)/3(~)/38(-)	0(+)/1(~)/41(-)	1(+)/11(~)/30(-)	0(+)/0(~)/42(-)
GC	—	—	—	9(+)/16(~)/17(-)	5(+)/19(~)/18(-)	28(+)/9(~)/5(-)	0(+)/0(~)/42(-)
IODC	—	—	—	—	10(+)/14(~)/18(-)	30(+)/6(~)/6(-)	0(+)/0(~)/42(-)
TK	—	—	—	—	—	31(+)/7(~)/4(-)	0(+)/0(~)/42(-)
WCRV	—	—	—	—	—	—	0(+)/0(~)/42(-)

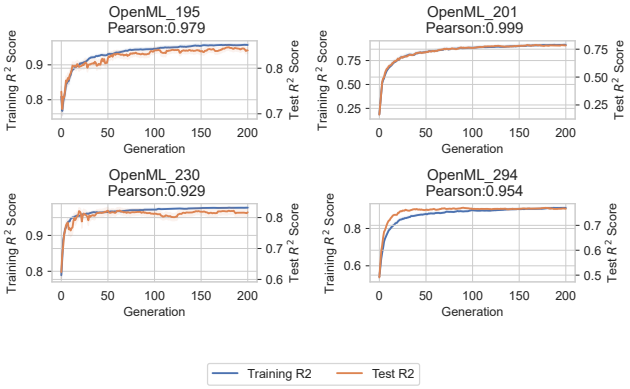

Figure 6: Evolutionary plots of the *training and test* R^2 *scores* for VR.

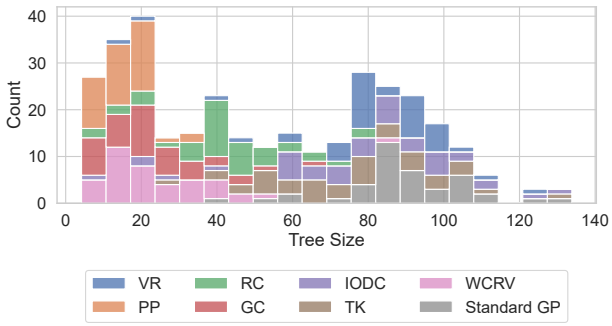

Figure 7: Distribution of *tree sizes* over the 42 datasets when optimizing different model complexity measures.

5.4 Training Time

A comparison of training times between using VR and other algorithms is presented in Figure 8. Calculating the variance of GP trees is more time-consuming than traditional complexity measures, such as parsimonious pressure, because semantics on the data with noise need to be computed. The training time for VR centers around 1950 seconds, while that for GP without regularization concentrates around 574 seconds. This indicates that the training time with VR is three times longer than that of training without regularization. However, it is important to note that effectively addressing overfitting cannot be achieved solely by increasing the training time. Considering the improvements in generalization performance, the increase in consumed time is still acceptable.

6 FURTHER ANALYSIS

In this section, we conduct a further analysis of the proposed algorithm. Specifically, we present two alternative loss functions designed without the guidance of bias-variance decomposition to highlight the importance of bias-variance decomposition-guided overfitting control.

6.1 Variance based on Ground Truth (VGT)

In this paper, we adhere to the bias-variance decomposition framework. A practical question arises: Can we replace the target in the variance term, which is the GP predictions $LM(\Phi(x))$, with the ground truth Y, because $LM(\Phi(x))$ is not the actual target, whereas

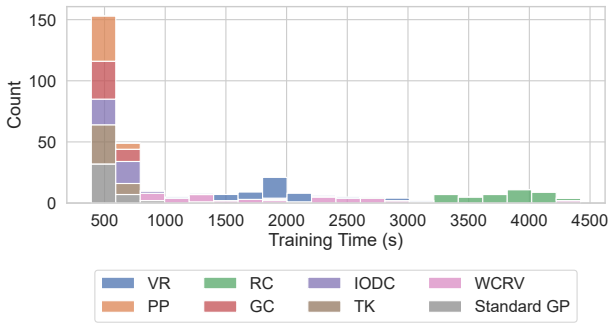

Figure 8: Distribution of *training time* over the 42 datasets when optimizing different complexity measures.

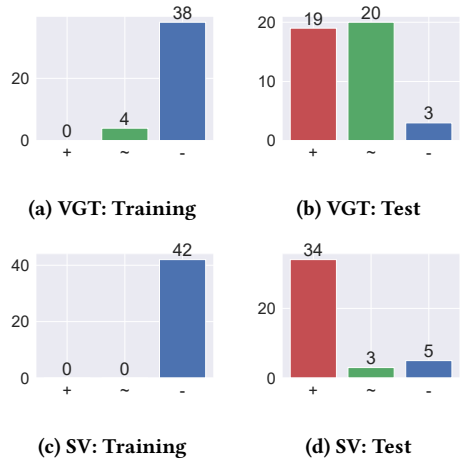

Figure 9: Statistical comparison of *training* R^2/*test* R^2 *scores* between VR and VGT/SV. "+"/"~"/"-" indicate VR outperforms VGS/SV on the corresponding number of datasets.

the ground truth Y is. This is formally defined as follows:

$$\text{minimize} \begin{cases} O_1(\Phi) = \frac{1}{|X|} \sum_{x \in X} (LM(\Phi(x)) - Y)^2 \\ O_2(\Phi) = \frac{1}{|X|} \sum_{x \in X} (LM(\Phi(x + \epsilon)) - Y)^2 \end{cases} \tag{9}$$

This approach appears reasonable, as it would be beneficial if the predictions on noisy data $LM(\Phi(x + \epsilon))$ approximating the ground truth Y. The experimental results for training and test R^2 scores are presented in Figure 9a and Figure 9b, respectively. The test R^2 scores show that replacing $LM(\Phi(x))$ with Y in Equation (5) significantly decreases test R^2 scores on 19 datasets, while it improves performance on only 3 datasets. Regarding training R^2 scores, using Y as the target leads to higher training R^2 scores. These combined results indicate that overfitting occurs when using Y as the target. In fact, the two objectives in Equation (9) are highly correlated, mainly reflecting bias rather than variance. In other words, individuals with lower $(LM(\Phi(x)) - Y)^2$ could also have lower $(LM(\Phi(x + \epsilon)) - Y)^2$. Therefore, the training loss has been overemphasized, resulting in the final model overfitting the training data.

6.2 Surrogate Variance (SV)

To estimate variance through bootstrapping, one possible solution is to use a surrogate variance [14], as shown in Equation (10).

$$\text{minimize} \begin{cases} O_1(\Phi) = \frac{1}{|X|} \sum_{x \in X} (LM(\Phi(x)) - Y)^2 \\ O_2(\Phi) = \text{STD}_{X^*}(\frac{1}{|X|} \sum_{x \in X^*} (LM(\Phi(x)) - Y)^2) \end{cases} \tag{10}$$

Here, STD_{X^*} represents the standard deviation of errors across the bootstrap samples of X. The final model is determined based on the multiplication of two objectives, i.e., $\arg\min_\Phi O_1(\Phi) \times O_2(\Phi)$ [14]. The experimental results are presented in Figure 9c and Figure 9d, respectively. These results reveal that using surrogate variance leads to a significant overfitting problem, where training R^2 scores improve but test R^2 scores decrease. It is clear that the gap between surrogate variance and real variance cannot be ignored. Our proposed VR method, which measures the actual variance, significantly outperforms the surrogate variance in terms of overfitting control.

Based on the results from these two variants, the advantage of the proposed variance estimation method, with a theoretical basis, compared to empirically designed optimization objectives, has been further confirmed.

7 CONCLUSIONS

This paper proposes optimizing the variance of the regression outputs within a bias-variance decomposition framework to mitigate the overfitting issue in GP-based evolutionary feature construction methods for regression tasks. The experimental results across 42 regression datasets demonstrate that optimizing based on bias-variance decomposition yields superior performance compared to standard GP, as well as six overfitting control methods in GP. The results confirm that controlling overfitting is more about controlling variance rather than simply reducing tree size. Moreover, further analysis highlights the importance of reducing variance based on bias-variance decomposition theory, as it outperforms several empirically designed optimization objectives for overfitting control.

For future work, it is worth investigating the effectiveness of the proposed method in classification or other learning problems. Careful derivation of bias-variance decomposition for other loss functions should be undertaken in advance. Another avenue is to employ a reference-guided evolutionary algorithm [8] to search in the interested region, which could potentially strike a better balance between the two objectives while saving computational resources.

ACKNOWLEDGMENTS

This work was supported in part by the Marsden Fund of New Zealand Government under Contract VUW1913, Contract VUW1914, and Contract VUW2016; in part by the Science for Technological Innovation Challenge (SfTI) Fund under Grant E3603/2903; in part by the MBIE Data Science SSIF Fund under Contract RTVU1914; in part by Huayin Medical under Grant E3791/4165; and in part by the MBIE Endeavor Research Programme under Contract C11X2001 and Contract UOCX2104.

REFERENCES

[1] Harith Al-Sahaf, Ying Bi, Qi Chen, Andrew Lensen, Yi Mei, Yanan Sun, Binh Tran, Bing Xue, and Mengjie Zhang. 2019. A survey on evolutionary machine learning. *Journal of the Royal Society of New Zealand* 49, 2 (2019), 205–228.

[2] Peter L Bartlett, Nick Harvey, Christopher Liaw, and Abbas Mehrabian. 2019. Nearly-tight VC-dimension and pseudodimension bounds for piecewise linear neural networks. *The Journal of Machine Learning Research* 20, 1 (2019), 2285–2301.

[3] Geoffrey F Bomarito, Patrick E Leser, NCM Strauss, Karl M Garbrecht, and Jacob D Hochhalter. 2022. Bayesian model selection for reducing bloat and overfitting in genetic programming for symbolic regression. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. 526–529.

[4] Qi Chen, Bing Xue, and Mengjie Zhang. 2020. Improving symbolic regression based on correlation between residuals and variables. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*. 922–930.

[5] Qi Chen, Bing Xue, and Mengjie Zhang. 2022. Rademacher Complexity for Enhancing the Generalization of Genetic Programming for Symbolic Regression. *IEEE Transactions on Cybernetics* 52, 4 (2022), 2382–2395.

[6] Qi Chen, Mengjie Zhang, and Bing Xue. 2017. Feature selection to improve generalization of genetic programming for high-dimensional symbolic regression. *IEEE Transactions on Evolutionary Computation* 21, 5 (2017), 792–806.

[7] Qi Chen, Mengjie Zhang, and Bing Xue. 2018. Structural risk minimization-driven genetic programming for enhancing generalization in symbolic regression. *IEEE Transactions on Evolutionary Computation* 23, 4 (2018), 703–717.

[8] Ran Cheng, Yaochu Jin, Markus Olhofer, and Bernhard Sendhoff. 2016. A reference vector guided evolutionary algorithm for many-objective optimization. *IEEE Transactions on Evolutionary Computation* 20, 5 (2016), 773–791.

[9] Wei-Yu Chiu, Gary G Yen, and Teng-Kuei Juan. 2016. Minimum manhattan distance approach to multiple criteria decision making in multiobjective optimization problems. *IEEE Transactions on Evolutionary Computation* 20, 6 (2016), 972–985.

[10] Fabrício Olivetti de França. 2023. Alleviating overfitting in transformation-interaction-rational symbolic regression with multi-objective optimization. *Genetic Programming and Evolvable Machines* 24, 2 (2023), 13.

[11] Fabricio Olivetti de Franca and Gabriel Kronberger. 2023. Reducing Overparameterization of Symbolic Regression Models with Equality Saturation. In *Proceedings of the Genetic and Evolutionary Computation Conference*. 1064–1072.

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

[13] Junlan Dong, Jinghui Zhong, Wei-Neng Chen, and Jun Zhang. 2022. An efficient federated genetic programming framework for symbolic regression. *IEEE Transactions on Emerging Topics in Computational Intelligence* (2022). https://doi.org/10.1109/TETCI.2022.3201299

[14] Jeannie Fitzgerald, R Muhammad Atif Azad, and Conor Ryan. 2013. A bootstrapping approach to reduce over-fitting in genetic programming. In *Proceedings of the 15th Annual Conference Companion on Genetic and Evolutionary Computation*. 1113–1120.

[15] Ivo Gonçalves and Sara Silva. 2013. Balancing learning and overfitting in genetic programming with interleaved sampling of training data. In *Genetic Programming: 16th European Conference, EuroGP 2013, Vienna, Austria, April 3-5, 2013. Proceedings 16*. Springer, 73–84.

[16] Thomas Helmuth, Lee Spector, and James Matheson. 2014. Solving uncompromising problems with lexicase selection. *IEEE Transactions on Evolutionary Computation* 19, 5 (2014), 630–643.

[17] Ting Hu. 2023. Genetic Programming for Interpretable and Explainable Machine Learning. In *Genetic Programming Theory and Practice XIX*. Springer, 81–90.

[18] Maarten Keijzer and Vladan Babovic. 2000. Genetic programming, ensemble methods and the bias/variance tradeoff–introductory investigations. In *European Conference on Genetic Programming*. Springer, 76–90.

[19] Krzysztof Krawiec. 2002. Genetic programming-based construction of features for machine learning and knowledge discovery tasks. *Genetic Programming and Evolvable Machines* 3 (2002), 329–343.

[20] William La Cava, Thomas Helmuth, Lee Spector, and Jason H Moore. 2019. A probabilistic and multi-objective analysis of lexicase selection and ε-lexicase selection. *Evolutionary Computation* 27, 3 (2019), 377–402.

[21] William La Cava, Sara Silva, Kourosh Danai, Lee Spector, Leonardo Vanneschi, and Jason H Moore. 2019. Multidimensional genetic programming for multiclass classification. *Swarm and Evolutionary Computation* 44 (2019), 260–272.

[22] Jason Liang, Elliot Meyerson, Babak Hodjat, Dan Fink, Karl Mutch, and Risto Miikkulainen. 2019. Evolutionary neural automl for deep learning. In *Proceedings of the Genetic and Evolutionary Computation Conference*. 401–409.

[23] Luis Muñoz, Leonardo Trujillo, Sara Silva, Mauro Castelli, and Leonardo Vanneschi. 2019. Evolving multidimensional transformations for symbolic regression with M3GP. *Memetic Computing* 11 (2019), 111–126.

[24] Kourosh Neshatian, Mengjie Zhang, and Peter Andreae. 2012. A filter approach to multiple feature construction for symbolic learning classifiers using genetic programming. *IEEE Transactions on Evolutionary Computation* 16, 5 (2012), 645–661.

[25] Ji Ni, Russ H Drieberg, and Peter I Rockett. 2012. The use of an analytic quotient operator in genetic programming. *IEEE Transactions on Evolutionary Computation* 17, 1 (2012), 146–152.

[26] Ji Ni and Peter Rockett. 2014. Tikhonov regularization as a complexity measure in multiobjective genetic programming. *IEEE Transactions on Evolutionary Computation* 19, 2 (2014), 157–166.

[27] Miguel Nicolau and Alexandros Agapitos. 2021. Choosing function sets with better generalisation performance for symbolic regression models. *Genetic programming and evolvable machines* 22, 1 (2021), 73–100.

[28] Randal S Olson, William La Cava, Patryk Orzechowski, Ryan J Urbanowicz, and Jason H Moore. 2017. PMLB: a large benchmark suite for machine learning evaluation and comparison. *BioData mining* 10, 1 (2017), 1–13.

[29] Caitlin A Owen, Grant Dick, and Peter A Whigham. 2020. Characterizing genetic programming error through extended bias and variance decomposition. *IEEE Transactions on Evolutionary Computation* 24, 6 (2020), 1164–1176.

[30] Caitlin A Owen, Grant Dick, and Peter A Whigham. 2022. Standardization and Data Augmentation in Genetic Programming. *IEEE Transactions on Evolutionary Computation* 26, 6 (2022), 1596–1608.

[31] Michael O'Neill, Leonardo Vanneschi, Steven Gustafson, and Wolfgang Banzhaf. 2010. Open issues in genetic programming. *Genetic Programming and Evolvable Machines* 11 (2010), 339–363.

[32] Sara Silva, Leonardo Vanneschi, Ana IR Cabral, and Maria J Vasconcelos. 2018. A semi-supervised Genetic Programming method for dealing with noisy labels and hidden overfitting. *Swarm and Evolutionary Computation* 39 (2018), 323–338.

[33] Matthew G Smith and Larry Bull. 2005. Genetic programming with a genetic algorithm for feature construction and selection. *Genetic Programming and Evolvable Machines* 6 (2005), 265–281.

[34] Keith M Sullivan and Sean Luke. 2007. Evolving kernels for support vector machine classification. In *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation*. 1702–1707.

[35] Clíodhna Tuite, Alexandros Agapitos, Michael O'Neill, and Anthony Brabazon. 2011. Early stopping criteria to counteract overfitting in genetic programming. In *Proceedings of the 13th Annual Conference Companion on Genetic and Evolutionary Computation*. 203–204.

[36] Leonardo Vanneschi and Mauro Castelli. 2021. Soft target and functional complexity reduction: A hybrid regularization method for genetic programming. *Expert Systems with Applications* 177 (2021), 114929.

[37] Leonardo Vanneschi, Mauro Castelli, and Sara Silva. 2010. Measuring bloat, overfitting and functional complexity in genetic programming. In *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*. 877–884.

[38] Marco Virgolin, Tanja Alderliesten, and Peter AN Bosman. 2020. On explaining machine learning models by evolving crucial and compact features. *Swarm and Evolutionary Computation* 53 (2020), 100640.

[39] Chunyu Wang, Qi Chen, Bing Xue, and Mengjie Zhang. 2023. Shapley Value Based Feature Selection to Improve Generalization of Genetic Programming for High-Dimensional Symbolic Regression. In *Australasian Conference on Data Science and Machine Learning*. Springer, 163–176.

[40] Zitong Yang, Yaodong Yu, Chong You, Jacob Steinhardt, and Yi Ma. 2020. Rethinking bias-variance trade-off for generalization of neural networks. In *International Conference on Machine Learning*. PMLR, 10767–10777.

[41] Byoung-Tak Zhang and Heinz Mühlenbein. 1995. Balancing accuracy and parsimony in genetic programming. *Evolutionary Computation* 3, 1 (1995), 17–38.

[42] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. 2021. Understanding deep learning (still) requires rethinking generalization. *Commun. ACM* 64, 3 (2021), 107–115.

[43] Hengzhe Zhang, Qi Chen, Bing Xue, Wolfgang Banzhaf, and Mengjie Zhang. 2023. Modular Multi-Tree Genetic Programming for Evolutionary Feature Construction for Regression. *IEEE Transactions on Evolutionary Computation* (2023).

[44] Hengzhe Zhang, Qi Chen, Bing Xue, Wolfgang Banzhaf, and Mengjie Zhang. 2023. A Semantic-Based Hoist Mutation Operator for Evolutionary Feature Construction in Regression. *IEEE Transactions on Evolutionary Computation* (2023).

[45] Hengzhe Zhang, Aimin Zhou, Qi Chen, Bing Xue, and Mengjie Zhang. 2023. SR-Forest: A Genetic Programming based Heterogeneous Ensemble Learning Method. *IEEE Transactions on Evolutionary Computation* (2023).

[46] Hengzhe Zhang, Aimin Zhou, and Hu Zhang. 2022. An Evolutionary Forest for Regression. *IEEE Transactions on Evolutionary Computation* 26, 4 (2022), 735–749.