A General Feature-Informed Crossover for Two-Stage Feature Selection in Symbolic Regression

Hengzhe Zhang, Qi Chen*, Bing Xue, Wolfgang Banzhaf, Mengjie Zhang
Centre for Data Science and Artificial Intelligence & School of Engineering and Computer Science
Victoria University of Wellington, Wellington 6140, New Zealand
{hengzhe.zhang, qi.chen, bing.xue, mengjie.zhang}@ecs.vuw.ac.nz
Department of Computer Science and Engineering, College of Engineering and BEACON Center
Michigan State University, East Lansing, MI 48824, USA
banzhafw@msu.edu

2025 IEEE Congress on Evolutionary Computation (CEC) | 979-8-3315-3431-8/25/$31.00 ©2025 IEEE | DOI: 10.1109/CEC65147.2025.11043089

***Abstract*—Genetic programming-based symbolic regression is a widely used machine learning technique, but its effectiveness can be limited as the number of input features increases. In genetic programming, two-stage feature selection has been extensively applied to enhance performance when dealing with a large number of input features. Existing two-stage feature selection methods typically require reinitializing new GP trees based on the selected features after feature selection, which disrupts the building blocks accumulated during evolution. In this paper, we propose a crossover operator that is aware of the selected features to leverage the feature selection results, thereby bypassing the need for reinitialization. This operator guides the crossover process to prioritize selected features, gradually eliminating unimportant features while preserving evolved building blocks. Experimental results validate the proposed method across three different feature-selection mechanisms on 98 datasets, demonstrating its effectiveness and broad applicability across various feature-selection strategies.**

***Index Terms*—Symbolic Regression, Feature Selection, Genetic Programming**

I. Introduction

Symbolic regression is a widely used interpretable machine learning technique [1]. The core idea behind symbolic regression is to generate a mathematical expression, denoted as $f(X)$, that accurately maps the input data X to the target output Y. Genetic programming (GP) has been extensively used for symbolic regression [2], [3] as it is a gradient-free, population-based optimization method capable of creating and optimizing variable-length symbolic expressions, making it naturally suited for this task.

GP possesses an inherent ability to perform feature selection. However, as the dimensionality of the dataset increases, GP often faces challenges in maintaining effectiveness due to the rapid growth of the search space. To mitigate this issue, feature selection has been proposed as a strategy to reduce the size of feature set, thereby improving both the efficiency and effectiveness of GP [4].

*Corresponding author: Qi Chen.

979-8-3315-3431-8/25/$31.00 ©2025 IEEE

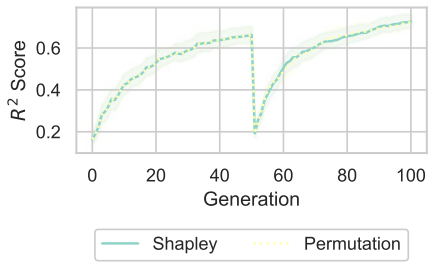

Fig. 1: A significant drop in performance in an existing two-stage feature selection-based symbolic regression algorithm.

A commonly used approach for feature selection is a two-stage framework [5], [6]. In the first stage, a standard GP is run on all available features. The top-performing individuals are then analyzed to assess feature usage, and feature importance is computed using methods such as frequency analysis [4], permutation importance [5], Shapley values [6], or Maximal Information Coefficient (MIC) [7] values. In the second stage, new GP trees are initialized using only the selected features, and a subsequent GP run is performed to derive the final symbolic expression.

A major limitation of this approach lies in the random initialization of trees with the selected features at the beginning of the second stage, which disrupts the effective building blocks accumulated during the first stage. This disruption can reduce effectiveness due to the loss of evolved building blocks [8]. As shown in Fig. 1, this disruption significantly hinders the evolutionary process. Ideally, it would be more beneficial to retain and refine these building blocks after feature selection rather than completely discarding them and regenerating entirely new GP trees using the selected features.

To address this limitation, we propose a feature-informed crossover (FIC) operator that eliminates the need for reinitialization. The key idea behind this operator is to prioritize subtrees that contain features identified as important at the first stage. Specifically, the operator replaces subtrees with fewer selected features with those containing more selected features.

Authorized licensed use limited to: Michigan State University. Downloaded on June 30,2025 at 05:26:08 UTC from IEEE Xplore. Restrictions apply.

This approach gradually eliminates irrelevant features while focusing the search on the most important features without requiring reinitialization. The main contributions of this paper are summarized as follows:

- To improve effectiveness, we propose a feature-informed crossover operator that encourages the use of selected features without reinitializing the population after feature selection.
- To ensure the crossover operator accounts for the importance of selected features, we propose a subtree importance measurement method, allowing the crossover operator to prioritize subtrees containing selected features.
- To examine the generality of the proposed crossover operator, we evaluate its performance across three types of two-stage feature selection mechanisms. The results demonstrate its broad applicability.

The remainder of this paper is structured as follows: In Section II, we review related work on feature selection mechanisms and crossover operators in GP. Section III introduces the proposed importance measurement method and the feature-informed crossover operator. Experimental settings are described in Section IV, and experimental results are presented in Section V. Finally, Section VI concludes the paper and outlines future directions.

II. Related Work

GP has an inherent capability to select features [9] and has demonstrated superior feature selection performance to traditional metrics like Information Gain [10]. Notably, GP can serve as a feature construction technique that simultaneously performs feature selection and feature construction, enhancing the learning performance of classical machine learning algorithms [11], [12]. However, when the search space becomes large, this intrinsic mechanism may be insufficient, necessitating additional strategies to effectively perform feature selection.

A widely used approach is the two-stage feature selection framework [5], [6]. In this framework, a GP algorithm is first executed on the whole set of features. Then, feature importance is calculated based on the top individuals to identify potentially relevant features. The importance mechanisms include frequency analysis [4], permutation importance [5], Shapley values [6], and Maximal Information Coefficient (MIC) [7]. The importance values of all features used by the top-performing individuals are aggregated, and the top-ranked features are selected. In the second stage, a new population is randomly initialized using only these selected features.

Feature selection in GP has been successfully applied across various other domains, including classification [13] and hyper-heuristic generation [8]. Similar to feature selection in symbolic regression [5], features are replaced with a constant to evaluate their removal impact, and only the top-ranked features are retained for use in the second stage. However, the issue of performance degradation due to random initialization has been observed in the job shop scheduling domain. In job shop scheduling, randomly initializing GP trees in the second stage led to significantly worse performance [8]. To address this, the authors proposed randomly initializing a large number of individuals at the beginning of the second stage and selecting individuals with behavior similar to the best-performing individual from the first stage to continue in the second stage. While this strategy alleviates the disruption caused by random initialization and achieves better results, it requires additional computational effort to determine the behavior of individuals, which can be expensive.

Beyond applying feature selection during random initialization, feature selection can also be incorporated into mutation operators. Specifically, when generating a random subtree to replace the current tree, the features used to generate the subtree can be selected based on a probability vector rather than through random sampling. This strategy has been widely applied in symbolic regression [14], classification [15], ensemble learning [16], and workflow scheduling [17]. However, a limitation of this approach is that mutation operators are typically applied with a low probability, limiting the overall influence of feature selection in this paradigm.

Context-aware crossover is a promising approach for designing a crossover operator that leverages feature selection results, where crossover points are intelligently chosen instead of being selected randomly [18]. The selection point can be determined based on enumeration search [18] or the correlation between subtree semantics and individual semantics [19]. Despite advancements in context-aware crossover operators, there remains a lack of operators specifically designed to integrate effectively with feature selection techniques. Developing such operators could help guide GP in evolving more interpretable models and may reduce the cost of collecting numerous features during the test phase, representing a key area for research.

III. The Proposed Method

In this section, we propose a novel crossover operator to be embedded into a two-stage feature-selection-based symbolic regression algorithm. First, the overall algorithm framework is introduced. Next, in Section III-B, we describe the subtree importance measurement method, which assigns weights to each subtree based on the feature selection results. Finally, in Section III-C, we present the feature-informed crossover (FIC) operator, which leverages the feature selection results to guide the evolutionary process.

A. Algorithm Framework

The proposed method is built upon a two-stage symbolic regression framework. The first stage operates on the full set of features, while the second stage focuses on a subset of selected features, as illustrated in Fig. 2. The transition between stages occurs at the midpoint of evolution without reinitializing the population. The algorithm consists of the following steps:

- **Population Initialization:** A population of individuals is randomly initialized to form the starting point of the evolutionary process. For each individual, a single GP tree is generated using the ramped half-and-half method.

Authorized licensed use limited to: Michigan State University. Downloaded on June 30,2025 at 05:26:08 UTC from IEEE Xplore. Restrictions apply.

- **Parent Selection:** Lexicase selection [20] is employed to identify promising individuals for reproduction. This selection method evaluates individuals based on a sequence of test cases, each serving as a criterion. For each criterion, individuals must meet or exceed a threshold defined as $\min_L + \epsilon$, where $\min_L$ is the minimum loss achieved by any individual in the population on that criterion, and ϵ is the median absolute deviation.
- **Offspring Generation:** Offspring are generated using random crossover, feature-informed crossover, and random mutation operators:
  - **Random Crossover (First Stage):** A random subtree from one parent individual is selected to replace a subtree in a randomly chosen tree from the other parent individual.
  - **Feature-Informed Crossover (FIC, Second Stage):** The feature-informed subtree crossover operator assigns higher probabilities to subtrees with greater utilization of selected variables for selection as donors. These donor subtrees replace randomly chosen subtrees in the other parent. This approach encourages the use of selected features while gradually eliminating irrelevant features. Further details are provided in Section III-C.
  - **Random Mutation (First Stage/Second Stage):** A subtree is randomly selected and replaced with a newly generated subtree. In the first stage, the newly generated subtree uses the full set of features. In the second stage, the new subtrees are generated based only on the selected features. Thus, in the second stage, random subtree mutation also encourages the use of selected features.
- **Offspring Evaluation:** During the evaluation stage, each GP tree is executed on the training data. Linear scaling [21] is applied to improve alignment between the GP tree and the target. Specifically, for a GP tree ϕ with output $\phi(X)$, the final output is calculated as $\Phi(X) = \beta\phi(X) + \alpha$, where β is the scaling coefficient and α is the intercept. The coefficient β is regularized using the L2 norm, and an efficient leave-one-out cross-validation procedure [22] is employed to determine the optimal regularization coefficient. During evaluation, the outputs are the efficient leave-one-out cross-validation predictions $\hat{y}$, which help mitigate overfitting. The fitness value is computed using the mean squared error (MSE), defined as $\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$, to identify the best-performing individual. Additionally, for lexicase selection, the unaggregated loss values are retained.
- **Elitism:** The best individual from the population is explicitly preserved and used to generate offspring for the next generation. Additionally, the best individual is used to make predictions at the end of the evolutionary process.
- **Feature Selection:** At the end of the first stage, feature selection is performed using a feature importance calculation method. These methods include frequency analysis, permutation importance, or Shapley values, as detailed in Section IV-A. The selected feature importance method identifies the most significant features based on the top-N individuals in the population. Feature importance values are aggregated from these individuals. Given k original features, the top $\mathrm{round}(\log(k))$ features are selected for use in the second stage, which has been empirically shown to perform well in existing research [23].

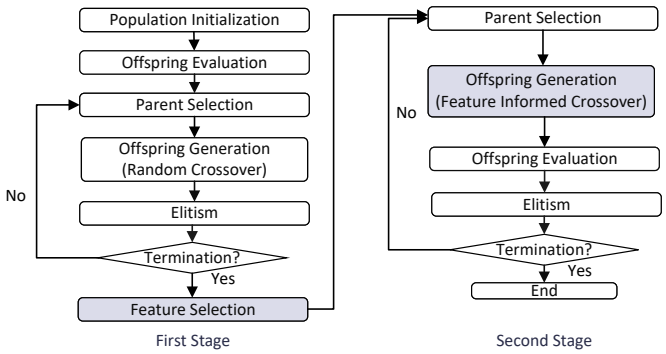

Fig. 2: Workflow of the two-stage feature selection process. In this framework, population reinitialization is not required at the beginning of the second stage.

The process of population initialization, parent selection, offspring generation, offspring evaluation, and elitism is repeated iteratively until a predefined number of iterations is reached.

B. Subtree Importance Measurement

Based on the selected features, we calculate the importance of each subtree. This importance guides the crossover operator to gradually eliminate subtrees with irrelevant features by replacing them with subtrees containing selected features during the crossover process. As shown in Fig. 3, the importance of each subtree ψ within an individual Φ is based on its composition of terminals. The weight w_ψ of subtree ψ is formally defined as Eq. (1):

$$w_\psi = \frac{\omega_w}{\sigma_w}, \tag{1}$$

where σ_w represents the total number of terminal nodes in subtree ψ, and ω_w is the total weight assigned to nodes within ψ. A node j contributes a weight of 1 if it is a selected terminal from the terminal set V or a constant, and 0 otherwise. This normalization by σ_w ensures that larger subtrees do not inherently have greater importance due to their size. The pseudocode for the measurement process is provided in Algorithm 1 [1] .

C. Feature Informed Crossover

Instead of reinitializing the entire population, the proposed method leverages the crossover and mutation operators to gradually fill the population with GP trees using selected features. This approach allows the evolutionary process to

[1]Source Code: https://github.com/hengzhe-zhang/EvolutionaryForest/blob/master/evolutionary_forest/component/crossover/marking_weights.py

Authorized licensed use limited to: Michigan State University. Downloaded on June 30,2025 at 05:26:08 UTC from IEEE Xplore. Restrictions apply.

Algorithm 1 Subtree Importance Measurement

Input: Tree T, Feasible Variables V
Output: Weights of Each Subtree W
Initialize $W \leftarrow []$
for each node $n \in T$ **do**
  Determine the subtree ψ rooted at n
  Initialize $\sigma_w \leftarrow 0$ and $\omega_w \leftarrow 0$
  for each node $m \in \psi$ **do**
    if m is a terminal node **then**
      $\sigma_w \leftarrow \sigma_w + 1$
      if $m \in V$ or m is a constant **then**
        $\omega_w \leftarrow \omega_w + 1$
      end if
    end if
  end for
  $w_\psi \leftarrow \frac{\omega_w}{\sigma_w}$
  Append w_ψ to W
end for
Return: W

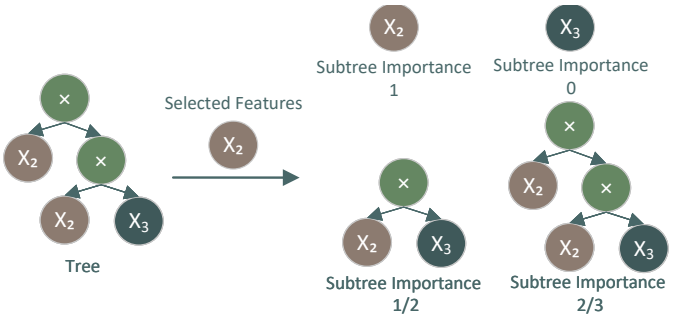

Fig. 3: Measuring subtree importance based on selected features.

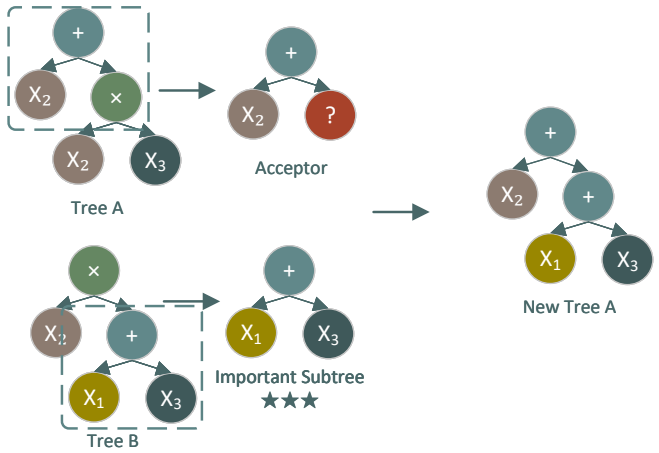

Fig. 4: Feature-Informed Crossover. Assuming X_1 and X_3 are the selected features, while X_2 is unselected and considered an irrelevant feature.

continue after the feature selection stage without discarding useful building blocks. With awareness of feature selection results, the crossover is performed as follows:

Algorithm 2 Subtree Crossover with Importance-Aware Selection

Input: Parent individuals Φ_1, Φ_2 with subtrees $\Psi_{\Phi_1}, \Psi_{\Phi_2}$ and weights w_{Φ_1}, w_{Φ_2}
Output: Offspring Φ_1', Φ_2'
$p_{\Phi_1} \leftarrow \text{NormalizeWeights}(\Psi_{\Phi_1}, w_{\Phi_1})$ ▷ Get probabilities
$p_{\Phi_2} \leftarrow \text{NormalizeWeights}(\Psi_{\Phi_2}, w_{\Phi_2})$
$\psi_d \sim \text{Sampling}(\psi \in \Psi_{\Phi_1})$ with probabilities $\{p_\psi\}_{\psi \in \Psi_{\Phi_1}}$ ▷ Sample donor subtree from Φ_1
$\psi_a \sim \text{Uniform}(\psi \in \Psi_{\Phi_2})$ ▷ Sample acceptor subtree from Φ_2
$\Phi_2' \leftarrow \text{Replace}(\Phi_2, \psi_a \rightarrow \psi_d)$ ▷ Replace ψ_a in Φ_2 with ψ_d from Φ_1
$\psi_d \sim \text{Sampling}(\psi \in \Psi_{\Phi_2})$ with probabilities $\{p_\psi\}_{\psi \in \Psi_{\Phi_2}}$ ▷ Sample donor subtree from Φ_2
$\psi_a \sim \text{Uniform}(\psi \in \Psi_{\Phi_1})$ ▷ Sample acceptor subtree from Φ_1
$\Phi_1' \leftarrow \text{Replace}(\Phi_1, \psi_a \rightarrow \psi_d)$ ▷ Replace ψ_a in Φ_1 with ψ_d from Φ_2
return (Φ_1', Φ_2')

1) **Probability Conversion:** The weights of subtrees in each individual Φ are first converted into a probability vector p_Φ, which is then used to sample donor subtrees. Since these weights fall within the range $[0, 1]$ but do not necessarily sum to 1, normalization is required to transform them into probabilities. Specifically, each weight w_ψ is incremented by 1 to smooth the probability distribution and ensure that even terminal nodes, which might otherwise be inadvertently filtered out by feature selection, retain a nonzero probability of selection. This adjustment enhances the robustness of the search process.
2) **Selection of Subtrees:** Subtrees are selected based on their normalized probability:
   - **Donor:** A subtree that utilizes more selected features, ψ_d, is sampled from parent Φ_1 with a probability proportional to its normalized probability p_ψ.
   - **Acceptor:** A randomly chosen subtree, ψ_a, is selected uniformly from the subtrees Ψ_{Φ_2} of parent Φ_2.
3) **Replacement:** The selected acceptor subtree ψ_a in one parent is replaced with the donor subtree ψ_d from the other parent. The crossover operation is illustrated in Fig. 4, and the pseudocode is presented in Algorithm 2 [2].

This mechanism ensures that the crossover process preferentially propagates subtrees containing more important features while discarding less significant subtrees composed of unimportant variables.

[2]Source Code: https://github.com/hengzhe-zhang/EvolutionaryForest/blob/master/evolutionary_forest/component/crossover/feature_informed_crossover.py

Authorized licensed use limited to: Michigan State University. Downloaded on June 30,2025 at 05:26:08 UTC from IEEE Xplore. Restrictions apply.

IV. Experimental Settings

A. Baseline Algorithms

The proposed crossover operator is general and can be extended to any existing two-stage feature selection-based symbolic regression framework. In this paper, we consider three feature selection algorithms. These algorithms share the same evolutionary process; the only difference lies in the method used to calculate feature importance:

- **Frequency-based Feature Selection:** Feature importance is determined based on the frequency of a feature appearing in the top individuals. Features with higher frequency are considered more important.
- **Permutation-based Feature Selection [5]:** Permutation importance is evaluated by randomly shuffling the values of one feature at a time while keeping all other features unchanged, then measuring the impact on model performance. The importance of each feature X_j is calculated as:

$$I(X_j) = \mathbb{E}_{\text{data}}\left[R^2_{\text{original}} - R^2_{\text{permuted}(X_j)}\right], \tag{2}$$

  where R^2_{original} is the R^2 score of the GP model on the original data, and $R^2_{\text{permuted}(X_j)}$ is the R^2 score of the GP model when X_j is permuted.
- **Shapley-based Feature Selection [6]:** This method employs Shapley values to assess feature importance. The importance of a feature X_j is given by:

$$I(X_j) = \sum_{S \subseteq N \setminus \{j\}} \frac{|S|!(|N| - |S| - 1)!}{|N|!} \left(\Phi(S \cup \{j\}) - \Phi(S)\right), \tag{3}$$

  where N is the whole set of features, S is a subset of features not including j, and $\Phi(S)$ is the output of the model for the subset S. The calculation of Shapley values is computationally expensive, as it requires evaluating all possible coalitions of features to determine their marginal contributions to the prediction.

B. Parameter Settings

In this paper, common settings for GP are employed, as detailed in Table I. For instance, a crossover rate of 0.9 is used to encourage the exchange of building blocks, while a mutation rate of 0.1 is applied to enable the discovery of new blocks [14]. To prevent division by zero errors, the analytical quotient [24] is utilized in place of the traditional division operator. The analytical quotient is defined as $AQ(a, b) = \frac{a}{\sqrt{1+b^2}}$. We implement our standard GP algorithm using the DEAP library [25], employing standard random subtree crossover in the baseline methods.

C. Datasets

The experimental datasets are selected from the Penn Machine Learning Benchmark (PMLB) [26]. Due to computational constraints, only datasets with fewer than 2000 instances are included. Based on this criterion, a total of 98 datasets are used in this paper.

TABLE I: Parameter settings for GP.

Parameter	Value
Population Size	100
Initial Tree Depth	0-6
Number of Generations	50 (First Stage)+50 (Second Stage)
Maximum Tree Depth	10
Crossover and Mutation Rates	0.9 and 0.1
Elitism (Number of Individuals)	1
Functions	+, -, *, AQ, Square, Log, Sqrt, Max, Min, Sin_π, Cos_π, Abs, Negative

D. Evaluation Protocol

For evaluation, all datasets are split into training and test sets with a ratio of 80:20. All features are standardized before training. The R^2 score is used as the evaluation metric, which is defined as:

$$R^2 = 1 - \frac{\sum(y - \hat{y})^2}{\sum(y - \bar{y})^2} \tag{4}$$

where y represents the ground truth values, $\hat{y}$ represents the predicted values, and $\bar{y}$ is the mean of the ground truth values. The R^2 score is derived from the mean squared error but normalized by the variance of the target variable. R^2 provides a scale-invariant measure, with an optimal value of 1, making it particularly suitable for comparing performance across datasets with different scales. To ensure stable experimental results, all experiments are repeated 30 times using 30 different random seeds. Specifically, these 30 runs consist of six groups of five-fold cross-validation. The Wilcoxon signed-rank test with a significance level of 0.05 is used to compare the performance of different algorithms [27]. Additionally, the Kruskal-Wallis test with Bonferroni correction is applied to assess statistical differences between algorithms.

V. Experimental Results

A. Comparison of Training R^2 Scores

A comparison of training R^2 scores with and without the use of FIC (the proposed crossover operator) is presented in Table II. The results demonstrate that the FIC operator outperforms baseline algorithms in terms of training R^2 scores. Specifically, by using permutation importance, the FIC crossover operator significantly improves the training R^2 scores on 73 datasets, and it performs no worse on any dataset. This highlights the effectiveness of the FIC operator compared to random initialization, as the crossover operator preserves the building blocks of solutions. The effectiveness of FIC compared to random initialization is also demonstrated in Fig. 5 using four example datasets from PMLB, each containing 25 features. As shown by the convergence curve of training R^2 scores, reinitialization leads to a significant drop at the midpoint of the evolutionary process. Although feature selection ensures that the new population starts from a higher baseline compared to initializing with the full set of original features, the substantial drop in performance requires several generations for the new population to recover after reinitialization, which limits the benefits brought by feature

Authorized licensed use limited to: Michigan State University. Downloaded on June 30,2025 at 05:26:08 UTC from IEEE Xplore. Restrictions apply.

TABLE II: Statistical comparison of **training** R^2 **scores** across different feature selection methods. The notation "a(+)/b(∼)/c(-)" indicates the number of datasets where the row algorithm performed statistically significantly better (+), not significantly different (∼), or significantly worse (-) than the column algorithm.

	Permutation	**Shapley+FIC**	**Shapley**	**Frequency+FIC**	**Frequency**
Permutation+FIC	73(+)/25(∼)/0(-)	0(+)/98(∼)/0(-)	70(+)/28(∼)/0(-)	0(+)/98(∼)/0(-)	77(+)/21(∼)/0(-)
Permutation	—	0(+)/27(∼)/71(-)	4(+)/94(∼)/0(-)	0(+)/24(∼)/74(-)	12(+)/86(∼)/0(-)
Shapley+FIC	—	—	72(+)/26(∼)/0(-)	0(+)/98(∼)/0(-)	78(+)/20(∼)/0(-)
Shapley	—	—	—	0(+)/28(∼)/70(-)	3(+)/95(∼)/0(-)
Frequency+FIC	—	—	—	—	75(+)/23(∼)/0(-)

TABLE III: Statistical comparison of **test** R^2 **scores** using different feature selection methods.

	Permutation	**Shapley+FIC**	**Shapley**	**Frequency+FIC**	**Frequency**
Permutation+FIC	42(+)/56(∼)/0(-)	0(+)/98(∼)/0(-)	38(+)/60(∼)/0(-)	0(+)/98(∼)/0(-)	43(+)/54(∼)/1(-)
Permutation	—	0(+)/57(∼)/41(-)	3(+)/95(∼)/0(-)	0(+)/55(∼)/43(-)	7(+)/91(∼)/0(-)
Shapley+FIC	—	—	39(+)/59(∼)/0(-)	0(+)/98(∼)/0(-)	41(+)/57(∼)/0(-)
Shapley	—	—	—	0(+)/58(∼)/40(-)	3(+)/94(∼)/1(-)
Frequency+FIC	—	—	—	—	41(+)/56(∼)/1(-)

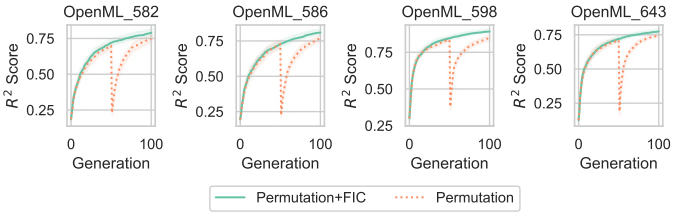

Fig. 5: Evolutionary plots of **training** R^2 **scores** comparing the FIC operator with random initialization when incorporating **Permutation Importance**.

selection. The results using Shapley values and frequency-based importance measures are provided in Section A of the supplementary material [3], showing similar trends and further demonstrating the effectiveness of the FIC operator in improving GP compared to random reinitialization.

B. Comparison of Test R^2 Scores

A comparison of test R^2 scores is shown in Table III. Similar to the training performance, the test scores also significantly improved with the use of the proposed FIC operator. Specifically, the FIC operator significantly improves the test R^2 scores on 42 datasets when using permutation importance to calculate importance scores. Likewise, the FIC operator significantly improves the test R^2 scores on 39 datasets when using Shapley importance to calculate importance scores. Furthermore, the FIC operator significantly improves the test R^2 scores on 41 datasets when using frequency importance to calculate importance scores. These results indicate that the FIC operator not only enhances training performance but also improves generalization performance. The convergence curve of test R^2 scores in Fig. 6 shows that random initialization results in a significant drop in test performance. In contrast, the proposed FIC operator avoids this issue and drives GP toward better final models. While the FIC operator may not completely eliminate irrelevant features, its ability in retaining building blocks and continuing evolution outweighs the drawbacks of complete reinitialization. Consequently, the proposed FIC achieves superior final generalization performance compared to the reinitialization method. Moreover, not completely eliminating filtered-out features during random initialization can be beneficial if the feature importance mechanism is imperfect. Under the FIC paradigm, these filtered-out features still have a chance to survive in the final model. Thus, even if some useful features are filtered out due to poor feature selection mechanisms, the FIC operator can still utilize them during the evolutionary process.

Although the proposed method generally outperforms existing two-stage feature selection approaches across various datasets, the significance test in Table IV indicates that it does not significantly outperform standard GP. This can be attributed to two main factors. First, the datasets used in this evaluation have relatively low dimensionality, which aligns with findings from existing GP feature selection studies. These studies suggest that when the number of features is modest, feature selection primarily enhances model interpretability rather than performance [8]. Second, the two-stage approach does not fully exploit the benefits of feature selection, and a multi-stage feature selection strategy that adapts in real-time could potentially yield better results. Nonetheless, as shown in the next section, the proposed method effectively reduces the number of features without sacrificing accuracy, making it a favorable outcome.

C. Number of Used Features

Feature selection reduces the number of features used in the final model [8]. Random initialization ensures a reduced feature set since only the selected features are utilized. In contrast, the FIC operator does not reinitialize the population, potentially allowing some features to persist in the final model. In this section, we compare the number of features in the final models when applying the random initialization strategy versus the FIC operator. When counting, all random constants are treated as a single feature. Fig. 7 presents the number of features in the final models. The results indicate no

[3]Supplementary Material: https://github.com/hengzhe-zhang/CEC-2025-Supplementary-Material

Authorized licensed use limited to: Michigan State University. Downloaded on June 30,2025 at 05:26:08 UTC from IEEE Xplore. Restrictions apply.

TABLE IV: Statistical comparison of **test** R^2 **scores** using the Kruskal-Wallis test with Bonferroni correction for p-values.

	Permutation	**Shapley+FIC**	**Shapley**	**Frequency+FIC**	**Frequency**	**Standard GP**
Permutation+FIC	0.0000	1.0000	0.0000	1.0000	0.0000	1.0000
Permutation	-	0.0000	1.0000	0.0000	1.0000	0.0000
Shapley+FIC	-	-	0.0000	1.0000	0.0000	1.0000
Shapley	-	-	-	0.0000	1.0000	0.0000
Frequency+FIC	-	-	-	-	0.0000	1.0000
Frequency	-	-	-	-	-	0.0000

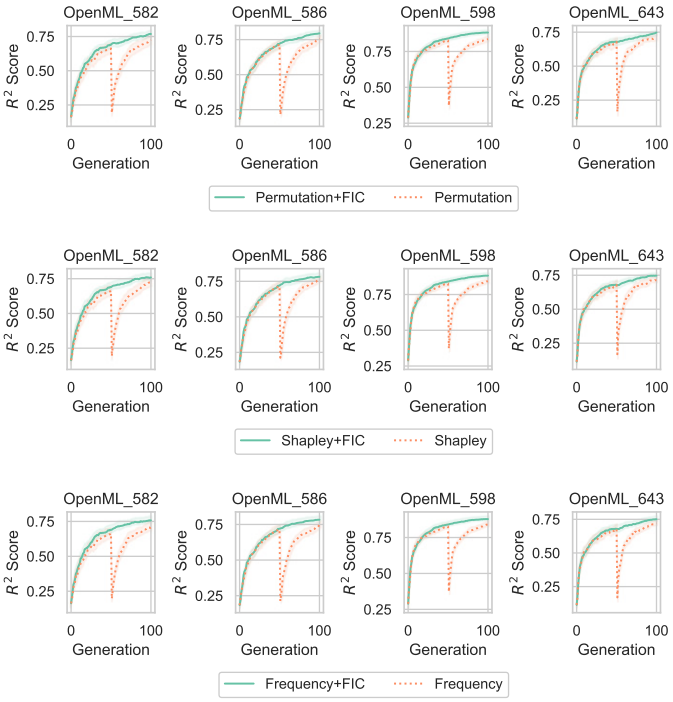

Fig. 6: Evolutionary plots of **test** R^2 **scores** comparing the use of the FIC operator with random initialization when incorporating **Permutation Importance, Shapley Values, and Frequency**.

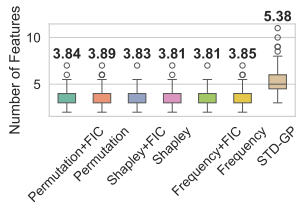

Fig. 7: Distribution of the number of features selected in the final model across 98 datasets and 30 runs.

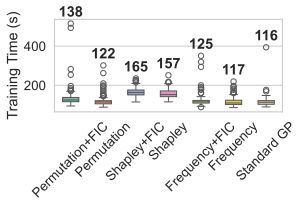

Fig. 8: Distribution of training times across 98 datasets and 30 runs.

significant difference in the number of features between using the crossover operator and reinitialization. This suggests that, while the FIC operator may not theoretically eliminate all irrelevant features, it empirically demonstrates strong performance in reducing irrelevant features and maintaining a compact feature subset in the final model. Compared to standard GP, the proposed algorithm significantly reduces the number of features used. This indicates that the proposed method effectively enhances interpretability without compromising the final model's effectiveness. Another potential benefit is that the reduced number of features may lower the cost of collecting and maintaining features for unseen data.

The trajectories of the number of features in the best individuals during the evolutionary process for both the proposed method and the baseline reinitialization are shown in Fig. 9. As observed in Fig. 9, the proposed method effectively preserves useful building blocks. Consequently, although the number of features decreases after feature selection, it remains at a reasonable level. In contrast, the baseline algorithm discards useful building blocks and repeatedly follows a pattern similar to the initial generation, gradually evolving to a level comparable to the proposed approach. Overall, these results confirm that the reinitialization process unnecessarily reduces the number of features too drastically, leading to the loss of useful building blocks. In contrast, the proposed method effectively retains these building blocks, resulting in superior performance.

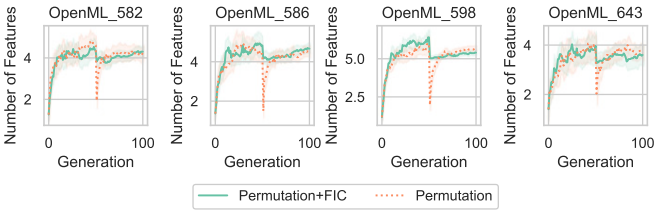

Fig. 9: Distribution of the number of features used in the final model across 98 datasets over 30 runs.

D. Comparison of Training Time

A comparison of training time is presented in Fig. 8. The experimental results show that the proposed method does not significantly increase training time compared to standard GP. The most time-consuming method is the Shapley value approach with the proposed crossover operator. The main reason for this is that Shapley value computation is inherently time-consuming, leading to an increase in training time. In fact, the baseline method, random initialization with Shapley value-based feature selection, also results in longer training times, indicating that the increased computational cost is not primarily caused by the proposed operator. Compared to using the proposed operator with random initialization to leverage feature selection results, the proposed method is slightly more

Authorized licensed use limited to: Michigan State University. Downloaded on June 30,2025 at 05:26:08 UTC from IEEE Xplore. Restrictions apply.

time-consuming. This is mainly because random initialization may start from simpler trees, which evaluate faster than the proposed method, which inherits useful building blocks that might be more complex. Overall, the conclusion is that the proposed method does not significantly increase computational cost on its own. However, to make the proposed method as efficient as standard GP, developing an efficient yet reliable feature selection method should be a key focus.

VI. Conclusions

In this paper, to address the limitations of existing two-stage feature-selection-based GP for symbolic regression algorithms, we propose a feature-informed crossover operator that leverages feature selection information while preserving building blocks. Specifically, the proposed crossover operator encourages the use of subtrees with selected features to replace randomly selected subtrees, thereby gradually eliminating irrelevant features during the evolution process. The experimental results demonstrate that the proposed method significantly improves the two-stage feature selection process compared to the random initialization approach, achieving both higher training R^2 scores and test R^2 scores. Furthermore, as shown by the results on the number of used features, the proposed method achieves on-par performance in terms of the overall distribution of the number of features utilized in the final model. These results highlight the effectiveness of the feature-informed crossover operator.

For future work, since the proposed operator maintains building blocks after feature selection, it can be straightforwardly extended to a multi-stage feature selection framework without re-initialization. Specifically, features eliminated in earlier stages could be reintroduced in later stages, while initially retained features could be eliminated in subsequent stages. This iterative approach may further enhance performance.

References

[1] D. J. Bartlett, H. Desmond, and P. G. Ferreira, "Exhaustive symbolic regression," *IEEE Transactions on Evolutionary Computation*, 2023. doi: 10.1109/tevc.2023.3280250 .

[2] J. Zhong, L. Feng, W. Cai, and Y.-S. Ong, "Multifactorial genetic programming for symbolic regression problems," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 50, no. 11, pp. 4492–4505, 2018.

[3] Y.-H. Sun, T. Huang, J.-H. Zhong, J. Zhang, and Y.-J. Gong, "Symbolic regression-assisted offline data-driven evolutionary computation," *IEEE Transactions on Evolutionary Computation*, 2024. doi: 10.1109/tevc.2024.3482326 .

[4] Q. Chen, B. Xue, B. Niu, and M. Zhang, "Improving generalisation of genetic programming for high-dimensional symbolic regression with feature selection," in *2016 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2016, pp. 3793–3800.

[5] Q. Chen, M. Zhang, and B. Xue, "Feature selection to improve generalization of genetic programming for high-dimensional symbolic regression," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 5, pp. 792–806, 2017.

[6] C. Wang, Q. Chen, B. Xue, and M. Zhang, "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, 2023, pp. 163–176.

[7] M. Rimas, Q. Chen, and M. Zhang, "Feature selection for gpsr based on maximal information coefficient and shapley values," in *2024 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2024, pp. 1–8.

[8] F. Zhang, Y. Mei, S. Nguyen, and M. Zhang, "Evolving scheduling heuristics via genetic programming with feature selection in dynamic flexible job-shop scheduling," *IEEE Transactions on Cybernetics*, vol. 51, no. 4, pp. 1797–1811, 2020.

[9] S. Stijven, W. Minnebo, and K. Vladislavleva, "Separating the wheat from the chaff: on feature selection and feature importance in regression random forests and symbolic regression," in *Proceedings of the 13th Annual Conference Companion on Genetic and Evolutionary Computation*, 2011, pp. 623–630.

[10] F. Viegas, L. Rocha, M. Gonçalves, F. Mourão, G. Sá, T. Salles, G. Andrade, and I. Sandin, "A genetic programming approach for feature selection in highly dimensional skewed data," *Neurocomputing*, vol. 273, pp. 554–569, 2018.

[11] B. Tran, B. Xue, and M. Zhang, "Genetic programming for feature construction and selection in classification on high-dimensional data," *Memetic Computing*, vol. 8, pp. 3–15, 2016.

[12] H. Zhang, A. Zhou, and H. Zhang, "An evolutionary forest for regression," *IEEE Transactions on Evolutionary Computation*, vol. 26, no. 4, pp. 735–749, 2021.

[13] J. Ma and X. Gao, "Designing genetic programming classifiers with feature selection and feature construction," *Applied Soft Computing*, vol. 97, p. 106826, 2020.

[14] H. Zhang, Q. Chen, B. Xue, W. Banzhaf, and M. Zhang, "A semantic-based hoist mutation operator for evolutionary feature construction in regression," *IEEE Transactions on Evolutionary Computation*, 2023. doi: 10.1109/tevc.2023.3331234 .

[15] A. Friedlander, K. Neshatian, and M. Zhang, "Meta-learning and feature ranking using genetic programming for classification: Variable terminal weighting," in *2011 IEEE Congress of Evolutionary Computation (CEC)*. IEEE, 2011, pp. 941–948.

[16] H. Zhang, A. Zhou, Q. Chen, B. Xue, and M. Zhang, "Sr-forest: A genetic programming based heterogeneous ensemble learning method," *IEEE Transactions on Evolutionary Computation*, 2023. doi: 10.1109/tevc.2023.3243172 .

[17] Y. Yang, G. Chen, H. Ma, S. Hartmann, and M. Zhang, "Dual-tree genetic programming with adaptive mutation for dynamic workflow scheduling in cloud computing," *IEEE Transactions on Evolutionary Computation*, 2024. doi: 10.1109/tevc.2024.3392968 .

[18] H. Majeed and C. Ryan, "On the constructiveness of context-aware crossover," in *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation*, 2007, pp. 1659–1666.

[19] F. Zhang, Y. Mei, S. Nguyen, and M. Zhang, "Correlation coefficient-based recombinative guidance for genetic programming hyperheuristics in dynamic flexible job shop scheduling," *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 3, pp. 552–566, 2021.

[20] W. La Cava, T. Helmuth, L. Spector, and J. H. Moore, "A probabilistic and multi-objective analysis of lexicase selection and ε-lexicase selection," *Evolutionary Computation*, vol. 27, no. 3, pp. 377–402, 2019.

[21] M. Virgolin, T. Alderliesten, and P. A. Bosman, "Linear scaling with and within semantic backpropagation-based genetic programming for symbolic regression," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2019, pp. 1084–1092.

[22] H. Zhang, A. Zhou, H. Qian, and H. Zhang, "Ps-tree: A piecewise symbolic regression tree," *Swarm and Evolutionary Computation*, vol. 71, p. 101061, 2022.

[23] C. Wang, Q. Chen, B. Xue, and M. Zhang, "Improving generalization of genetic programming for high-dimensional symbolic regression with shapley value based feature selection," *Data Science and Engineering*, pp. 1–16, 2024.

[24] J. Ni, R. H. Drieberg, and P. I. Rockett, "The use of an analytic quotient operator in genetic programming," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 1, pp. 146–152, 2012.

[25] F.-A. Fortin, F.-M. De Rainville, M.-A. G. Gardner, M. Parizeau, and C. Gagné, "Deap: Evolutionary algorithms made easy," *The Journal of Machine Learning Research*, vol. 13, no. 1, pp. 2171–2175, 2012.

[26] P. Orzechowski, W. La Cava, and J. H. Moore, "Where are we now? a large benchmark study of recent symbolic regression methods," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2018, pp. 1183–1190.

[27] H. Zhang, Q. Chen, B. Xue, W. Banzhaf, and M. Zhang, "Modular multi-tree genetic programming for evolutionary feature construction for regression," *IEEE Transactions on Evolutionary Computation*, 2023. doi: 10.1109/tevc.2023.3318638 .

Authorized licensed use limited to: Michigan State University. Downloaded on June 30,2025 at 05:26:08 UTC from IEEE Xplore. Restrictions apply.