Genetic Programming and Evolvable Machines (2024) 25:2
https://doi.org/10.1007/s10710-023-09465-z

A geometric semantic macro-crossover operator for evolutionary feature construction in regression

Hengzhe Zhang[1] · Qi Chen[1] · Bing Xue[1] · Wolfgang Banzhaf[2] · Mengjie Zhang[1]

Received: 2 June 2023 / Revised: 24 August 2023 / Accepted: 10 October 2023 /
Published online: 8 December 2023

Abstract
Evolutionary feature construction has been successfully applied to various scenarios. In particular, multi-tree genetic programming-based feature construction methods have demonstrated promising results. However, existing crossover operators in multi-tree genetic programming mainly focus on exchanging genetic materials between two trees, neglecting the interaction between multi-trees within an individual. To increase search effectiveness, we take inspiration from the geometric semantic crossover operator used in single-tree genetic programming and propose a macro geometric semantic crossover operator for multi-tree genetic programming. This operator is designed for feature construction, with the goal of generating offspring containing informative and complementary features. Our experiments on 98 regression datasets show that the proposed geometric semantic macro-crossover operator significantly improves the predictive performance of the constructed features. Moreover, experiments conducted on a state-of-the-art regression benchmark demonstrate that multi-tree genetic programming with the geometric semantic macro-crossover operator can significantly outperform all 22 machine learning algorithms on the benchmark.

Keywords Evolutionar feature construction · Genetic programming · Geometric semantic genetic programming

1 Introduction

Automated feature construction is a key technique in the machine learning domain. The goal is to automatically construct features $\phi(X)$ to improve the learning performance of a machine learning algorithm on a specific task $(X, Y) = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ [1]. Here, X represents input data, Y represents the corresponding target labels, and n represents the number of instances. Automated feature construction techniques have been successfully applied to various

Extended author information available on the last page of the article

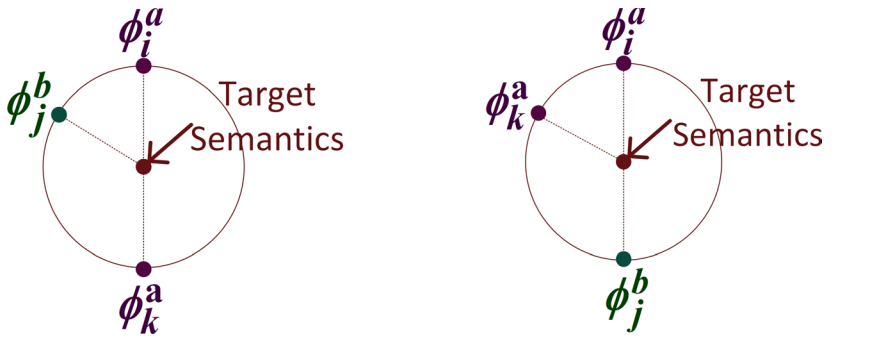
areas, including regression [1], classification [2], clustering [3] and manifold learning [4]. Among existing evolutionary feature construction techniques, genetic programming-based feature construction techniques have shown promising results in constructing expressive features for improving the predictive performance of existing machine learning algorithms [1, 5].

Genetic programming (GP) is a gradient-free and variable-length evolutionary algorithm [6]. For evolutionary feature construction tasks, tree-based GP is particularly suitable since the constructed features can be naturally represented by GP trees. However, a single tree may not contain all the necessary information to fit a good machine learning model. Therefore, multi-tree GP has been developed to achieve better performance. The general idea of multi-tree GP is to simultaneously construct multiple relatively simple, informative, and complementary features for enhancing the learning performance of a machine learning algorithm. Multi-tree GP has been successfully applied to improve the predictive performance of machine learning algorithms on regression [7] and classification tasks [2].

Although multi-tree GP has achieved great success in feature construction tasks, the existing crossover operators in multi-tree GP rarely consider the complementarity between features during crossover. Formally, assuming $\mathcal{L}$ is the loss function, $\mathcal{A}$ is the learning algorithm, $\phi_i(X), \phi_j(X)$ are two constructed features, if $\mathcal{L}(\mathcal{A}, \{\phi_i(X), \phi_j(X)\}) > max\{\mathcal{L}(\mathcal{A}, \phi_i(X)), \mathcal{L}(\mathcal{A}, \phi_j(X))\}$, then these two features are considered as complementary. For example, if a pair of constructed features $\phi_i(X), \phi_j(X)$ have symmetrical semantics with respect to the target semantics Y, then the linear regression technique can determine a coefficient β to make the equation $\phi_i(X) + \beta\phi_j(X) = Y$ hold, thereby achieving the exact target semantics. For a traditional crossover operator in multi-tree GP, it randomly selects two GP trees ϕ_i^a, ϕ_j^b from two parents Φ_a, Φ_b for crossover, no matter whether they are complementary or not. To discover a set of complementary features, it is sensible to utilize a geometric semantic crossover operator to assemble GP trees under the guidance of semantics. The primary focus of geometric semantic crossover in this context is not only to consider ϕ_i^a, ϕ_j^b during crossover, but also considering interaction between features such as ϕ_k^a in Φ_a. More specifically, if ϕ_i^a in Φ_a is complementary to ϕ_k^a in Φ_a, as illustrated in Fig. 1a, it is advisable to avoid disrupting ϕ_i^a and ϕ_k^a during crossover. Conversely, if the complementarity between ϕ_i^a and ϕ_k^a is not strong, and ϕ_i^a in Φ_a is complementary to ϕ_j^b in Φ_b, as depicted in Fig. 1b, performing a crossover to replace ϕ_k^a in Φ_a with ϕ_j^b in Φ_b is an optimal strategy.

In order to achieve this goal, we aim to propose a geometric semantic macro-crossover operator (GSMX) for evolutionary feature construction.[1] Different from the traditional crossover operator which focuses on exchanging genetic materials between different GP trees, the GSMX operator focuses on how to combine GP trees from different parents to get a set of relevant and complementary features. GSMX as a macro-crossover operator will perform crossover on the tree level instead of the subtree level. Specifically, we focus on the following objectives:

[1] Source code: https://tinyurl.com/MAPMX-GPFC



(a) Complementary features ϕ_i^a and ϕ_k^a in the same individual (Φ_a).

(b) Complementary features ϕ_i^a and ϕ_j^b in different individuals (Φ_a, Φ_b).

Fig. 1 Two situations of complementarity need to be considered for crossover in multi-tree GP, where the complementarity is measured by the degree of symmetry with respect to the target semantics

- To explicitly integrate relevant and complementary features, we develop a macro geometric semantic crossover operator that considers the geometric semantic relationships between GP trees in the crossover process.
- To make the constructed features relevant to the target label and complementary to each other, we propose two ways of implementing the GSMX operator: one based on the angle-driven selection operator (ADS) and the other on the multi-dimensional archive of phenotypic elites algorithm (MAP-Elites).
- To make GSMX compatible with traditional crossover operators, we develop two modes to apply the GSMX operator in GP, which are parallel mode and sequential mode. The difference between them depends on whether an individual can be modified simultaneously by both GSMX and traditional crossover operations in one generation.

The remainder of this paper is organized as follows: Sect. 2 presents related work on semantic GP and crossover operators for multi-tree GP. In Sect. 3, the proposed approach is introduced in detail. Section 4 describes the experimental settings. Section 5 reports the experimental results to demonstrate the effectiveness of the proposed algorithm. Section 6 further analyzes the effectiveness of the proposed operator under different settings. Finally, Sect. 7 provides the conclusion and future work.

2 Related work

2.1 Geometric semantic operators

Over recent years, geometric semantic operators have become a popular topic in GP [8]. Unlike traditional genetic operators that randomly exchange genetic materials,

geometric semantic operators crossover or mutate genetic materials to fulfill the desired semantics. A pioneering work on geometric semantic operators is the geometric semantic crossover (GSX) and geometric semantic mutation (GSM) [9]. GSX combines two GP trees to form a new GP tree, which ensures that the semantics of the offspring lie on the line connecting the semantics of the parents. Formally, supposing the semantics of two parents as $\vec{P}_1$ and $\vec{P}_2$, the semantics of the offspring $\vec{O}$ should satisfy $\|\vec{P}_2 - \vec{P}_1\| = \|\vec{O} - \vec{P}_1\| + \|\vec{P}_2 - \vec{O}\|$. GSM combines a GP tree with a randomly generated GP tree to perturb the semantics of a GP tree within a certain radius r. Formally, supposing the semantics of the parent as $\vec{P}$, the semantics of the offspring $\vec{O}$ should satisfy $\|\vec{O} - \vec{P}\| \leq r$. However, a problem with GSX and GSM is that using them repeatedly will make GP trees grow exponentially. To solve this issue, maintaining a table in memory to store semantics can significantly reduce the computational cost [10, 11], and hashing can be used to merge identical subtrees, alleviating the exponential growth problem of the GSX operator [12]. Nonetheless, the size of the final GP tree can still be up to 10,000 nodes. To obtain an interpretable model, approximate semantic crossover operators have been developed to address the exponential growth problem [13]. These operators search for subtrees that satisfy desired semantics from an external library to ensure that the offspring satisfy the desired semantics of geometric crossover [14], back-propagated target semantics [15], and projected semantics [16]. The approximated semantic crossover operators not only reduce model sizes but also improve the generalization performance of the discovered model [17]. Alternatively, performing geometric semantic crossover on the subtree level instead of the tree level can also alleviate the exponential growth problem without an external library [18]. Moreover, geometric semantic operators have been studied from the perspective of variation rate self-tuning [19], efficient large data mining [20], and binary classification using the sigmoid function [21]. Although geometric semantic operators have been widely studied, geometric semantic operators for multi-tree GP still lack investigation and are worth exploring.

2.2 Crossover operators in multi-tree genetic programming

Unlike single-tree GP, which has only one GP tree in each individual in the population, multi-tree GP has multiple candidates to be crossed and mutated, making the problem more complex. For multi-tree GP, the simplest crossover operator randomly selects an index across a parent individual and then uses the traditional crossover operator on the selected parent trees [1, 22]. Although these methods are intuitive, randomly selecting a GP tree to crossover may be more likely to disrupt good GP trees or to leverage bad GP trees. In some algorithms, a random crossover operator has been proposed to cross two whole GP trees rather than two subtrees [22, 23], but these operators do not explicitly consider the semantics of GP trees, making it difficult to maintain complementary features. In the data imputation task [24], the importance of GP trees is considered when performing crossover, and experimental results show that using a higher probability to mutate important features is the most effective approach. In a job shop scheduling task, a self-competitive crossover and mutation operator [25] is designed to modify the worst GP tree in each individual.

Although several crossover operators have been designed for multi-tree GP, these crossover operators focus on exploiting important GP trees based on the importance value of GP trees, whereas the semantics of each GP tree are not explicitly considered, leaving room for improving effectiveness using semantics.

2.3 Evolutionary feature construction

Evolutionary feature construction is an automatic feature construction technique that has a history of more than 20 years [26] and has shown excellent performance in recent years [27, 28]. Generally, evolutionary feature construction techniques can be categorized into three types based on the evaluation method: filter-based, wrapper-based, and embedded methods. Filter-based feature construction methods use simple metrics instead of a learning algorithm to measure the quality of features with respect to the target labels, such as information gain [29], Pearson correlation [30], and other information-theoretic measures [31]. The filter-based feature construction methods are fast, and the constructed features can generalize well to different machine learning algorithms. However, since they do not rely on any machine learning algorithms, performance improvement might be limited. In contrast, wrapper-based feature construction methods evaluate features based on specific machine learning algorithms, such as a decision tree [7], a linear model [22], a random forest [1], or a heterogeneous ensemble model [32, 33]. They are more time-consuming than filter-based methods, but they can achieve better performance on the specific algorithm. Finally, the embedded methods embed feature construction in the learning process [34]. Representative examples include GP-based symbolic regression methods, which construct features and models simultaneously using the GP method.

3 The proposed method

3.1 Overall framework

In this paper, we propose a geometric semantic crossover operator (GSMX) for a multi-tree GP-based evolutionary feature construction algorithm. The overall algorithm consists of four parts, as follows:

- Population Initialization: The population initialization stage randomly generates N GP individuals, each with m trees, using the ramped half-and-half method, where m GP trees represent m constructed features.
- Solution Evaluation: The solution evaluation stage uses all GP trees $\{\phi_1 \dots \phi_m\}$ to construct a set of features $\{\phi_1(X) \dots \phi_m(X)\}$, with ridge regression determining the coefficients of these features. To encourage generalization on unseen data, a leave-one-out cross-validation method is applied to the training data to obtain predictions $\{\hat{y_1}, \dots, \hat{y_n}\}$ for fitness evaluations. In this paper, squared errors between predicted values and target values, i.e., $\{(\hat{y_1} - y_1)^2, \dots, (\hat{y_n} - y_n)^2\}$, are

defined as fitness cases. Finally, the mean value of all fitness cases determines the fitness value for model selection.

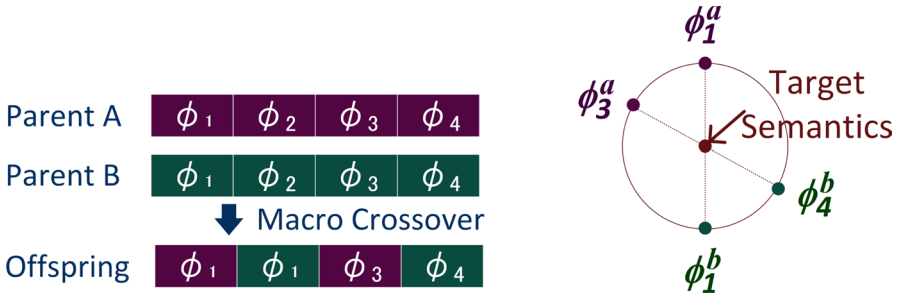
- Parent Selection: Parents are selected using the automatic ϵ-lexicase selection operator [35]. The general idea of lexicase selection is to initialize a candidate pool P based on all individuals in the current population, and then construct multiple filters based on the fitness cases to filter out individuals in the candidate pool until only one individual remains, which is then selected as the parent. In this paper, the filter is defined as $min_{p \in P} \mathcal{L}_k(p) + \epsilon_k$ [36], where $\mathcal{L}_k(p)$ represents the fitness value of individual p on training instance k, and ϵ_k represents the median absolute deviation of fitness values on training instance k for all individuals in the candidate pool.
- Offspring Generation: Once two parents are selected, the offspring are generated using the random subtree crossover and mutation operators. For an m-tree GP, the crossover and mutation operators are repeated $\frac{m}{/}2$ and m times , respectively, to encourage the exploration of GP trees.

Although the simplest way to crossover parents Φ_a and Φ_b is to exchange subtrees ψ_a and ψ_b between randomly selected GP trees ϕ_a and ϕ_b in Φ_a and Φ_b, a more effective approach is to swap GP trees based on their semantics to take relevancy and complementarity into consideration. Specifically, the goal is to best approximate the target semantics with the weighted average of all GP trees in an individual : $\sum_{i=1}^{m} w_i \phi_i(X) \approx y$, where weights w_i are determined by linear regression.

To achieve this goal, the proposed geometric semantic crossover operator (GSMX) crosses two GP individuals based on their semantics $\{\phi_1^a(X), \ldots, \phi_m^a(X)\} \cup \{\phi_1^b(X), \ldots, \phi_m^b(X)\}$ to obtain a set of relevant and complementary GP trees $\{\phi'_1 \ldots \phi'_m\}$ as a new individual that approximates the target semantics. Figure 2 presents an example of the ideal behavior of GSMX in genotype and semantic space. Figure 2a shows the view from the genotype space, where GSMX selects $\{\phi_1^a, \phi_3^a\}$ from parent A and $\{\phi_1^b, \phi_4^b\}$ from parent B to form an offspring. The reason for selecting these two pairs of trees is illustrated by the geometric relationship between the trees in Fig. 2b. As it shows, ϕ_1 and ϕ_3 in parent A, and ϕ_1 and ϕ_4 in parent B, are complementary with respect to the target semantics, as their cosine similarities are exactly -1. Therefore, selecting these four trees allows the linear combination of the constructed features to equal the target semantics. The following sections will introduce how to implement such an operator.

3.2 Sequential mode and parallel mode

The GSMX operator is compatible with the traditional crossover and mutation operators, and it can be applied in either a sequential mode or a parallel mode. The key difference between the two modes is whether the GSMX operator and the traditional variation operator can be used on the same individual. The purpose of having



(a) Genotype Space. (b) Semantic Space.

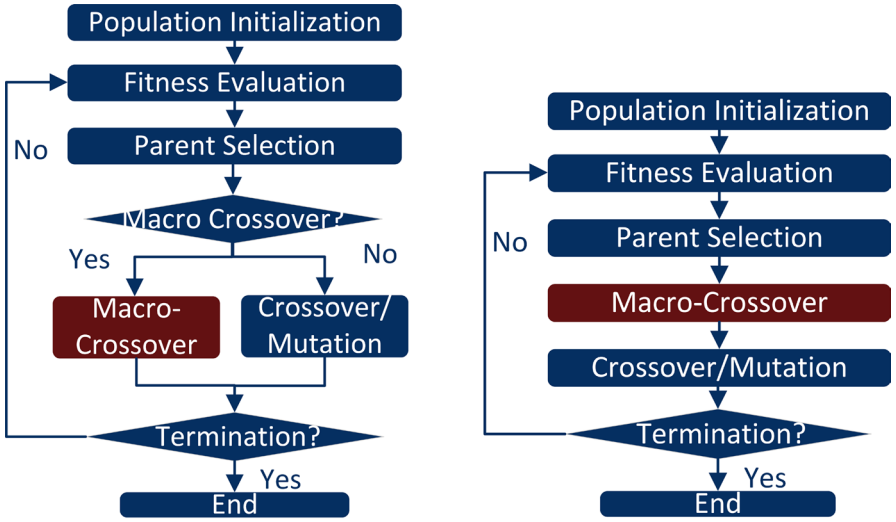
Fig. 2 View of GSMX from genotype space and semantic space

these two modes is to explore the best way to combine GSMX with the traditional crossover and mutation operators. An illustrative example is presented in Fig. 3. The details for these two modes are described as follows:

- Parallel Mode: The GSMX operator is applied in parallel with standard crossover and mutation operators. The choice between using the GSMX operator or standard genetic operators is determined by a predefined probability p_{GSMX}. For each pair of parents, based on the probability p_{GSMX}, either GSMX or standard operators are chosen, resulting in a new population generated by a combination of GSMX and standard operators. Another key aspect is that the GSMX operator produces only a single offspring from two parents, as the set of less-important features is discarded.
- Sequential Mode: The GSMX operator is invoked with a probability of p_{GSMX} before applying standard genetic operators. Firstly, this mode invokes lexicase selection four times to select two pairs of parents $\Phi_A, \Phi_B, \Phi_C, \Phi_D$. Then, it combines relevant and complementary features from two pairs of parents $\{\Phi_A, \Phi_B\}, \{\Phi_C, \Phi_D\}$ to produce two offspring Φ_{O1}, Φ_{O2}. Finally, traditional genetic operators can make further changes on Φ_{O1}, Φ_{O2} to fine-tune GP trees.

3.3 Two implementations of GSMX

In this section, we presents two alternative approaches for implementing macro-crossover in genetic programming: MAP-Elites and Angle-Driven Selection. Both methods aim to consider complementarity by measuring the cosine similarity between GP trees during macro-crossover, thereby improving the search effectiveness.



(a) Parallel Mode. (b) Sequential Mode.

Fig. 3 Workflow of different modes for GSMX

3.3.1 MAP-elites for macro-crossover

The goal of the GSMX operator is to form a set of relevant and complementary GP trees in the offspring individuals. To achieve this goal, we propose using MAP-Elites, a niching algorithm that has shown promising results in many quality-diversity optimization tasks [37, 38]. The general idea of MAP-Elites is to map individuals into a low-dimensional behavioral space and then discretize the behavioral space into multiple cells. On each cell, the elite can be chosen according to their fitness values. By identifying elites located at different regions of the behavior space, we can maintain a high-quality and complementary set of GP trees as the offspring.

The process of using MAP-Elites for GSMX can be divided into four steps:

- *Step 1*. Data processing: At the beginning of MAP-Elites, semantics of GP trees are centralized according to the target semantics, which ensures that the cosine similarity is calculated using the target semantics as a central point.
- *Step 2*. Dimensionality reduction: The second step is dimensionality reduction. This is necessary because the curse of dimensionality can make the number of elite cells very large and fail to select elites. In this paper, we use cosine kernel principal component analysis (KPCA) to reduce the semantic space to a two-dimensional space, which has shown superior performance to other dimensionality reduction methods in MAP-Elites for evolutionary ensemble learning

[39]. To be specific, the reason for using a cosine kernel is because using cosine similarity as the distance metric in MAP-Elites can make it explicitly consider the complementarity of different features. In an ideal scenario, if two features that have an exact cosine similarity of -1 with respect to the target semantics have been identified, they can be used to accurately estimate the target semantics through a linear combination method.

- *Step 3*. Discretization: After reducing the semantic space from $\mathbb{R}^n$ to a behavioral space of $\mathbb{R}^2$, MAP-Elites discretizes the behavioral space into a $k \times k$ cell with equal distance between the cells.
- Step 4 Elite Selection: For each cell $g_{i,j}$, where $i, j \in [1, k]$, MAP-Elites selects the most important GP tree $e_{i,j}$ in $g_{i,j}$, where the importance value of each GP tree ϕ is determined by the absolute value of its coefficient β_ϕ in the linear model. If no GP tree is in the cell, $g_{i,j}$, $e_{i,j}$ is marked as a nonelement. To avoid the scale difference of constructed features influencing the magnitude of coefficients, all constructed features are standardized before training the linear model.

At the end of the MAP-Elites selection, we rank all elements in the grid $\{e_{i,j} \mid i, j \in [1, k]\}$ according to their importance values in a descending order. Then, we simply select the top-m trees to form an offspring individual with m GP trees. However, the number of elites in the elite grid may be less than m, making the above process unable to select enough features. If this occurs, GSMX repeats the MAP-Elites process until enough features have been selected.

3.3.2 Angle-driven selection for macro-crossover

The GSMX operator is a general concept and there are multiple ways to implement it. The Angle-Driven Selection (ADS) is another way to implement the GSMX operator, which consists of three steps:

- *Step 1*. Redundant feature elimination: At the start of the macro-crossover process, redundant features are eliminated to ensure that no semantically equivalent features appear in the offspring. For speedup, the hash value $h(\phi(X))$ is calculated using SipHash [40] for compressing the semantics of each feature ϕ, and then the semantic equivalence is checked based on hash values.
- *Step 2*. Roulette wheel selection: In this stage, the roulette wheel selection operator is used to select a feature ϕ based on the normalized importance values of all features. The importance value is the same as the importance value defined in MAP-Elites, which is the absolute value of the corresponding coefficient in the linear model.
- *Step 3*. Angle-driven pairing: Based on the selected feature ϕ, the angle-driven selection mechanism selects the feature ϕ' with the maximal cosine distance with respect to ϕ. This is because the feature with the maximal cosine distance is the feature that has the most symmetrical semantics with respect to the target semantics. In the ideal scenario, the angle-driven pairing process can identify a pair of features that have a cosine similarity of -1. This enables an exact estimation of the target semantics by using a linear combination of the complementary features.

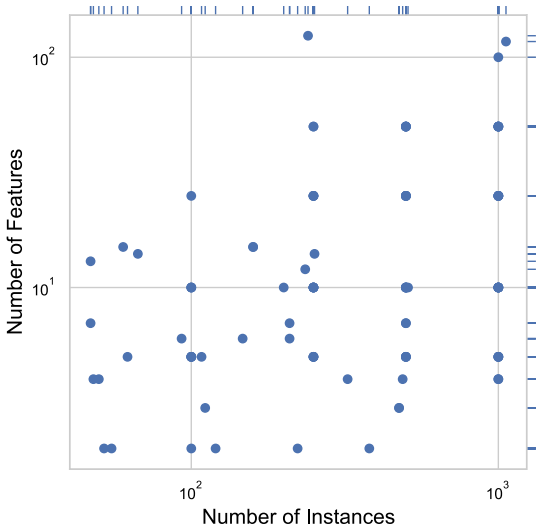


Fig. 4 Properties of the regression benchmark

To generate a child individual, Steps 2 and 3 are executed repeatedly $\frac{m}{2}$ times until m features have been selected.

4 Experimental settings

4.1 Datasets

In this paper, we conduct experiments on Penn Machine Learning Benchmark (PMLB) [41]. Due to limited computational resources, in ablation studies, 98 datasets with less than 2000 data instances in PMLB are used. The properties of these datasets are presented in Fig. 4. Figure 4 shows that the largest dataset has 1059 instances, whereas the smallest dataset only has 47 instances. As for the number of features, they are within the range of 2 to 124. When comparing with machine learning algorithms, all 120 regression datasets from PMLB are used, resulting in the maximum number of instances in the experimental datasets being 1,000,000.

4.2 Parameter settings

The parameter settings are shown in Table 1, and most of them are common settings used in the existing literature. The population size is set to 30 times the number of features D, with a maximum limit of 300. This is because high-dimensional datasets require more computational resources to discover good high-order features. To avoid the problem of zero division, the analytical quotient (AQ) operator [42] is used to replace the division operator.

Table 1 Parameter settings for GP

Parameter	Value
Population Size	$30 \times D$
Maximal Number of Generations	100
Crossover and Mutation Rates	0.9 and 0.1
Maximum Tree Depth	8
Maximum Initial Tree Depth	2
Number of Trees in Each Individual	20
Elitism (Number of Individuals)	1
Macro-Crossover Rate	0.2
Functions	Add, Sub, Mul, AQ, Sin, Cos, Abs, Max, Min, Negative

4.3 Comparison methods

4.3.1 Macro-crossover operators

This paper proposes two methods for implementing the Geometric Semantic Macro-Crossover (GSMX) algorithm: Angle-Driven Selection (ADS) and MAP-Elites. Both methods can be applied in either sequential or parallel mode, resulting in four macro-crossover operators. These operators are named MAP-Elites Macro-Crossover Operator in parallel (MAPMX-P), Angle-Driven Macro-Crossover Operator in parallel (ADMX-P), MAPMX in sequential (MAPMX-S), and ADMX in sequential (ADMX-S). The standard Genetic Programming (STD-GP) serves as the baseline for comparison.

4.3.2 Micro-crossover operators

As GSMX is a macro-crossover operator, it is compatible with traditional crossover operators in GP. In this paper, we also investigate the performance gain brought by using MAPMX-P in conjunction with different micro-crossover operators. In the default implementation, we use a random crossover operator to crossover GP trees by randomly selecting a GP tree in each GP individual. To ensure sufficient variation for m GP trees in each individual, we repeat the random crossover operator m times under the control of crossover probability. However, existing literature offers more advanced crossover operators for multi-tree GP than the random selection operator. In this paper, we consider two micro-crossover operators:

- Self-Competitive Crossover (SCX) [25]: This is a biased crossover operator [43] that replaces a subtree of the least important tree in parent A with a subtree of the most important tree in parent B, while leaving the most important subtree in parent B intact. This biased crossover operator ensures exploration without disrupting important GP trees.

- Probability Best Index Crossover (PBIX) [24]: This crossover operator selects the tree to be crossed with a probability proportional to the importance value of each tree. The importance of each tree in PBIX is defined as the Kolmogorov-Smirnov distance between the features constructed by a GP tree in the source domain and the target feature in the target domain [24].

It is important to note that both operators require defining the important trees in each individual. In this paper, the importance value of each GP tree is defined as the absolute value of the coefficient for each GP tree in each individual. While the original implementation of PBIX and SCX operators are invoked only once for multi-tree GP, it is possible to invoke PBIX operator m times to facilitate the exchange of genetic materials. Consequently, this section considers five crossover operators:

- SCX: SCX uses the most important and the most unimportant GP tree in each individual.
- PBIX: Performing PBIX once.
- PBIX-All: Performing PBIX m times.
- RIX: Performing Random Crossover once.
- RIX-All: Performing Random Crossover m times.

4.4 Evaluation protocol

The evaluation protocol used in this paper follows the conventional protocol used in the machine learning domain. Specifically, datasets are split into training and test sets in a ratio of 80:20. For each combination of algorithms and datasets, experiments are conducted for 30 independent runs with different random seeds. Based on the experimental results, a Wilcoxon signed rank test is performed with a significance level of 0.05.

5 Experimental results

5.1 Comparisons of macro-crossover methods

Table 2 provides a statistical comparison of the test R^2 scores of four macro-crossover operators, namely MAPMX-P, ADMX-P, MAPMX-S, and ADMX-S, with the standard GP (STD-GP). The results show that MAPMX-P outperforms STD-GP on 37 datasets while only performing worse on 5 datasets, validating the effectiveness of the proposed semantic macro-crossover operator in multi-tree GP. In contrast, the angle-driven macro-crossover operator only outperforms STD-GP on 18 datasets, indicating that the MAP-Elites selection mechanism is crucial for the improvement of the effectiveness of the semantic macro-crossover operator. Additionally, changing from parallel mode to sequential mode can degrade the performance of both ADMX-P and MAPMX-P. MAPMX-S is significantly worse than MAPMX-P on 27 datasets, whereas it is only better on 5 datasets. As for

Table 2 Statistical comparisons of **test R^2 scores** using different macro-crossover operators.

	ADMX-P	MAPMX-S	ADMX-S	STD-GP
MAPMX-P	23(+)/71(~)/4(-)	27(+)/66(~)/5(-)	35(+)/57(~)/6(-)	37(+)/56(~)/5(-)
ADMX-P	–	6(+)/90(~)/2(-)	9(+)/88(~)/1(-)	18(+)/78(~)/2(-)
MAPMX-S	–	–	8(+)/87(~)/3(-)	22(+)/74(~)/2(-)
ADMX-S	–	–	–	11(+)/87(~)/0(-)

("+","~", and "-" indicate using the method in a row is better than, similar to or worse than using the method in a column.)

ADMX-S, it performs worse than ADMX-P on 9 datasets, while is only better on 1 dataset. Thus, it is not recommended to use sequential mode for MAPMX and ADMX.

To gain a deeper understanding of the success of GSMX, the evolutionary plot of test R^2 scores with respect to different macro-crossover operators on four representative datasets is presented in Fig. 5. The plot demonstrates that GSMX significantly improves the test R^2 scores over standard GP in early generations. This can be explained by the MAP-Elites mechanism in GSMX tries to explicitly maintain a set of relevant and complementary features. Thus, despite features in early generations have low predictive performance, they can still work well as long as they are complementary with each other with respect to the target semantics.

Moreover, Table 3 shows that GP with ADMX leads to significantly better fitness values than standard GP on 94 out of the 98 datasets and is not worse on any dataset. Combining these results with those presented in Table 2, we can conclude the performance degradation of GSMX compared to STD-GP on 5 datasets might be caused by overfitting. For example, GSMX outperforms STD-GP on the "sleuth case" dataset [44], which has 6 features but only 147 instances, in terms of fitness values, but performs worse on test R^2 scores. Thus, if the overfitting problem of GSMX can be alleviated in the future, we can expect that GSMX will further improve the test R^2 scores of GP.

It is worth noting that MAPMX-P performs well on both low-dimensional and high-dimensional datasets. As shown in Fig. 6, MAPMX-P still outperforms STD-GP on datasets with 50 dimensions and even with 100 dimensions.

5.2 Comparisons of micro-crossover methods

The experimental results for the micro-crossover operator are presented in Table 4. The results indicate that MAPMX-P can significantly improve test R^2 scores when used with any of the micro-crossover operators. For instance, when applied with the PBIX operator, MAPMX-P significantly improves the prediction performance on 41 datasets while only exhibiting worse performance on 6 datasets. Interestingly, applying micro-crossover operators multiple times can be better than using them once only. One potential reason is that applying

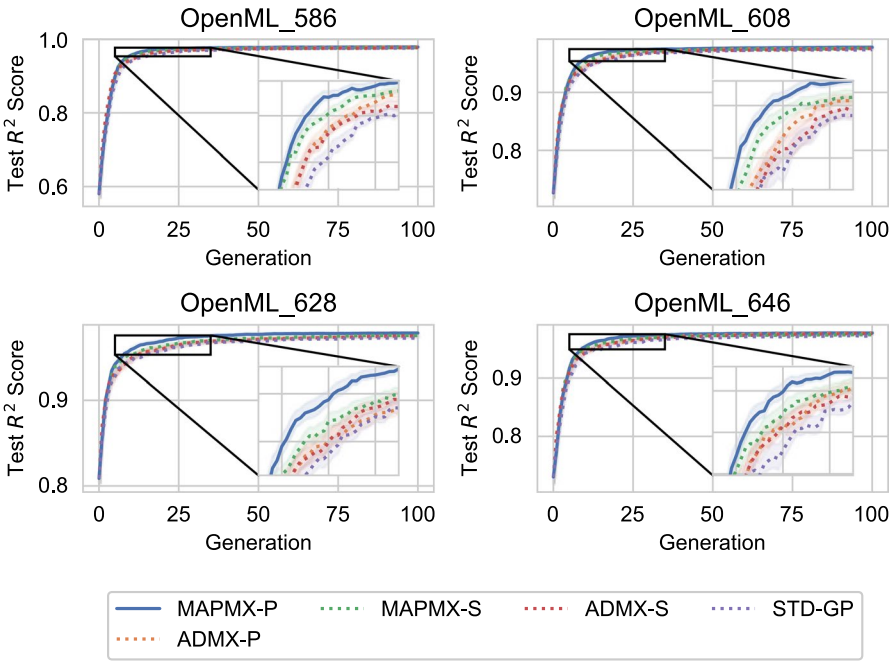


Fig. 5 Evolutionary plot of test R^2 scores for different macro-crossover operators

Table 3 Statistical comparisons of **best fitness values** on training set using different macro-crossover operators

	ADMX-P	MAPMX-S	ADMX-S	STD-GP
MAPMX-P	91(+)/5(~)/2(-)	91(+)/7(~)/0(-)	91(+)/6(~)/1(-)	94(+)/4(~)/0(-)
ADMX-P	–	8(+)/77(~)/13(-)	60(+)/38(~)/0(-)	80(+)/17(~)/1(-)
MAPMX-S	–	–	62(+)/32(~)/4(-)	78(+)/19(~)/1(-)
ADMX-S	–	–	–	55(+)/42(~)/1(-)

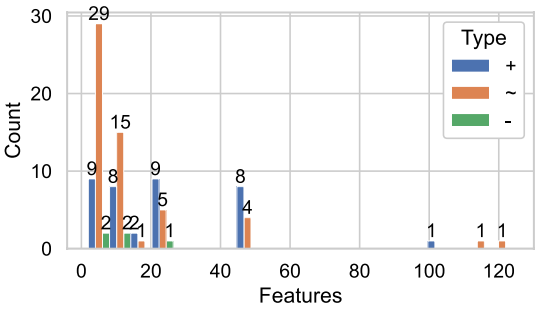


Fig. 6 Summary of the statistical comparison between MAPMX-P and STD-GP on 98 datasets with respect to the number of features

Table 4 Statistical comparisons of **test** R^2 **scores** with different micro-crossover operators and using GSMX or not

	RIX	PBIX	RIX-ALL
SX	6(+)/77(~)/15(-)	9(+)/79(~)/10(-)	0(+)/86(~)/12(-)
RIX	–	8(+)/83(~)/7(-)	1(+)/87(~)/10(-)
PBIX	–	–	0(+)/89(~)/9(-)
RIX-ALL	–	–	–
PBIX-ALL	–	–	–
MAPMX+SX	–	–	–
MAPMX+RIX	–	–	–
MAPMX+PBIX	–	–	–
MAPMX+RIX-ALL	–	–	–

	PBIX-ALL	MAPMX+SX	MAPMX+RIX
SX	15(+)/65(~)/18(-)	4(+)/60(~)/34(-)	3(+)/54(~)/41(-)
RIX	27(+)/58(~)/13(-)	11(+)/52(~)/35(-)	6(+)/50(~)/42(-)
PBIX	18(+)/74(~)/6(-)	12(+)/52(~)/34(-)	12(+)/40(~)/46(-)
RIX-ALL	24(+)/72(~)/2(-)	15(+)/65(~)/18(-)	12(+)/54(~)/32(-)
PBIX-ALL	–	14(+)/38(~)/46(-)	17(+)/38(~)/43(-)
MAPMX+SX	–	–	4(+)/78(~)/16(-)
MAPMX+RIX	–	–	—
MAPMX+PBIX	–	–	—
MAPMX+RIX-ALL	–	–	—

	MAPMX+PBIX	MAPMX+RIX-ALL	MAPMX+PBIX-ALL
SX	3(+)/56(~)/39(-)	1(+)/39(~)/58(-)	1(+)/35(~)/62(-)
RIX	3(+)/49(~)/46(-)	3(+)/34(~)/61(-)	1(+)/37(~)/60(-)
PBIX	6(+)/51(~)/41(-)	4(+)/36(~)/58(-)	2(+)/40(~)/56(-)
RIX-ALL	9(+)/66(~)/23(-)	5(+)/56(~)/37(-)	5(+)/55(~)/38(-)
PBIX-ALL	15(+)/35(~)/48(-)	10(+)/36(~)/52(-)	10(+)/31(~)/57(-)
MAPMX+SX	2(+)/79(~)/17(-)	2(+)/49(~)/47(-)	2(+)/50(~)/46(-)
MAPMX+RIX	5(+)/90(~)/3(-)	2(+)/72(~)/24(-)	3(+)/75(~)/20(-)
MAPMX+PBIX	–	1(+)/73(~)/24(-)	2(+)/71(~)/25(-)
MAPMX+RIX-ALL	–	–	4(+)/90(~)/4(-)

micro-crossover operators multiple times can provide sufficient variation to discover good enough solutions within a limited number of generations. Conversely, applying micro-crossover operators only once may only modify an unimportant feature of an individual, with little impact on fitness improvement. Thus, RIX-ALL outperforms RIX on 10 datasets and is only worse on 1 dataset. Based on this advantage, combining MAPMX-P with RIX-ALL achieves top-notch performance compared to all competitors. Specifically, this operator is significantly better than all other crossover operators except for combining MAPMX-P with PBIX-ALL, which has a similar performance.

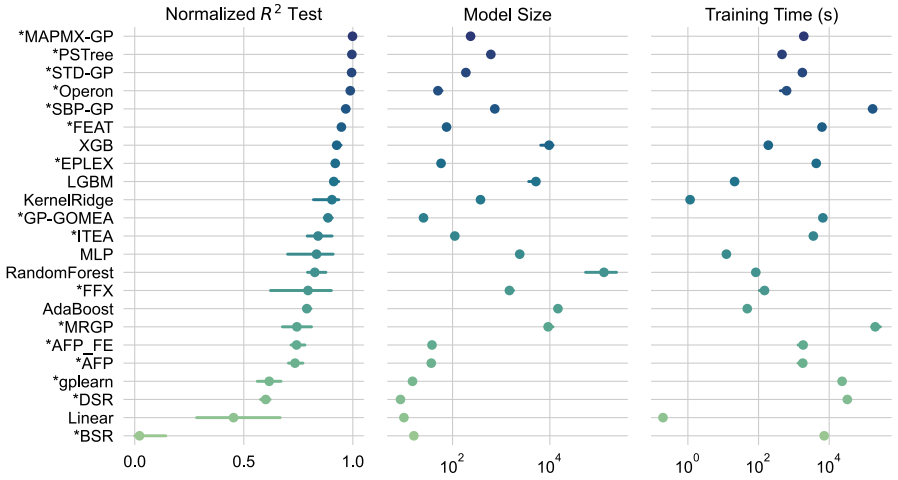


Fig. 7 Median normalized R^2 scores, model sizes and training time on 120 regression problems

5.3 Comparisons with machine learning algorithms

To fully investigate the performance improvement brought by using the GSMX operator, we name a standard multi-tree GP algorithm (STD-GP) with the proposed MAPMX-P operator as MAPMX-GP, against 22 state-of-the-art symbolic regression and machine learning algorithms on 120 datasets. Figure 7 presents the median normalized test R^2 score, model size, and training time over the 120 datasets. Here, the test R^2 scores are normalized to eliminate the differences in difficulty between different datasets. Specifically, the test R^2 scores are normalized using $\frac{s-s_{min}}{s_{max}-s_{min}}$ based on the maximum score s_{max} and the minimum score s_{min} that the 23 algorithms can achieve on each dataset. The experimental results show that MAPMX-GP achieves the best normalized test R^2 scores among the 23 algorithms and has a smaller model size than PS-Tree, which is the state-of-the-art symbolic regression algorithm in terms of test R^2 scores. With respect to the training time, although MAPMX-GP takes more time to train than PS-Tree, its training time is shorter than that of SBP-GP. These results indicate that MAPMX-GP achieves top-notch performance without incurring overly complex models and excessive time consumption. Figure 8 presents the statistical comparison results of normalized test R^2 scores with Benjamini-Hochberg correction [45] to make the results of multiple testing more reliable. The experimental results show that MAPMX-GP significantly outperforms other methods. It is worth noting that STD-GP is the same as MAPMX-GP in generation operators, selection operators and the fitness evaluation function, with the only difference being the lack of use of the MAPMX-P operator. Figures 7 and 8 demonstrate that MAPMX-GP significantly outperforms PS-Tree, whereas STD-GP is significantly worse than PS-Tree, indicating that the MAPMX-P operator is the key component for the outstanding performance of MAPMX-GP. Moreover, MAPMX-GP has the same order of magnitude in computational cost as STD-GP. This is reasonable since MAPMX-GP does not require additional fitness evaluations,

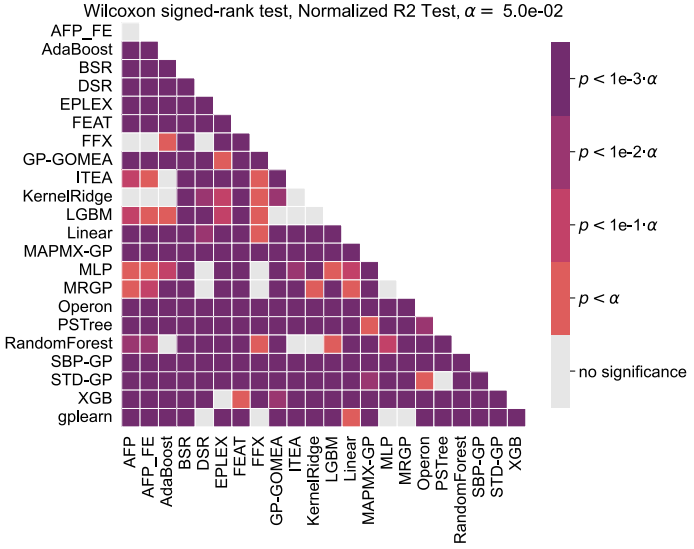


Fig. 8 Pairwise statistical comparisons of normalized R^2 test scores on 120 regression problems

and the extra computation cost only occurs during the crossover process. Also, although MAPMX-GP is based on the philosophy of geometric semantics, it does not suffer from the problem of exponential growth, because MAPMX-GP discards some GP trees during the crossover process. Thus, as shown in Fig. 8, it has the same order of magnitude in model size as STD-GP.

6 Further analysis

6.1 Analysis of success rate

To gain a comprehensive understanding of why GSMX outperforms other crossover operators and why macro-crossover is beneficial for multi-tree GP, evolutionary plots of successful crossover rates on four representative datasets for both the macro-crossover operator and the micro-crossover operator are presented in Figs. 9 and 10, respectively. The successful crossover rate is calculated as the ratio between the number of successful crossovers and the total number of crossovers. In both MAPMX-P and ADMX-P, a successful macro-crossover is defined as when the macro-crossover operator generates an offspring that is better than all its parents. In both MAPMX-S and ADMX-S, if an individual is produced by macro-crossover and is superior to its parents, it is deemed a successful macro-crossover, regardless of subsequent micro-crossover/mutation.

The success rate of macro-crossover operators is shown in Fig. 9. The experimental results demonstrate that MAPMX-P has a success rate of 0.5, even in later generations, which is significantly higher than the success rate of other macro-crossover operators. The larger success rate of MAPMX-P over ADMX-P confirms the

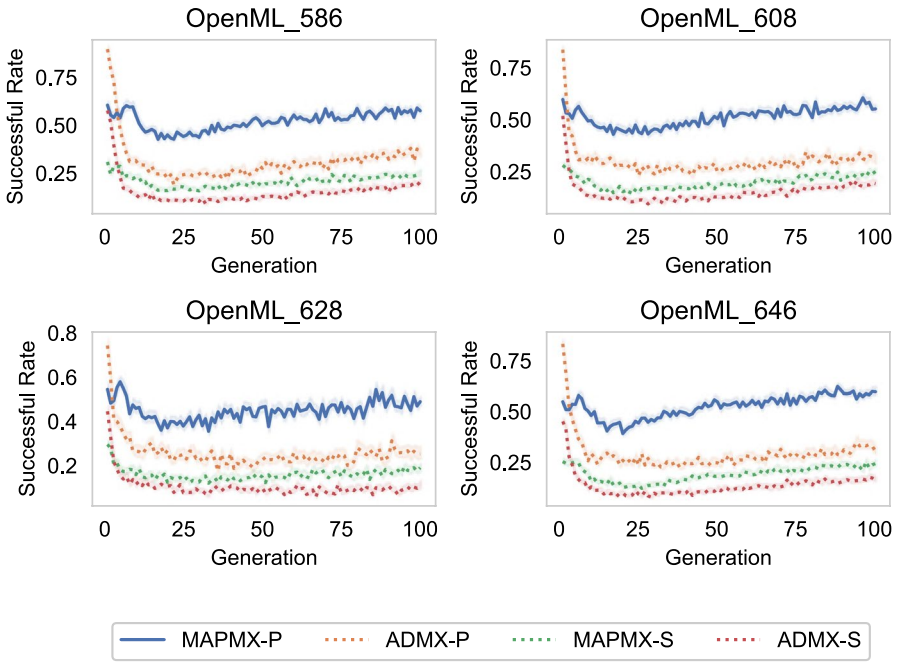


Fig. 9 Evolutionary plot of success rate for different macro-crossover operators

benefits of using MAPMX-P. Additionally, both MAPMX-S and ADMX-S have a lower success rate than their parallel variants, indicating that successful macro-crossover will be disrupted by the micro-crossover/mutation operators if applying micro-variation operators immediately after macro-crossover. Therefore, it is more appropriate to apply the macro-crossover variation in parallel with the micro-crossover operator to achieve good search effectiveness.

Figure 10 further shows the successful variation rates of the micro-variation operator. The results indicate that the success rate of the micro-variation operator does not vary significantly, regardless of the accompanying macro-crossover operator. Moreover, it is clear that the micro-crossover operator only has a success rate of 0.05, which is significantly lower than the success rate of macro-crossover operators. These results reveal that MAPMX-P has a higher crossover success rate than traditional genetic operators, meaning that applying a macro-crossover operator on a pair of parent individuals is more likely to improve them than applying traditional genetic operators. This can explain the performance improvement brought by MAPMX-P over other traditional genetic operators.

6.2 Impact of macro-crossover probability

In this paper, macro-crossover is recommended to be used in parallel with the standard crossover operator. In this case, balancing between the macro-level

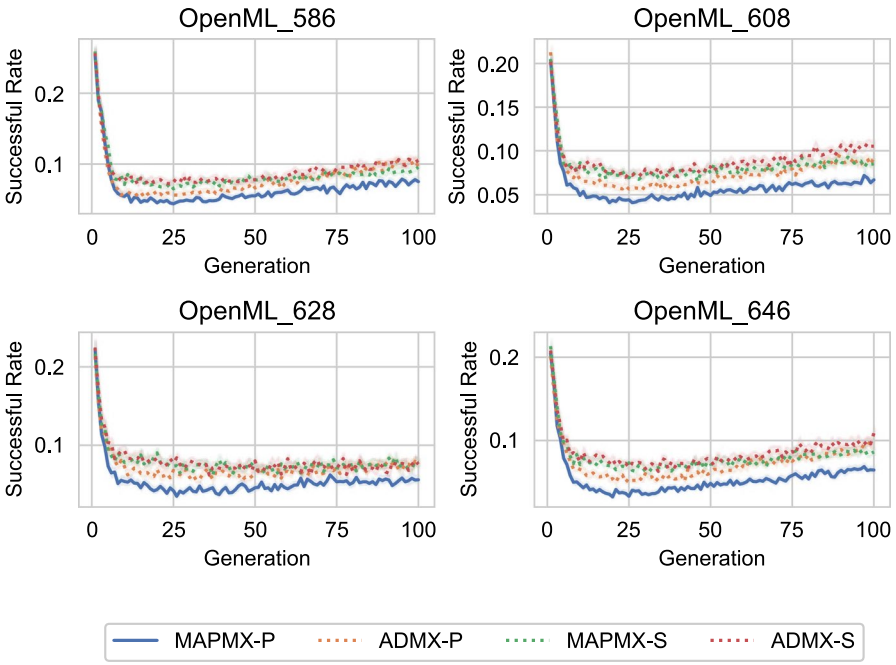


Fig. 10 Evolutionary plot of success rate for micro-crossover/mutation operators when combined with different macro-crossover operators

crossover and micro-level crossover is important. To investigate the impact of different crossover probabilities, we conduct experiments for four possible probabilities $\{0.01, 0.05, 0.2, 0.5\}$. Table 5 presents the results of using different macro-crossover probabilities. The test results for R^2 scores suggest that a probability of 0.2 is suitable for macro-crossover. Specifically, when compared to a probability of 0.05, a probability of 0.2 outperforms significantly on 14 datasets and underperforms on 4 datasets. Similarly, compared to a probability of 0.5, a probability of 0.2 shows significantly better results on 11 datasets while performing worse on 4 datasets. Nevertheless, using MAPMX-P is always better than not using it. As shown in Table 5, even a low probability like 0.01 can be significantly better than not using MAPMX-P on 8 out of the 98 datasets, without performing worse on any dataset.

7 Conclusions

The aim of this paper is to improve the search effectiveness of GP-based evolutionary feature construction methods by leveraging semantic information in crossover. This is achieved by proposing a GSMX operator that uses MAP-Elites to maintain a set of complementary features during crossover.

The performance of the proposed GSMX operator was validated on 98 datasets. Experimental results show that using the GSMX operator in multi-tree GP can significantly improve search effectiveness and yield better R^2 scores compared to using

Table 5 Statistical comparisons of test R^2 scores with different macro-crossover probabilities

	0.2	0.05	0.01	0
0.5	4(+)/83(~)/11(-)	18(+)/65(~)/15(-)	33(+)/52(~)/13(-)	47(+)/35(~)/16(-)
0.2	–	14(+)/80(~)/4(-)	30(+)/62(~)/6(-)	37(+)/56(~)/5(-)
0.05	–	–	21(+)/75(~)/2(-)	22(+)/76(~)/0(-)
0.01	–	–	–	8(+)/90(~)/0(-)

only the standard crossover operator. Furthermore, a comparison between four variants of the GSMX operator reveals that applying the GSMX operator in parallel with the standard crossover operator is the best choice, and using MAP-Elites in macro-crossover is superior to using angle-driven selection in macro-crossover. Finally, experimental results on the state-of-the-art symbolic regression benchmark demonstrate that multi-tree GP with GSMX can outperform all other 22 algorithms, highlighting the superior performance of using GSMX in multi-tree GP-based evolutionary feature construction algorithms. Moreover, experimental results with micro-crossover operators indicate that the proposed macro-crossover operator can significantly improve the performance when combined with five different types of micro-crossover operators.

In this work, GSMX is applied for regression tasks. In the future, it would be interesting to investigate whether such a macro-crossover operator can be applied to other applications, such as classification [2] and clustering problems [3]. For these problems, the lack of target semantics is the major challenge, and using state-of-the-art operational research algorithms to estimate target semantics may be a feasible approach.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s10710-023-09465-z.

Acknowledgements The authors would like to acknowledge the assistance of the volunteer evaluators and the helpful comments of the reviewers, which have significantly improved the paper.

Author Contributions Hengzhe Zhang, Qi Chen, and Mengjie Zhang designed the algorithm and experimental protocol. Hengzhe Zhang implemented the code and conducted the experiments. All authors analyzed the results. Hengzhe Zhang drafted the paper, and all authors edited the manuscript.

Funding This work was supported in part by the Marsden Fund of New Zealand Government under Contracts VUW1913, VUW1914, VUW2016, MBIE Data Science SSIF Fund under the contract RTVU1914, Huayin Medical under grant E3791/4165, and MBIE Endeavor Research Programme under contracts C11X2001 and UOCX2104.

Declarations

Conflict of interest The authors are not aware of any competing interests.

References

1. H. Zhang, A. Zhou, H. Zhang, An evolutionary forest for regression. IEEE Trans. Evol. Comput. **26**(4), 735–749 (2022)

2. B. Tran, B. Xue, M. Zhang, Genetic programming for multiple-feature construction on high-dimensional classification. Pattern Recogn. **93**, 404–417 (2019)
3. A. Lensen, B. Xue, M. Zhang, Genetic programming for evolving similarity functions for clustering: Representations and analysis. Evol. Comput. **28**(4), 531–561 (2020)
4. A. Lensen, M. Zhang, B. Xue, Multi-objective genetic programming for manifold learning: balancing quality and dimensionality. Genet. Program. Evolvable Mach. **21**(3), 399–431 (2020)
5. W. La Cava, J.H. Moore, Learning feature spaces for regression with genetic programming. Genet. Program. Evolvable Mach. **21**, 433–467 (2020)
6. J.R. Koza, Genetic programming as a means for programming computers by natural selection. Stat. Comput. **4**(2), 87–112 (1994)
7. H. Zhang, A. Zhou, H. Qian, H. Zhang, PS-Tree: a piecewise symbolic regression tree. Swarm Evol. Comput. **71**, 101061 (2022)
8. L. Vanneschi, M. Castelli, S. Silva, A survey of semantic methods in genetic programming. Genet. Program Evolvable Mach. **15**, 195–214 (2014)
9. A. Moraglio, K. Krawiec, C.G. Johnson, Geometric semantic genetic programming. In: International Conference on Parallel Problem Solving from Nature. pp. 21–31. Springer (2012)
10. L. Vanneschi, M. Castelli, L. Manzoni, S. Silva, A new implementation of geometric semantic GP and its application to problems in pharmacokinetics. In: Genetic Programming: 16th European Conference, EuroGP 2013, Vienna, Austria, April 3-5, 2013. Proceedings 16. pp. 205–216. Springer (2013)
11. M. Castelli, S. Silva, L. Vanneschi, A c++ framework for geometric semantic genetic programming. Genet. Program. Evolvable Mach. **16**, 73–81 (2015)
12. J.F.B. Martins, L.O.V. Oliveira, L.F. Miranda, F. Casadei, G.L. Pappa, Solving the exponential growth of symbolic regression trees in geometric semantic genetic programming. In: Proceedings of the Genetic and Evolutionary Computation Conference. pp. 1151–1158 (2018)
13. K. Krawiec, T. Pawlak, Approximating geometric crossover by semantic backpropagation. In: Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation. pp. 941–948 (2013)
14. K. Krawiec, T. Pawlak, Locally geometric semantic crossover: a study on the roles of semantics and homology in recombination operators. Genet. Program. Evolvable Mach. **14**, 31–63 (2013)
15. T.P. Pawlak, B. Wieloch, K. Krawiec, Semantic backpropagation for designing search operators in genetic programming. IEEE Trans. Evol. Comput. **19**(3), 326–340 (2014)
16. Q. Chen, B. Xue, M. Zhang, Improving generalization of genetic programming for symbolic regression with angle-driven geometric semantic operators. IEEE Trans. Evol. Comput. **23**(3), 488–502 (2018)
17. T.P. Pawlak, B. Wieloch, K. Krawiec, Review and comparative analysis of geometric semantic crossovers. Genet. Program. Evolvable Mach. **16**, 351–386 (2015)
18. Q.U. Nguyen, T.A. Pham, X.H. Nguyen, J. McDermott, Subtree semantic geometric crossover for genetic programming. Genet. Program. Evolvable Mach. **17**, 25–53 (2016)
19. M. Castelli, L. Manzoni, L. Vanneschi, S. Silva, A. Popovič, Self-tuning geometric semantic genetic programming. Genet. Program. Evolvable Mach. **17**, 55–74 (2016)
20. M. Castelli, L. Vanneschi, L. Manzoni, A. Popovič, Semantic genetic programming for fast and accurate data knowledge discovery. Swarm Evol. Comput. **26**, 1–7 (2016)
21. I. Bakurov, M. Castelli, F. Fontanella, A.S. di Freca, L. Vanneschi, A novel binary classification approach based on geometric semantic genetic programming. Swarm Evol. Comput. **69**, 101028 (2022)
22. W. La Cava, T.R. Singh, J. Taggart, S. Suri, J.H. Moore, Learning concise representations for regression by evolving networks of trees. In: International Conference on Learning Representations (2018)
23. L. Muñoz, L. Trujillo, S. Silva, M. Castelli, L. Vanneschi, Evolving multidimensional transformations for symbolic regression with M3GP. Memetic Comput. **11**, 111–126 (2019)
24. B. Al-Helali, Q. Chen, B. Xue, M. Zhang, Multitree genetic programming with new operators for transfer learning in symbolic regression with incomplete data. IEEE Trans. Evol. Comput. **25**(6), 1049–1063 (2021)
25. S. Nguyen, D. Thiruvady, M. Zhang, D. Alahakoon, Automated design of multipass heuristics for resource-constrained job scheduling with self-competitive genetic programming. IEEE Trans. Cybern. **52**(9), 8603–8616 (2021)

26. K. Krawiec, Genetic programming-based construction of features for machine learning and knowledge discovery tasks. Genet. Program. Evolvable Mach. **3**, 329–343 (2002)
27. K. Neshatian, M. Zhang, P. Andreae, A filter approach to multiple feature construction for symbolic learning classifiers using genetic programming. IEEE Trans. Evol. Comput. **16**(5), 645–661 (2012)
28. K. Nag, N.R. Pal, Feature extraction and selection for parsimonious classifiers with multiobjective genetic programming. IEEE Trans. Evol. Comput. **24**(3), 454–466 (2019)
29. M. Muharram, G.D. Smith, Evolutionary constructive induction. IEEE Trans. Knowl. Data Eng. **17**(11), 1518–1528 (2005)
30. I. Arnaldo, U.M. O'Reilly, K. Veeramachaneni, Building predictive models via feature synthesis. In: Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation. pp. 983–990 (2015)
31. J. Ma, X. Gao, A filter-based feature construction and feature selection approach for classification using genetic programming. Knowl.-Based Syst. **196**, 105806 (2020)
32. Y. Bi, B. Xue, M. Zhang, Genetic programming with a new representation to automatically learn features and evolve ensembles for image classification. IEEE Trans. Cybern. **51**(4), 1769–1783 (2020)
33. H. Zhang, A. Zhou, Q. Chen, B. Xue, M. Zhang, SR-Forest: a genetic programming based heterogeneous ensemble learning method. IEEE Trans. Evol. Comput. https://doi.org/10.1109/TEVC.2023.3243172 (2023)
34. Q. Chen, M. Zhang, B. Xue, Genetic programming with embedded feature construction for high-dimensional symbolic regression. In: Intelligent and Evolutionary Systems: The 20th Asia Pacific Symposium, IES 2016, Canberra, Australia, November 2016, Proceedings. pp. 87–102. Springer (2017)
35. W. La Cava, L. Spector, K. Danai, Epsilon-lexicase selection for regression. In: Proceedings of the Genetic and Evolutionary Computation Conference 2016. pp. 741–748 (2016)
36. W. La Cava, T. Helmuth, L. Spector, J.H. Moore, A probabilistic and multi-objective analysis of lexicase selection and ϵ-lexicase selection. Evol. Comput. **27**(3), 377–402 (2019)
37. J.B. Mouret, J. Clune, Illuminating search spaces by mapping elites. arXiv preprint arXiv:1504.04909 (2015)
38. A. Cully, J. Clune, D. Tarapore, J.B. Mouret, Robots that can adapt like animals. Nature **521**(7553), 503–507 (2015)
39. H. Zhang, Q. Chen, A. Tonda, B. Xue, W. Banzhaf, M. Zhang, MAP-Elites with cosine-similarity for evolutionary ensemble learning. In: Genetic Programming: 26th European Conference, EuroGP 2023, Held as Part of EvoStar 2023, Brno, Czech Republic, April 12–14, 2023, Proceedings. pp. 84–100. Springer (2023)
40. J.P. Aumasson, D.J. Bernstein, Siphash: a fast short-input prf. In: Progress in Cryptology-INDOCRYPT 2012: 13th International Conference on Cryptology in India, Kolkata, India, December 9-12, 2012. Proceedings 13. pp. 489–508. Springer (2012)
41. J.D. Romano, T.T. Le, W. La Cava, J.T. Gregg, D.J. Goldberg, P. Chakraborty, N.L. Ray, D. Himmelstein, W. Fu, J.H. Moore, PMLB v1.0: an open-source dataset collection for benchmarking machine learning methods. Bioinformatics **38**(3), 878–880 (2022)
42. J. Ni, R.H. Drieberg, P.I. Rockett, The use of an analytic quotient operator in genetic programming. IEEE Trans. Evol. Comput. **17**(1), 146–152 (2012)
43. N.F. McPhee, M.K. Dramdahl, D. Donatucci, Impact of crossover bias in genetic programming. In: Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation. pp. 1079–1086 (2015)
44. F. Ramsey, D. Schafer, The statistical sleuth: a course in methods of data analysis. Cengage Learning (2012)
45. Q.U. Nguyen, T.H. Chu, Semantic approximation for reducing code bloat in genetic programming. Swarm Evol. Comput. **58**, 100729 (2020)

Authors and Affiliations

Hengzhe Zhang[1] · Qi Chen[1] · Bing Xue[1] · Wolfgang Banzhaf[2] · Mengjie Zhang[1]

✉ Qi Chen
qi.chen@ecs.vuw.ac.nz

Hengzhe Zhang
hengzhe.zhang@ecs.vuw.ac.nz

Bing Xue
bing.xue@ecs.vuw.ac.nz

Wolfgang Banzhaf
banzhafw@msu.edu

Mengjie Zhang
mengjie.zhang@ecs.vuw.ac.nz

[1] Center for Data Science and Artificial Intelligence and School of Engineering and Computer Science, Victoria University of Wellington, PO Box 600, Wellington 6140, New Zealand

[2] Department of Computer Science and Engineering, Michigan State University, East Lansing 48824, MI, USA