Improving Generalization of Genetic Programming for Symbolic Regression With Angle-Driven Geometric Semantic Operators

Qi Chen, Bing Xue, *Member, IEEE*, and Mengjie Zhang, *Senior Member, IEEE*

***Abstract*—Geometric semantic genetic programming (GP) has recently attracted much attention. The key innovations are inducing a unimodal fitness landscape in the semantic space and providing a theoretical framework for designing geometric semantic operators. The geometric semantic operators aim to manipulate the semantics of programs by making a bounded semantic impact and generating child programs with similar or better behavior than their parents. These properties are shown to be highly related to a notable generalization improvement in GP. However, the potential ineffectiveness and difficulties in bounding the variations in these geometric operators still limits their positive effect on generalization. This paper attempts to further explore the geometry and search space of geometric operators to gain a greater generalization improvement in GP for symbolic regression. To this end, a new angle-driven selection operator and two new angle-driven geometric search operators are proposed. The angle-awareness brings new geometric properties to these geometric operators, which are expected to provide a greater leverage for approximating the target semantics in each operation, and more importantly, be resistant to overfitting. The experiments show that compared with two state-of-the-art geometric semantic operators, our angle-driven geometric operators not only drive the evolutionary process to fit the target semantics more efficiently but also improve the generalization performance. A further comparison between the evolved models shows that the new method generally produces simpler models with a much smaller size and is more likely to evolve toward the correct structure of the target models.**

***Index Terms*—Generalization, genetic programming (GP), geometric semantic operator, symbolic regression.**

I. Introduction

STANDARD genetic programming (GP) [1] generally allows search operators such as crossover and mutation to consider syntax but ignore the semantic properties of swapping/replacing elements. In GP, semantics refers to a description of what the GP program or subprogram does [2]. The definition of semantics in GP is domain dependent. This paper focuses mainly on symbolic regression, the task of which is to find the underlying relationship between the input variables and output variable(s), and express this relationship in mathematical or symbolic models. In symbolic regression, the semantics of a program F is defined as a vector S [2]. The elements of S are the corresponding outputs of the program with respect to a set of input X, i.e., $S(F) = \{F(X_1), F(X_2), \ldots, F(X_n)\}$.

Unlike traditional GP, semantic GP (SGP) [2], [3], which is a recently developed variant of GP, utilizes semantic awareness to generate offsprings that are highly correlated with their parents in behaviors. SGP assumes that taking the detailed behavioral information of solutions into account can increase the effectiveness of GP.

As one particular branch of SGP, geometric SGP (GSGP) [2], [4], [5] further utilizes the semantics of GP individuals. While SGP uses the semantics as a guideline for the evolutionary process to evolve toward a program with satisfactory performance, GSGP aims to manipulate the semantics directly and has the target of generating a program with (near) optimal semantics. In GSGP, the semantics of the possible solutions form a semantic space, where different distance metrics can be employed to measure the fitness of GP individuals. By the properties of the distance metrics, the surfaces of the semantic space can be in different conic forms. An important consequence of this property is that the semantic space is unimodal, i.e., the minimum error can only be obtained at the target point, and no plateaus exist. Thus when it is possible to manipulate the semantic directly, searching on such a space will become much easier than on the search space of GP *in principle*. However, it is not that easy *in practise*, since the program space instead of the semantic space is the space being searched. Changes (swapping or replacing) in the programs do not correspond to the desired move in the semantic space. Geometric semantic operators attempt to make it possible to search directly in the semantic space.

The geometry of the semantics space is attractive for enhancing GP. However, searching directly in the semantic

Manuscript received November 3, 2017; revised April 1, 2018 and July 5, 2018; accepted August 22, 2018. Date of publication September 13, 2018; date of current version May 29, 2019. This work was supported in part by the Marsden Fund of New Zealand Government under Contract VUW1509 and Contract VUW1615, in part by the Huawei Industry Fund under Grant E2880/3663, and in part by the University Research Fund at Victoria University of Wellington under Grant 209862/3580 and Grant 213150/3662. *(Corresponding author: Qi Chen.)*

The authors are with the Evolutionary Computation Research Group, School of Engineering and Computer Science, Victoria University of Wellington, Wellington 6140, New Zealand (e-mail: qi.chen@ecs.vuw.ac.nz; bing.xue@ecs.vuw.ac.nz; mengjie.zhang@ecs.vuw.ac.nz).

This paper has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the author. To further investigate the advance of the proposed method, more experimental comparisons between the benchmark and the proposed methods have been conducted. Due to the page limits of the paper, the results of these experiments and detailed analyses are presented in the supplemental file. This file is 1.09 MB in size.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TEVC.2018.2869621

space is difficult. Therefore, GSGP provides a formal *theoretical framework* for designing geometric search operators [4]. The framework defines the desired semantic properties of the offspring generated by the geometric semantic operators. Specifically, the *geometry requirement* in the semantic space is that, the child points generated by the geometric semantic crossover stand in the segment connecting the two parent points (i.e., the semantics of the child programs is the intermediate of the parent semantics). And the child program generated by the geometric semantic mutation stand within the interval bound defined by the parent (i.e., the semantics of the child is not too different from the semantics of the parent).

Guided by the theoretical framework (see Section II-A), a number of *implementation algorithms* have been proposed for formalizing the geometric properties and various forms of geometric semantic operators (geometric operators for short) have been presented. GSGP has shown good generalization ability in [6] and [7]. Generalization, which reflects the prediction ability of the learned model, is particularly important in GP for symbolic regression. The impressive generalization gain brought by the geometric operators has been indicated by their geometric properties, which lead to a more constructive reproduction (i.e., more likely to generate child programs with better fitness than their parents) and a small variation in the semantic space. With these geometric operators, the evolutionary process approaches much smoother and maintains higher semantic locality (i.e., small changes in structures leads to small differences in semantics). All these aspects contribute to improve the generalization of GP.

However, these geometric operators have a number of limitations, which may restrict their effect on the generalization of GP. The potential *ineffectiveness* of geometric semantic crossover should be noted. The geometric crossover produces child programs that outperform both parents only when the target semantics are between the semantics of the two parents. This is true not only on the training data but also on the test data. Moreover, the geometric crossover becomes more ineffective along with the increase of the number of data instances, since it leads to higher dimensional semantic spaces. For geometric semantic mutation, it is difficult to determine the mutation step and how tight the variation should be bounded. Overly large mutation steps might lead to a decrease in the training error but an increase in the test error (overfitting), while small mutation steps decrease the exploration/search ability of GP, thus might lead to underfitting and poor generalization performance. Much research has been devoted to find an optimal mutation step, such as adaptive geometric mutation [6]. However, the effect of these geometric mutations on enhancing the generalization of GP is still limited.

The overall goal of this paper is to develop new geometric operators for addressing the above issues and improving the generalization ability in GSGP for symbolic regression. The new geometric operators will be designed to further utilize the geometric properties of the semantic space and incorporate angle-awareness. Specifically, this paper has three research questions/objectives as follows.

1) Whether the proposed geometric operators can improve the learning performance of GSGP for symbolic regression over the state-of-the-art GSGP methods.
2) Whether and how the proposed geometric operators can further improve the generalization ability of GP for symbolic regression over the state-of-the-art GSGP methods.
3) Whether and how the proposed geometric operators influence the size and interpretability of the evolved programs in GP.

This paper is based on our preliminary investigations [8], [9]. However, both of these two studies rely on the existing GSGP methods and have their own limitations. This paper significantly extends [8], [9] by developing a new selection operator and a new method to formalize the semantic requirement. More experiments and deep analyses have been conducted on the evolved programs in this paper.

II. Related Work

This section introduces concepts related to this paper, and reviews the state-of-the-art GSGP methods and recent methods proposed to improve the generalization of GP.

A. Geometric Semantic Genetic Programming

In recent years, GSGP [4] has gained increasing attention due to its conic fitness landscape. Crucially, this conic landscape is unimodal, i.e., the minimum error value is obtained at the target semantic point only. The definition of theoretical framework in GSGP, which reflects the geometry property of the geometric operators, is given as follows [4].

Definition 1 [Geometric Semantic Crossover (Geometric Crossover)]: Given two parent individuals P_1 and P_2, a geometric crossover generates offspring $O_j (j \in 1, 2)$ having semantics $\vec{O}_j (j \in 1, 2)$ in the segment between the semantics of their parents, i.e., $\|\vec{P}_2 - \vec{P}_1\| = \|\vec{O}_j - \vec{P}_1\| + \|\vec{P}_2 - \vec{O}_j\|$.

Definition 2 [Geometric Semantic Mutation (Geometric Mutation)]: Given a parent P, r-geometric mutation produces offspring O within a ball of radius r centered in P, i.e., $\|\vec{O} - \vec{P}\| \leq r$.

The implementation of geometric operators can be grouped into two categories, i.e., the *exact* geometric operators [4] and the *approximated* geometric operators [2], [10], [11]. These geometric operators have their own advantages and drawbacks. The exact geometric operators, which rely on the convex combination of the genotype of the parent(s) to manipulate the semantics of the programs directly, *guarantees* the geometry of the offspring in the semantic space. An implementation method of the geometric crossover and mutation was proposed in [4]. This implementation is a convex or linear combination of the parent(s) and one or two random programs, which results in obtaining the desired semantics exactly for the new generations. The exact geometric operators make GP have the ability to search directly in the semantic space instead of only using semantics as a guide for the evolutionary search. However, the exact geometric operators have a critical drawback of producing offspring with unmanageable size, i.e., the exact geometric semantic crossover leads to an exponential growth in the size of offspring, while the geometric mutation causes a linear growth. The over-grown offsprings are expensive to execute in both memory and time. This unavoidably leads to a low interpretability of the evolved models and an unaffordable

computational cost, which is an obstacle to the application of GSGP to data having a large number of instances.

An improved implementation of the exact GSGP has been presented [7], which does not generate the offspring explicitly. It stores the reference to the information needed to construct GP individuals, i.e., the initial population and a set of random trees. At the end of the evolutionary process, the best-of-the-run individual can be generated from the reference. This implementation makes GSGP better applicable to real-world problems. However, the final evolved models are still over-complex and hard to prune and interpret. In [12] and [13], a new exact GSGP method named error space alignment GP (ESAGP) was proposed. ESAGP introduced two concepts, which are error vectors and error space based on the semantics, with the aim of searching for two aligned individual, i.e., two individuals having the smallest angle between their error vectors in the error space. ESAGP assumes that if it is possible to find two optimal aligned individuals, then with a proportionality factor, it is able to generate a child individual with the target semantics. The angle definition of their work is similar to that in this paper. However, this paper considers various types of angles and the aims of introducing angle-awareness are different.

The approximated geometric operators work in the genotype space and *approximates* behaviors geometrically in the corresponding semantic space. The approximating geometric operators such as locally geometric semantic crossover [10], approximately geometric semantic crossover (AGX) [14], and random desired operator (RDO) (mutation) [11] do not lead to over-grown offsprings, since they generally rely on various mechanisms, such as semantic backpropagation (SB) [11] to approximately satisfy the semantic requirements. These operators will be briefly reviewed below.

B. Semantic Backpropagation, Approximately Geometric Crossover, and Random Desired Operators

Designing search operators that work in the genotype space and behave geometrically in the corresponding semantic space is not trivial. Therefore, rather than guaranteeing the geometric behavior, approximating it seems to be more sensible. Based on this assumption, various approximated geometric operators have been developed. Krawiec and Pawlak [14] proposed an AGX. In AGX, the desired semantics of the offspring is defined to be the midpoint on the segment of the two parents. SB is proposed to obtain the desired semantics. The rationale behind SB is that achieving a set of (simple) subtargets (which form the original target) should be easier and more efficient than accomplishing the whole target. Specifically, SB randomly selects a node (i.e., the crossover point in AGX) in a parent tree, which divides the tree into a *suffix* and *prefix*. The suffix is the subtree that contains the root node, while the prefix is subtree rooted at the selected node. Accordingly, the desired semantics for the whole tree is also split into two parts, which are the semantics of the suffix and the desired semantics of the prefix, i.e., a subtarget semantics. To generate a child tree with the desired semantics, SB keeps the suffix while replacing the prefix with a new subtree with the subtarget semantics. The key components in SB are obtaining the subtarget semantics of the new subtree and finding this new subtree. To calculate the subtarget semantics, the algorithm needs to backpropagate through a chain of nodes from the root to the selected node. A semantic library is formed by collecting subtrees with distinct semantics from the population of GP trees. This library is maintained every several generations. Based on certain distance metrics, SB then searches and selects new subtrees with the (approximate) subtarget semantics from this semantic library. Later, Pawlak *et al.* [11] further developed SB and proposed a geometric mutation operator named RDO. When operating on programs, RDO aims to explicitly use the target semantics, which is assumed to be the most useful information in a training set. The target semantics is considered as the unique desired semantics of the new generation. The programs evolved by AGX and RDO are much smaller than those produced by the exact geometric operators. However, these programs are still too large to be interpreted and the unique desire semantics in RDO has a potential drawback of leading to a low semantic diversity and a greedy nature in fitting the target semantics, which limit its potential in improving the generalization of GP.

C. Generalization of GP for Symbolic Regression

Generalization is the ability with which a learned model can obtain good prediction results on unseen test data. It is one of the desired abilities and the key performance criterion for measuring the effectiveness of learning algorithms. Although generalization has been deeply investigated in many fields of machine learning [15], [16], it was not the case in GP for symbolic regression for quite a long time in the past. Before 2000, symbolic regression was mostly treated as an ordinary optimization problem, where all the available data was used for evolving the models and the generalization performance of the models on unseen data was not considered at all. Recently, an increasing number of algorithms have been proposed to improve the generalization capability of GP for symbolic regression [17]–[20].

Since overfitting is the contrasting phenomena to generalization, i.e., poor generalization, the majority of research on generalization of GP for symbolic regression has been devoted to reducing/controlling model complexity or avoiding overfitting. Astarabadi and Ebadzadeh [19] proposed a multiobjective GP algorithm to enhance the generalization ability of GP. The multiobjective GP employs the first order derivative of the evolved models as a measure of model complexity. The root mean square error (RMSE) between the first order derivative of the evolved models and that of the target model was treated as one objective, and the training error of the models was the second objective. The results showed that their method generalized better than standard GP. Gonçalves and Silva [21] proposed a GP method using interleaved sampling strategy training data to achieve a tradeoff between controlling overfitting and presenting enough information for the evolutionary process. The interleaved sampling of the training data was processed in both deterministic and probabilistic ways, respectively. The experiments have shown that most variants of the proposed method can improve generalization and reduce overfitting of GP consistently.

Introducing semantic-awareness into GP methods has been shown to have a positive effect on promoting its generalization [8], [9], [22]. Uy *et al.* [22], [23] proposed new semantic-aware crossover operators, which imposed various requirements on the semantic distance between subtrees rooted on the crossover points in the two parents. Only subtrees that have similar (but not equivalent) semantics are allowed to swap. These semantic-aware crossover operators maintained high locality thus yielding a better generalization of GP. Gonçalves *et al.* [6] compared the generalization ability of GSGP methods employing sole geometric crossover, geometric mutation, and bounded geometric mutation [7]. They claimed that geometric crossover contributed little to improve the generalization ability, while geometric mutation had a good effect on promoting the generalization. They also attributed the advance generalization ability of GSGP to the bounded geometric mutation, which produced a small variation to the offspring.

In summary, different from traditional methods that improve the generalization of GP by evolving structurally simpler programs, GSGP methods enhance the generalization of GP by controlling the semantics of the programs during the evolutionary process. In GSGP, the semantic-awareness drives GP search in a smoother fitness space and the geometry of the semantic space makes GP search more effectively, both of which lead to a better generalization gain. However, most existing GSGP methods still favor the training performance and their generalization ability is still limited. In addition, the evolved models are still over-complex and very hard to interpret, and the structures of the evolved models are still far from the target models, which also limit the generalization capability. This paper explores novel ways to make significant improvements on these points, which is to be described in the next section.

III. Proposed Angle-Driven GSGP

The theoretical framework in GSGP was shown to be positive in enhancing the generalization of GP. However, the geometric operators defined by the framework still have some potential limitations, which might restrict their effectiveness in gaining better generalization ability. This paper attempts to further explore the geometry of these search operators to make them more constructive. To this end, we incorporate the angle-awareness into GSGP and make the angle-awareness a main force to drive the evolutionary process of GSGP. The proposed GSGP method is thus named angle-driven GSGP (ADGSGP). The angle-awareness is expected to make geometric operators more effective and help the evolutionary process converge to the target semantics much more accurately and faster.

Our preliminary work [8] has shown that incorporating angle-awareness into GSGP has an impressive effect on promoting its learning and generalization performance. Chen *et al.* [8] introduced an angle-aware mating scheme to geometric crossover. Given a set of candidate parents that have won the tournament selection, the mating scheme drives geometric crossover operating on pairs of parents, which have the largest angle-distance between the relative semantics of the parents and the target. The large angle-distance between these relative semantics helps reduce semantic duplications in the offspring and increase the effectiveness of crossover. Mating between individuals having a large angle-distance increases the exploration ability of GP and makes the evolutionary process converge to the target semantics much faster. However, the mating scheme has an underlying limitation. During the evolutionary process, the set of candidate parents that have won the tournament selection, are more and more likely to overlap with each other in the semantic space. Accordingly, it becomes increasingly difficult to find parents with a large angle-distance. To address this limitation and further utilize the angle-awareness for selecting better parents, this paper proposes a new angle-driven selection (ADS) operator (see Section III-B) for geometric crossover. The new selection operator selects pairs of parents that have large angle-distances in semantic space from the whole population directly, instead of from a set of winners of the tournament selection. Furthermore, we develop two new geometric semantic operators, i.e., perpendicular crossover (PC) and random segment mutation (RSM). The rationale behind the proposal of PC and RSM is to have two geometric operators which are more effective not only on learning but also on generalization. More specifically, PC is developed to address the ineffectiveness of the geometric crossover by further exploring the geometric properties, while RSM is proposed to solve the difficulty in determining the mutation step by generating child individuals that are highly correlated to their parent and approximating the target semantics in the right direction. The target semantics is assumed to be the most informative aspect in the dataset. Explicitly utilizing such information can further improve the performance of the geometric operators. A preliminary investigation of the effect of the two operators has been shown in [9]. However, the implementation of the angle-aware geometric operators relies on SB, while this paper will develop a new *implementation algorithm* to address the potential drawbacks of SB. The details of the new geometric operators will be presented in Sections III-C–III-E.

A. Angle-Distance Measurement

Before presenting the details of the proposed method, a brief introduction on how to measure the angle-distance between the semantics of two individuals and between two relative semantics is necessary. As the semantics of individuals in GSGP for symbolic regression is defined as a vector, the angle-distance between the semantics of two individuals is defined as the angle between the two vectors. In an n-dimensional space (e.g., the semantic space of a symbolic regression problem with n training instances), the angle γ between two vectors $\vec{V}_1$ and $\vec{V}_2$ is the arc-cosine of the dot product of their normalized vectors. The definition is given as follows:

$$\gamma = \arccos\left(\frac{\vec{V}_1}{\|\vec{V}_1\|} \cdot \frac{\vec{V}_2}{\|\vec{V}_2\|}\right) \tag{1}$$

where the normalized vector/semantics is

$$\frac{\vec{V}_j}{\|\vec{V}_j\|} = \frac{\sum_{i=1}^{n} v_{j,i}}{\sqrt{\sum_{i=1}^{n} v_{j,i}^2}}, \quad j \in \{1, 2\}$$

and $v_{j,i}$ is the ith dimensional value of $\vec{V}_j$.

For the angle between the relative vectors (a relative vector is the vector between one parent's semantics and the target

semantics), the only change in the definition is to replace the normalized vectors with the normalized relative vectors. Given three vectors $\vec{V}_1$, $\vec{V}_2$, and $\vec{V}_3$, the angle between the two relative semantics, $(\vec{V}_3 - \vec{V}_1)$ and $(\vec{V}_3 - \vec{V}_2)$, is defined as

$$\gamma_r = \arccos\left(\frac{(\vec{V}_3 - \vec{V}_1)}{\|\vec{V}_3 - \vec{V}_1\|} \cdot \frac{(\vec{V}_3 - \vec{V}_2)}{\|\vec{V}_3 - \vec{V}_2\|}\right) \tag{2}$$

where the normalized relative semantics is

$$\frac{(\vec{V}_3 - \vec{V}_k)}{\|\vec{V}_3 - \vec{V}_k\|} = \frac{\sum_{i=1}^{n}(v_{3,i} - v_{k,i})}{\sqrt{\sum_{i=1}^{n}(v_{3,i} - v_{k,i})^2}}, \quad k \in \{1, 2\}.$$

$v_{3,i}$ and $v_{k,i}$ are the values of $\vec{V}_3$ and $\vec{V}_k$ in the ith dimension.

B. Angle-Driven Selection

In ADGSGP, a new selection operator named *ADS* is proposed to select pairs of parents for geometric crossover. The pseudo code of ADS is shown in Algorithm 1. ADS selects the first parent p_1 according to the tournament selection. An iterative procedure is performed to select the second parent p_2 according to its angle distance from p_1. During this procedure, each candidate of p_2 is selected by tournament selection, then its angle-distance from p_1 is calculated. This procedure will terminate when reaching the maximum number of iterations *nt* or finding a candidate with the angle lager than the predefined threshold *ta*. To achieve a tradeoff between finding optimal parents (e.g., parents with an angle distance near to 180°) and the high computational cost, after a preliminary investigation, the two parameters are set to $nt = 10$ and $ta = 90$.

Different from the commonly used selection operators (e.g., tournament selection and truncation selection), which select parents according to their fitness values only, ADS selects a pair of parents that not only have good fitness values, but also are far away from each other regarding the angle-distance of their relative semantics (to the target semantics) in the semantic space. Introducing angle-awareness into the selection operator and selecting parents with large angle-distance bring several benefits to the evolutionary process. First, it helps decrease the semantic duplicates. Since these far away parents generally have different semantics, and the interval between their semantics, i.e., the segment between the two parent points in the semantic space, is much larger than that of the nearby parents. The semantics of the two children, which stand in this larger segment, are more likely to differ from their parents and from each other. This can potentially maintain/increase the semantic diversity of the population. Second, the convex hull of the far-away parents becomes larger, which will increase the probability of covering the target semantics, and have a more accurate fitting to the target semantics. The benefits of a larger angle-distance between parents to the evolutionary process have been investigated and confirmed in our preliminary work [8]. Compared with the angle-aware mating scheme in [8], the advantage of ADS is in increasing the probability of finding parents with a large angle. ADS selects parents directly from the population. These parent pairs satisfy the angle-distance and fitness requirement simultaneously, while the mating scheme in [8] selects the satisfied parent pairs from the winners of the tournament selection.

Algorithm 1: ADS

Input : a population of individuals, the target semantics $\vec{T}$, the number of pairs *np* to be selected, the maximum number of trails *nt*, the threshold angle-distance *ta*
Output: a list *l* containing all the selected pairs
for *g*: = 1 *to np* **do** *Selection loop*
  Setting the flag of finding good enough pair $f = false$, clear the candidate list *cl*;
  Select the first parent p_1 by tournament selection;
  for *t*: = 1 *to nt* **do** *Selecting the second parent* p_2 *loop*
    Select a candidate parent individual cp_2 by tournament selection;
    Calculate the angle γ_r between the relative semantics $\vec{T} - \vec{p}_1$ and $\vec{T} - \vec{p}_2$ according to Equation (2);
    if $\gamma_r > ta$ **then**
      $p_2 = cp_2, f = true$;
      Stop the loop;
    else
      Put cp_2 into *cl*;
  if $f == false$ **then**
    Select the maximum angle value from *al*;
    Set p_2 to be the individual with the largest γ_r from *cl*;
  Put the selected pair p_1 and p_2 into *l*
return *l*;

C. Perpendicular Crossover

The desired semantics of the offspring in existing geometric operators is highly correlated to the semantics of their parents. In *exact* geometric crossover, the semantics of the children stands in the segment of their parents, while in AGX, it is the (approximately) middle point of this segment. Neither of them has considered improving the effectiveness of the geometric crossover by introducing the geometry of the target semantics. A better geometric crossover can produce offspring that are not only highly correlated to the parent semantics but also effective in approximating the target semantics. To this end, *PC* is proposed in this paper. PC imposes a more precise semantic requirement than exact geometric crossover. In this way, it aims to drive the search toward convergence to the target semantics much faster. Given two parent individuals, PC generates a child point standing on the line crossing the two parents, which follows the theoretical framework of GSGP. Moreover, the relative vector given by this child point and the target semantics is perpendicular to the vector defined by the two parents. Suppose the target semantics is $\vec{T}$, and the semantics of the two parents are $\vec{P}_1$ and $\vec{P}_2$. As shown in Fig. 1(a)–(c), the three points define a triangle. α refers to the angle between the relative semantics of $(\vec{P}_2 - \vec{P}_1)$ and $(\vec{T} - \vec{P}_1)$, while β is its counterpart to $\vec{P}_2$. Given $\vec{T}$, $\vec{P}_1$, and $\vec{P}_2$, according to (2), it is straightforward to obtain the values of these two angles.

Obtaining the semantics of the offspring $\vec{O}$ is to calculate the position of $\vec{O}$, which is the base of the perpendicular dropped from $\vec{T}$ to the relative semantics $(\vec{P}_1 - \vec{P}_2)$. As shown in Fig. 1(a)–(c), according on the values of α and β, there are three possible position of $\vec{O}$. In the first case [as shown in Fig. 1(a)], when α and β are both smaller than 90°, the offspring $\vec{O}$ (represented by the green point in the figure) stands

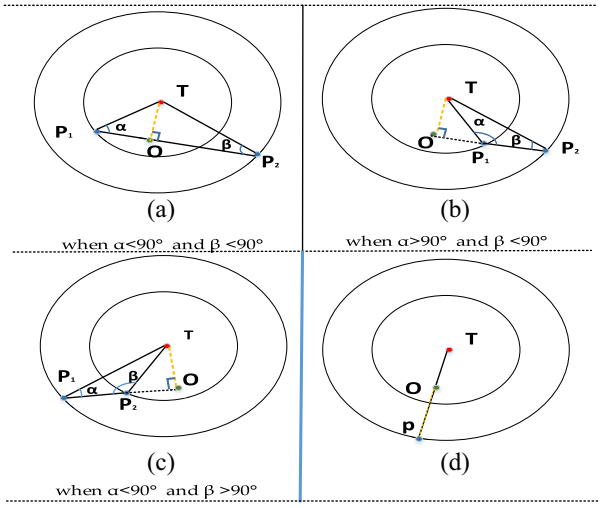

Fig. 1. Offspring generated by (a)–(c) PC and (d) RSM.

in the segment of $(\vec{P}_1 - \vec{P}_2)$. In the other two cases where either α or β is larger than 90°, the offspring stands along the segments on the $\vec{P}_1$ or $\vec{P}_2$ side.

To calculate the position of $\vec{O}$, i.e., a particular point in the line crossing the two parent points, we adopt the parametric equation, which is the most versatile equation to define a line in an n-dimensional space, to express this line. Specifically, suppose that the line is given by two points $\vec{P}_1$ and $\vec{P}_2$ in an n-dimensional space, the particular point $\vec{O}$ is defined by (3)–(5) [corresponding to Fig. 1(a)–(c)] as follows.

1) When $\vec{O}$ stands on the segment between $\vec{P}_1$ and $\vec{P}_2$, i.e., $\alpha < 90$ and $\beta < 90$

$$\begin{aligned}\vec{O} &= \vec{P}_1 + \frac{\|\vec{P}_1 - \vec{O}\|}{\|\vec{P}_1 - \vec{P}_2\|} \cdot \left(\vec{P}_1 - \vec{P}_2\right), \|\vec{P}_1 - \vec{O}\| \\ &= \|\vec{P}_1 - \vec{T}\| \cdot \cos(\alpha). \end{aligned} \tag{3}$$

2) When $\vec{O}$ is outside the segment on the $\vec{P}_1$ side, i.e., $\alpha > 90$

$$\begin{aligned}\vec{O} &= \vec{P}_1 - \frac{\|\vec{P}_1 - \vec{O}\|}{\|\vec{P}_1 - \vec{P}_2\|} \cdot \left(\vec{P}_1 - \vec{P}_2\right), \|\vec{P}_1 - \vec{O}\| \\ &= \|\vec{P}_1 - \vec{T}\| \cdot \cos(180 - \alpha). \end{aligned} \tag{4}$$

3) When $\vec{O}$ is outside on the $\vec{P}_2$ side, i.e., $\beta > 90$

$$\begin{aligned}\vec{O} &= \vec{P}_2 + \frac{\|\vec{P}_2 - \vec{O}\|}{\|\vec{P}_1 - \vec{P}_2\|} \cdot \left(\vec{P}_1 - \vec{P}_2\right), \|\vec{P}_2 - \vec{O}\| \\ &= \|\vec{P}_2 - \vec{T}\| \cdot \cos(180 - \beta) \end{aligned} \tag{5}$$

where $(\vec{P}_2 - \vec{P}_1)$ gives the direction of the line, the elements of which are defined as $\{(p_{2,1} - p_{1,1}), (p_{2,2} - p_{1,2}), \ldots, (p_{2,n} - p_{1,n})\}$. $\|\vec{P}_1 - \vec{O}\|$ and $\|\vec{P}_2 - \vec{O}\|$ are the relative distance between $\vec{P}_1$ ($\vec{P}_2$) and $\vec{O}$.

As the evaluation of a program in GSGP is generally defined as the distance from the target semantics, when the Euclidean metric is adopted, the exact geometric crossover produces offspring that are not worse than the worse parent, but PC guarantees that the offspring program is better than both of its parents.

Algorithm 2: Obtaining the Desired Semantics in RSM

Input : Target semantics $\vec{T}$, and the semantics of the parent $\vec{P}$.
Output: The desired semantics of the offspring $\vec{O}$.
Calculate the relative semantics between the parent and the target semantics $\vec{P} - \vec{T}$ and the norm $\|\vec{P} - \vec{T}\|$;
Obtain a random real number $k \in (0, 1)$;
Calculate $\vec{O}$ according to $\vec{O} = \vec{P} + k \cdot (\vec{T} - \vec{P})$, which will make $\vec{O}$ stand in the segment of $\vec{P}$ and $\vec{T}$;
Return $\vec{O}$;

D. Random Segment Mutation

Inspired by RDO and to have a better control of the semantic variation induced by geometric mutation, we aim to utilize the target semantics. We propose an angle-driven geometric mutation operator named RSM. By operating RSM, the desired semantics of the offspring is standing on the segment connecting the parent and the target point in the semantic space. The pseudo code of the procedure to obtain the desired semantics in RSM is shown in Algorithm 2. Given a parent $\vec{P}$, RSM first needs to find the segment between the target semantics $\vec{T}$ and $\vec{P}$. Then a random point is obtained along this segment, which is treated as the desired semantics of the offspring $\vec{O}$. Then the desired semantics is calculated according to (6), which has the same principle as (3)–(5)

$$\vec{O} = \vec{P} + k \cdot \left(\vec{T} - \vec{P}\right), k \in (0, 1). \tag{6}$$

This requirement on the semantics of the offspring follows the theoretical requirement in geometric mutation, meanwhile it makes the angle between the relative semantics of $(\vec{O} - \vec{P})$ and $(\vec{T} - \vec{O})$ exactly 180°. In this way, RSM has a more precise semantic control on the offspring than the exact geometric mutation, which drives the fitting of the target semantics in the "right" direction. Compared with RDO, RSM generates offspring with different desired semantics, which are random points standing on the intervals between the parents and the target semantics. This helps increase semantic diversity of the population and leads to a better exploration ability in ADGSGP than approximate GSGP (AGSGP).

E. Semantic Context Replacement

Once the desired semantics of the offspring in the two geometric operators is obtained, a further step is to generate offspring to fulfil these semantics. In our preliminary work [9], SB is employed, which aims to achieve the desired semantics by replacing the prefix subtree with a new one with the subtarget semantics. In SB, a parent tree is split into two parts, i.e., a suffix and a prefix, by a randomly selected node in the tree. Then SB keeps the suffix, which generally contains the root node of the tree. Meanwhile, SB replaces the prefix expressed by the subtree rooted at the selected node with a new subtree with the (approximate) subtarget semantics from a semantic library. While the idea of SB is sensible, it generally produces over-complex models (but these models are much smaller/simpler than those in exact GSGP). Obtaining the subtarget semantics by propagation backward through these complex trees is still too expensive to afford. Meanwhile, over-complex models are often prone to overfit the training set

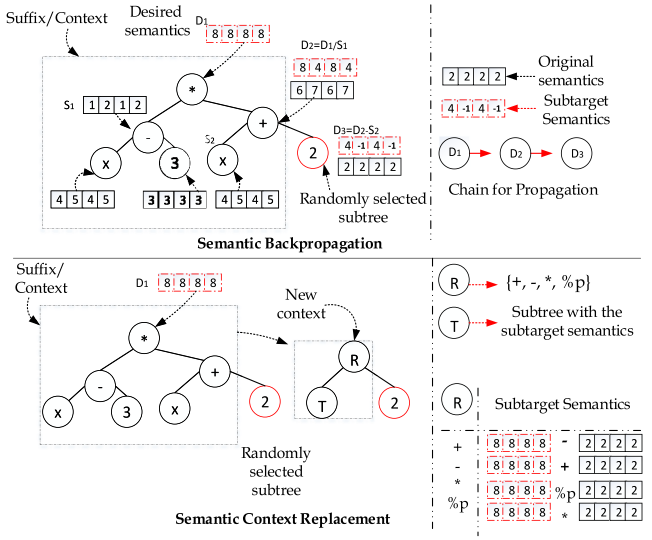

Fig. 2. SB and SCR.

and could not generalize well on the unseen data. All these limitations prevent SB from being applied to problems with a larger semantic space.

To avoid the potential limitations of SB, this paper proposes a new approximately semantic operator named *semantic context replacement (SCR)* to fulfil the semantic requirement. The procedure of fulfilling the desired semantics in SB and SCR is shown in Fig. 2. The major difference between SCR and SB is that SB maintains the suffix/context of the parent individual and replaces the prefix, while SCR aims to provide a better suffix/context, which consists of a new root node (selected from a set of binary operators) and a new subtree with subtarget semantics, and preserves the semantics of the prefix to the children. The motivation of making this change is the importance of the context of GP individual to its fitness, which has been confirmed by previous research [24]. Meanwhile, we expect this small but important change will bring many benefits during the process of achieving the desired semantics. First, compared with backpropagation in SB to obtain the desired subtree semantics, SCR is more straightforward to calculate the desired semantics for the context. As shown in Fig. 2, SB needs several subsequent steps of backpropagation (from the parent of the selected node to the root node), while SCR only needs one step. Once the new root node R is decided, SCR inverts the execution of the new root node R to get the subtarget semantics. The rules for the inverted operators are very simple, i.e., $+$ and $-$, and $*$ and $\%p$ (protected division) are inverted with each other. Furthermore, replacing the context with a newly constructed one, which is generally smaller than the original one, can potentially decrease the complexity of the offspring programs.

As shown in Algorithm 3, after splitting the tree into the context/suffix and the prefix, SCR replaces the context/suffix of a tree by two steps, where the first step is to select the root node and the second step is to select a subtree with the subtarget semantics from the semantic library. The selection is according to the angle-distance between the semantics of candidate trees [according to (1)] and the subtarget

Algorithm 3: SCR

Input : The parent individual P, the maximum depth of the GP individuals MD, an angle-distance list AL.
Output: A new subtree with the subtarget semantics.
Randomly select a node in the parent tree P to split the tree into suffix and prefix ;
Replace the suffix/context by selecting a binary operator from the function set to be the root node R randomly, i.e., select one operator among $\{+, -, *, \%protected\}$;
Calculate the subtarget semantics for the rest part of the context T according to the inverted execution of R ;
for *each ct in the semantic library* **do**
  Calculate the angle between ct and T according to Equation (1), and put it into AL;
Obtain the tree TR with the smallest angle value in AL;
Perform a linear scale to TR, and make it as $(a + b \cdot TR)$, where a and b are set according to Equation (7)

semantics. SCR selects the tree TR having the smallest relative angle-distance. In this paper, the dynamic semantic library is employed, which contains all the semantically unique subtrees collected from all the individuals in the current generation. It needs to be maintained and updated at every generation.

We hypothesize that the subtree with the closest angle-distance to the subtarget semantics is most likely to have the desired structure but not the fitted coefficients. A simple form of linear scaling [25] has shown to be effective to find a set of better coefficients. With a linear scaling on the output of the selected subtree, it could help find better coefficients to the selected subtree to fit the subtarget semantics better. Therefore, after finding TR, a linear scaling [25] is performed to TR, i.e., inducing two coefficients a and b to TR to scale it to $(a+b\cdot TR)$, where a and b are the intercept and slope of the linear scaling, respectively. According to [25], the values of a and b are defined as follows:

$$b = \frac{\sum_{i=1}^{n}\left[\left(t_i - \bar{t}\right)\left(ct_i - \overline{ct}\right)\right]}{\sum_{i=1}^{n}\left[\left(ct_i - \overline{ct}\right)^2\right]}, \quad a = \bar{t} - b \cdot \overline{ct} \tag{7}$$

where t_i is the value of the subtarget semantics in the ith dimension and $\bar{t}$ is the mean of all the t_i values. ct_i is the value of the semantics of TR in the ith dimension and $\overline{ct}$ is the mean value of all ct_i.

IV. Experimental Design

To investigate and confirm the effectiveness of our new method ADGSGP, a set of experiments have been conducted to compare ADGSGP with two state-of-the-art GSGP methods and standard GP. The three benchmark GP methods are as follows.

1) GSGP refers to the exact GSGP method which uses the exact geometric crossover and geometric mutation. The improved implementation of GSGP [7] is used in the experiments.
2) AGSGP [11] employs two state-of-the-art geometric operators: 1) AGX and 2) RDO. Both of these two operators are based on SB.
3) Standard GP, which employs standard crossover and mutation, is also used as a baseline for comparison.

TABLE I
SYNTHETIC DATASETS

The training samples are drawn using regular intervals from the interval range, and the test samples are drawn randomly within the same interval.

Problem	Function	Range	#Points (Training,Test)
Nguyen-7	$log(x+1)+log(x^2+1)$	[0,2]	(20, 100)
Keijzer-11	$(x*y)+sin((x-1)*(y-1))$	[-1,1]	(25, 100)
Keijzer-14	$8.0/(2+x^2+y^2)$	[-1,1]	(25, 100)
Pagie1	$\frac{1}{1+x^{-4}}+\frac{1}{1+y^{-4}}$	[-5,5]	(625, 1000)
Keijzer1	$0.3xsin(2\pi x)$	training:[-1,1]	(20, 1000)
Koza2	(x^5-2x^3+x)	test:[-1.1,1.1]	
Nonic	$\sum_{i=1}^{9} x^i$		(20, 1000)
R1	$(x+1)^3/(x^2-x+1)$	training:[-2,2]	
R2	$(x^5-3x^3+1)/(x^2+1)$	test:[-2.2,2.2]	
Mod_quartic	$4x^4+3x^3+2x^2+x$		

TABLE II
REAL-WORLD PROBLEMS

Name	#Features	#Instances		
		Total	Training	Test
BHouse	13	506	354	152
Concrete	9	1030	712	309
Wine	12	1402	980	422
DLBCL	7400	240	160	80

All the four GP methods are implemented under the GP framework provided by Distributed Evolutionary Algorithms in Python (DEAP) [26].

A. Benchmark Problems

Following the previous research on geometric operators, we investigate the performance of the GP methods on a set of synthetic datasets. We test the GP methods on ten synthetic datasets. The target functions and sampling strategies are shown in Table I. They are taken from [8], [11], and [27]. Furthermore, we are also interested in testing the proposed method on real-world datasets, which have not been widely used in research on GSGP. The detailed information of the datasets are shown in Table II. The training set and the test set are given in DLBCL, while each of the other three datasets is randomly split with 70% of the data for training and the rest 30% for test in each GP run. This is a widely accepted way of splitting a dataset in machine learning [6], [7], [28], [29]. The splitting is different in each independent GP run, but is the same for the four GP method in one run. For a further comparison, we have also examined the methods using fivefold cross validation, which shows almost the same patterns. Interested readers are referred to the online supplementary material for the detailed analysis.

B. Parameter Settings

The parameter settings for all the four GP methods are summarized in Table III. Most of these parameters are common settings in GP and GSGP [1], [4]. The crossover and mutation rates in GP and GSGP are different. Standard GP generally has a higher crossover probability, since crossover is considered to be much more important than mutation for the progress of the evolutionary process. A high mutation rate is desired in GSGP methods to promote the search in semantic spaces more efficiently. Exact GSGP does not have a depth limitation. For the other three GP methods, the maximum tree depth is 17. RMSE is adopt as the fitness function. Each GP method has been conducted for 50 independent runs on each problem. Note that, as pointed out in the previous study [30], in some cases, the comparison under the same number of generations, which is the mainstream in the literature of GP, might be not a fair comparison. The comparison between algorithms adopting a variable-size of representation should take the computational effort into consideration.

TABLE III
PARAMETER SETTINGS FOR GP RUNS

Parameter	Value
Population Size	100
Maximal Number of Generations	100
Crossover and Mutation Rates	GP(0.9 and 0.1) and GSGPs (0.5 and 0.5)
Elitism	1
Maximum Tree Depth	17
Initialisation	Ramped half-and-half
Initialisation Depth	2-6
Function Set	$+$, $-$, $*$, %protected
Fitness Function	Root Mean Square Error (RMSE)
Crossover and Mutation Node Selection	Uniform depth node selector
Mutation Step for Exact GSGP	1

V. RESULTS AND DISCUSSION

This section compares and discusses the results obtained by the four GP methods. The comparisons will be presented in terms of the training performance, the generalization ability and the size of the evolved programs in the GP methods. The computational costs will also be shown. The main comparison is conducted between GSGP methods. A nonparametric statistical significance test—the Wilcoxon test, with a significance level of 0.05, is used to compare the training RMSEs and test RMSEs of the best-of-run models. Two sets of statistical significance tests have been conducted. One is between GP and the three GSGP methods. The other is among the three GSGP methods, i.e., comparing ADGSGP with the other two GSGP methods.

A. Overall Results

The distribution of RMSEs obtained by the best-of-run programs on the training sets and their corresponding test RMSEs are shown in Fig. 3. Red boxes are for the training sets, while the green ones are for the test sets. Table IV presents the results of the two sets of statistical significance tests. "$-$" stands for ADGSGP (GP) performs significantly better than the compared method, "$+$" indicates ADGSGP (GP) is significantly worse, and "$=$" means no significant difference.

On three of the first four synthetic datasets (except for Keijzer14), where the training instances and test instances are drawn from the same distribution, the three GSGP methods generally have much lower RMSEs than standard GP on both training sets and test sets. On Keijzer14 and the other six synthetic datasets, the exact GSGP method produces higher training and generalization errors than standard GP. The training differences are significant on six of these seven training sets, while the generalization differences are all significant. On most of the synthetic datasets, AGSGP and ADGSGP significantly outperform standard GP in terms of both learning ability and generalization performance. On

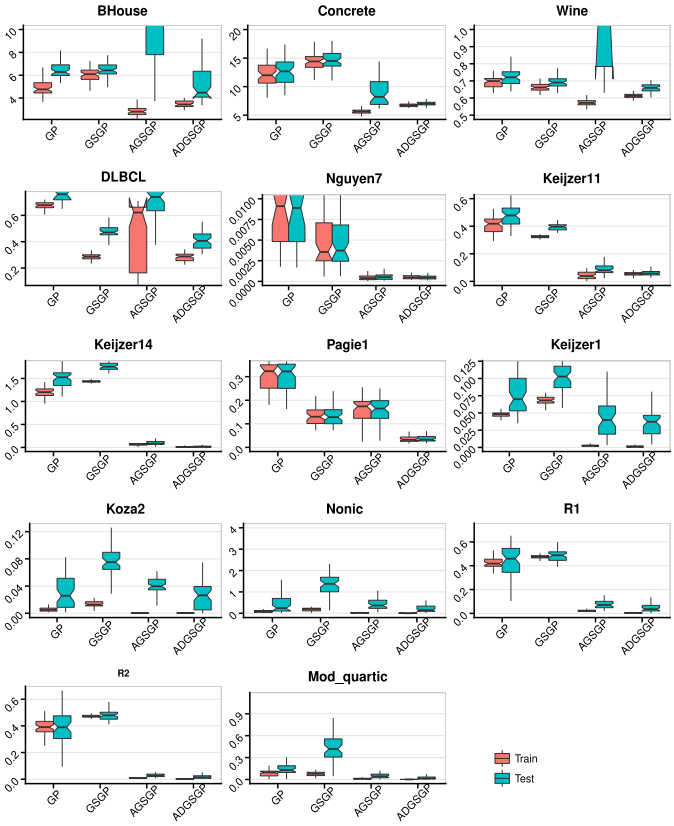

Fig. 3. Distribution of the training errors and the corresponding test errors of the best-of-run models.

TABLE IV
RESULTS OF STATISTICAL SIGNIFICANCE TESTS

Datasets	GP(training, test)			ADGSGP(training, test)	
	GSGP	AGSGP	ADGSGP	GSGP	AGSGP
Nguyen-7	(+, +)	(+, +)	(+, +)	(−, −)	(=, =)
Keijzer-11	(+, +)	(+, +)	(+, +)	(−, −)	(=, −)
Keijzer-14	(−, −)	(+, +)	(+, +)	(−, −)	(−, −)
Pagie1	(+, +)	(+, +)	(+, +)	(−, −)	(−, −)
Keijzer-1	(−, −)	(+, +)	(+, +)	(−, −)	(=, −)
Koza2	(−, −)	(+, −)	(+, =)	(−, −)	(=, −)
Nonic	(−, −)	(=, =)	(=, +)	(−, −)	(=, −)
R1	(−, −)	(+, +)	(+, +)	(−, −)	(=, −)
R2	(−, −)	(+, +)	(+, +)	(−, −)	(=, =)
Mod_quartic	(=, −)	(+, +)	(+, +)	(−, −)	(=, −)
BHouse	(−, =)	(+, −)	(+, +)	(−, −)	(+, −)
Concrete	(−, −)	(+, +)	(+, +)	(−, −)	(+, −)
Wine	(+, +)	(+, −)	(+, +)	(−, −)	(+, −)
DLBCL	(+, +)	(+, =)	(+, +)	(=, −)	(−, −)

all the synthetic datasets, ADGSGP outperforms exact GSGP on both learning and generalization performance. ADGSGP achieves significantly smaller training errors on Pagie1 and Keijzer14 than AGSGP, and no significant difference on their training performances was found on the other training sets. More importantly, ADGSGP has lower test errors than AGSGP on all the synthetic datasets, where on eight of the ten test sets (except for Nguyen-7 and R2) the advantage is significant. Meanwhile, ADGSGP has more robust performance than the other two GSGP methods in both learning and generalization performance, which is indicated by the shorter whiskers in the boxplots. Overall, the newly proposed method ADGSGP is undoubtedly the winner on the synthetic datasets.

The pattern on the real-world datasets is different to that on the synthetic datasets. The superiority of GSGP methods (particularly in AGSGP) over standard GP is not as obvious as on the synthetic datasets. GSGP performs worse than standard GP on Concrete regarding both training and test performance. On BHouse, GSGP also obtains much higher training errors than standard GP while no significant difference on their test performances. On Wine and DLBCL, the advantage of GSGP over standard GP is significant on both training set and test sets. On the four real-world problems, AGSGP obtains significantly better training gain than standard GP. However, it has a significantly worse generalization performance than GP on two test sets (BHouse and Wine), while having comparable generalization performance with GP on DLBCL, and only generalizing better than standard GP on Concrete. Different from the two counterparts, our proposed method ADGSGP achieves significantly smaller RMSEs than standard GP on *all* the training sets and the test sets of the real-world problems.

Regarding the comparison between the three GSGP methods on the real-world datasets, the superiority of AGSGP on the learning performance clearly contrasts with its poor generalization performance on the real-world problems, which indicates the occurrence of overfitting. ADGSGP is not always the winner of the training performance, but it outperforms the other two GSGP methods on the generalization performance in all cases. ADGSGP is the only GSGP method that consistently outperforms standard GP in all the examined datasets. The notably shorter whiskers in the boxplots of ADGSGP show the robustness of the performance. Statistical significance tests confirm that ADGSGP generalizes the best on *all* the real-world datasets among *all* the four GP methods.

To sum up, the overall pattern is that the new proposed method ADGSGP has comparable training performance to AGSGP, and much better than standard GP and GSGP. More importantly, regarding the generalization ability, ADGSGP is undoubted the winner among the four GP methods. It performs significantly better than all three other GP methods on all cases except for Nguyen-7 and R2, where it performs similar to AGSGP.

B. Learning Performance

To have a closer view on the training performance, the evolutionary training plots are presented in Fig. 4 (due to the page limit, the evolutionary plots on the last six synthetic datasets are shown in Fig. 1 in the supplementary material). These plots are drawn using the median RMSEs obtained by the best-of-generation programs on every generation.

As seen from these evolutionary training plots, AGSGP and ADGSGP are generally the most effective methods in fitting the training data. The progress of the evolution is due to that the percentage of effective breeding, i.e., the number of children which have better learning performance than their parents, has increased. Among all the training sets, ADGSGP consistently achieves a faster decrease in the training error, which indicates that it learns much faster than GSGP and AGSGP. The geometry property of PC drives ADGSGP converging much faster than the other three GP methods in the early phase of the evolutionary process. Along with the evolutionary process, it becomes more difficult for ADGSGP to get closer to the target semantics, which is indicated by the very

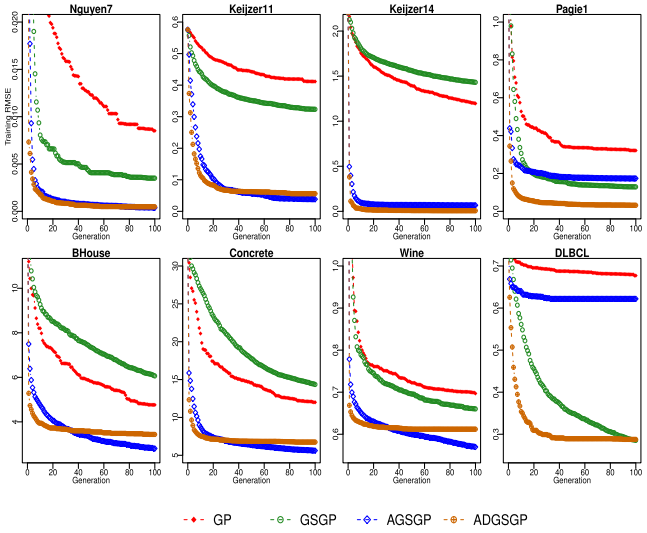

Fig. 4. Evolutionary plots on the *Training* median RMSEs of the best-of-generation models.

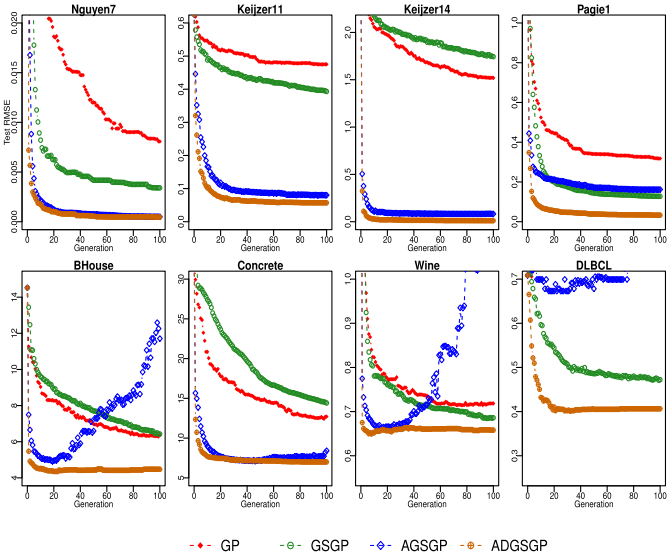

Fig. 5. Evolutionary plots on the corresponding *Test* RMSEs of the best-of-generation models.

slow decrease in the training errors after the first around 20 generations. This might be due to the difficulty in finding pairs of parents that are far away regarding their angle-distance, which limits the effectiveness of the new geometric crossover. In addition, this is an unavoidable phenomenon caused by the geometries of PC and RSM. When the whole population becomes closer to the target semantics, the distance between the parent(s) and the target semantics becomes smaller. The smaller relative distance leads to a smaller movement toward the target semantics. AGSGP suffers from the same difficulty as ADGSGP in most training sets. However, the case is different on the three real-world datasets (i.e., BHouse, Concrete, and Wine). On these datasets, the number of training instances is much higher than the rest five training sets. So correspondingly their semantic spaces are much larger in dimension than the other datasets. Approximating the target semantics is more difficult in this scenario. RDO in AGSGP, which aims to produce offspring highly correlated with the target semantics and approximates the target semantics greedily, is able to fit the target semantics much better than the other GP methods in this scenario. On other datasets, this advantage is not very clear. Exact GSGP generally learns much slower and produces much higher training errors than AGSGP and ADGSGP over generations. Particularly, GSGP fits the target semantics much slower than ADGSGP on almost all the training sets. On DLBCL, GSGP outperforms AGSGP and standard GP from the very early several generations, and this advantage increases over generations.

To summarize, introducing the semantic information into the evolutionary process does not always bring improvement, which is indicated by the much slower learning speed of exact GSGP than that of standard GP on six of the 14 datasets. Utilizing the semantic information is an important factor that influences the learning speed and learning performance. ADGSGP equipped with angle-awareness, which explores the geometry properties of the search operators in the semantic space, generally fits the target semantic much faster and more accurately than GP and exact GSGP.

C. Generalization Performance

Compared with the training performance, we are more interested in the generalization ability of the GSGP methods. Fig. 5 (and Fig. 1 in the supplementary material) shows the evolutionary plots on the median generalization errors of the best-of-generation models on the test sets. As shown in the figures, GSGP continues the advantage over standard GP only on five of the 14 test sets. While on the other nine datasets, where GSGP has worse training performance, it also generalizes worse than standard GP. On BHouse and R1, along with the increase of the generations, the difference between the generalization performance of GSGP and standard GP has decreased. At the end of the evolutionary process, they have almost the same generalization errors.

For AGSGP and ADGSGP, on the synthetic test sets, the overall pattern is very similar to that on the training sets. Both of them can generalize well on these test sets. Meanwhile, ADGSGP generalizes the best on ten test sets. It generalizes significantly better than GSGP on all these test sets, and significantly better than AGSGP on eight of the ten synthetic test sets. However, for AGSGP, the overall pattern on the test sets of the real-world datasets is very different from that on the training sets. AGSGP loses the advantage and is difficult to generalize well on the unseen data of the four real-world problems. In AGSGP, overfitting happens on all the four problems, where a severe overfitting occurs on BHouse and Wine, and a weak overfitting appears on DLBCL and Concrete. Unlike AGSGP, the proposed ADGSGP method can generalize well and resist to overfitting on *all* the test sets. A possible reason is, compared with RDO in AGSGP, RSM in ADGSGP produces offspring highly correlated with their parent. Along with the evolutionary process, when both AGX in AGSGP and PC in ADGSGP have difficulties bringing further progress to the search process, RSM brings a smaller variation to the offspring than RDO, so it has the potential to limit the deterioration of the generalization error. In other words, RSM is less greedy when approximating the target semantics, which drives

ADGSGP to generalize well and resist to overfitting. This can be shown by the fact that the stage where AGSGP advances ADGSGP on the training set is right the time when overfitting happens in AGSGP. Another possible contribution to the generalization is from SCR, which is our new method to fulfil the semantic requirement. Compared with SB in AGSGP, SCR induces simpler programs that are more likely to generalize better on the unseen data. This will be confirmed later in the analysis of the evolved programs in Section VI.

In general, on the majority of the test sets, the three GSGP methods generalize much better than standard GP, which confirms that the geometric properties of the geometric operators, i.e., the error committed by the child program is bounded by the worst (or best for ADGSGP) parents, also hold on the unseen data. This is especially the case for the synthetic datasets, where the problem is easier than the real-world problems with a large number of instances to some extent. These smaller semantic spaces have less or no noise, so the pattern of the target semantics in the training set can represent the actual pattern of the problem. This also explains why the overall pattern on the test sets of the synthetic datasets and DLBCL (where the number of instance is much smaller than other three real-world datasets) is almost the same as that of the training sets.

On the other hand, the geometric operators in GSGP can also bring negative variations to GP individuals, which could increase the generalization errors of the offspring. When the majority of the variations produced by these operators are negative, overfitting occurs. Compared with RDO, the variations imposed to the parents are bounded and smaller in RSM, which makes the deterioration (increase) of the generalization error to be slower and consequently lead to a better generalization in ADGSGP.

D. Comparisons on Program Size and Computational Time

Table V (and Table I in the supplementary material) shows the mean and minimum program sizes in terms of the number of nodes in the best-of-run GP individuals. The computational costs are also presented in the form of the average and minimum computation time (in seconds) of one GP run in the evolutionary training process, where N/A means that we did not consider the computational cost of exact GSGP, since the implementation method of GSGP does not generate the GP trees explicitly.

It is clearly shown that exact GSGP and AGSGP have notably larger program sizes than standard GP on most of the datasets. Different from the other three GP methods, exact GSGP does not have a limitation on the maximum height of the GP tree. The exact geometric operators lead to a exponential or linear growth on the size of the offspring. So it has a much larger program size than the other three GP methods. On the real-world problems, the mean program sizes of exact GSGP are in millions. In this case, the evolved programs are impossible to be interpreted by human experts and lose the advantage of GP over the traditional machine learning algorithms such as support vector regression [31], which perform a black-box regression. In AGSGP, the size of the evolved programs is 3–10 times larger than their counterparts in standard GP. This might be due to the fact that SB is prone to find more complex trees with the desired (or approximate) semantics to replace the original prefix of the tree. The complexity difference between the new subtree and the original prefix accumulates over generations, which will lead to a much larger GP individual. As expected, ADGSGP has a much smaller program size than AGSGP in all the examined cases. This confirms our hypothesis that replacing the context of the tree by SCR has a benefit over SB in restricting the increase of the program size. On the other hand, the average size of the evolved programs in ADGSGP is 1.3 to around three times larger than standard GP. On Keijzer14, it even has a smaller mean program size than standard GP. This indicates that ADGSGP will not decrease the interpretability of GP programs too much and the increase of the computation cost is also acceptable. More importantly, the program simplification methods [32], [33] might work for ADGSGP (which is not so easy for oversize programs in AGSGP). This will help address the open issue of over-grown program size in GSGP methods. Another pattern in the program size is that, compared with GP and AGSGP, the size of the programs in ADGSGP is more stable. This is indicated by a much smaller difference between the mean and the minimum program size. ADGSGP does not produce programs with extremely large/small size.

An interesting phenomenon is that these oversize/complex programs in GSGP methods usually generalize better than their counterparts in standard GP on unseen data. This conflicts with the widely accepted theories, such as the minimum description length principle [34] and Occam's razor [35], which claim that complex programs are difficult to generalize well.

TABLE V
PROGRAM SIZE AND COMPUTATIONAL TIME

Benchmarks	Method	Program size (Node)	Time(in second)
		Mean(Minimum)	Mean(Minimum)
BHouse	GP	142.36(11)	47.0(18.28)
	GSGP	1.46E17(1.04E10)	N/A
	AGSGP	1029.68(331)	2237.04(1512.26)
	ADGSGP	225.4(97)	614.84(384.2)
Wine	GP	103.36(39)	44.07(29.06)
	GSGP	2.89E21(2.78E19)	N/A
	AGSGP	1222.71(443)	5052.86(4298.66)
	ADGSGP	258.36(129)	673.58(411.79)
DLBCL	GP	54.6(1)	21.51(3.48)
	GSGP	4.56E24(4.85E23)	N/A
	AGSGP	175.04(1)	550.08(97.98)
	ADGSGP	150.52(75)	440.15(306.91)
Concrete	GP	146.24(63)	54.1(28.62)
	GSGP	3.89E12(1.85E8)	N/A
	AGSGP	1191.4(637)	4342.31(3636.02)
	ADGSGP	196.88(107)	628.43(485.42)
Nguyen7	GP	157.32(35)	53.8(26.34)
	GSGP	2.51E23(8.2E13)	N/A
	AGSGP	1542.88(275)	963.21(332.65)
	ADGSGP	317.2(111)	215.66(140.18)
Keijzer11	GP	254.28(143)	82.36(46.67)
	GSGP	7.85E23(6.8E21)	N/A
	AGSGP	946.08(489)	767.76(492.48)
	ADGSGP	382.68(187)	322.62(203.3)
Keijzer14	GP	265.75(175)	91.6(52.85)
	GSGP	3.08E20(1.12E17)	N/A
	AGSGP	592.24(175)	636.12(207.32)
	ADGSGP	195.0(33)	212.05(84.62)
Pagie1	GP	159.92(47)	59.52(29.47)
	GSGP	1.83E25(4.25E22)	N/A
	AGSGP	1238.92(1)	4482.86(168.99)
	ADGSGP	243.0(99)	370.81(273.0)

TABLE VI
EXAMPLES OF GP INDIVIDUALS ON KEIJZER14:8.0/$(2+x^2+y^2)$

Method	Evolved Model	Simplified Model
GP	+(%(+(%(+(%(*(x, +(x, +(x, x))), -(*(+(-(+(+(y, y), y), y), %(y, +(x, y))), y), y)), +(%(*(+(y, y), y), *(*(+(-(+(+(y, y), y), -(-(%(*(x, +(y, x)), x), -(x, +(-(y, y), x))), -(-(y, %(-(%(x, x), -(x, x)), y)), +(+(y, %(-(x, y), x)), *(*(%(y, y), *(x, y)), +(y, y))))))), %(y, +(y, y))), y), y)), *(+(+(y, y), +(%(y, y), +(%(y, y), %(y, *(+(%(y, y), y), y))))), y))), +(y, %(*(+(x, x), x), -(x, y)))), +(%(*(x, x), *(+(%(%(y, +(y, y)), *(y, y)), *(+(+(*(x, x), *(x, x)), %(y, x)), y)), %(+(%(*(x, x), -(*(x, %(y, x)), y)), +(%(*(x, x), *(*(+(y, %(%(*(-(x, x), x), *(x, y)), +(y, y))), y), y)), y))), *(+(y, %(y, x)), y))), -(*(+(-(+(+(y, y), y), y), y), y), %(y, y))), +(%(*(-(%(x, x), x), x), *(+(%(y, y), %(y, *(+(%(y, y), y), y))), y)), *(+(%(%(y, y), y), %(y, *(+(%(y, y), y)))), y)))	$((y^2+xy^2)(x+2x^3y^3+2y^4)(y^3+y^4+x^2)+2x^4y^6)/(x(3y^2-1)(2y^4+2x^3y^3+x)(y^4+y^3+x^2))+((3x^2(x^2-y^2))/(y(2xy+2y^2-x))+(y(x-y)(2y^2+4y+3))/((1+y)+(4xy(x-y)/4xy^2-2x^2y-2x-xy+2y^2-4x^2y^3))/((2x^2+xy-y^2)(3y^2-1))+(1.5x(x-1)(y+1)+3y(y+2))/(1.5y(y+2))$
AGSGP	%(%(3.96, y), -(*(%(x, +(*(-(%(%(x, -(*(*(x, +(*(%(y, y), %(x, %(-(2.33, y), x))), x)), %(y, y)), y)), -(*(y, x), +(x, x))), %(x, %(+(*(-(+(*(y, x), %(x, x)), %(x, %(y, %(x, y)))), %(*(*(-(x, y), +(y, x)), x), -(+(-(x, y), *(%(x, 0.459), x)), x))), *(x, *(*(x, x), x))), %(x, y)))), +(*(%(%(*(%(+(x, +(5.107e-15, %(*(1.156, x), %(%(x, y), x)))), %(%(1.108, *(x, y)), %(x, y)))), +(*(x, y), *(y, x))), *(-(+(x, x), *(x, x)), *(+(+(x, 0.999), *(x, x)), x))), +(*(-(+(*(y, x), %(*(-(+(y, y), -(y, y)), *(+(y, y), *(x, y))), -(-(*(x, x), *(y, y)), %(%(y, y), -(x, x)))))), %(x, %(y, %(x, y)))), %(*(-(*(x, x), *(y, y)), x), *(-(%(x, -(*(y, x), +(x, x))), -(-(*(y, 3.291), +(y, x)), %(+(y, y), %(x, x)))), %(-(+(%(y, x), %(y, y)), %(-(y, x), x)), *(%(%(y, 0.5), -(x, x)), +(%(x, x), +(x, y))))))), *(x, +(5.107e-15, x)))), x), x)), *(x, %(x, -(*(*(x, x), %(y, y)), %(y, *(x, *(*(%(x, 1.663), +(y, y)), y))))))), x), -0.232))	$x^2/((x/((x^3/(2.33-y)+x^2-y)(xy-2x))-(x^2y(2.178x^2-y))/((xy^3+y^2-x^2)(x^3-xy^2)+x^4y^2(2.178x^2-y)))*(1.805x^4y(1.156xy+x+5.107E^{-15})/(x^3(x^2-y^2)/(y^2(1/(y-2)-0.291y+x)*(x+y+1)+(x^2+5.107e^{-15}x)))+x)+2x^4y/(2x^4y-1.663))+0.232$
ADGSGP	*(*(%(106.146, *(3.893, +(-0.004, *(106.653, *(0.069, +(0.066, *(4.357, +(0.123, *(0.115, +(0.368, *(0.483, +(0.581, *(1.914, +(*(y, y), *(x, x))))))))))))), +(-0.027, *(15.572, %(0.066, *(0.258, *(0.069, +(0.066, *(4.357, +(0.123, *(0.115, +(0.368, *(0.483, +(0.581, *(1.914, +(*(y, y), *(x, x))))))))))))), *(25.004, %(0.04, *(0.011, *(1.064, +(0.341, *(0.603, %(+(0.581, *(1.914, +(*(y, y), *(x, x)))), +(*(y, y), *(x, x))))))))))	$7.99/(2+x^2+y^2)*(124.49/(2+x^2+y^2)-0.027)*(1/(0.0175+0.004/(x^2+y^2)))$
GP	-(+(%(-(%(+(%(-(%(*(+(+(+(x, x), x), %(%(+(x, %(+(y, x), *(y, x))), x), *(y, *(y, y)))), +(x, %(%(+(x, %(x, *(y, %(y, x)))), x), *(x, *(%(x, x), +(x, x))))))), -(+(%(-(%(*(+(x, x), x), y), x), y), %(x, %(x, +(-(x, +(x, *(x, y))), -(x, +(-(x, x), x))))))), %(*(*(y, -(x, y)), -(%(%(x, x), *(y, *(y, x))), %(y, +(y, *(%(x, x), y))))), y))), %(%(*(+(-(+(x, y), +(-(-(x, y), %(y, x)), %(x, *(y, y)))), x), -(%(%(x, x), *(y, -(x, y))), y)), x), +(x, x))), y), *(x, +(%(y, x), y))), -(+(%(-(%(*(+(+(x, y), x), x), y), x), %(*(+(-(+(x, y), *(+(y, x), x)), x), -(%(%(x, x), *(y, %(y, y)))), y)), x)), x), %(*(+(-(+(x, y), *(+(x, x), x)), x), -(%(%(x, x), *(y, +(y, +(x, x)))), y)), x))), *(y, x)), y), *(x, +(x, %(y, *(+(+(x, y), x), x))))), %(%(*(+(-(+(x, y), +(-(-(x, y), %(y, x)), %(x, x))), x), -(%(%(x, y), *(y, -(x, *(+(x, x), x)))), %(y, y))), x), +(x, x)))	$((y^2+xy^2+3x)/y+((2x^2y+x+y)/(y^3(2x^3y+y^2+x^3))))*((x(1-y^3)(2x+y-x^2-xy))/(2x^4y-(1-y^3)(2x+y-x^2-xy)((2x+y-2x^2)*(1-y^2(x+y))/(2xy+y^2)-x^2)))-x+y/(2x+y)+x^2$
AGSGP	+(-(-(x, %(+(-0.007, y), *(y, x))), -(-(+(*(x, x), x), %(+(0.003, y), *(y, x))), %(%(0.276, *(x, %(%(%(+(%(%(+(x, %(%(+(x, %(x, y)), *(y, y)), y)), -(*(y, *(%(y, x), +(-(x, y), x))), x)), %(*(x, x), *(y, y))), +(x, %(%(+(x, x), *(y, x)), %(1.018, *(y, y))))), %(1.018, *(%(+(y, y), -(*(*(x, +(*(x, y), *(x, x))), %(x, %(*(x, x), *(y, y)))), %(*(*(*(y, x), -(x, y)), x), y))), +(%(y, *(*(y, *(%(y, x), +(x, x))), x)), %(+(y, y), %(y, x)))))), %(1.018, *(%(%(+(y, y), *(+(y, y), %(y, y))), x), %(%(+(y, y), x), x)))), y))), *(%(+(y, y), *(%(y, x), +(y, y))), *(*(y, *(*(*(*(%(+(y, y), +(*(y, y), *(x, %(%(+(x, x), 1.481), %(x, %(1.29, x)))))), y), y), y), y)), y))))), %(3.523, %(*(%(y, *(y, y)), %(+(y, y), *(y, y))), *(%(%(%(+(x, x), 1.481), %(0.736, y)), *(%(%(%(%(+(x, x), %(*(%(%(+(x, x), 1.481), %(x, %(y, x))), x), *(*(*(y, y), +(x, x)), %(%(-(+(x, x), -(%(y, 0.368), x)), *(-(x, y), *(*(y, x), y)))), *(%(y, y), %(+(y, y), *(y, y))))))), %(*(x, x), *(%(+(y, y), -(*(-(+(x, x), -(y, x)), %(x, %(*(x, x), *(y, y)))), %(*(-(+(x, x), +(y, x)), x), y))), y)), y), y), %(%(x, x), +(y, y)))), %(%(%(+(x, x), -(*(y, *(%(y, x), +(x, x))), x)), x), %(*(x, x), *(y, y))), y))))))	$(4.391x-4.358y-3x^4y+2.72x^3y^2)/(xy(3x-2.72y))-((0.48x(y+1)(y^3-x^2)(2x^2y-xy^2-x^3))/(y^3(y^4+y+1+xy^2(y^3-x^2)+1.96y^3(y^3-x^2))*(4x^2y+(y+1)(2x^2y-xy^2-x^3)))$
ADGSGP	%(8543.368, +(-0.003, *(2290.162, *(0.004, +(0.007, *(8543.193, *(0.062, +(0.039, *(1.649, +(-0.257, *(101.807, +(-0.056, *(0.019, +(3.206, *(0.069, +(*(y, y), *(x, x))))))))))))))))	$8.11/(2.01+x^2+y^2)$

A possible explanation on this phenomenon is that not only the size/complexity of the programs matters, but also the way how they are generated is also important. These theories might not hold among models that are produced in different manners. Previous research considers these oversize programs as ensembles of programs [6]. They claimed that there might exist some overfitted subprograms in the final evolved programs (the same as in an ensemble). However, their contribution to an increased generalization error can be reduced/eliminated by their counterparts in the programs that generalize well.

Considering the comparison of the computational cost between standard GP, AGSGP, and ADGSGP, AGSGP and ADGSGP have a much higher computational cost than standard GP. The maintenance and the search procedure of the semantic library in AGSGP and ADGSGP takes a large amount of time. In addition, along with the growth of the program size, the procedure of calculating the desired semantics in SB in AGSGP and SCR in ADGSGP becomes more costly, particularly in SB which needs to propagate backward from the root node to the selected node in the tree. In the worst case, it needs to backpropagate through almost the whole tree. In SCR, only one step propagation is needed after the root node of the context is decided. This explains why the computational cost in ADGSGP is much smaller than that in AGSGP. Another important reason is the much smaller program size in ADGSGP than AGSGP. ADGSGP needs additional expense for the angle-awareness in selection and breeding process. However, compared with the much higher cost in searching for the desired contexts (or the desired subtree in AGSGP), this additional cost can be neglected.

E. Analysis of Evolved Models

A further examination of the models evolved by the three GP methods is conducted (since the models evolved by GSGP is too large to show and analyze, they are omitted from the analysis). We randomly sample one run from the 50 GP runs and examine the evolved models of the three GP methods in that run, and we repeat this process twice to check two (random) groups of models on Keijzer14, where the target function is known, and both AGSGP and ADGSGP achieve good learning and generalization performance. To make the analysis clearer, we also present the mathematically simplified form of these models. The two forms of models are displayed in the second and the third columns of Table VI, respectively.

It is clear that the evolved models in ADGSGP are much simpler than those in AGSGP. The difference between the simplified models, which indicate the behavior of the evolved models, is even more apparent. In the two examples, the behavior of the models in ADGSGP is not only much smoother than their counterparts in standard GP and AGSGP, but also much more similar to the target functions (ADGSGP actually finds the most important building block $x^2 + y^2$ on Keijzer14, and the only difference from the target function is in the parameters of the models). The difference on the simplified models of AGSGP and ADGSGP also indicates that the reasons why the two GSGP methods can outperform standard GP might be different. The advantage of AGSGP over standard GP might come from the same strength as ensemble learning, while ADGSGP is able to find the target model, which not only has better learning performance but also can generalize very well.

VI. Further Analysis

To further investigate the effect of the three angle-driven operators proposed in this paper, PC with standard selection operator (i.e., tournament) (denoted as PC-SS), PC with the proposed ADS (PC), and RSM are tested on the eight selected datasets individually. They are compared with exact GSGP and AGSGP. Therefore, in this section, the tested GSGP methods are: GSGP, AGSGP, PC-SS, PC, and RSM.

As shown in Fig. 6, on the four synthetic datasets, PC-SS, PC, and RSM all have significantly better training performance than GSGP. Compared with AGSGP, on Keijzer11, PC-SS and PC have significantly larger training RMSEs, while RSM has slightly larger training RMSEs. On the other three synthetic datasets, PC-SS, PC, and RSM all have better training performance than AGSGP. On Nguyen7, the differences between them are not significant, while on the other two training sets, PC-SS, PC, and RSM all have significantly better training performance than AGSGP.

On three of the four real-world datasets (except for DLBCL), PC-SS, PC, and RSM all have notable improvement on the training performance than GSGP, but are much worse than AGSGP. On DLBCL, PC-SS and PC have slightly larger training errors than GSGP, which are not significant. RSM has significantly better training performance than GSGP. On DLBCL, PC-SS, PC, and RSM all have better training performance than AGSGP. While PC-SS has slightly better performance, the advantage of PC and RSM over AGSGP is significant.

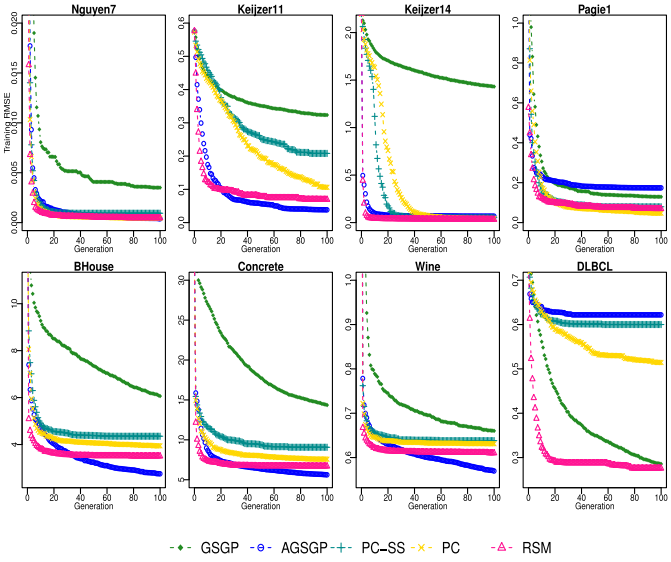

Fig. 6. Evolutionary plots on the *Training* RMSEs of the best-of-generation models in GSGP, AGSGP, and GSGP with three operators solely.

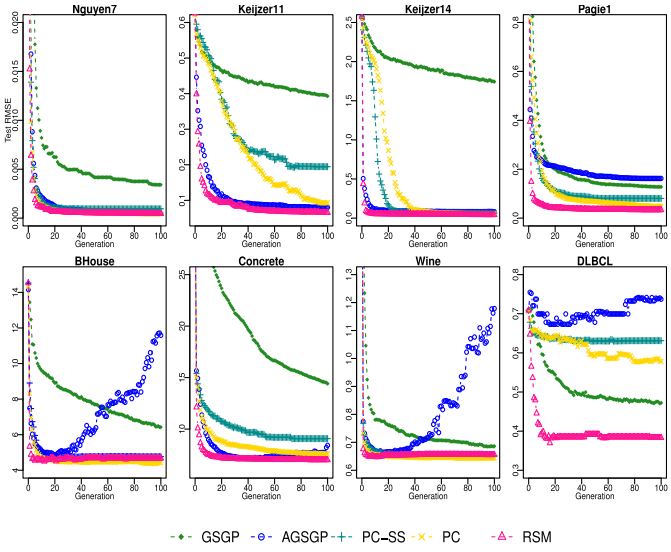

Fig. 7. Evolutionary plots on the *Test* RMSEs of the best-of-generation models in GSGP, AGSGP, and GSGP with three operators solely.

The evolutionary plots on the generalization performance in Fig. 7 show that the overall pattern on the four synthetic test sets is very similar to that on the training sets. PC-SS, PC, and RSM all have better generalization performance than GSGP and AGSGP. On Keijzer11, AGSGP loses the advantage on the generalization performance. PC has similar generalization performance with AGSGP, while RSM has the best generalization performance. On the four real-world datasets, the pattern is very different from that on the training sets. On the four datasets, where AGSGP suffers from serious overfitting, PC-SS, PC, and RSM generally generalize well. Among the five GSGP methods, PC has the best test performance on BHouse and Wine, while on the other two datasets, RSM is the winner.

In summary, GSGP with any of the three proposed operators solely can outperform GSGP on most of the examined datasets on both learning performance and generalization ability. When compared with AGSGP, they have comparable learning ability

but much better generalization performance. PC and RSM have comparable training and generalization performance, which are much better than PC-SS. The only difference between PC-SS and PC is in the selection operator. PC which employs the proposed ADS can improve the performance of GSGP much better than PC-SS (where standard tournament selection is used). This is a good evidence for the effectiveness of ADS. In addition, as can be seen from Figs. 4–7 (using the performance of GSGP and AGSGP as baseline for comparison), compared with employing PC or RSM individually, combining them (i.e., ADGSGP) has a better effect on promoting both training and generalization performance of GP.

The performance of GPAGLX, i.e., the method proposed in our preliminary work [8], has also been examined. Due to the page limit, the results and the detailed analysis are present in the supplementary material. A brief summary is that GPAGLX has comparable training performance but significantly worse generalization than ADGSGP on most of the examined datasets.

VII. Conclusion

The goal of further exploring the geometric properties of geometric operators to obtain a greater generalization gain in GP for symbolic regression has been achieved. The proposed angle-driven geometric operators give an impressive generalization improvement for GP. With angle-awareness, the error of the child programs produced by PC is bounded by their best parents, and RSM provides a small variation to the parent programs but consistently move toward the target semantics. These new geometric properties offer a productive leverage for approximating the target semantics in each operation.

To investigate and confirm the effectiveness of the proposed ADGSGP method, this paper conducted a comprehensive comparison among ADGSGP, exact GSGP and AGSGP. To the best of our knowledge, this is the first work filling the gap of the comparison between these two variants of GSGP. Standard GP was used as a baseline for the comparison. Compared with GP, the GSGP methods generally evolved models with better learning performance and generalization ability. However, the worse learning and generalization performance (than standard GP) in exact GSGP on some datasets and the overfitting trend in AGSGP on the real-world datasets also indicate that introducing the semantics into the evolutionary process does not always bring benefits. The way to utilize the semantic information has a high influence on the performance of GP. The angle-aware geometric operators eliminate the potential ineffectiveness in GSGP and increase the chance of effective breeding, which make ADGSGP advance exact GSGP and AGSGP dramatically on learning performance with a much faster learning speed. More importantly, in ADGSGP, RSM is less greedy (than RDO) in approximating the target semantics and brings a smaller variation to the parent programs. The SCR consistently produces much simpler/smaller programs, which are more likely to contain the right structure than the two state-of-the-art GSGP methods. All these characteristics make the solutions more resistant to overfitting and generalize better on the unseen data.

Another interesting finding is the models in the GSGP methods usually generalize well but overcomplex, which conflicts with the common claim that complex programs are difficult to generalize well. The better generalization gain of these complex models might come from the same strength as ensembles. Furthermore, the simplified form of the evolved models in ADGSGP are closer to the target model (e.g., containing the most important building blocks) and represent the correct pattern, thus generalize well on unseen data.

To sum up, the comparisons among four GP methods (including two of state-of-the-art GSGP methods) indicates that the proposed method has a faster convergence rate, a good interpret ability, and requires less computational effort. It is very suitable for real-world applications requiring a fast responsibility and aiming to have a good insight of the underlying data generating process.

For future work, we plan to further improve ADGSGP by reducing the computational cost and making it more efficient. As mentioned above, the major expense of the method is in maintaining and searching in the semantic library. Instead of exhaustive search used in ADGSGP, a heuristic search method will be explored to improve the efficiency while not decreasing the effectiveness of the process of searching for the desired subtrees. Moreover, the selection operator in ADGSGP is based on an indirect manner. We will consider to measure the distribution of the population and select the GP individuals according to their angle distributions around the target semantics in the future.

References

[1] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, vol. 1. Cambridge, MA, USA: MIT Press, 1992.

[2] K. Krawiec and P. Lichocki, "Approximating geometric crossover in semantic space," in *Proc. 11th Annu. Conf. Genet. Evol. Comput. Conf. (GECCO)*, 2009, pp. 987–994.

[3] L. Beadle and C. G. Johnson, "Semantically driven crossover in genetic programming," in *Proc. Evol. Comput.*, 2008, pp. 111–116.

[4] A. Moraglio, K. Krawiec, and C. G. Johnson, "Geometric semantic genetic programming," in *Proc. Int. Conf. Parallel Problem Solving Nat.*, 2012, pp. 21–31.

[5] L. Vanneschi, "An introduction to geometric semantic genetic programming," in *Proc. NEO*, 2017, pp. 3–42.

[6] I. Gonçalves, S. Silva, and C. M. Fonseca, "On the generalization ability of geometric semantic genetic programming," in *Proc. 18th Eur. Conf. Genet. Program. (EuroGP)*, 2015, pp. 41–52.

[7] L. Vanneschi, M. Castelli, L. Manzoni, and S. Silva, "A new implementation of geometric semantic GP and its application to problems in pharmacokinetics," in *Proc. 16th Eur. Conf. Genet. Program. (EuroGP)*, 2013, pp. 205–216.

[8] Q. Chen, B. Xue, Y. Mei, and M. Zhang, "Geometric semantic crossover with an angle-aware mating scheme in genetic programming for symbolic regression," in *Proc. 20th Eur. Conf. Genet. Program. (EuroGP)*, 2017, pp. 229–245.

[9] Q. Chen, M. Zhang, and B. Xue, "New geometric semantic operators in genetic programming: Perpendicular crossover and random segment mutation," in *Proc. 19th Annu. Conf. Genet. Evol. Comput. Companion (GECCO)*, 2017, pp. 223–224.

[10] K. Krawiec and T. Pawlak, "Locally geometric semantic crossover: A study on the roles of semantics and homology in recombination operators," *Genet. Program. Evol. Mach.*, vol. 14, no. 1, pp. 31–63, 2013.

[11] T. P. Pawlak, B. Wieloch, and K. Krawiec, "Semantic backpropagation for designing search operators in genetic programming," *IEEE Trans. Evol. Comput.*, vol. 19, no. 3, pp. 326–340, Jun. 2015.

[12] S. Ruberto, L. Vanneschi, M. Castelli, and S. Silva, "Esagp—A semantic GP framework based on alignment in the error space," in *Proc. Eur. Conf. Genet. Program.*, 2014, pp. 150–161.
[13] M. Castelli, L. Vanneschi, S. Silva, and S. Ruberto, "How to exploit alignment in the error space: Two different GP models," in *Genetic Programming Theory and Practice XII*. Cham, Switzerland: Springer, 2015, pp. 133–148.
[14] K. Krawiec and T. Pawlak, "Approximating geometric crossover by semantic backpropagation," in *Proc. 15th Annu. Conf. Genet. Evol. Computat. Conf. (GECCO)*, 2013, pp. 941–948.
[15] S.-I. Amari and S. Wu, "Improving support vector machine classifiers by modifying kernel functions," *Neural Netw.*, vol. 12, no. 6, pp. 783–789, 1999.
[16] D. Cohn, L. Atlas, and R. Ladner, "Improving generalization with active learning," *Mach. Learn.*, vol. 15, no. 2, pp. 201–221, 1994.
[17] Q. Chen, B. Xue, L. Shang, and M. Zhang, "Improving generalisation of genetic programming for symbolic regression with structural risk minimisation," in *Proc. 17th Annu. Conf. Genet. Evol. Comput. Conf. (GECCO)*, 2016, pp. 709–716.
[18] M. A. Haeri, M. M. Ebadzadeh, and G. Folino, "Improving GP generalization: A variance-based layered learning approach," *Genet. Program. Evol. Mach.*, vol. 16, no. 1, pp. 27–55, 2015.
[19] S. S. M. Astarabadi and M. M. Ebadzadeh, "Avoiding overfitting in symbolic regression using the first order derivative of GP trees," in *Proc. 17th Annu. Conf. Genet. Evol. Comput. Conf. (GECCO)*, 2015, pp. 1441–1442.
[20] Q. Chen, M. Zhang, and B. Xue, "Feature selection to improve generalization of genetic programming for high-dimensional symbolic regression," *IEEE Trans. Evol. Comput.*, vol. 21, no. 5, pp. 792–806, Oct. 2017.
[21] I. Gonçalves and S. Silva, "Balancing learning and overfitting in genetic programming with interleaved sampling of training data," in *Proc. 16th Eur. Conf. Genet. Program. (EuroGP)*, 2013, pp. 73–84.
[22] N. Q. Uy, N. T. Hien, N. X. Hoai, and M. O'Neill, "Improving the generalisation ability of genetic programming with semantic similarity based crossover," in *Proc. 13th Eur. Conf. Genet. Program. (EuroGP)*, 2010, pp. 184–195.
[23] N. Q. Uy, N. X. Hoai, M. O'Neill, R. I. McKay, and E. Galván-López, "Semantically-based crossover in genetic programming: Application to real-valued symbolic regression," *Genet. Program. Evol. Mach.*, vol. 12, no. 2, pp. 91–119, 2011.
[24] H. Majeed and C. Ryan, "A less destructive, context-aware crossover operator for GP," in *Proc. 9th Eur. Conf. Genet. Program. (EuroGP)*, 2006, pp. 36–48.
[25] M. Keijzer, "Improving symbolic regression with interval arithmetic and linear scaling," in *Proc. 6th Eur. Conf. Genet. Program. (EuroGP)*, 2003, pp. 70–82.
[26] F.-A. Fortin *et al.*, "DEAP: Evolutionary algorithms made easy," *J. Mach. Learn. Res.*, vol. 13, pp. 2171–2175, Jul. 2012.
[27] J. McDermott *et al.*, "Genetic programming needs better benchmarks," in *Proc. 14th Annu. Conf. Genet. Evol. Comput. Conf. (GECCO)*, 2012, pp. 791–798.
[28] H. Al-Sahaf, M. Zhang, A. Al-Sahaf, and M. Johnston, "Keypoints detection and feature extraction: A dynamic genetic programming approach for evolving rotation-invariant texture image descriptors," *IEEE Trans. Evol. Comput.*, vol. 21, no. 6, pp. 825–844, Dec. 2017.
[29] E. Hancer, B. Xue, and M. Zhang, "Differential evolution for filter feature selection based on information theory and feature ranking," *Knowl. Based Syst.*, vol. 140, pp. 103–119, Jan. 2018.
[30] F. Fernández, M. Tomassini, and L. Vanneschi, "An empirical study of multipopulation genetic programming," *Genet. Program. Evol. Mach.*, vol. 4, no. 1, pp. 21–51, 2003.
[31] H. Drucker, C. J. C. Burges, L. Kaufman, A. J. Smola, and V. Vapnik, "Support vector regression machines," in *Proc. Adv. Neural Inf. Process. Syst.*, 1997, pp. 155–161.
[32] M. Zhang, P. Wong, and D. Qian, "Online program simplification in genetic programming," in *Simulated Evolution and Learning*. Heidelberg, Germany: Springer, 2006, pp. 592–600.
[33] P. Wong and M. Zhang, "Algebraic simplification of GP programs during evolution," in *Proc. 8th Annu. Conf. Genet. Evol. Comput. Companion (GECCO)*, 2006, pp. 927–934.
[34] J. Rissanen, "Modeling by shortest data description," *Automatica*, vol. 14, no. 5, pp. 465–471, 1978.
[35] W. M. Thorburn, "The myth of Occam's razor," *Mind*, vol. 27, no. 107, pp. 345–353, 1918.

AQ2

Qi Chen received the B.E. degree in automation from the University of South China, Hengyang, China, in 2005, the M.E. degree in software engineering from the Beijing Institute of Technology, Beijing, China, in 2007, and the Ph.D. degree in computer science from the Victoria University of Wellington (VUW), Wellington, New Zealand, in 2018.

Since 2014, she has joined the Evolutionary Computation Research Group, VUW, where she is currently a Post-Doctoral Research Fellow with the School of Engineering and Computer Science. Her current research interests include genetic programming for symbolic regression, machine learning, evolutionary computation, feature selection, feature construction, transfer learning, domain adaptation, and statistical learning theory.

Dr. Chen serves as a Reviewer for international conferences, including the IEEE Congress on Evolutionary Computation, and international journals, including the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION and the IEEE TRANSACTIONS ON CYBERNETICS.

Bing Xue (M'10) received the B.Sc. degree from the Henan University of Economics and Law, Zhengzhou, China, in 2007, the M.Sc. degree in management from Shenzhen University, Shenzhen, China, in 2010, and the Ph.D. degree in computer science from the Victoria University of Wellington, Wellington, New Zealand, in 2014.

She is currently a Senior Lecturer with the School of Engineering and Computer Science, Victoria University of Wellington. She has over 100 papers published in fully refereed international journals and conferences and most of them are on evolutionary feature selection and construction. Her current research interests include evolutionary computation, feature selection, feature construction, multiobjective optimization, image analysis, transfer learning, data mining, and machine learning.

Dr. Xue is an Associate Editor/Editorial Board Member for five international journals, and a Reviewer of over 50 international journals. She is the Finance Chair of the IEEE Congress on Evolutionary Computation 2019, the Program Co-Chair of the 31st Australasian AI 2018, ACALCI 2018, and the 7th International Conference on SoCPaR2015, and the Tutorial Chair, the Special Session Chair, or the Publicity Chair for several other international conferences. She is currently the Chair of the IEEE Task Force on Evolutionary Feature Selection and Construction, IEEE Computational Intelligence Society (CIS), and the Vice-Chair of the IEEE CIS Data Mining and Big Data Analytics Technical Committee and the IEEE CIS Task Force on Transfer Learning and Transfer Optimisation.

Mengjie Zhang (M'04–SM'10) received the B.E. and M.E. degrees from the Artificial Intelligence Research Center, Agricultural University of Hebei, Baoding, China, in 1989 and 1992, respectively, and the Ph.D. degree in computer science from RMIT University, Melbourne, VIC, Australia, in 2000.

He is currently a Professor of Computer Science, the Head of the Evolutionary Computation Research Group, and the Associate Dean (Research and Innovation) with the Faculty of Engineering, Victoria University of Wellington, Wellington, New Zealand. He has published over 350 research papers in refereed international journals and conferences. His current research interests include evolutionary computation, particularly genetic programming, particle swarm optimization, and learning classifier systems with application areas of image analysis, multiobjective optimization, feature selection and reduction, job shop scheduling, and transfer learning.

Prof. Zhang is currently the Chair of the IEEE Computational Intelligence Society (CIS) Intelligent Systems and Applications Technical Committee, the immediate Past Chair of the IEEE CIS Emergent Technologies Technical Committee and the Evolutionary Computation Technical Committee, and a member of the IEEE CIS Award Committee. He is the Vice-Chair of the IEEE CIS Task Force on Evolutionary Feature Selection and Construction and the Task Force on Evolutionary Computer Vision and Image Processing, and the Founding Chair of the IEEE Computational Intelligence Chapter in New Zealand. He is also a Committee Member of the IEEE NZ Central Section. He is a fellow of the Royal Society of New Zealand and has been a Panel Member of the Marsden Fund (New Zealand Government Funding). He is also a member of ACM.