Structural Risk Minimization-Driven Genetic Programming for Enhancing Generalization in Symbolic Regression

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

***Abstract*—Generalization ability, which reflects the prediction ability of a learned model, is an important property in genetic programming (GP) for symbolic regression. Structural risk minimization (SRM) is a framework providing a reliable estimation of the generalization performance of prediction models. Introducing the framework into GP has the potential to drive the evolutionary process toward models with good generalization performance. However, this is tough due to the difficulty in obtaining the Vapnik–Chervonenkis (VC) dimension of nonlinear models. To address this difficulty, this paper proposes an SRM-driven GP approach, which uses an *experimental* method (instead of theoretical estimation) to measure the VC dimension of a mixture of linear and nonlinear regression models for the first time. The *experimental* method has been conducted using uniform and nonuniform settings. The results show that our method has impressive generalization gains over standard GP and GP with the 0.632 bootstrap, and that the proposed method using the nonuniform setting has further improvement than its counterpart using the uniform setting. Further analyzes reveal that the proposed method can evolve more compact models, and that the behavioral difference between these compact models and the target models is much smaller than their counterparts evolved by the other GP methods.**

***Index Terms*—Generalization, genetic programming (GP), structural risk minimization (SRM), symbolic regression, Vapnik–Chervonenkis (VC) dimension.**

I. Introduction

GENERALIZATION is one of the most important performance criteria for machine learning techniques [1] which reflects their prediction performance on unseen data. More specifically, the expected generalization error $\mathrm{Err}_T = E[L(Y, f(X))|_T]$ measures the prediction error of the learned model over a set of unseen/test data for a given training set T. Here, $L(Y, f(X))$ refers to the loss function between the target output Y and the output of the model $f(X)$. A set of input X and output Y pairs are considered to be drawn from an underlying distribution $P(X, Y) = P(Y|X)P(X)$, where $P(X)$ is the distribution of the input X and the conditional distribution $P(Y|X)$ is based on the input–output relation. The joint distribution $P(X, Y)$ is needed in order to measure Err_T, which is typically unknown in real-world learning tasks. Thus, many learning algorithms rely on the empirical risk minimization principle [2]. This principle consists of computing the errors of a set of candidate models over the training set, and then selects the one that obtains the minimum training error among the set of models. The empirical/training error is expected to be a good indicator of the expected test error. However, in many cases, this indicator does not work well, particularly when over-complex models have been learned, and/or the number of training samples is too small to represent the real distribution of $P(X, Y)$.

Genetic programming (GP) [3] solves regression problems by evolving models with the best-fitted structures and coefficients. GP does not require any prior assumption on the data and has a flexible representation, which make it a very suitable approach for symbolic regression. However, poor generalization is still an open issue in GP. The evolutionary process, which is guided by chasing the lowest empirical error, might suffer from severe overfitting.

Structural risk minimization (SRM) [4] in learning theory provides a powerful framework to estimate the generalization ability of models. SRM defines an upper bound of the generalization error, which is a combination of the *empirical risk/error* and the *confidence interval*. The confidence interval, which estimates a difference between the empirical risk/error and the expected risk/error, is determined by the size of the training set and the model complexity measured by Vapnik–Chervonenkis dimension (VC-dimension) [5]. For a fixed size training set, the confidence interval is determined purely by the VC-dimension. Previous research has confirmed that the model complexity directly influences its generalization ability [6], [7]. A widely accepted agreement is that, given the same training set, complex models generally have a larger difference between the training error and the test error than their simple counterparts [6], [8], [9]. Therefore, the

Manuscript received May 31, 2017; revised November 24, 2017 and April 22, 2018; accepted October 29, 2018. Date of publication November 15, 2018; date of current version July 30, 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; mengjie.zhang@ecs.vuw.ac.nz; bing.xue@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 material is 288 KB 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.2018.2881392

learning process under SRM, which tries to select the models having a good tradeoff between the empirical error and VC-dimension (i.e., model complexity), can lead to models with better generalization ability.

Despite its solid theoretical foundation and the ability in assessing the expected test error, SRM is seldom considered in GP for symbolic regression. The underlying reason is the great difficulty in obtaining a tight theoretical estimation of the VC-dimension of nonlinear models. Vapnik *et al.* [9] developed an *experimental*[1] method to measure the VC-dimension of a learning machine for classification. Our preliminary work extended it to GP for symbolic regression and developed a method named GPSRM [10]. Despite the notable generalization improvement of GPSRM over standard GP, it still has some drawbacks. An obvious one is that the *experimental* method measuring the VC-dimension of regression models was conducted under a uniform setting,[2] which potentially limits the accuracy of the estimated generalization error. This drawback will be explained in detail in Section III.

This paper aims to address the above limitation of GPSRM and performs a more comprehensive investigation on the generalization ability of SRM-driven GP on both synthetic and real-world datasets.

A. Goals

The overall goal of this paper is to develop a new GP method named GP with optimized[3] SRM (GPOPSRM) for further improving the generalization performance of GP for symbolic regression. The new method represents a substantial improvement over the previous method GPSRM proposed in [10]. Specifically, this paper has the following research objectives.

1) Whether and how the proposed SRM-driven GP approach influences the training performance of GP.
2) Whether SRM-driven GP can lead to a notably better generalization capability than GP and GP with other generalization estimation methods.
3) How SRM-driven GP influences the complexity and behavior of the evolved models.
4) Whether SRM with a nonuniform setting can outperform its counterpart with a uniform setting in improving the training and generalization performance of GP.

II. Background

This section introduces GP for symbolic regression and reviews state-of-the-art methods on improving the generalization of GP for symbolic regression. Then we introduce two key concepts in learning theory, i.e., VC-dimension and SRM, which are the crucial components of the method to be proposed in this paper. This is followed by discussing existing implementation of SRM in different learning algorithms, including GP.

A. GP for Symbolic Regression

When addressing symbolic regression problems, GP starts from a population of randomly created regression models. Then the population is progressively evolved in an iterative way through evaluation, selection, and breeding generation by generation. The evolutionary process will continue until a predefined termination criterion has been met. Without any predefined model structure, GP is able to evolve the structure of regression models with a set of good parameter values simultaneously. This capability makes GP a suitable approach to symbolic regression. There have been many successful applications of GP for symbolic regression to date [11], [12].

B. Generalization in GP for Symbolic Regression

Generalization is a key performance criterion for measuring the goodness of learning algorithms. Although generalization has been deeply investigated in many other fields [13], [14], it had not received much attention in GP for symbolic regression for quite a long time. Before 2000, symbolic regression was mainly considered to be an optimization problem, which used all the available data for evolving the models and did not report the generalization performance of the models on unseen data. In recent years, an increasing number of approaches to promoting the generalization ability of GP for symbolic regression have been proposed [15]–[19].

Since overcomplex models can easily lead to overfitting, i.e., poor generalization, many methods try to reduce the complexity of models in order to eliminate or reduce overfitting and improve generalization on unseen data. Vanneschi *et al.* [7] and Silva *et al.* [20] introduced an equalization genetic operator to GP to control the distribution of model size, and can work well for controlling bloat and reducing overfitting. The equalization operator and its dynamic version are shown to be effective to increase the generalization ability of GP. Astarabadi and Ebadzadeh [17] 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. In addition to use the training error of the models as one objective, the root mean square error (RMSE) between the first order derivative of the models and the corresponding value of the target model is used as the second objective. The results on four symbolic regression problems show that their method can generalize better than standard GP. Vladislavleva *et al.* [6] introduced a complexity measure named *order of nonlinearity* to GP. Their method approximates the GP solutions by the Chebyshev polynomials [21] to a certain accuracy. The minimum degree of these Chebyshev polynomials is considered to the complexity of the GP solution. The measure is utilized in a Pareto GP algorithm as a competing objective to the training error, thus to generate much smoother models and lead to better generalization gain. One common limitation of the above methods is to measure the complexity of various approximations (to GP solutions) but not directly on the solutions themselves.

[1] "Experimental" was used in the original paper [9] to emphasize that the method is not a theoretical estimation.

[2] The "uniform setting" refers to the setting in the *experimental* method to obtain the maximum deviations of the error rates, which are a set of key values for measuring the VC-dimension. The uniform setting means that the number of experiments to obtain the maximum deviation of errors is the same for all datasets regardless the number of instances.

[3] "Optimized" is used in this paper, since for a set of design points used in the experiments to measure the VC-Dimension, the process of search for a better setting is an optimization process. It does not always mean the best one.

Assessing the expected test error of GP individuals properly during the evolutionary process is an intuitive way to enhance the generalization performance. One way to achieve this is bias-variance decomposition, which decomposes the expected test error into the bias error and the variance error [22]. A lower variance error indicates that the models are less sensitive to the training data, thus can potentially generalize well on unseen data. Typically the variance error is estimated by a Bootstrap method [23]. Various GP methods have employed Bootstrap techniques [24]–[26], where the GP population is trained on a list of bootstrap samples, and individuals with a lower variance error are selected.

C. Vapnik–Chervonenkis Dimension and Structural Risk Minimization

Statistical learning theory and probably approximately correct [4] define a general measure for the complexity of a learning machine, which is VC-dimension [5]. The original definition of the VC-dimension is for a set of indicator functions $\{I(X, \alpha)\}$, where X are the input vectors and α is a set of parameters. The VC-dimension h of functions $\{I(X, \alpha)\}$ is equal to the maximal number of input vectors $X_1, X_2, \ldots, X_h$ that can be *shattered* by $\{I(X, \alpha)\}$ [27]. In other words, with proper α, $\{I(X, \alpha)\}$ always can perfectly separate these vectors into two classes in all the 2^h possible ways. Late, this definition was extended for a set of real-value functions $\{R(X, \alpha)\}$, where $A \leq \{R(X, \alpha)\} \leq B$. The VC-dimension of $\{R(X, \alpha)\}$ is defined as the VC-dimension of its indicator functions $\{I(R(X, \alpha) - \beta)\}$ [4], where $\beta \in (A, B)$.

After the proposal of VC-dimension, various assessments on the *expected generalization risk* (i.e., expected test error) have been developed [28]. SRM is one of these approaches. SRM estimates the generalization error bound using the *empirical error/risk* and the *confidence interval*. While the empirical error is the error of the models on the training data, the confidence interval is determined by the size of the training set and the model complexity, i.e., the VC-dimension of the model. SRM intends to minimize the generalization error of the models in the way of taking both the empirical error and the confidence interval into consideration. In [4] and [28], a practical form of VC generalization bound for regression problems is proposed. It is defined as

$$R_{\exp}(h) \leq R_{\text{emp}}(h)\left(1 - \sqrt{p - p\ln p + \frac{\ln n}{2n}}\right)_{+}^{-1} \tag{1}$$

where $R_{\exp}(h)$ is the expected test risk, $R_{\text{emp}}(h)$ stands for the empirical risk/error of the model, $(1 - \sqrt{p - p\ln p + [\ln n/2n]})_{+}^{-1}$ represents the confidence interval ("+" denotes the positive part of $1 - \sqrt{p - p\ln p + [\ln n/2n]}$). In the confidence interval, $p = h/n$. h is the VC-dimension of the model, and n is the size of the training set. Accordingly, when learning from a fixed number of training samples, a higher VC-dimension h is more likely to lead to a larger generalization bound $R_{\exp}(h)$.

Let a set of k regression models be evaluated by SRM. These models form a nested sequence with increasing estimated generalization errors, $R_{\exp 1} < R_{\exp 2} < \cdots < R_{\exp k}$. SRM then chooses models with a lower $R_{\exp}$. These models usually have a good balance between the empirical error and the model complexity. They are expected to generalize well on unseen data. A key component and the most difficult part of SRM is how to accurately estimate of the VC-dimension of the models.

D. Implementation of the SRM Principle in Learning Algorithms

Two kinds of constructive approaches have been found to implement SRM directly into learning algorithms. The first approach is to keep the empirical error fixed and minimize the confidence interval. The design of support vector machines (SVMs) [27] follows this rule. SVMs maps the data into a high dimensional input space through some nonlinear mapping, and its kernel functions and parameters are selected to minimize the VC generalization bound. Via regularization operators, the kernel function in SVMs is associated with a flatness property. Among a set of functions which approximate the target outputs within a given precision, the flattest functions are chosen.

The second important approach to implementing SRM is to keep the confidence interval fixed and try to minimize the empirical error. This strategy is widely used in neural networks [29]. For a given number of training examples, the confidence interval of the networks is determined by the VC-dimension h of the functions for the neurons. The training process finds the weights to minimize the empirical error. Thus, in neural networks, selecting an appropriate structure for the neurons is an important task, since it will lead to a good tradeoff between underfitting and overfitting. A lot of research has been conducted to estimate a more accurate VC bound for neural networks [30], [31].

E. Implementing SRM in GP

Implementing SRM in GP is a challenging task, and only a few works can be found in the literature. When implementing SRM into GP, the decision of a tradeoff between an approximate complexity of the model (i.e., VC-dimension) and the minimal empirical error should be automatically made during the evolutionary process, since it is impossible to have a fixed confidence interval for the evolved models.

Borges *et al.* [32] and Montaña *et al.* [33] are the only work that can be found before our initial work [10]. In their work, SRM is introduced as a new fitness function to GP for symbolic regression. The VC-dimension of the evolved models is measured by a simplified estimator, which counts the number of nonscalar nodes (i.e., nodes that are not operated by the functions $\{+, -\}$) in a GP tree. They have shown the advantage of SRM in enhancing the generalization performance of GP. However, the relationship between the number of nonscalar nodes and the VC-dimension of the model needs further investigation.

Compared with a rough approximation, measuring the VC-dimension of the evolved models through a well-designed *experimental* method is more reliable and forms the major difference between methods in [32] and [33] and the proposed methods in this paper.

III. Proposed Method—GP With Optimized Structural Risk Minimization

In GP for symbolic regression, obtaining a small empirical/training error does not guarantee a good generalization performance in many scenarios, such as when the number of available training instances is small or when learning from training data with noise. In these scenarios, an accurate estimation of the expected generalization error of the evolved models is more reliable. Based on this hypothesis, in [10] we introduced SRM into GP to propose a GP approach named GPSRM. GPSRM employs SRM as the fitness function, and intends to achieve a good tradeoff between the accurate approximation on the training data and the lower complexity of these models. As mentioned above, when adopting SRM in GP, the crucial and most difficult aspect is to obtain the VC-dimension of the evolved models. Different from the existing methods [32], [33] that approximate the VC-dimension by counting the number of specific nodes in the models, we extended an *experimental* method to calculate the VC-dimension of the evolved models. The effectiveness of GPSRM on promoting the generalization of GP has been investigated and confirmed in [10]. However, it still has limitations.

One major limitation of GPSRM is the *uniform setting* in the method that measures the VC-dimension of the evolved models. More specifically, the uniform setting refers to the same number of experiments conducted on all the datasets to get the experimental maximum deviations. This setting does not consider the fact that these datasets are randomly generated and have different numbers of instances. The uniform setting and the variability of the random datasets potentially restrict the accuracy of the measured VC-dimension of the evolved models. It accordingly limits the effect of SRM on improving the generalization of GP. To address this problem, a more precise and reliable setting is needed. In applied statistics, there is an important research topic: experimental design, which aims to construct the optimal design for experiments. In the previous work [34], [35], the experimental design is iteratively improved by exchanging the design points according to the optimality criteria. In [36], this idea is introduced into the process of measuring the VC-dimension of linear models and shown its effectiveness.

Motivated by the idea of constructing an optimal design and the promising results achieved in [36], this paper proposes an improved method to measure the VC-dimension of evolved models in GP to increase the accuracy of the estimated generalization errors. The proposed method is named GPOPSRM. The details of the method are presented as follows. We first describe the fitness function used in SRM-driven GP. Then for self-contained purposes, we overview the implementation of SRM in GP under the uniform setting, which forms the basis of this paper.

A. Fitness Function in SRM-Driven GP

When introducing SRM into GP, the major change is the fitness function for measuring the performance of the evolved models. In SRM-driven GP, the solutions are evaluated by the estimated generalization error given by SRM. Assuming the VC generalization error bound is tight, the fitness function is defined as

$$\mathrm{Err}_{\exp} = \frac{\mathrm{RMSE}}{\left(1 - \sqrt{p - p\ln p + \frac{\ln n}{2n}}\right)_+} \tag{2}$$

where

$$\mathrm{RMSE} = \sqrt{\frac{\sum_{i=1}^{n}(f(X_i) - Y_i)^2}{n}}$$

RMSE is the error of the evolved model on the training data, $(1 - \sqrt{p - p\ln p + (\ln n/2n)})_+^{-1}$ is the confidence interval between the empirical/training error and the estimated generalization error. $p = h/n$, h is the VC-dimension of the model and n is the number of training instances. When learning from a given training set, i.e., n is fixed, the confidence interval of a model is determined solely by h. In other words, for a given n, a higher h leads to a larger p, which according cause a larger confidence interval $(1 - \sqrt{p - p\ln p + (\ln n/2n)})_+^{-1}$. Consequently, given the same/similar values of RMSE, a higher h will lead to a larger generalization bound $R_{\exp}(h)$. Moreover, when GP adopts the metric of SRM, the evolved models, which have slightly smaller empirical errors but are over complex (large h), are less likely to be selected to generate new individuals. Those models and their offspring generally incorporate too much information from the training data, thus are over-adapted to the training set and difficult to generalize well on unseen data. By assigning a higher estimated generalization error to those models and decreasing the probability of selecting them for breeding, our new GP method is expected to eliminate or decrease the trend of overfitting thus open opportunities to generalize well.

B. GPSRM: Measuring the VC-Dimension Using Uniform Setting

The evaluation process in GPSRM is shown in Fig. 1. The fruitfulness of SRM in promoting the generalization of GP highly depends on how precise of the measured VC-dimension of evolved models. The theoretical approximation of the VC-dimension is easy to obtain for linear models. The meaningful complexity index for linear models is $N + 1$, where N is the number of free parameters in the model [27]. However, this complexity index is not appropriate for nonlinear models [27]. Thus, Vapnik *et al.* [9] proposed an *experimental* method to measure the VC-dimension of a learning machine for classification, which is suitable for both linear and nonlinear models. Since the population of GP consists of a mixture of linear and nonlinear models, it is difficult to measure the VC-dimension of GP by a theoretical analysis. Before our preliminary work [10], there was no existing work measuring the VC-dimension of the evolved models in GP experimentally, which could be more accurate and reliable than any simple theoretical estimation. Therefore, we decided to extend the method in [9] to regression models in GP. As mentioned above, the VC-dimension of a real-value/regression function $f(X, \alpha)$ is equal to the corresponding value of its indicator functions

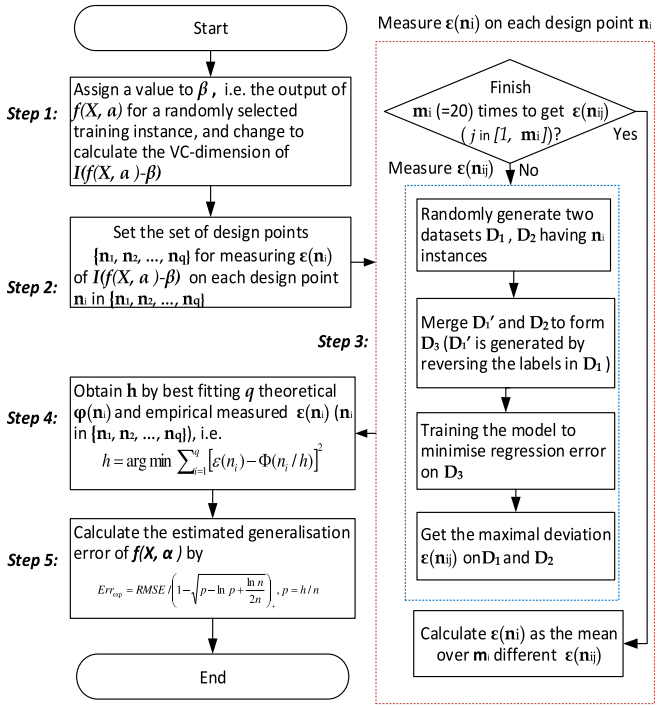

Fig. 1. Evaluation process in GPSRM for each individual $f(X, \alpha)$.

$I(f(X, \alpha) - \beta)$ [4], where $A \leq \{f(X, \alpha)\} \leq B$ and $\beta \in (A, B)$. The value of β can be obtained by calculating the corresponding output of the regression model for a randomly selected training example. By assigning a possible value of $f(X, \alpha)$ to β, the problem of measuring the VC-dimension of a regression model $f(X, \alpha)$ is easily changed to the VC-dimension of the indicator function $I(f(X, \alpha) - \beta)$. Then from steps 2–4 in Fig. 1, the VC-dimension of the model is measured. The method to measure VC-dimension is described as follows.

1) Main Idea: Vapnik and Chervonenkis [37] derived a criterion to decide whether a learning process is consistent. Being consistent means the maximum deviation between the empirical error and the expected error does not exceed a small value ε (the probability of it approaches zero). Specifically, for the indicator function $I(X, \alpha)$ with a VC-dimension h, to decide whether the learning process is consistent, Vapnik and Chervonenkis [37] defines the bound of this probability as follows:

$$P\{\sup[R_{\text{exp}} - R_{\text{emp}}] > \varepsilon\} < \min\left\{1, \exp\left[\left(C_1 \frac{\ln 2n/h + 1}{n + h} - C_2\varepsilon^2\right)n\right]\right\} \quad (3)$$

where C_1 and C_2 are two constants, and $C_1 <= 1$ and $C_2 > 0.25$. This bound is independent of the conditional distribution $P(Y|X)$. The bound also leads to an inequality as follows, i.e., for any given constant δ, there exist a number n_l so that when the number of training instances $n > n_l$, the following inequality holds:

$$P\{\sup[R_{\text{exp}} - R_{\text{emp}}] > \varepsilon\} < \exp\left[\left(-(C_2 - \delta)\varepsilon^2\right)n\right]. \quad (4)$$

Later researchers improved the value of the constant to $C_2 = 2$. For a large n, the bound is found to be close to the value gave by the Kolmogorov–Smirnov law [38], which defines the distribution law of the maximum derivation between the training error and the test error of a simple linear function. The law is formulated as: when learning the target function: $f(x, \alpha) = I(x - \alpha)$, for sufficiently large number of instances, the equality

$$P\{\sup[R_{\text{exp}} - R_{\text{emp}}] > \varepsilon\} = \exp\left(-2\varepsilon^2\right)n - 2\sum_{k=2}^{\infty}(-1)^k \exp(-2\varepsilon k n) \quad (5)$$

holds. Note that compared with the value of the first term, the value of the second term is very small. The close of above bound and the value of the Kolmogorov–Smirnov law indicates that the bound is tight and close enough to the exact value. Based on this observation, [9] also assumed that there exit a value for C_1 to make the bound to be tight for both small and large numbers of instances. In this scenario, the function $\Phi(n/h)$, which defines the expected maximum deviation between the error rates, is independent of the conditional distribution $P(Y|X)$. The definition of $\Phi(n/h)$ is

$$\Phi\left(\frac{n}{h}\right) = E\{\sup[R_{\text{exp}} - R_{\text{emp}}]\}. \quad (6)$$

Based on the assumption that it is able to derive $\Phi(n/h)$, the idea of the *experimental* method to measure the VC-dimension was proposed in [9]. Suppose that $\Phi(n/h)$, which is determined by the VC-dimension h and the number of training instances n, can then be derived successfully. The *experimental* estimation of the maximum deviation between the expected test error and the empirical error is also available. The VC-dimension h of a model can be measured by finding the value that can achieve a good fitting between the theoretical values given by $\Phi(n/h)$ and the maximum derivations obtained *experimentally*. Fig. 2 visualizes this process. As it shows, for a list of $\epsilon(n_i)$ (i.e., the orange dots, the values of which are obtained from the *experimental* method), try to find the parameter h of the curve $\Phi(n/h)$ to make it best fit the given $\epsilon(n)$ values. These $\epsilon(n)$ values are obtained based on a set of design points, where each design point determines the size of randomly generated datasets [9]. Note that in practice, it is impossible to obtain the error on an infinite number of test instances. Instead, measuring the maximum deviation of the errors on two independently generated paired datasets is a reasonable choice. Here, these paired datasets refer to two datasets having the same number of instances. The inputs in the two datasets follow the same distribution and the outputs are generated randomly. The derivation of $\Phi(n/h)$ and the process of obtaining the *experimental* values of the maximum difference between the errors will be presented in detail as follows. (Because of the page limit, we briefly introduce the major derivation process of $\Phi(n/h)$ here. Readers who are interested in the detailed procedure are referred to [9] or our online supplementary material for more information.)

2) Theoretical Formula of the Maximum Deviation: To estimate a bound on the expectation of the maximum deviation between errors, we need to formulate this maximum deviation at first. For a set of indicator functions $I(X, \alpha)$ with a VC-dimension h, given a set of samples $Z^{2n} =$

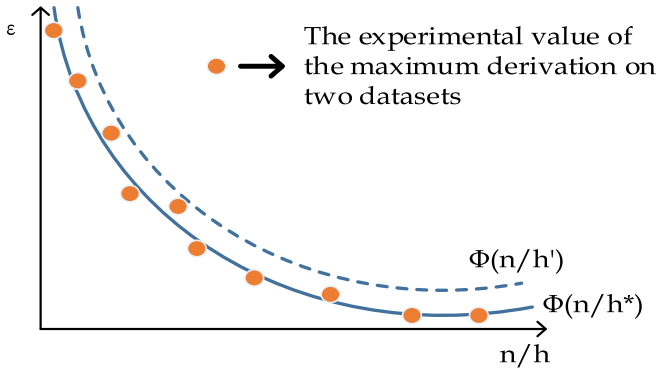

Fig. 2. h is measured by searching for a better fitting (from h' to h^*) between $\Phi(n/h)$ and the *experimental* maximum deviation.

$X_1, Y_1, X_2, Y_2, \ldots, X_{2n}, Y_{2n}$ where X_i is the input vector and $Y_i \in (0, 1)$ is the label, let $Pe_1(Z^{2n})$ denote the error rate on the first n samples, and $Pe_2(Z^{2n})$ denote the error rate on the other n samples. The error rate is obtained by

$$Pe_i(Z_{2n}, \alpha) = \frac{1}{n}\left(\sum_{j=1}^{n}\left|Y_j - I(X_j, \alpha)\right|\right). \tag{7}$$

Then the maximum derivation $\epsilon(n)$ between the errors obtained by $I(X, \alpha)$ is defined as

$$\epsilon(n) = \sup\left[Pe_1\left(Z^{2n}, \alpha\right) - Pe_2\left(Z^{2n}, \alpha\right)\right] \tag{8}$$

where sup is the supremum (least upper bound) of the set of derivations.

The expectation of $\epsilon(n)$ is bound as follows:

$$E\{\epsilon(n)\} \leq \begin{cases} 1 & \text{if } \frac{n}{h} <= 0.5 \\ C_1 \frac{\ln(2n/h)+1}{n/h} & \text{if } 0.5 < \frac{n}{h} <= 8 \\ C_2\sqrt{\frac{\ln(2n/h)+1}{n/h}} & \text{if } \frac{n}{h} > 8 \end{cases} \tag{9}$$

where C_1 and C_2 are two constants. 0.5 and 8 are values to distinguish datasets with large and small n/h values [9].[4] There exist constants C_1 and C_2 that make the bound tightly holds. Using a continuous approximation, the right side of the bound can be defined as

$$\Phi\left(\frac{n}{h}\right) = \begin{cases} 1 & \text{if } \frac{n}{h} <= 0.5 \\ a\frac{\ln(2\frac{n}{h})+1}{\frac{n}{h}-k}\left(\sqrt{1+\frac{b(\frac{n}{h}-k)}{\ln(2\frac{n}{h})+1}}+1\right) & \text{otherwise} \end{cases} \tag{10}$$

where the parameters a and b determine $\Phi(n/h)$ that can cover the region of large ($[n/h] > 8$) and small ($0.5 < (n/h) <= 8$) values of n/h. The values of a and b are obtained by fitting (10) to the experimental maximum deviations of linear models on datasets with various n/h, since the VC-dimension h of these linear models are known. Accordingly, it found that $a = 0.16$ and $b = 1.2$ in [9]. Then according to $\Phi(0.5) = 1$, it is easy to get $k = 0.14928$.

3) Experimental Measure of the Maximum Deviation: Step 3 in Fig. 1 shows the procedure of obtaining $\epsilon(n_i)$ on all the design points $\{n_1, n_2, \ldots, n_q\}$. On each design point, $\epsilon(n_i)$ is the average value over a set of $\epsilon(n_{i,j})$ ($j \in \{1, 2, \ldots, m_i\}$). These values are obtained from $m_i(= 20)$ times of independent

[4]For more details of the bound and the parameters, readers are referred to the online supplementary material of this paper or [9].

Algorithm 1: Measuring a Set of Maximum Deviations Obtained by $f(X,\alpha)$

Input: a regression model $f(X,\alpha)$, p represents the number of distinct nodes in a GP tree, i.e., the regression function $f(X,\alpha)$, and u is the number of distinct features in $f(X,\alpha)$
Output: a set of maximum deviations $\epsilon(n_i)$
Randomly select one training example X_1 and set β=$f(X_1,\alpha)$.
Set $h' = p$
Calculate the set of $\{n_1, n_2, ..., n_q\}$ according to a set of design points n_i/h'={0.5,0.8,1.0,1.2,2,2.5,3,3.5,5,6.5,8,10,15,20,30}, $i \in [1, q]$ (as recommend in [4], it needs a bunch of different n_i to make sure the range of n_i/h' to cover a wide enough range $0.5 < n_i/h' < 32$)
for $i := 1$ *to* q **do** *Obtaining the maximum deviations on one design point loop*
  for $j := 1$ *to* m_i **do** *Measuring one maximum deviation loop*
    Randomly generate two classification datasets D_1 and D_2. Each dataset has n_i instances and a feature set X containing u features/variables, and the label Y of each instance is generated randomly.
    Reverse the labels in D_1 to form a new dataset D_1' and merge D_1' and D_2 to form another new dataset D_3.
    Training the model $\{f(X,\alpha)-\beta\}$ to minimize its MSE on D_3 using mini-batch gradient descent, then calculate its error rates on D_1 and D_2 according to Equations (7) and (8).
    Calculate $\epsilon(n_{ij})$ that is the deviation of the error rates obtained by $\{f(X,\alpha)-\beta\}$ on D_1 and D_2.
  end
  $\epsilon(n_i) = \sum_{j=1}^{m_i}\epsilon(n_{ij})/m_i$
end
Return $\epsilon(n_i)$

experimental measure. The average of these values is used, since it is considered to be able to reduce the influence of randomness. The blue dashed box in step 3 in Fig. 1 shows how the maximum deviation of errors is obtained for one time, i.e., on two randomly generated datasets.

To get the maximum deviation $\epsilon(n_{i,j})$ on the two datasets, it needs maximizing the error rate on the first dataset, while minimizing the error rate on the second dataset at the same time. It is important to note that the error rate is a measure for the performance of an indicator/classification function, which is not suitable for regression models — the focus of this paper. Therefore, we change the task from calculating the VC-dimension of $f(X, \alpha)$ to obtain the corresponding value of the model $I(f(X, \alpha) - \beta)$, which is a binary classification model. The pseudo-code of this procedure is shown in Algorithm 1.

As shown in Algorithm 1, for a given regression model $f(X, \alpha)$ with u distinct input variables, the detailed procedure of getting the maximum deviation $\epsilon(n_i)$ of $I(f(X, \alpha) - \beta)$ on two independent datasets is described as follows.

1) Generate two random datasets D_1 and D_2, each of which has n instances. The length/dimensionality of the input vectors is equal to u. The inputs are drawn randomly within a uniform distribution over the interval $[-1, 1]$. The labels of the instances are created according to the conditional probability distributions $P(Y|X) = 0.5$ for $Y = 0$ and $P(Y|X) = 0.5$ for $Y = 1$.
2) Merge D_1' and D_2 to form a new dataset D_3, which has $2n$ instances. Here, D_1' refers to a new dataset where the instances are generated by reversing the labels in D_1 (D_1 and D_1' have the same input vectors but the opposite output values, i.e., for an instance with a label $Y = 0$ in D_1, the corresponding instance in D_1' has the label of $Y = 1$).

3) Training the model $\{f(X, \alpha) - \beta\}$ to minimize its MSE on dataset D_3 ($D_3 = D_1' \cup D_2$), thus $I(f(X, \alpha) - \beta)$ can get a maximum deviation of errors on D_1 and D_2.
4) Calculate the $\epsilon(n_{ij})$ over the two datasets D_1 and D_2 according to (7) and (8).

In step 3 of the above procedure, *mini-batch gradient descent* [39] is used to train the coefficients in the model to obtain the minimal MSE on D_3. Minimizing the error of the model on D_3 is equivalent to getting the maximum derivation $\epsilon(n)$ on D_1 and D_2 (since $D_3 = D_1' \cup D_2$).[5] The deviation is largely independent of the distribution $P(Y|X)$, thus we use $P(Y|X) = 0.5$ to generate random labels. However, $P(Y|X)$ can be any other values. Repeat steps 1–4 for m_i times independently ($m_i = 20$ is suggested in [9], which is considered to be large enough to make the average value represents the central tendency of $\epsilon(n_i)$). The mean value of the $\epsilon(n_{ij}), j \in [0, m_i]$ is treated as the maximum derivation on the design point n_i.

Then the whole procedure is repeated on q design points. Different design points refer to different numbers of instances, i.e., n_i from $\{n_1, n_2, \ldots, n_q\}$. The selection of $\{n_1, n_2, \ldots, n_q\}$ should cover the range of $0.5 \leq (n_i/h') \leq 32$ (it is the recommended setting in [9]), where 0.5 is the starting point of the definition of $\Phi(n/h)$ shown in (10) and 32 is set to make sure that the range of (n_i/h') is big enough for various n_i. h' is an initial guess of the VC-dimension of the model. In this paper, it is set to be the number of distinct nodes in the GP model. A larger q means more design points, which would lead to a more accurate fitting between $\epsilon(n)$ and $\Phi(n/h)$ and a tighter VC bound. However, it also lead to more computational cost. Thus, to achieve a good balance, the trail experiments in our preliminary work show that 15 is a reasonable value for q, (i.e., for a given h', setting 15 n_i to make $n_i/h' = \{0.5, 0.8, 1.0, 1.2, 2, 2.5, 3, 3.5, 5, 6.5, 8, 10, 15, 20, 30\}$). This repeated procedure is under a uniform setting, i.e., the number of experiments repeated on each of the q design points is the same, i.e., $m_1 = m_2 = \cdots = m_q = 20$, which is recommended in [9].

After getting all the maximum deviation [i.e., $\epsilon(n_i)$] values, the VC-dimension of the model can then be approximated by choosing the h that can create a good fit between the set of $\epsilon(n_i)$ and the function $\Phi(n/h)$ according to $h = \arg\min \sum_{i=1}^{q} [\epsilon(n_i) - \Phi(n_i/h)]^2$, i.e., choosing an approximate h to minimize the error (such as the MSE used in this paper) between the set of $\epsilon(n)$ and $\Phi(n/h)$.

C. GPOPSRM: Measuring the VC-Dimension Using Optimised Setting

The uniform setting of the experiments for measuring the maximum deviation and the variability of random instances in each pair of independent datasets can potentially lead to a not reliable enough VC-dimension h, and correspondingly generate a loose generalization bound in SRM-driven GP. Therefore, we aim to make further improvement on measuring the maximum deviation by employing a better setting.

[5]For a more detail approve of this, readers are referred to the online supplementally material or [9].

To improve the setting in the procedure of measuring the VC-dimension, the idea of optimal experiment design from applied statistics [34], [35] is employed. The process of searching for a better setting starts from a well-designed setting, then repeats a process of constructing neighbor settings, which have better performance than current setting, until an "optimal" setting is achieved (i.e., no any better neighboring setting is available) or the stop criterion is satisfied.

To develop a better setting to measure the VC-dimension, the original uniform setting is a good starting point. The target of this optimization process is to minimize the error between a set of $\epsilon(n)$ and $\Phi(n, h)$, thus the evaluation criterion for a setting is set as

$$\text{MSE} = \sum_{i=1}^{q} \sum_{j=1}^{m_i} \left(\epsilon\left(n_{i,j}\right) - \Phi\left(n_i, h^*\right)\right)^2 / (m_i * q) \qquad (11)$$

where $\epsilon(n_{i,j})$ is the jth maximum deviation on the design point n_i. m_i is the number of maximum deviation values obtained from the repeated experiments on n_i, $i \in [1, q]$. q is the number of design points ($q = 15$ in this paper). The number of instances on each design point is different, typically $n_1 < n_2 < \cdots < n_q$. A better setting leads to a lower MSE between $\epsilon(n)$ and $\Phi(n, h)$.

Improving the setting is to adjust m_i for each design point appropriately. To find a better setting, the neighbors of current setting are obtained and evaluated. A neighboring setting can be reached by decreasing the number of experiments by one on the worst design point while adding one more time on the best design point. The goodness of a design point n_i relies on the contribution of $(\epsilon(n_i) - \Phi(n_i, h))$ to the overall MSE in (11), the smaller the better. The procedure of measuring VC-dimension using the optimized setting is presented as follows.

1) Under the uniform setting, measure $m_i * q$ different $\epsilon(n_i)$ and $\Phi(n_i)$ values, where $n_i \in \{n_1, n_2, \ldots, n_q\}$, $m_i = 20$ and $q = 15$ (initialization).
2) Calculate the VC-dimension h^* by finding the best fit among all the various $\epsilon(n_{ij})$ and $\phi(n_i/h)$. Here, each $\epsilon(n_{ij})$ participates in the fitting instead of using the average $\epsilon(n_i)$ over m_i times in the uniform setting, since the values of m on $\{n_1, n_2, \ldots, n_q\}$ are potentially different now.
3) Calculate the MSE according to (11).
4) Rank the design points $n_1, n_2, \ldots, n_q$ according to its contribution $CB(n_i)$ to MSE, $CB(n_i) = (\text{MSE}(\text{remove} n_i) - \text{MSE})/n_i$ (the contribution is normalized by the number of instances).
5) Construct a neighboring setting by adjusting the number of experiments on the design points, which is to add one experiment on the best point, while removing one experiment from the worst point.
6) Calculate the new MSE* between the set of $\epsilon(n)$ and $\Phi(n, h)$ on the new setting. If MSE* is higher than MSE, then reverse to the former setting and add the removed experiment to the 2nd (or 3rd, 4th,…, until find the one yields to lower MSE*).
7) Repeat steps 2–6 until no design point has positive contribution on MSE or the number of experiments on the

positive points reaches a predefined threshold, which is to prevent the situation that all the experiments are allocated on a single design point, not a set of design points.

The proposed GPOPSRM algorithm employs the nonuniform setting for measuring the VC-dimension of evolved models in GP. This forms the major difference between GPOPSRM and our preliminary method GPSRM. In addition, the advance of the nonuniform setting over the uniform setting is expected to bring benefit to SRM-driven GP, since the advance will lead to a tighter VC generalization bound, which is crucial to the success of SRM-driven GP.

Note that it is only necessary to measure the VC-dimension for a number of top individuals ranked according to their empirical errors (i.e., RMSE in this paper) in both GPSRM and GPOPSRM. This is because the difference between the confidence interval of the top individuals and their worse counterparts ranges within the interval $[0, 1]$, so that it can be ignored when the empirical risk difference between two sets of individuals is large. On the other hand, these worst individuals have a very low probability to win the tournament selection to be parents of the new individuals in GP. Moreover, measuring the VC-dimension of evolved models is expensive. Therefore, we define a parameter γ in GPSRM and GPOPSRM, so that only the top γ percent of individuals in the candidate population will be measured. For the rest $1-\gamma$ percent of the population, their VC-dimension is assigned to be a random big value (i.e., 50 in this paper). The setting is to make the evolutionary process focus on the comparison of estimated generalization error between the top γ percent of individuals and make the method more efficient.

IV. Experiment Design

To investigate the generalization ability of GPOPSRM, a set of experiments have been conducted. The experiment design, in particular, the selection of benchmark problems, the benchmark methods for comparison, and the parameter settings for GP runs, is presented in detail.

A. Benchmark Problems

Due to the lack of benchmarks (datasets) specially designed for testing the generalization ability of GP for symbolic regression, in this paper, we examine the methods on eight synthetic symbolic regression problems and two real-world high-dimensional regression datasets, which are taken from previous research on GP for symbolic regression [6], [40], [41]. These benchmark problems have been shown to be prone to overfitting, therefore generalization estimation during the training process is desired.

The details of the target functions and the sampling strategies for the training data and the test data of the eight synthetic regression datasets are shown in Table I. The first four functions are taken from [6]. Despite the low dimensionality, they are claimed to be difficult regression tasks. The rest four problems are from [40]. For all these eight problems, a small number of training points is obtained to simulate the real-world situation, where GP is prone to overfitting. The number of training data points is 50 for the first four problems and 20 for the other four problems.

TABLE I
Sampling Strategies for the Training Data and the Test Data

The notation *rnd[a,b]* denotes the variable is randomly sampled from the interval *[a,b]*, while the notation *mesh([start:step:stop])* defines the set is sampled using regular intervals.

Target function	Training	Test
$f_1=e^{-x}x^3cosxsinx(cosxsin^2x-1)$	50 points x=rnd[0.05,10]	221 points x=mesh([-0.5:0.05:10.5])
$f_2=30\frac{(x_1-1)(x_3-1)}{x_2^2(x_1-10)}$	50 points x_1, x_3 =rnd[0.05,2] x_2=rnd[1,2]	2701 points x_1, x_3= mesh([-0.05:0.15:2.1]) x_2=mesh([0.95:0.1:2.05])
$f_3=6sinx_1cosx_2$	50 points x_1, x_2 =rnd[0.1,5.9]	961 points x_1, x_2= mesh([0.05:0.02:6.05])
$f_4=\frac{(x_1-3)^4+(x_2-3)^3-(x_2-3)}{(x_2-2)^4+10}$	50 points x_1, x_2 =rnd[0.05,6.05]	1157 points x_1, x_2= mesh([-0.25:0.2:6.35])
$f_5=\frac{x_1x_2+sin((x_1-1)(x_2-1))}{}$	20 points x_1, x_2 =rnd[-3,3]	361,201 points x_1, x_2 =mesh([-3:0.01:3])
$f_6=x_1^4-x_1^3+x_2^2/2-x_2$		
$f_7=8/(2+x_1^2+x_2^2)$		
$f_8=x_1^3/5+x_2^3/2-x_2-x_1$		

TABLE II
Real-World Problems

Name	# Features	#Total Instances	#Training Instances	#Test Instances
LD50	626	234	163	71
DLBCL	7399	240	160	80

We also test the methods on two high-dimensional real-world regression datasets as shown in Table II. The first dataset is from the field of pharmacokinetics [42]. The task is to predict the value of a kind of pharmacokinetics parameter, i.e., the median lethal dose (represented as LD50). It has been used in many recent papers [7], [41], [43] to investigate the generalization of GP. LD50 is split randomly with 70% of instances for training and the other 30% for test. The second dataset is the Diffuse Large-B-Cell Lymphoma (DLBCL) dataset [44]. The task is to predict the survival time of patients who have DLBCL and received chemotherapy. In DLBCL, the training set and the test set are provided.

B. Benchmark Algorithms for Comparison

To further investigate and confirm the effect of SRM on estimating the generalization performance, comparisons between GPOPSRM and the following three GP methods have been approached.

1) Standard GP, which is a baseline for comparison.
2) GP with 0.632 Bootstrap (BGP) refers a GP method which also estimates the generalization error using the 0.632 bootstrap [45]. In [10], the bias/variance error decomposition (BVGP) [24] was compared. In BVGP, the generalization error of a GP model is assessed by two aspects, the bias error and the variance error. Bias error refers to the error over the training set, while the variance error is considered as the sensitivity of a model to the training data. However, the experiment results show that BVGP generally has worse generalization performance than GPSRM [10]. This might due to

TABLE III
PARAMETERS FOR THE FOUR GP METHODS

parameter	Values
Population Size	512
Generations	51
Crossover Rate	0.9
Mutation Rate	0.1
Elitism(number of individual)	1
Maximum Tree Depth	11
Initialisation	Ramped-Half&Half
Initialisation	
— Minimum Depth	2
— Maximum Depth	6
Selection Operator	Tournament Selection with a size of 7
Basic Function Set	$+, -, *$, %protected, $Square, Sqrt, Negative$
— f_1	$e^x, e^{-x}, sinx, cosx$
— f_3	e^x, e^{-x}
Percentage of Top Individuals — γ	20%

the potential overlap of instances in bootstrap datasets and the original training set, which leads to an inaccurate estimation of variance error [2]. Therefore, this paper compares with an improved version of BVGP employing the 0.632 bootstrap. Under 0.632 bootstrap, the definition of the estimation of generalization error as follows:

$$R_{\text{est}} = 0.368 * R_{\text{emp}} + 0.632 * V_{\text{err}}$$
$$V_{\text{err}} = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{|C^{-i}|}\sum_{b\in C^{-i}}\left(E\left(D^{*b}\right)\right) \tag{12}$$

where R_{est} is the estimated generalization error, R_{emp} is the empirical error, and V_{err} is the variance error. n is the number of training instances. C^{-i} refers to the set of bootstrap samples which does not contain the training instance i, and $|C^{-i}|$ is the number of such bootstrap samples. $E(D^{*b})$ is the error on each bootstrap sample in C^{-i}. For computing V_{err}, we should choose the total number of bootstrap samples B to be large enough to ensure $|C^{-i}|$ is larger than zero (i.e., to make sure some bootstrap set contain samples that are not used for training). In this paper, we set $B = 200$, which is a recommended setting in [2].

3) GPSRM. It is proposed in our recent work [10], which is the preliminary investigation on the generalization of SRM-driven GP for symbolic regression.

These four GP methods use different indicators for the generalization performance. Standard GP relies on the empirical risk/error, while BGP uses the variance error estimated by the 0.632 Bootstrap. The GPSRM and GPOPSRM use the confidence interval. The comparison focuses mainly on the effect of these indicators on the generalization of GP. All the examined GP methods are implemented under the ECJ GP framework [46].

C. Parameter Settings

The parameter settings for the four GP methods (GP, BGP, GPSRM, and GPOPSRM) can be found in Table III, which are common settings in GP [3]. Note that the tournament selection for GPSRM and GPOPSRM is also a standard one. To select a parent, the tournament selection operator *randomly* samples 7 candidate individuals from the population and the one with the smallest estimated generalization error is selected as a parent for genetic operators. Following the settings in [6] and [40], the function set is different for different benchmark problems. For the same benchmark problem, all the four methods have the same function set. According to our preliminary work [10], the parameter γ is set to 20%, which is sufficiently large for not missing individuals that have potentially good generalization ability while can reduce the computational cost.

In each method, 100 independent runs have been conducted on each problem. Therefore, 4000 (i.e., 4*10*100) experiments have been run for the four methods on ten datasets, and 8000 (i.e., 4000*2) training and test results are used here to discuss the training and generalization performance of the four methods.

TABLE IV
RESULTS OF STATISTICAL SIGNIFICANCE TESTS

Datasets	GPOPSRM(training, test)			GP(training, test)	
	GP	BGP	GPSRM	BGP	GPSRM
F1	$(=, -)$	$(-, -)$	$(-, -)$	$(-, -)$	$(-, +)$
F2	$(+, -)$	$(+, =)$	$(-, -)$	$(+, +)$	$(-, +)$
F3	$(=, -)$	$(=, -)$	$(-, =)$	$(=, -)$	$(-, +)$
F4	$(+, -)$	$(+, -)$	$(=, =)$	$(-, +)$	$(-, +)$
F5	$(=, -)$	$(+, -)$	$(=, =)$	$(=, -)$	$(-, +)$
F6	$(+, -)$	$(+, =)$	$(-, =)$	$(+, +)$	$(=, +)$
F7	$(+, -)$	$(+, =)$	$(=, =)$	$(+, +)$	$(-, +)$
F8	$(+, -)$	$(=, -)$	$(=, -)$	$(-, +)$	$(-, +)$
LD50	$(+, -)$	$(+, =)$	$(=, -)$	$(-, +)$	$(-, +)$
DLBCL	$(+, -)$	$(=, -)$	$(=, =)$	$(-, =)$	$(-, +)$

V. RESULTS AND DISCUSSIONS

The section presents and discusses the results on the ten datasets. The distributions of RMSEs of the 100 best-of-run individuals on both the training sets and the test sets are presented. To examine the generalization performance in more detail, the evolutionary plots drawing the median test RMSE of the 100 best individuals on every generation are provided. Further analyzes on model size and model behavior are also presented.

The Wilcoxon test, which is a nonparametric statistical significance test, is conducted to compare the 100 best training RMSEs and the corresponding test RMSEs. The Wilcoxon test is performed on the comparisons between GPOPSRM and the other three methods (GP, BGP, and GPSRM) in pairs, and also between GP and BGP and GPSRM (i.e., GP versus BGP and GP versus GPSRM). The significance level is 0.05.

A. Overall Results

The distributions of the RMSEs of the 100 best-of-run models on the training sets and the test sets are shown in the box plots in Figs. 3 and 4, respectively. The overall pattern is that the SRM-driven GP methods generally has a worse learning performance (showed in Fig. 3) but much better generalization performance (demonstrated in Fig. 4) than standard GP and BGP on the examined datasets. Table IV presents the results of the statistical significance tests. While "−" stands for GPOPSRM (GP) performs significantly better than the compared method, "+" indicates GPOPSRM (GP) is significantly worse, and "=" means no significant difference.

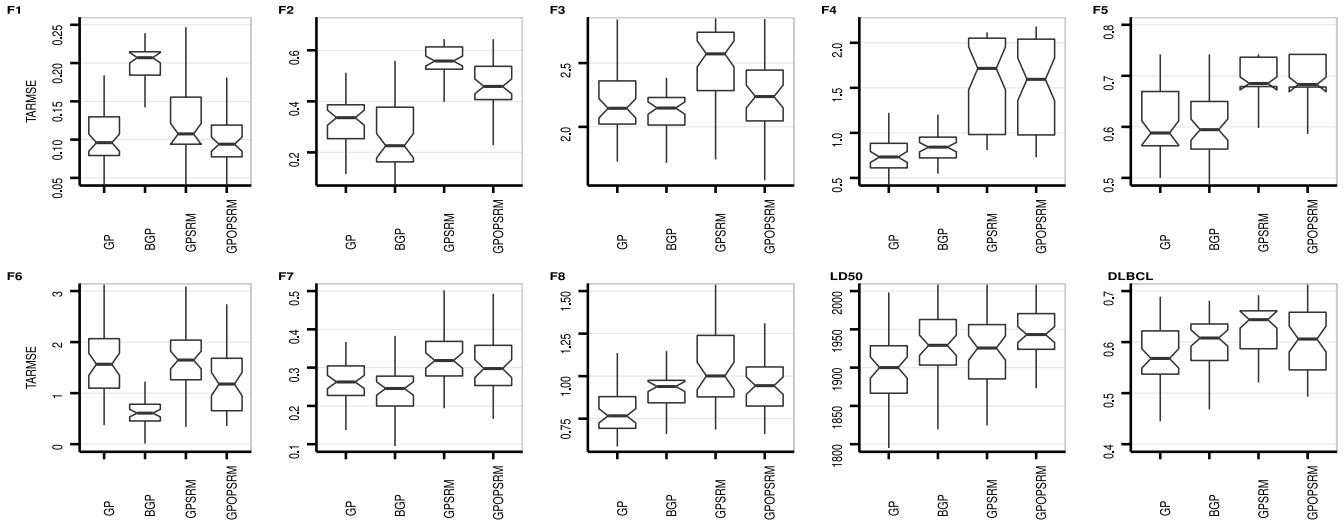

Fig. 3. Distribution of RMSE of the 100 best-of-runs individuals on the *training sets*.

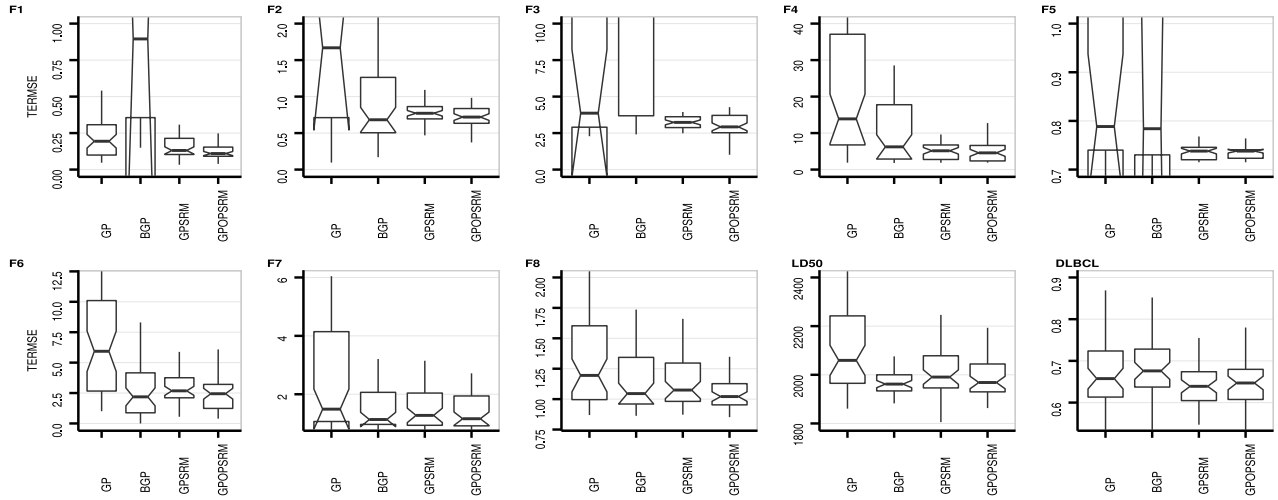

Fig. 4. Distribution of the corresponding test RMSE of the 100 best-of-runs individuals.

1) Learning Performance on the Training Sets: As shown in Fig. 3, on most of the ten training sets, the two SRM-driven GP methods both have a worse training performance than standard GP. On seven of the ten training sets, GPOPSRM has much higher training RMSEs than GP (except on f_1, f_3, and f_6). The training advantage of standard GP over GPOPSRM on these seven datasets is significant. On the other three datasets, f_1, f_3, and f_6, while GPOPSRM has a better training performance (in median) than GP, the difference between the training RMSEs in the two methods is not significant. When compared with BGP, GPOPSRM has significantly higher RMSEs on six training sets, which are f_2, f_4, f_5, f_6, f_7, and $LD50$. On f_1, it has a smaller training error than BGP, which is significant. The training RMSEs of BGP and GPOPSRM on the other three training sets have a similar distribution, and no significant difference can be found. Compared with GPSRM, GPOPSRM has smaller training errors on most of the datasets. On four training sets (f_1, f_2, f_3, and f_6), GPOPSRM has significantly smaller training errors than GPSRM. On the other six datasets, GPOPSRM has a smaller training RMSE than GPSRM, but the gaps are not significant.

It is not very surprising that standard GP outperforms the two SRM-driven GP methods on most of the training sets. This is due to the underlying objective in the two SRM-driven GP methods, which is to restrict the model complexity. This restricted objective has a tendency to conflict with the lower training errors, particularly when over-complex models with smaller training errors and smoother models with larger training errors are competing in the GP population. This is also the reason that GPOPSRM has a worse learning performance than BGP. The variance error in BGP is not related to the model complexity directly. Therefore, the conflict between the variance error and the empirical/training error is not as severe as it counterpart in SRM. This is confirmed by the fact that on three of these training sets (f_2, f_6, and f_7), BGP can have better training performance than GP.

2) Generalization Performance: Compared with the training performance, we are more interested in the generalization performance, which is a more important criterion for the success of the learned model. The overall pattern is very clear in Fig. 4. Both GPSRM and GPOPSRM have significantly better generalization performance, i.e., a much smaller RMSE,

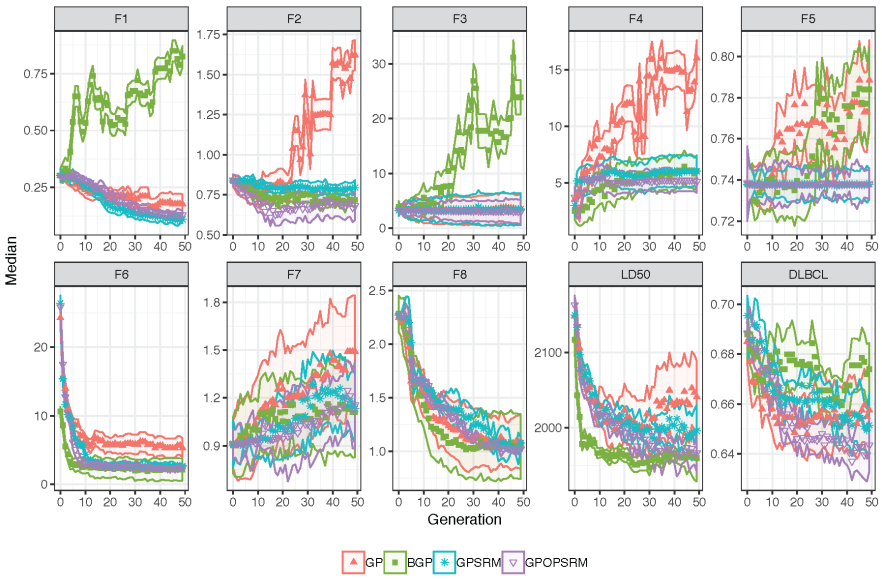

Fig. 5. Evolution plots of the *test* median RMSE and the 95% confidence interval.

than GP and BGP on almost all the ten test sets. This is very different from the pattern in the training sets. As shown in Fig. 4, on most of the test sets, GPOPSRM has much lower median values than GP, which indicates that GPOPSRM has much better generalization performance than GP. In addition, the smaller whiskers in the boxes of GPOPSRM represent a much smaller standard deviation than that of GP, which indicates that GPOPSRM outperforms GP on all the test sets in a stable way. The Wilcoxon test results confirm that, the new method can enhance the generalization of GP significantly on all the ten datasets.

Compared with BGP, GPOPSRM has much smaller test errors on five of the eight synthetic datasets (f_1, f_3, f_4, f_5, and f_8). On the other three synthetic datasets, no significant generalization difference between the two methods can be found. GPOPSRM outperforms BGP on the majority of the test sets, which indicates the advantage of SRM over bootstrap on estimating the generalization ability of GP solutions, particularly when the number of training instances is small. In this case, the bootstrap sets and the training set are more likely to have instances in common, thus bootstrap is difficult to provide a good estimation of the generalization performance. Compared with BGP on the two real-world datasets with a larger number of training instances, GPOPSRM has significantly better generalization gain on DLBCL, and slightly larger test RMSEs on LD50, but not significant. These two datasets have a similar number of training instances (which is 163 in LD50 and 160 in DLBCL), but the number of features in DLBCL is much larger than LD50 (i.e., 7399 versus 626). The available information in DLBCL is much less than LD50. This makes BGP, which relies on extracting information from the training set during the evolutionary process, lose the advantage on DLBCL, while it can perform well on LD50.

Compared with GP and BGP, SRM-driven GP methods generally achieve a better generalization performance on most of the test sets, particularly on the first five synthetic datasets. The target functions of these five datasets contain trigonometric or exponential functions and have a smaller number of training instances. So the first five datasets are more difficult than the other three synthetic datasets. On four of the last five test sets (except for LD50), the two SRM-driven GP methods still outperform GP and BGP in a smooth and significant way.

In terms of the comparison between the two SRM-driven GP methods, GPOPSRM has a better generalization performance than GPSRM on all datasets. GPOPSRM has significantly smaller test RMSEs than GPSRM on f_1, f_2, f_8, and LD50. On the other six test sets, GPOPSRM still outperforms GPSRM, although not at a significantly level. The advantage of GPOPSRM over GPSRM is due to the nonuniform setting for measuring VC-dimension of evolved models, which is the major difference between the two methods. The detailed comparison between the two methods will be presented in the following section.

B. Evolution of Generalization Performance

Since the capability of generalization is the focus of the paper, we will examine the generalization performance in more detail. The evolutionary plots on the test sets in Fig. 5 are drawn using the median corresponding test RMSE and the 95% confidence interval over the 100 best-of-generation models. On every generation, the generalization performance of the best-of-generation model is recorded but *never* takes any part in the evolutionary process.

It can be observed that overfitting occurs in GP in most cases. GP has an increasing generalization error after decreasing over the first few generations on most of the test sets, except for f_6 and f_1. On f_2, f_3, f_4, f_5, and f_7, it suffers from a serious overfitting, while on the other three datasets, it slightly overfits over several final generations.

On most of the datasets (i.e., on f_2, f_3, f_4, f_5, and f_7), where GP overfits severely and quickly, BGP can not eliminate/reduce overfitting effectively either. This is due to the

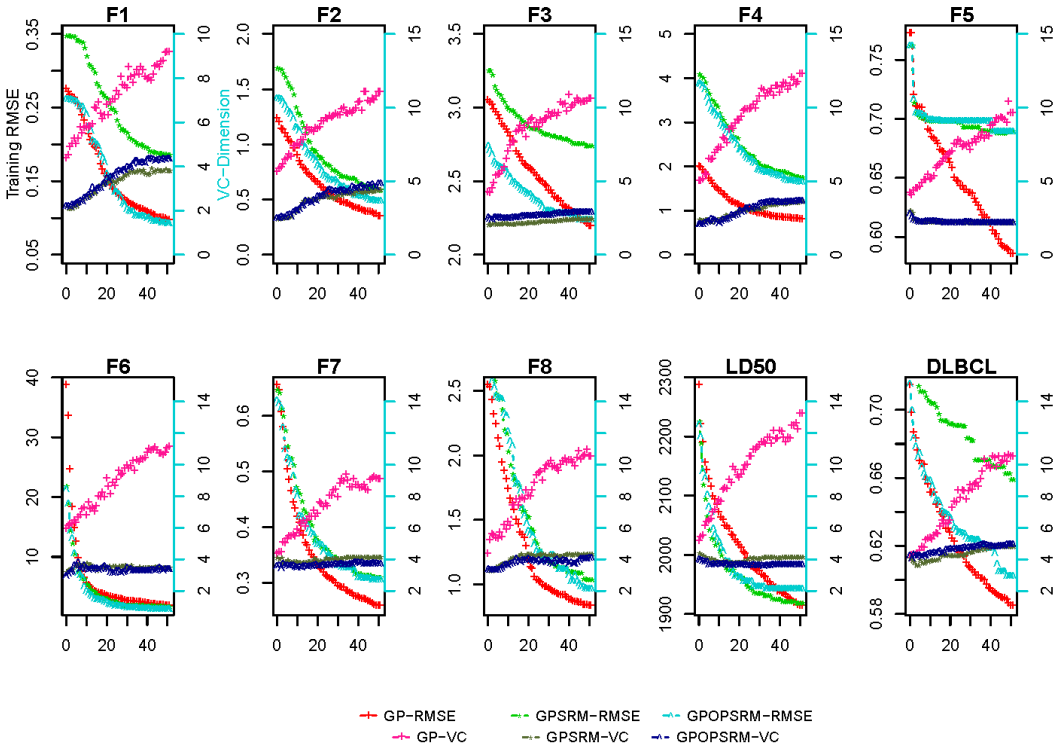

Fig. 6. Evolutionary plots on training RMSEs and VC-dimensions.

small number of training instances and/or the smaller ratio of instances over the number of features in the training sets. BGP, which relies on the bootstrap of the training instances to estimate the variance error, fails to generalize beyond the training sets in this case. In some test sets (i.e., on f_1, f_3, and f_5), it performs even worse than GP.

Different from GP and BGP, the two SRM-driven GP methods generalize well on most of the test sets. On f_2, f_3, f_4, and f_5, where GP overfits severely, the two methods can eliminate overfitting and do not have the overfitting trend. On f_7, the two methods can reduce generalization errors significantly, but still overfit. On the other test sets, where GP does not overfit or overfits slightly, the two SRM-driven GP methods generalize very well. The pattern of generalization errors in the evolutionary process confirms the advantage of SRM principle over the empirical risk minimization principle. In other words, guiding the evolutionary process by the estimated generalization error typically leads to a better generalization ability than by the purely empirical risk. This might be due to the less greedy nature for chasing a lower training error of SRM-driven GP, which encourages a better exploration of the search space.

Considering the comparison between GPSRM and GPOPSRM, on most of the examined benchmarks, GPOPSRM generalizes better than GPSRM. The improvement on the generalization performance is brought by the nonuniform setting in GPOPSRM, which outperforms the uniform setting in GPSRM in two aspects. First, the optimized setting can reduce the random variability of the measured VC-dimension by removing the relatively large MSE as defined in (11). Second, compared with the uniform setting, the nonuniform setting generally has more experiments on design points having a larger number of instances. This will decrease the difference between the theoretical and the experimental maximum deviation of errors. Both the two aspects will lead to a more accuracy VC-dimension of evolved models, therefore will achieve a better generalization estimation. The advantage of GPOPSRM confirms the expectation that a better estimation ability on the VC-dimension of evolved models can lead to a lower generalization error.

C. Further Analysis

Further analysis on the evolved models has been approached with respect to their structures and behaviors. We also have an analysis on the expensive computational cost in SRM-driven GP.

1) Structural Level: To examine how SRM influences the model complexity in GP, we draw the evolutionary plots on the relationship between RMSEs and VC-dimensions in both GP and the SRM-driven GP.[6] These plots are drawn using the median RMSE of the best-of-generation programs and the median VC-dimension of these programs. Note that the best solution is selected according to their fitness value, i.e., the estimated generalization error in the SRM-driven GP and the RMSE in standard GP. So it is possible that the best solutions in these GP methods are total different from each other from the first generation (e.g., on $F1$, $F2$, $F3$, and $F4$). For standard GP, the VC-dimension of the best solution is measured and recorded, but they never play a role in the evolutionary process.

As Fig. 6 shows, the over pattern is that the VC-dimension of the best solution increases along with the generation, while

[6]More information on the relationship between RMSE and the confidence intervals can be seen from our online supplementary material.

TABLE V
EXAMPLE OF BEST-OF-THE-RUN MODELS FOR PROBLEM $f_6 = x_1^4 - x_1^3 + x_2^2/2 - x_2$

	Method	Evolved Model	Simplified Model	Evolved Model	Simplified Model
f_6	GP	$(+(+(0-(sqrt(*(0-(sqrt (*(*-4.604x_2)x_1)))x_1))) (*(sqrt(+(%1.0(*(sqrt(+ -4.604(*-10.18x_1))) (*(*(%1.0x_2)x_1)(+x_2 x_1))))(*-10.182x_1))) (*(*(*0.343x_1) (%1.0x_2))x_1)(0-(+(0-x_2) (*-4.604x_2))))))) (*0.343(*(*(*0.343x_1) (%1.0x_2))x_1)))$	$1.92x_1^2 * \left(\frac{1}{\left(x_1+\frac{x_1^2}{x_2}\right)*(-10.18x_1+4.6)^{1/2}} - 10.18x_1\right)^{1/2} + \frac{0.117x_1^2}{x_2} - (-x_1-(-4.6x_1x_2)^{1/2})^{1/2}$	$(+(+(0-x_2)(*(0-(*x_1 x_1))(+(0-(*x_1x_1))x_1)))(sqrt(+(0- (+(0-x_2)2.093))(+ (+(0-(+(+(0-x_2) (+(0-x_2)2.093))2.093)) (+(0-(+(0-x_2)(+(0- x_2)2.093)))(0-(0-x_2))))(sqrt(0-(+(0-x_2)(+ (0-x_2)2.093))))))))$	$x_1^4-x_1^3-x_2+ (6x_2-8.372)^{1/2} +(2x_2-2.093)^{1/2}$
	BGP	$(+(+(*(0--0.332)(sqrt(0- -0.332)))-0.95)(+(*(+(sqrt (0-x_2))(*-0.978x_1))(+(*x_1 x_1)x_1))(+(+(+(*(*x_1x_1) (*x_1x_1))(*-0.978x_1))(sqrtx_2)) (0-x_2))(sqrtx_2))))$	$x_1^4-0.978x_1^3-0.978x_1^2-0.978x_1+2\sqrt{x_2}-x_2+0.758$	$(+(*(*x_1x_1)(sqrt (+(+(+(*(*x_1x_1)(*x_1x_1)) (*(\%1.0x_1)(+(\%1.0x_1)x_2))) (*(\%1.0x_1)(0-x_1))) (*(\%1.0x_1)(+x_1x_2)))))) (*(0-x_1)(+(sqrt(+(*(*x_1x_1) (*x_1x_1))(*(\%1.0x_1)(+x_1x_2)))) (sqrt(\%1.0(+(+(0--0.456) (+(\%1.00.191)(+x_1x_2))) (+x_2(+(*x_1x_1) (+x_1x_2)))))))))$	$x_1*(x_1^6+2x_1x_2+1)^{1/2} -x_1*(x_1^4+x_2/x_1+1)^{1/2}$
	GPSRM	$(+(0-(*(*x_1x_1)x_1)) (+(sqrt(+(*(0-(+x_2 x_2))(sqrt(+(0-(+(+x_2 x_2)x_2))(*(0-(+x_2 x_2))(*x_1x_1))))) (+x_2x_2)))(*(*x_1x_1) (*x_1x_1))))$	$x_1^4-x_1^3+\sqrt{2x_2\left(1-\sqrt{x_2\left(-2x_1^2-3\right)}\right)}$	$(+(+(0-(*(sqrtx_1)x_2)) (*(*x_1x_1)(0-x_1))) (+(*(*x_1x_1)(*x_1 x_1))(+(sqrt(*(*(sqrt (*(sqrtx_1)x_2))x_2)x_2)) (+(*(*(+(0-(*(sqrtx_1) x_2))(*(*x_1x_1) (0-x_1)))x_2)(+(0-x_1) x_1))(+(0-x_1)x_1)))))$	$x_1^4-x_1^3-x_2\sqrt{x_1} +\sqrt{(x_2)^2\sqrt{x_2\sqrt{x_1}}}$
	GPOPSRM	$(+(sqrt(+(sqrt(*(0-(+ x_2(+x_2x_2)))(+(*(*(+ x_2x_2)x_2)(*x_2(+x_2x_2))) (sqrt(*x_1x_1)))))(0-(sqrt(0-(*(+x_2x_2)(+x_2 x_2)))))))(*(+(0-x_1) (*x_1x_1))(*x_1x_1)))$	$x_1^4-x_1^3+\sqrt{\sqrt{-3x_2(x_1+6(x_2)^2)}}$	$(+(*(*(sqrt(*(*(*(0- -0.192)(0--0.192))x_2) (*(0--0.192)(0- -0.192))))x_2) x_2)(+(*x_1(* (*x_1x_1)x_1)) (0-(*x_1x_1) x_1)))$	$x_1^4-x_1^3+0.036x_2^{5/2}$

RMSE keeps decreasing. As expected, on the ten datasets, standard GP has consistently larger VC values than the SRM-driven GP on all the datasets. In standard GP, where there is no any restriction on the model complexity, the VC-dimension increases very fast. On most generations, it grows linearly. In some cases, at the final stage of the evolutionary process, the growth of the VC-dimension does not bring any benefit on the training performance (This is obvious on *F*4 and *F*6). SRM-driven GP has a different pattern. On most of the datasets, the VC-dimension increases slowly. The upward trend of the VC-dimension and the downward trend of the RMSE are consistent, i.e., a larger increase in the VC-dimension brings a larger decrease to the RMSE. This increase in the model-complexity in SRM-driven GP is effective in reducing the training errors.

Another finding is that the different pattern on the measured VC-dimensions in GPSRM and GPOPSRM. In some cases, this small difference brings a big difference in RMSE (e.g., on *F*1, *F*2, *F*3, LD50, and DLBCL). On these datasets, the difference between the VC-dimension of GPOPSRM and standard GP is large, but the RMSE difference is small. This indicates there might exist certain threshold for the model complexity. Under this threshold, the increase in the model complexity brings notable benefits for the train performance. When the complexity above the threshold, it is difficult to improve the performance by increasing the model complexity. We will investigate this in the future work.

2) Behavioral Level: On the behavioral level, we examine some evolved models in detail and try to find why the models evolved by GPOPSRM outperform those of the other three methods. We randomly took two groups of the best-of-run models from the 100 group candidates on f_6, where the four algorithms all have good generalization performance. They are displayed in the *Evolved Model* column of Table V. To make the behavior of the evolved models more obvious, we present the mathematically simplified form of these models. The original evolved models confirms that SRM-driven GP methods can evolve more compact models with a simpler structure. The behavior of the evolved models can be seen from the simplified form of the models. The similarity between the simplified models and the target model ($f_6 = x_1^4 - x_1^3 + x_2^2/2 - x_2$) indicates why all the four methods can generalize well on f_6. It is clear that the example models evolved by GP on f_6 are more complex than the target models. The other three methods can reduce the model complexity to different levels. Compared with BGP and GPSRM, GPOPSRM can evolve simpler models. Moreover, these simpler models generally contain the same components as those in the target function, such as x_1^4, x_1^3, and x_1^2. This indicates that the behavioral similarity between these models and the target models is higher than their counterparts in BGP and GPSRM.

3) Computational Cost for Measuring SRM: The computation cost in both GPSRM and GPOPSRM is much higher than standard GP, which is usually more than ten times higher. Here, we summarize the additional computational effort needed when introducing SRM into GP under the uniform setting. Using the nonuniform setting needs more effort.

The major cost in SRM-driven GP is spend on measuring the VC-dimension of the solutions experimentally. To measure the VC-dimension of GP solutions, every generation needs $102 * (3000 + 600)$ additional evaluations. Specifically, on every generation, the VC-Dimension of 102 individuals (20% of 512 individuals) needs to be measured. Each individual needs 3000 times training before obtaining the maximum deviations, and 600 additional evaluations to calculate the maximum deviations of the errors. The additional training process uses mini-batch gradient descent. For each time of training, it needs to calculate the gradient of the solution on a subset of instances. All of these are time consuming. On the other hand, as we mentioned above, the model complexity/size in SRM-driven GP is much smaller than standard GP. This saves some effort on the evaluation.

VI. Conclusion

This paper proposed a new GP method by incorporating SRM into GP to improve the generalization ability of GP for symbolic regression. SRM has solid theoretical foundation and is able to provide tight generalization error bound for models. This paper extents the *experimental* method to measure the VC-dimension for regression models and make SRM available for a mixture of linear and nonlinear regression models in GP for the first time.

The results show that SRM-driven GP has impressive generalization gain over standard GP on all the ten examined datasets. In addition, the better generalization performance in SRM-driven GP methods than BGP confirms the advantage of SRM as a framework to estimate generalization error. This paper also conducts a comparison between GPOPSRM and GPSRM, which are SRM-driven GP methods using nonuniform and uniform setting, respectively. The results confirm that GPOPSRM outperforms GPSRM in both the training performance and the generalization ability on most of the examined problems. Further analyzes of the evolved models show that GPOPSRM not only evolves more compact models but also approximate the behavior of the target functions better than the other methods.

However, the overall computational cost of SRM-driven GP methods is much higher than standard GP. In future work, we will try to solve this problem and speed up SRM-driven GP. We also would like to develop a context-aware mechanism to estimate the number of solutions to evaluate for VC-dimension. The new mechanism is expect to save effort on the unnecessary measure of VC-dimension while maintain the effective of SRM in GP. In addition, in this paper, we touched the difference between model size and complexity, but have not analyzed the difference by experiments. We will compare GPOPSRM with some bloat control methods, such as the parsimony method. This paper focus mainly on the benefit of SRM with the experimentally measured VC-dimension to standard GP. However, the SRM should be effective for more domains (e.g., neural networks) with the demand of measuring the model complexity. Moreover, we also would like to broaden our study on the generalization of GP, which involves various types of regression problems, e.g., GP with multiple world models to partition data in [47], GP for solving discontinuous regression problems [48] and Island Model for function construction [49].

References

[1] R. S. Michalski, *Machine Learning: An Artificial Intelligence Approach*, vol. 2, Los Altos, CA, USA: Kaufmann, 1986.

[2] J. Friedman, T. Hastie, and R. Tibshirani, *The Elements of Statistical Learning* (Springer Series in Statistics), vol. 1. Berlin, Germany: Springer, 2001.

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

[4] V. N. Vapnik, *Statistical Learning Theory*, vol. 1. New York, NY, USA: Wiley, 1998.

[5] V. N. Vapnik and S. Kotz, *Estimation of Dependences Based on Empirical Data*, vol. 40. New York, NY, USA: Springer-Verlag, 1982.

[6] E. J. Vladislavleva, G. F. Smits, and D. Den Hertog, "Order of nonlinearity as a complexity measure for models generated by symbolic regression via Pareto genetic programming," *IEEE Trans. Evol. Comput.*, vol. 13, no. 2, pp. 333–349, Apr. 2009.

[7] L. Vanneschi, M. Castelli, and S. Silva, "Measuring bloat, overfitting and functional complexity in genetic programming," in *Proc. 12th Annu. Conf. Genet. Evol. Comput. Conf. (GECCO)*, 2010, pp. 877–884.

[8] V. Cherkassky and Y. Ma, "Comparison of model selection for regression," *Neural Comput.*, vol. 15, no. 7, pp. 1691–1714, 2003.

[9] V. Vapnik, E. Levin, and Y. LeCun, "Measuring the VC-dimension of a learning machine," *Neural Comput.*, vol. 6, no. 5, pp. 851–876, 1994.

[10] Q. Chen, B. Xue, L. Shang, and M. Zhang, "Improving generalisation of genetic programming for symbolic regression with structural risk minimisation," in *Proc. 18th Annu. Conf. Genet. Evol. Comput. Conf. (GECCO)*, 2016, pp. 709–716.

[11] Q. N. Huynh, S. Chand, H. K. Singh, and T. Ray, "Genetic programming with mixed-integer linear programming-based library search," *IEEE Trans. Evol. Comput.*, vol. 22, no. 5, pp. 733–747, Oct. 2018.

[12] Q. Chen, B. Xue, and M. Zhang, "Improving generalisation of genetic programming for symbolic regression with angle-driven geometric semantic operators," *IEEE Trans. Evol. Comput.*, to be published. [Online]. Available: https://doi.org/10.1109/TEVC.2018.2869621

[13] 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.

[14] D. Cohn, L. Atlas, and R. Ladner, "Improving generalization with active learning," *Mach. Learn.*, vol. 15, no. 2, pp. 201–221, 1994.

[15] I. Gonçalves, S. Silva, and C. M. Fonseca, "On the generalization ability of geometric semantic genetic programming," in *Genetic Programming*. Cham, Switzerland: Springer, 2015, pp. 41–52.

[16] 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.

[17] 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.

[18] 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 *Genetic Programming*. Heidelberg, Germany: Springer, 2010, pp. 184–195.

[19] 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.

[20] S. Silva, S. Dignum, and L. Vanneschi, "Operator equalisation for bloat free genetic programming and a survey of bloat control methods," *Genet. Program. Evol. Mach.*, vol. 13, no. 2, pp. 197–238, 2012.

[21] J. C. Mason and D. C. Handscomb, *Chebyshev Polynomials*. Hoboken, NJ, USA: CRC Press, 2002.

[22] M. Keijzer and V. Babovic, *Genetic Programming, Ensemble Methods and the Bias/Variance Tradeoff–Introductory Investigations*. Heidelberg, Germany: Springer, 2000.

[23] B. Efron and R. J. Tibshirani, *An Introduction to the Bootstrap*. Boca Raton, FL, USA: CRC Press, 1994.

[24] A. Agapitos, A. Brabazon, and M. O'Neill, "Controlling overfitting in symbolic regression based on a bias/variance error decomposition," in *Parallel Problem Solving from Nature-PPSN XII*. Heidelberg, Germany: Springer, 2012, pp. 438–447.
[25] J. Fitzgerald, R. M. A. Azad, and C. Ryan, "Bootstrapping to reduce bloat and improve generalisation in genetic programming," in *Proc. 15th Annu. Conf. Genet. Evol. Comput. Conf. (GECCO)*, 2013, pp. 141–142.
[26] G. Folino, C. Pizzuti, and G. Spezzano, "Ensemble techniques for parallel genetic programming based classifiers," in *Proc. 6th Eur. Conf. Genet. Program. (EuroGP)*, 2003, pp. 59–69.
[27] V. Vapnik, *The Nature of Statistical Learning Theory*. New York, NY, USA: Springer, 1995.
[28] V. S. Cherkassky and F. M. Mulier, *Learning From Data: Concepts, Theory, and Methods*. Hoboken, NJ, USA: Wiley, 2007.
[29] K.-I. Funahashi, "On the approximate realization of continuous mappings by neural networks," *Neural Netw.*, vol. 2, no. 3, pp. 183–192, 1989.
[30] G. Lappas, "Estimating the size of neural networks from the number of available training data," in *Proc. Int. Conf. Artif. Neural Netw.*, 2007, pp. 68–77.
[31] H. Neuner, "Design of artificial neural networks for change-point detection," in *Proc. 1st Int. Workshop Qual. Geodetic Observation Monitor. Syst. (QuGOMS)*, 2015, pp. 139–144.
[32] C. E. Borges, C. L. Alonso, and J. L. Montaña, "Model selection in genetic programming," in *Proc. 12th Annu. Conf. Genet. Evol. Comput. Conf. (GECCO)*, 2010, pp. 985–986.
[33] J. L. Montaña, C. L. Alonso, C. E. Borges, and J. De La Dehesa, "Penalty functions for genetic programming algorithms," in *Computational Science and Its Applications—ICCSA*. Heidelberg, Germany: Springer, 2011, pp. 550–562.
[34] M. E. Johnson and C. J. Nachtsheim, "Some guidelines for constructing exact D-optimal designs on convex design spaces," *Technometrics*, vol. 25, no. 3, pp. 271–277, 1983.
[35] W. W. Li and C. J. Wu, "Columnwise-pairwise algorithms with applications to the construction of supersaturated designs," *Technometrics*, vol. 39, no. 2, pp. 171–179, 1997.
[36] X. Shao, V. Cherkassky, and W. Li, "Measuring the VC-dimension using optimized experimental design," *Neural Comput.*, vol. 12, no. 8, pp. 1969–1986, 2000.
[37] V. N. Vapnik and A. Y. Chervonenkis, "On uniform convergence of the frequencies of events to their probabilities," *Teoriya Veroyatnostei i ee Primeneniya*, vol. 16, no. 2, pp. 264–279, 1971.
[38] A. Kolmogorov, "Sulla determinazione empirica di una lgge di distribuzione," *Istituto Italiano degli Attuari*, vol. 4, no. 1, pp. 83–91, 1933.
[39] Q. Chen, B. Xue, and M. Zhang, "Generalisation and domain adaptation in GP with gradient descent for symbolic regression," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, 2015, pp. 1137–1144.
[40] M. Keijzer, "Improving symbolic regression with interval arithmetic and linear scaling," in *Genetic Programming*. Heidelberg, Germany: Springer, 2003, pp. 70–82.
[41] L. Vanneschi, S. Silva, M. Castelli, and L. Manzoni, "Geometric semantic genetic programming for real life applications," in *Genetic Programming Theory and Practice XI*. New York, NY, USA: Springer, 2014, p. 191.
[42] F. Archetti, S. Lanzeni, E. Messina, and L. Vanneschi, "Genetic programming for computational pharmacokinetics in drug discovery and development," *Genet. Program. Evol. Mach.*, vol. 8, no. 4, pp. 413–432, 2007.
[43] 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.
[44] A. Rosenwald *et al.*, "The use of molecular profiling to predict survival after chemotherapy for diffuse large-B-cell lymphoma," *New England J. Med.*, vol. 346, no. 25, pp. 1937–1947, 2002.
[45] B. Efron and R. Tibshirani, "Improvements on cross-validation: The 632+ bootstrap method," *J. Amer. Stat. Assoc.*, vol. 92, no. 438, pp. 548–560, 1997.
[46] S. Luke *et al.* (Mar. 2004). *A Java-Based Evolutionary Computation Research System*. [Online]. Available: https://cs.gmu.edu/ẽclab/projects/ecj/
[47] J. A. Brown and D. Ashlock, "Using evolvable regressors to partition data," *Intell. Eng. Syst. Artif. Neural Netw.*, vol. 20, pp. 187–194, Jan. 2010.
[48] X. Shengwu, W. Weiwu, and L. Feng, "A new genetic programming approach in symbolic regression," in *Proc. 15th IEEE Int. Conf. Tools Artif. Intell. (ICTAI)*, 2003, pp. 161–165.
[49] D. Whitley, S. Rana, and R. B. Heckendorn, "Island model genetic algorithms and linearly separable problems," in *Proc. AISB Int. Workshop Evol. Comput.*, 1997, pp. 109–125.

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, Wellington, New Zealand, in 2018.

In 2014, she joined the Evolutionary Computation Research Group, Victoria University of Wellington, 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 of international conferences, including IEEE Congress on Evolutionary Computation, and international journals, including the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION and the IEEE TRANSACTIONS ON CYBERNETICS.

Mengjie Zhang (M'04–SM'10) received the B.E. and M.E. degrees from 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 CIS Intelligent Systems and Applications Technical Committee, the immediate Past Chair for 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 have been a Panel Member of the Marsden Fund (New Zealand Government Funding). He is a member of ACM.

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 currently the Chair of the IEEE Task Force on Evolutionary Feature Selection and Construction, IEEE Computational Intelligence Society (CIS), 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 Optimization. She is also an Associate Editor/member of editorial board for five international journals and a reviewer of over 50 international journals. She is the Finance Chair of IEEE Congress on Evolutionary Computation in 2019, the Program Co-Chair of the 31st Australasian AI in 2018, ACALCI in 2018, and the 7th International Conference on SoCPaR in 2015. She is also the tutorial chair, the special session chair, or the publicity chair for many other international conferences.