Improving the Generalisation of Genetic Programming for Symbolic Regression

by

Qi Chen

A thesis
submitted to the Victoria University of Wellington
in fulfilment of the
requirements for the degree of
Doctor of Philosophy
in Computer Science.

Victoria University of Wellington
2018

Abstract

Symbolic regression (SR) is a function identification process, the task of which is to identify and express the relationship between the input and output variables in mathematical models. SR is named to emphasise its ability to find the structure and coefficients of the model simultaneously.

Genetic Programming (GP) is an attractive and powerful technique for SR, since it does not require any predefined model and has a flexible representation. However, GP based SR generally has a poor generalisation ability which degrades its reliability and hampers its applications to science and real-world modeling. Therefore, this thesis aims to develop new GP approaches to SR that evolve/learn models exhibiting good generalisation ability.

This thesis develops a novel feature selection method in GP for high-dimensional SR. Feature selection can potentially contribute not only to improving the efficiency of learning algorithms but also to enhancing the generalisation ability. However, feature selection is seldom considered in GP for high-dimensional SR. The proposed new feature selection method utilises GP's build-in feature selection ability and relies on permutation to detect the truly relevant features and discard irrelevant/noisy features. The results confirm the superiority of the proposed method over the other examined feature selection methods including random forests and decision trees on identifying the truly relevant features. Further analysis indicates that the models evolved by GP with the proposed feature selection method are more likely to contain only the truly relevant features and have better interpretability.

To address the overfitting issue of GP when learning from a relatively

small number of instances, this thesis proposes a new GP approach by incorporating structural risk minimisation (SRM), which is a framework to estimate the generalisation performance of models, into GP. The effectiveness of SRM highly depends on the accuracy of the Vapnik-Chervonenkis (VC) dimension measuring model complexity. This thesis significantly extends an *experimental* method (instead of theoretical estimation) to measure the VC-dimension of a mixture of linear and nonlinear regression models in GP for the first time. The *experimental* method has been conducted using uniform and non-uniform settings and provides reliable VC-dimension values. The results show that our methods have an impressively better generalisation gain and evolve more compact model, which have much smaller behavioural difference from the target models than standard GP and GP with bootstrap, The proposed method using the optimised non-uniform setting further improves the one using the uniform setting.

This thesis employs geometric semantic GP (GSGP) to tackle the unsatisfied generalisation performance of GP for SR when no overfitting occurs. It proposes three new angle-awareness driven geometric semantic operators (GSO) including selection, crossover and mutation to further explore the geometry of the semantic space to gain a greater generalisation improvement in GP for SR. 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, to be resistant to overfitting. The results show that compared with two kinds of state-of-the-art GSOs, the proposed new GSOs not only drive the evolutionary process fitting the target semantics more efficiently but also significantly improve the generalisation performance. A further comparison on the evolved models shows that the new method generally produces simpler models with a much smaller size and containing important building blocks of the target models.

List of Publications

1. Qi Chen, Mengjie Zhang, Bing Xue. "Feature Selection to Improve Generalisation of Genetic Programming for High-Dimensional Symbolic Regression". *IEEE Transaction on Evolutionary Computation.* 2017. PP. 792-806, Vol.21, No. 5, DOI: 10.1109/TEVC.2017.2683489.

2. Qi Chen, Mengjie Zhang, Bing Xue. "Improving the Generalisation of Genetic Programming for Symbolic Regression with Angle-Driven Geometric Semantic Operators". Submitted to *IEEE Transaction on Evolutionary Computation*. 2017. (Passed the first-round review)

3. Qi Chen, Mengjie Zhang, Bing Xue. "Structural Risk Minimisation-Driven Genetic Programming for Enhancing Generalisation in Symbolic Regression" *IEEE Transaction on Evolutionary Computation*. 2017. (Conditional accepted). 15pp.

4. Qi Chen, Mengjie Zhang, Bing Xue. "Geometric Semantic Genetic Programming with Perpendicular Crossover and Random Segment Mutation for Symbolic Regression". *Proceedings of the 11th International Conference on Simulated Evolution and Learning*. Lecture Notes in Computer Science, **Vol. 10593**, Shenzhen, China. 10-13 November 2017. pp. 229-245.

5. Qi Chen, Bing Xue, Mengjie Zhang. "New Geometric Semantic Operators in Genetic Programming: Perpendicular Crossover and Ran-

dom Segment Mutation". *Proceedings of 2017 Genetic and Evolutionary Computation Conference (GECCO 2017 Companion)*. Berlin, Germany. 15-19 July, 2017. pp. 223-224.

6. Qi Chen, Mengjie Zhang, Bing Xue. "Angle-aware Geometric Semantic Crossover in Genetic Programming for Symbolic Regression". *Proceedings of the 20th European Conference on Genetic Programming (EuroGP 2017)*. Lecture Notes in Computer Science, **Vol. 10196**. Amsterdam, The Netherlands. 18-21 April 2017. pp. 229-245.

7. Qi Chen, Mengjie Zhang, Bing Xue. "Improving Generalisation of Genetic Programming for Symbolic Regression with Structural Risk Minimisation". *Proceedings of 2016 Genetic and Evolutionary Computation Conference (GECCO 2016)*. ACM Press. Denver, Colorado, USA. 20-24 July 2016. pp. 709-716.

8. Qi Chen, Bing Xue, Mengjie Zhang. "Improving Generalisation of Genetic Programming for High-Dimensional Symbolic Regression with Feature Selection". *Proceedings of 2016 IEEE World Congress on Computational Intelligence/ IEEE Congress on Evolutionary Computation (WCCI 2016 /CEC2016)*. Vancouver, Canada. 24-29 July, 2016. pp. 3793-3800.

9. Qi Chen, Bing Xue, Mengjie Zhang. "Generalisation and Domain Adaptation in GP with Gradient Descent for Symbolic Regression". *Proceeding of 2015 IEEE Congress on evolutionary computation (CEC2015)*. Sendai, Japan. 25-28 May,2015 pp. 1137-1144

10. Qi Chen, Mengjie Zhang, Bing Xue. "Genetic Programming with Embedded Feature Construction for High-Dimensional Symbolic Regression". *Proceedings of the 20th Asia-Pacific Symposium on Intelligent and Evolutionary Systems*. Canberra, Australia, November 16-18, 2016. pp. 87-102.

Acknowledgments

I would like to express my sincere gratitude to those who gave me the assistance and support during my PhD study.

My deepest gratitude goes to my supervisors, Prof Mengjie Zhang and Dr Bing Xue. They have spent dedicated time and efforts to train my research skills, and provided constructive feedback to improve my research work. This thesis would not have been possible without their encouragement, motivation, inspiration, and guidance.

I am very grateful for the Marsden Fund of New Zealand (VUW1209, VUW1509 and 16-VUW-111), School of Engineering and Computer Science, Faculty of Engineering , and Victoria University of Wellington for their financial support over the past more than three years.

I wish to thank my friends in the Evolutionary Computation Research Group (ECRG) for creating an active and interesting research environment. Thank you to my officemates for their help in my study and nice discussions.

I wish to thank my family for their constant support and encouragement throughout my study. Thank you to my parents who have always been there for me, offering their unconditional love and care. Last, but not least, I would like to thank my beloved husband Yu and my little one Shirley for their love, patience and support. You have always been the source of love that helps me complete this journey.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Problem Statement

Artificial Intelligence (AI), which is one of the most absorbing branches of Computer Science, has an important goal of developing intelligent systems to make rational decisions based on observations. Generally, a system is defined to be intelligent when it can handle tasks that usually are performed by human beings and require a significant degree of intelligence [195]. Another important property of intelligent systems is to improve themselves without external guidance, i.e. to learn and induce a general pattern from the observations. Specifically, learning methods in AI are expected to be able to generalise on unseen data of the same pattern with the given observations.

Symbolic regression is a function identification process, which aims to develop a model to best fit a given dataset [184]. More specifically, the task of symbolic regression is to identify the relationship between the input variables and the response variable(s) in the dataset, and express the relationship in mathematical models or symbolic descriptions. Symbolic regression is named to emphasise its target, which is to produce/find the symbolic description of a model, not only a set of coefficients for a predefined model. This is a sharp contrast with classical or statistical regres-

sion techniques, where the structure of the model is usually predefined [19, 162, 199] and the task is to find the best fitted coefficients for the model. Being free from predefined model structure, symbolic regression is less likely to be affected by unknown gaps in domain knowledge or human bias [15, 208].

Genetic programming (GP) [124] is an approach to automatically evolving computer programs to solve the problems at hand. As a paradigm of evolutionary computation (EC), GP is inspired by biological evolution and generally starts with a population of randomly initialised programs. The population is progressively evolved to gain higher fitness over a series of generations. At the end of the evolutionary process, a satisfactory solution is expected to be found to tackle the problems at hand.

GP has two attractive properties, the symbolic nature of solutions and free from prior knowledge, which make it very suitable for symbolic regression. GP based symbolic regression provides a way to efficiently and effectively convert data steams and/or data sets into knowledge and actionable insight, and it has been successfully applied to many real-world problems, such as industrial processing [75, 44, 90, 132, 197], software engineering [2, 108] and digital circuits design [4, 172].

In the past decades, GP based symbolic regression has many successful stories. However, its generalisation ability is still an open issue [173]. The generalisation ability/error $Err_T = E[L(Y, f(X))|_T]$ measures the prediction error of the learnt model over a set of unseen/test data for a given training set T. When tackling learning problems, particularly supervised learning problems, generalisation of solutions is one of the most important performance criteria for any learning algorithm. Compared with impressive training performance, generalising well on unseen data is usually much more important for learning algorithms, since it shows that the learning algorithm is not memorising the instances in the training set but learning a general model/pattern. GP has a flexible representation, which usually has no constraint on the structure of the evolved models.

This property makes GP good at learning complex patterns. However, the downside of the flexible representation is increasing the risk of overfitting in GP, i.e. the risk of a poor generalisation ability [173].

The contrary concept of generalisation is overfitting. Overfitting means poor generalisation performance. A more detailed definition of overfitting is given in [152]. According to this definition, a model overfits the training data when there exist some other models with worse training performance but better performance over the entire distribution of instances. Besides the downside of the representation of GP, there are many other factors that make GP overfit. One of them is the nature of GP to chase the lowest training error. An ideal learning process is to learn the underlying relationship between the input and output variables while ignoring the noise. Nevertheless, GP has a greedy nature in pursuing models with the lowest error on the training set. Then after the true relationship is learnt, GP might continue to fit the noise in the training set in order to minimise the training error. In this case, GP might generate models that overfit the training set. Furthermore, when the number of available observations/instances is too small to represent the true pattern of the desired function, GP is prone to produce models that overfit the training set.

Another issue, which might limit the generalisation of GP for symbolic regression, is the high-dimensionality of the data. The representation of observations often uses a large number of features, but only a small number of these features are relevant to the target/true model. Searching for an unknown model in a high-dimensional space with a small number of observations runs the risk of overfitting and leads to poor generalisation.

1.2 Motivations

Generalisation has been well-studied in many fields of machine learning for long. In these fields, such as *Artificial Neural Networks* [91, 60] and *Support Vector Machines* [210, 12], significant research has been dedicated

to investigate the factors effecting generalisation and develop methods to improve generalisation. Compared with these fields, the generalisation ability of GP for symbolic regression has received growing attention just in recent years [98, 42, 130, 218].

An important issue that leads to poor generalisation of GP for symbolic regression arises when the available data is high-dimensional. In theory, more features means more discriminative power. However, in practice excessive features may not only significantly slow down the learning process, but also cause the models overfit the training data since irrelevant or redundant features may confuse the learning algorithms. Feature selection is a process of choosing relevant features that are necessary and sufficient to describe the output variable(s). For high-dimensional datasets, feature selection is usually required in order to avoid the curse of dimensionality and reduce the risk of overfitting. While much work has been done on feature selection in machine learning [64, 139], the majority of the existing work focuses on classification problems [64, 240]. Not much related research on symbolic regression has been reported to date. This thesis aims to fill the gap and investigate the possibility of feature selection to improve the generalisation of GP for symbolic regression.

In previous studies that improve the generalisation of GP, two major categories of methods have been proposed [41]. Methods in the first category aim to tackle the problem of bloat [88, 141, 190, 246], which is a phenomenon that the growth on the size of GP solutions does not bring sufficient improvements in their learning performance [141]. These methods are based on the assumption that bloat and overfitting are related phenomena. It is well accepted that compact solutions will generalise better than their complex counterparts, since the latter have more space for noise thus overfit the training set [152]. However, recent research has observed that overfitting can occur in absence of bloat [41, 67]. Therefore, some researchers suggested that bloat and overfitting are two independent phenomena and should be tackled by separate mechanisms [67, 217].

Methods in the second category are to detect and avoid overfitting [112, 170]. Researchers found that compared with the size of solutions, the functional/behavioural complexity of solutions is a more appropriate and useful indicator of overfitting [224, 233]. Typically, behaviourally simple models for regression problems are more generalisable [65, 94]. However, measuring the complexity of the candidate models is not a trivial task. A number of model complexity measures for GP have been proposed [42, 217, 224, 233]. Some of them use the complexity of solutions as an indicator of overfitting, while others treat the complexity as an objective that needs to be optimised. However, these methods usually measure the complexity of various approximations (of GP solutions), such as *order of nonlinearity* [233], *curvature* [224] and *graph based complexity* [42], but not directly measure the solutions themselves. Moreover, some of these measures have shown to have a negative correlation with the generalisation performance [40]. Further investigation on how to measure the complexity of GP solutions and how the complexity measure can be used as an indicator of overfitting is fundamental for pursuing a good generalisation ability of GP for symbolic regression.

At the same time, overfitting and model complexity are well studied in the field of *statistical learning theory*, particularly *probably approximately correct* (PAC) [221, 231]. These fields counteract overfitting by providing theoretical tools to analyse learning accuracy. These theoretical tools estimate the expected test error of candidate models during learning process and provide a good way to detect overfitting in principle. For learning problems, especially when there are insufficient data instances, the statistical approaches such as *Akaike Information Criterion* (AIC) [8], *Final Prediction Error* (FPE) [8], *Bayesian Information Criterion* (BIC) [23] and *Structural Risk Minimisation* (SRM) [231], approximate the validation step analytically and estimate prediction error effectively. These methods utilise different measurements to fulfil the ability of assessing the generalisation error bound. For example, SRM is based on *Vapnik-Chervonenkis Dimension* [231], which

measures the model complexity. SRM derives formal formulae defining the difference between the generalisation and empirical/training errors for various classification and regression models. Regardless of the promising benefits of these statistical learning approaches in other fields, they have not been well-studied in GP yet. This thesis will introduce these theoretical tools into GP to evolve/learn more generalisable models for symbolic regression. We will investigate and analyse whether (and how) the theoretical tools could provide a reliable indicator of overfitting, and on the relationship between model complexity and generalisation will be conducted.

The previous research on promoting the generalisation ability of GP focuses mainly on avoiding overfitting. In some scenarios, GP might have poor performance on unseen data, but no overfitting occurs [62]. In these scenarios, the traditional methods on enhancing the generalisation of GP, which often focus only on reducing/eliminating overfitting, might not work. A potential solution is *semantic GP*, which is a new branch of GP. Semantic GP introduces semantic information into GP to utilise the knowledge extracted from the behaviour of GP solutions. Previous work [98, 43, 218] has shown that introducing semantic-awareness into the evolutionary process brings notable improvements to GP for symbolic regression on both the training and unseen data. Despite the promising performance of semantics GP methods, they have some potential limitations such as the overgrown offspring that may restrict the generalisation of GP. Moreover, the relationship between semantic control and generalisation is still unclear. More research is deserved. Further investigation on these issues is sensible for better improvements on generalisation ability of GP.

1.3 Research Goals

The overall goal of this thesis is to develop **new GP approaches for symbolic regression** which can evolve models exhibiting an **impressive generalisation ability**. Poor generalisation often occurs when there is overfit-

ting, but may also be affected by underfitting and other causes. This thesis would like to improve the generalisation performance of GP for symbolic regression, which means counteracting poor generalisation. This includes investigating the factors influencing the generalisation performance of GP for symbolic regression, and proposing substantial improvements on the basic components of GP with an expectation of notably enhancing its generalisation ability for symbolic regression. Specifically, to fulfil this goal, the following objectives are established to guide this research.

1. *Developing a new GP approach incorporating feature selection to high-dimensional symbolic regression.* The rationale behind incorporating a feature selection method into GP for high-dimensional symbolic regression is to prevent the evolutionary process from manipulating irrelevant features and learning from the noise, and hence improve the generalisation of GP. The new feature selection approach is expected to discard irrelevant features and select a subset of features, which are necessary and sufficient to describe the target variable(s). Using the set of selected features, GP for symbolic regression is expected to achieve significantly better generalisation performance.

   GP has the ability to automatically explore the search space to detect relevant features. But when learning in a large space, the built-in feature selection ability of GP is not strong enough to identify the relevant features. In this scenario, GP is prone to include irrelevant/noisy features in the evolved models. GP for high-dimensional symbolic regression therefore runs a high risk of overfitting. Feature selection is an important preprocessing step for high-dimensional datasets. However, most of the existing work is for classification tasks. Developing a novel feature selection approach to GP for high-dimensional regression problems is necessary. The feature selection process is expected to discard irrelevant/noise features while maintaining a subset of relevant features which are highly correlated with the target models. Learning from the selected subset of features

might be intuitively easier to produce models with good generalisation performance. This study will investigate a new feature relevance analysis method, which measures the degree of importance of each feature in the dataset. Accordingly, a subset of relevant features with a high importance value will be selected while removing other features.

2. *Developing a new GP approach implementing an estimating framework on generalisation performance based on learning theory and VC-dimension.* The proposed GP method attempts to detect and avoid overfitting. The estimated generalisation bound is expected to be an accurate indicator of the generalisation gain of GP and guide the evolutionary process towards a good trade-off between training accuracy and model complexity. This trade-off is crucial to reducing/avoiding overfitting and making GP solutions generalise well on unseen data.

   The greed in searching for models with the lowest training error makes GP based symbolic regression prone to overfitting, which unavoidably leads to poor generalisation performance. In order to generate generalisable models, it is necessary to detect and avoid/eliminate overfitting. Model complexity is an important indicator of overfitting, since excessively complex models usually have a tendency of overfitting. The complexity of a model is considered to be negatively related to its generalisation ability. This study aims to estimate the generalisation performance of GP solutions by developing new methods based on the framework of *Structural Risk Minimisation* (SRM). SRM estimates the generalisation bound of models based on the empirical error and the model complexity, which is measured by the *VC-dimension*. An accurate measure of the VC-dimension of models will result in a tight generalisation bound, which corresponds to a precise prediction on the performance of models on the unseen data. However, because of the difficulties in measuring the VC-dimension of nonlinear models, no existing work has been pro-

posed to implement SRM into GP. This study aims to fill the gap and utilise the estimation ability of SRM on generalisation performance to release/avoid overfitting.

3. *Developing a new Geometric Semantic GP (GSGP) approach with angle-awareness for symbolic regression to improve the generalisation performance of GP.* The proposed algorithm is expected to provide solutions to the limitations of the existing GSGP methods and further explore the geometry of the semantic space, and hence obtain a greater generalisation gain than the existing GSGP methods and standard GP.

   As we mentioned in the motivation section, overfitting is not the only reason for poor generalisation in GP. In many scenarios, methods, that aim to reduce overfitting thus improve generalisation do not work. In these scenarios, although the generalisation performance is poor, no overfitting occurs. It has been shown that *semantics-aware GP*, particularly GSGP, may help [98, 43, 218]. Models evolved by semantic GP generally exhibit a significantly better performance both on training and test data than those by standard GP. Although the exact reason why semantic GP can promote the generalisation ability of models is still unknown, the geometric properties of the geometric semantic operator are shown to be highly related to the impressive generalisation gain in GP for symbolic regression [98, 218]. These geometric semantic operators (appropriately) manipulate the semantics of programs. They usually make a bounded semantic impact and generate child programs with similar (but usually better) behaviour to their parents. However, the huge size of the models produced by the geometric operators, the highly expensive computational cost and the potential ineffectiveness of geometric semantic crossover are still major limitations of GSGP. This research will provide solutions to these limitations. It will develop new geometric operators, including selection, crossover and mutation, to drive the evolutionary process to fit the target semantics more effectively and

efficiently while overcoming the drawbacks of the existing GSGP methods.

1.4 Major Contributions

This thesis makes the following major contributions:

1. This thesis shows how feature selection can be introduced into GP to improve the generalisation performance of symbolic regression. This thesis develops a new feature selection approach to GP for high-dimensional symbolic regression. The feature selection method helps GP identify the truly relevant features effectively, and makes the regression process robuster by detecting and discarding noisy features. The results show that the proposed feature selection method outperforms some state-of-the-art feature selection methods in helping GP gain an impressively better generalisation ability. This work is one of the very first studies on developing feature selection algorithms for GP for high-dimension symbolic regression. Previous work focuses mainly on selecting features for classification problems.

   Part of this contribution has been published in:

   Qi Chen, Mengjie Zhang, Bing Xue. "Feature Selection to Improve Generalisation of Genetic Programming for High-Dimensional Symbolic Regression". *IEEE Transaction on Evolutionary Computation.* 2017. PP. 792-806, Vol.21, No. 5, DOI: 10.1109/TEVC.2017.2683489.

   Qi Chen, Bing Xue, Mengjie Zhang. "Improving Generalisation of Genetic Programming for High-Dimensional Symbolic Regression with Feature Selection". *Proceedings of 2016 IEEE World Congress on Computational Intelligence/ IEEE Congress on Evolutionary Computation (WCCI 2016 /CEC2016).* Vancouver, Canada. 24-29 July, 2016. pp. 3793-3800.

2. The thesis shows how learning theory and VC-dimension can be used to measure the program complexity in GP to significantly improve the generalisation performance for symbolic regression. The thesis develops a new GP approach to symbolic regression that can reduce overfitting by providing a reliable estimation of generalisation error during the evolutionary process. The new method for estimating generalisation error roots on the development of SRM, which is a theoretical tool from statistical learning theory, and makes the theoretical framework available for a mixture of linear and nonlinear regression models in GP for the first time. The results show that, compared with standard GP and GP equipped with the state-of-the-art generalisation estimating methods, the proposed GP approach can detect and reduce overfitting more effectively. The positive effect of SRM-driven GP on generalisation confirms that it is feasible to utilise SRM to achieve a proper trade-off between the model accuracy and complexity. This trade-off has long been desired but is lack of solution in the GP community. This is a very first work to demonstrate that the theory on VC-dimension is practically useful for GP to improve its generalisation performance.

   Part of this contribution has been published/shown in:

   Qi Chen, Mengjie Zhang, Bing Xue. "Improving Generalisation of Genetic Programming for Symbolic Regression with Structural Risk Minimisation". *Proceedings of 2016 Genetic and Evolutionary Computation Conference (GECCO 2016)*. ACM Press. Denver, Colorado, USA. 20-24 July 2016. pp. 709-716.

   Qi Chen, Mengjie Zhang, Bing Xue. "Structural Risk Minimisation-Driven Genetic Programming for Enhancing Generalisation in Symbolic Regression" *IEEE Transaction on Evolutionary Computation*. 2017. (Conditional accepted). 15pp.

3. This thesis shows how to use semantics and geometric information

in GP to improve generalisation performance for symbolic regression. This thesis proposes a new GSGP approach to enhancing the generalisation ability of GP, which covers the scenario when no overfitting occurs in GP. The benefit of semantic information to the generalisation of GP has been confirmed, and the underlying reasons for this phenomenon are analysed in depth. The results confirm that the proposed GSGP method increases the generalisation of GP more remarkably than the state-of-the-art GSGP methods. The shortcomings of the existing GSGP methods have been overcome, which makes GSGP more applicable on real-world regression problems with a large number of instances. This thesis also has conducted the first comprehensive comparison between different types of GSGP methods.

Part of this contribution has been published/shown in:

Qi Chen, Mengjie Zhang, Bing Xue. "Angle-aware Geometric Semantic Crossover in Genetic Programming for Symbolic Regression". *Proceedings of the 20th European Conference on Genetic Programming (EuroGP 2017)*. Lecture Notes in Computer Science, **Vol. 10196**. Amsterdam, The Netherlands. 18-21 April 2017. pp. 229-245.

Qi Chen, Bing Xue, Mengjie Zhang. "New Geometric Semantic Operators in Genetic Programming: Perpendicular Crossover and Random Segment Mutation". *Proceedings of 2017 Genetic and Evolutionary Computation Conference (GECCO 2017 Companion)*. Berlin, Germany. 15-19 July, 2017. pp. 223-224.

Qi Chen, Mengjie Zhang, Bing Xue. "Geometric Semantic Genetic Programming with Perpendicular Crossover and Random Segment Mutation for Symbolic Regression". *Proceedings of the 11th International Conference on Simulated Evolution and Learning (SEAL).* Lecture Notes in Computer Science, **Vol. 10593**, Shenzhen, China. 10-13 November 2017. pp. 229-245.

Qi Chen, Mengjie Zhang, Bing Xue. "Improving the Generalisa-

tion of Genetic Programming for Symbolic Regression with Angle-Driven Geometric Semantic Operators". Submitted to *IEEE Transaction on Evolutionary Computation*. 2017. (Passed the first-round review)

1.5 Organisation of Thesis

The remainder of this thesis is structured as follows. Chapter 2 presents the literature review of related work. The main contributions of the thesis are presented in Chapters 3-5. Each chapter addresses one of the research objectives. Chapter 6 concludes the thesis and highlights some promising future research directions.

Chapter 2 presents basic concepts and background on machine learning and classical regression, which is followed by an introduction to evolutionary computation, GP and symbolic regression. As GP based symbolic regression is the focus of this study, a detailed introduction of each component of GP are presented. A number of related works on GP for symbolic regression are highlighted, particularly those on improving the generalisation performance of GP.

Chapter 3 proposes a new feature selection approach to improving the generalisation of GP for high-dimensional symbolic regression, where the importance of a feature is measured by the impact on increasing/reducing the error of models in GP. A permutation measure is employed to select the truly relevant features. Then a subset of features are selected and fed to GP for symbolic regression. To investigate and confirm the effect of the proposed feature selection method on promoting generalisation of GP, experiments have been conducted to compare the regression performance of several GP for symbolic regression methods equipped with various state-of-the-art feature selection methods. The results are presented and analysed in detail.

Chapter 4 develops a new GP approach, which estimates the general-

isation error of the models based on the model complexity and empirical error during the evolutionary process. It investigates the relationship between the model complexity, training/empirical error and generalisation error. The chapter then proposes a new algorithm and evaluates the effect of the new algorithm on reducing overfitting and promoting generalisation ability. A set of experiments are conducted to compare the new algorithm with the state-of-the-art methods. The results and analysis are presented.

Chapter 5 develops a new GSGP approach, which introduces the angle-awareness into the evolutionary process. Three new geometric semantic operators, including a geometric crossover, a geometric mutation and a selection operator, are proposed. In addition, a new method to formalise the semantic requirement is developed. The proposed GSGP method is then examined and compared with two major variants of GSGP methods. Deep analyses are conducted on the results. Many important and interesting findings on the generalisation of GSGP and GP are reported.

Chapter 6 summarises the work and draws conclusions of this thesis. Key research points and major contributions of this thesis are also ascertained. Some future research directions are also pointed out.

Chapter 2

Literature Review

This chapter provides definitions of the basic concepts and covers essential background of machine learning, generalisation and overfitting. This is followed by an introduction of symbolic regression by means of comparing with classical regression. A brief introduction of evolutionary computation is also presented. As genetic programming (GP) is a core component in this thesis, a detailed introduction of each component of GP is presented. Semantic GP, which is a particularly new branch of GP with promising performance, is introduced. Some elements in learning theory, which are related to generalisation and will be further studied later in this thesis, are also introduced. This chapter then reviews typical works related to GP for symbolic regression with a focus on its generalisation, feature selection approaches and the implementation of the specific elements of learning theory.

2.1 Machine Learning

Machine learning is one of the most important areas in *Artificial Intelligence*. It can be broadly defined as computational methods using *experience* to improve performance or to make accurate predictions [153]. Mitchell [152] provided a more precise definition of learning as follows:

Definition 1. *A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.*

The *experience E* is the past information and may take different forms available to the learner, typically collected as electronic data. The items of the data are often termed *examples* or *samples* or *instances*. The set of attributes are *features*, which are usually represented as vectors. The quality and size of the data is crucial to the success of the learner in all cases.

The main objective of machine learning consists of designing accurate and efficient learning algorithms to generate accurate prediction for unseen items. These algorithms are data-driven methods combining fundamental concepts in computer science with ideas from statistics, optimisation and probability [153].

There are a number of ways to determine various types of learning. One of the most commonly used ways is based on the types of feedback. Three types of feedback determine the three main types of learning [192].

- *Supervised Learning*: The learner receives a set of labeled examples and learns a function to map from inputs to outputs. This is the most common learning scenario.

- *Unsupervised Learning*: The learner exclusively receives unlabelled examples and learns patterns in the inputs without feedback. In this scenario, it is difficult to evaluate the performance of the learner.

- *Reinforcement Learning*: In this scenario, in order to collect information, the learner interacts with the environment. During this process the learner learns from a series of reinforcements — punishments or rewards from the environment.

2.1.1 Learning Tasks

Learning tasks can be categorised into different types with respect to the desired output of machine learning systems. The most mature and widely researched ones are [25, 11]:

- *Classification*: Learning from a set of inputs which are divided into two or more kinds of classes, the learner must develop a model to assign the unseen items to each category. Email Spam filtering is a typical example of classification.
- *Regression*: The learner needs to explore the underlying form of relationship between inputs and outputs and predict a real value for each input item. Regression is widely used for forecasting and prediction. The difference between regression and classification is that the output of regression is continuous rather than discrete.
- *Clustering*: Partition a set of inputs into different groups. Different from classification, these groups are not known beforehand. Clustering is often used for analysing large datasets. For instance, in social network analysis, clustering attempts to identify the "community" within a large group of people [153].
- *Association Rules*: Learning association rules aim to identify the strong rules/regularities discovered in databases using different measures of interestingness [182, 5]. It can be applied to a number of industries, such as medicine, the retail industry and website traffic analysis.
- *Density Estimation*: Learning the underlying probability distribution that gives rise to the observed variables.
- *Dimension reduction*: Reducing the number of features under consideration, and can be divided into feature selection and feature extraction.

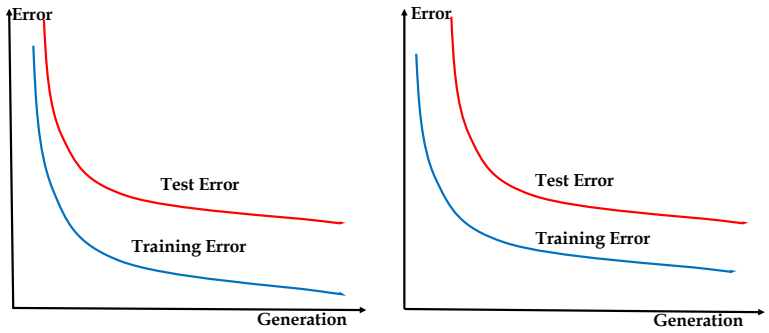
2.1.2 Training and Testing

The common procedure of learning consists of the training and testing processes. The process by which a learning algorithm uses observations to learn a new model is called the *training* process, while the process by which the learnt model is examined on unseen observations is called the *testing* process [152]. During the training process, a learning algorithm learns from a collection of observations obtained from the problem domain, which is called the *training set*. The algorithm learns important knowledge or patterns from the training set by building models and adjusting the corresponding parameters. The performance of the algorithm is then evaluated on the *test set*, which is also a collection of instances in the same problem domain, but these are *not used* and remain *unseen* during the training process.

During the training process, another set of instances called the *validation* set might also take a part. A validation set is a set of instances, which are independent of the training set. The learnt models obtained from the training set are evaluated over the validation instances with various purposes. The validation set is widely to avoid overfitting and improve generalisation (these two important concepts will be introduced later). Learning algorithms will stop training when the error on the validation dataset increases, since this indicates overfitting to the training set. The validation set also can be used for model selection (i.e. selecting the model has a higher potential to generalise well on unseen data). The use of the validation set is often subject to the number of instances. It is appropriate to use validation set when there are a sufficient number of instances available for training.

2.1.3 Generalisation and Overfitting

Generalisation is an ability with which the learnt models can represent the true underlying pattern in the training data, and provide accurate predic-



(a) The difference between training and test errors becomes **larger**

(b) The difference between training and test errors **keeps the same**

Figure 2.1: Poor Generalisation when No Overfit Occurs

tion on new unseen data. In machine learning, generalisation is one of the most desirable properties for learning algorithms [152], since the variability of instances exists in real-world data. The generalisation performance of a learning algorithm is usually measured by the prediction error of its learnt models on an independent test set.

Overfitting is the contrary concept of generalisation which generally occurs when the evolved model performs well on the training set but poorly on the test set. According to [152], a more specific definition of overfitting is given as: given a hypothesis space H, a hypothesis $h \in H$ is said to overfit the training data if there exists some other hypothesis $h^{'} \in H$, such that h has smaller error than $h^{'}$ on the training set, but $h^{'}$ has a smaller error than h over the entire distribution of instances. Overfitting means poor generalisation. In order to gain best generalisation for solutions, detecting and avoiding overfitting is desirable.

Overfitting definitely leads to poor generalisation, however poor generalisation is not necessarily caused by overfitting only. In some cases, as shown in Figure 2.1, over a series of generations, the difference between the training error and the test error might become bigger (Figure 2.1 (a)) or keeps the same (Figure 2.1 (b)), which means the generalisation of learn models is unsatisfied but no overfitting occurs.

2.1.4 Feature Selection

Real-world data is becoming extremely high-dimensional as the data collection technology evolves these days. With these high-dimensional data, the ability of learnt models to extract meaningful information has decreased. Some other important impacts of high-dimensional data to learning algorithms are:

- The curse of dimensionality: the performance of a learnt model depends on the interrelationship among number of features, sample size and model complexity [113]. Searching for optimal models using various learning techniques breaks down with the phenomenon of *curse of dimensionality* in high-dimensional data. For some learning techniques, the requirement of the number of the training instances might be an exponential function of the features. Thus, the ability of these learning techniques to converge to an adequate model degrades rapidly as the number of features increases.

- High computational cost: because of the high dimensionality of the input space, the cost of measuring the performance of models increases rapidly. Thus, converging to a true or correct model will generally become computationally intensive.

- Model overfitting: a further challenge for building models on high-dimensional data is to avoid overfitting. When modeling on high-dimensional data, the model search space can be extremely large. Searching for an unknown model in such a huge space with finite instances runs a high risk of overfitting.

In addition to all these issues, learning from all features does not necessarily ensure good performance due to the existence of noisy/irrelevant features. Thus, *feature selection* is desired when learning on high-dimensional data.

Feature selection, also known as variable selection or attribute selection, is the process of finding a minimal subset of features that is sufficient and necessary to represent the problem to be solved. A feature selection algorithm explores the search space of available feature combinations to find the best feature subset. It is an important data preprocessing technique. Feature selection brings dimensionality reduction by eliminating the irrelevant and redundant features in the dataset. Therefore, it can enhance the learning performance and make the learning process more efficient. Moreover, models generated using fewer features have lower probability to be overfitting and easier to interpret.

2.2 Classical Regression and Symbolic Regression

2.2.1 Classical Regression

Regression, as one of the most popular machine learning tasks, is concerned with modeling the relationship between variables via measuring the error of the prediction model in an iterative way [192]. The goal of regression is to find the mathematical model that best fits the given data. A regression problem can be specified with a set of inputs which are independent variables X and a dependent variable Y that stands for the desired value. The objective of the task is to approximate Y using X and coefficients W such that:

$$Y = f(X, W) + \epsilon \tag{2.1}$$

where ϵ is a term for random error and represents the part of the data that is unable to be modelled by $f(X, W)$.

With classical regression techniques, the functional form f is predefined. For example, for the most widely used regression model — *linear*

regression, f would be:

$$f(X, W) = w_0 + w_1 x_1 + w_2 x_2 + ... + w_m x_m \tag{2.2}$$

where the coefficients (w_0, w_1, ..., w_m) can be found by *least squares*. Some other examples of classical regression techniques are:

- *Back-Propagation Neural Network* [110]: A neural network that is typically represented by a network diagram is a two-stage regression or classification model. The main idea is to extract linear combination of the inputs as derived features, and then model the target as a nonlinear function of these features by a transfer function (usually a sigmoid function). The generic approach to seeking values for the unknown coefficients (usually called *weights*) in neural network models by gradient descent, is called *back-propagation*.

- *Support Vector Regression* [61, 71] (SVR): *Support Vector Machines* (SVM) are a set of powerful modelling techniques. The original version of SVM was developed for binary classification. In [71, 209] various SVM versions for regression were proposed. In SVR, the regression equation can be specified as:

$$f(X) = \beta_0 + \sum_{i=1}^{n} \alpha_i K(x_i, w) \tag{2.3}$$

  where the corresponding x_i are *support vectors*, and $K(x_i, w)$ is termed *kernel function*. $K(x_i, w)$ can be linear and nonlinear functions.

- *Multivariate Adaptive Regression Splines* [85] (MARS): MARS is an adaptive procedure for regression. Like neural networks, MARS uses surrogate features instead of the original predictors. However, MARS produces two different version of a feature in the models. The idea is to form a reflected pair for each feature X_k with knots at each observed value x_{ik} of that feature. The collection of reflected pairs is

$$C = \{(X_k - t)_+, (t - X_k)_+\} \tag{2.4}$$

where $t \in \{x_{1k}, x_{2k}, ..., x_{ik}\}$, $k = 1, 2, ...p$. The model building strategy is like a forward stepwise linear regression. However, instead of using the original features, functions in C are used. Thus the model has the form

$$f(X) = \beta_0 + \sum_{j=1}^{J} \beta_j h_j(X) \tag{2.5}$$

where each $h_j(X)$ is a function in C. The coefficients β_j are estimated by standard linear regression.

- Least absolute shrinkage and selection operator [215] (LASSO): LASSO is a method for linear regression, which shrinks some coefficients in the regression models (to 0) to retain good features in linear models. LASSO minimises residual sum of squares. This procedure is subject to the condition that the sum of the absolute values of the coefficients less than a constant t, i.e. given N instance with the target outputs $Y = \{y_1, y_2, \ldots, y_N\}$, for linear models $f(X) = \sum_{j=1}^{J} \beta_j X_j$, LASSO estimates the $\beta = (\beta_1, \beta_2, \ldots, \beta_J)^T$ by:

$$\beta = argmin \left\{ \sum_{i=1}^{N} \left(y_i - \sum_{j=1}^{J} \beta_j X_{ij} \right)^2 \right\}, \quad \sum_{j=1}^{J} |\beta_j| \leq t \tag{2.6}$$

2.2.2 Symbolic Regression

Symbolic regression is a kind of regression analysis, which performs a function identification. The task of symbolic regression is to identify the input variables in a given dataset that are related to output variables, and express the relationship in mathematical models or symbolic descriptions. Symbolic regression is named to emphasise that the target of the identification process is a mathematical model, not only a set of coefficients of a predefined model (as in classical regression). This is a sharp difference with classical regression techniques in which a specific model is predefined.

Symbolic regression provides a way to gain insights into the data generating process. It does not require a predefined function and has the objective of generating a model, which is a combination of primitive functions, independent variables and coefficients, and minimising the error of this model regarding the differences from the desired outputs. During the learning/modeling of symbolic regression, there are some other important objectives. When performing symbolic regression, it is also important to exclude irrelevant and redundant input variables during this process. The number of coefficients and the value of these coefficients is another problem that needs to be tackled during the deriving of the model. There is no prior knowledge of the shape and size of the function. This is another characteristic of the deriving process.

Symbolic regression is based on the existence of EC techniques (which will be discussed in the following section). The idea of solving various problems by symbolic regression by means of EC comes from Koza [124]. However, non-EC methods [146] have been proposed for symbolic regression recently. The existing methods which can solve symbolic regression are including:

- Genetic Programming (GP): GP is an EC techniques and most popular method for symbolic regression. GP for symbolic regression is the main focus of this thesis, so it is discussed in the following subsection.

- Analytic programming[245] (AP), which was inspired by GP and Hilbert spaces, aims to address one particularly important issue in symbolic regression, i.e. how to represent a symbolic model. AP constructs regression models by manipulating the predefined function set and terminal set. However, instead of the direct representation used in GP, AP uses an integer index to represent the individual. Then it uses the idea of functional spaces and building of resulting function by means of searching process from Hilbert spaces to rep-

resent the regression models. Note that, AP is not an independent algorithm, i.e. to perform symbolic regression, AP needs to work with an EC algorithm.

- Artificial immune system [114] (AIS): AIS was inspired from the way in which natural immune systems learn to respond to attacks on an organism. When using AIS for symbolic regression, it works in a similar way as GP, where the programs are represented as LISP parse-trees (the same as the standard manner of GP [124]) and various components of the evolutionary process of GP are translated into the immune metaphor.

- Fast Function Extraction [146] (FFX): FFX is a non-EC technique for symbolic regression. FFX uses a recently developed machine learning technique, pathwise regularised learning [84], to rapidly prune a huge set of candidate basis functions down to compact models [146]. Compared with EC techniques, FFX spends less computational cost. However, its performance is highly related to the set of basis functions.

EC methods, particularly various GP methods, are still the most popular and well-study techniques for symbolic regression. GP based symbolic regression describes the data effectively by developing symbolic functions. It evolves data-driven models, which are useful for predicting the response values while facilitating human insight and understanding of the data generating process. In the following two sections, we will introduce EC techniques and the detail of GP.

2.3 Evolutionary Computation

Evolutionary Computation (EC) consists of various population-based algorithms that simulate different aspects of evolution. Theses algorithms are

mainly categorized to *Evolutionary algorithms* (EAs) and *Swarm intelligence* (SI).

Inspired by Darwinian principles and survival of the fittest, EAs are global optimisation methods with a stochastic character and can be distinguished by the population based solutions. According to the representation of the solutions and the evolutionary operators, EAs are classified to *Evolutionary Programming* [80], *Evolutionary Strategies* [24], *Genetic Algorithms* [111] and *Genetic Programming* [124]. These algorithms share one fundamental commonality in that they use the same iterative progress. This progress involves random variant, reproduction, and selection of fittest individuals in a population [16]. Many aspects of the evolutionary process are stochastic since the variant is randomly chosen and the selection operator can be deterministic or stochastic.

Swarm intelligence (SI) considers the design of intelligent multi-agent systems that are inspired by the collective behaviours of social insects such as ants, bees, as well as by other animal societies such as flocks of birds or schools of fish [26]. Two typical SI techniques in the literature are *Particle Swarm Optimisation* (PSO) [118] and *Ant Colony Optimisation* (ACO) [70]. PSO incorporates swarm behaviours observed in flocks of birds, swarms of bees, schools of fish [1]. It is distinguished by its fast convergence when compared with many other evolutionary algorithms, for example, genetic algorithms. ACO is inspired from the foraging behaviours of ants and is typically used to solve discrete optimisation problems. In ACO, the indirect communication by means of chemical pheromone trials enables artificial ants to find the shortest path between their nest and food.

The application and development of EC algorithms has been one of the fast growing fields in computer science. In studies related to both EC and machine learning techniques, many attempts have been devoted to make various evolutionary algorithms to be efficient and effective machine learning techniques [79, 95, 96]. Since this thesis is focused mainly on GP for symbolic regression, we will only review GP below in detail.

2.4 Genetic Programming

Genetic Programming (GP) is an evolutionary computation method that is inspired by biological evolution. The very first statement of modern "tree-based" GP was given by Nichel L. Cramer in 1985 [102]. However the work of Koza [124] in 1992 marked the beginning of the field of GP. In [124], many problems in various fields were addressed by GP in a way of automatically finding computer programs.

GP is a domain-independent approach which can automatically address problems without requiring the specification of the structure and the size of the solutions in advance. GP starts from a population of randomly generated programs. This population of programs evolves generation by generation to gain a better fitness. During this evolutionary process, GP stochastically transforms a population of programs into a new population and simulates evolution by some forms of fitness based on selection and breeding. During this process, the fittest programs are expected to be found and survive.

2.4.1 Representation

In standard GP, computer programs are represented in an abstract form — a parse expression. Typically a parse tree is used to represent a candidate program.

In order to generate and represent a population of programs, a function set F and a terminal set T need to be prepared beforehand according to the tasks. For regression tasks, F can include standard arithmetical functions such as *addition*, *subtraction*, *multiplication*, *division*, transcendental functions and trigonometric functions. Each element in F has a fixed number of arguments. For example, function "-" has two arguments and "sin" has one argument. The terminal set T usually consists of independent variables/features and a number of random constants.

The selection of function set and terminal set should satisfy the *suffi-*

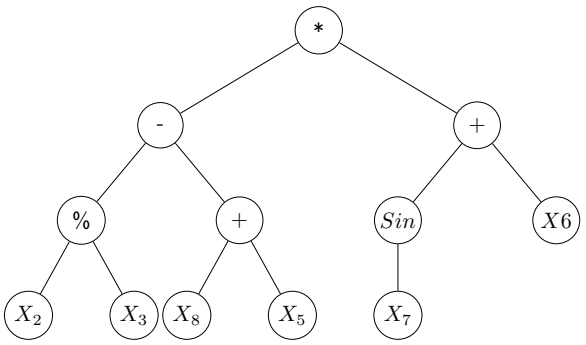


Figure 2.2: GP Tree for: $((x_2/x_3) - (x_8 + x_5)) * (sinx_7 + x_6)$.

ciency and *closure* property [17]. While the sufficiency property requires these two sets have enough expressive power to represent a solution for the problem, the closure property requires each function in F is capable to handle gracefully all possible inputs. When the terminal set and the function set have been prepared, candidate solutions are constructed from their elements. Fig. 2.2 describes a solution in GP by the parse tree.

In recent years, variants of GP using some other representations have been developed, such as grammatically-based GP [235] using a context free grammar and variants of linear GP using *linear* structures including *gene expression programming* [76] and *linear genetic programming* [30]. Although there are various ways to represent candidate solutions in GP, the tree-based representation is still the most popular one. Thus, this thesis focuses on tree-based GP.

2.4.2 Initialisation

In standard GP, initialisation of the population is the first step in performing a GP run. By using functions and terminals, it is possible to generate well-formed tree-like individuals in GP. There are several methods to build the trees.

Grow is the simplest one where a node is selected randomly from the function set or the terminal set before the maximum tree size or limited

depth is reach. In this way, the grow method can produce trees with various shapes and sizes.

In contrast, one another commonly used method — the *full* method only produces trees with a full size. The full method chooses nodes only from functions at the beginning when construction a GP tree. It does not select a terminal node until the maximum depth is reach.

In order to promote the population diversity of the initialisation, usually the *ramped-half-and-half* technique is employed by GP, which actually is a combination of the previous two methods. That is, in this method, half of the population is initialised with the full method and the rest half is produced by the grow method.

2.4.3 Evaluation

The main feedback to evolutionary algorithms is the performance measure of candidate solutions. The evaluation criterion of GP is called a fitness function. The fitness function is calculated on the training set. It should be designed to give a fine-grained differentiation between candidate solutions for GP. There are many ways to cast fitness functions. The most commonly used measures are:

- when applying GP for classification problems, *classification accuracy* and *classification error rate* are most widely used for performance measure. *Classification accuracy* is the number of correct predictions as a percentage of the total number of predictions. *Classification error rate* is the number of error predictions as a percentage of the total number of predictions.

- when GP is used for regression problems, generally *absolute error*, *squared error* and *scaled/normalised error* can be used. *Absolute error* calculates the sum of the absolute value with respect to the error between the given outputs and the desired values. *Squared error* is a

common alternative which calculates the sum of the squared difference. It usually has various forms such as *mean squared error* (MSE), and *root mean squared error* (RMSE). *Scaled/normalised error* refers to some functions that can amplify or damp smaller deviations from the desired outputs.

In GP, the fitness function is extremely important, since it is the primary mechanism in a high-level statement of the problem's requirements [184]. Moreover, the fitness measure is the main force of the evolutionary process in GP, as the breeding and survival of GP solutions are generally according to the fitness values.

2.4.4 Selection

After the performance of an individual has been evaluated, the selection operators should be used to give the better fitted solutions more opportunities to apply genetic operators. Choosing selection methods is one of the most important decisions in applying GP. Many standard evolutionary selection mechanisms can be used for selecting candidate parents in GP, such as *fitness-proportional selection*, *truncation selection*, and *tournament selection*. Note that these selection mechanisms are generally not greedy, i.e. the best individual in the population is not guaranteed to be selected and the worst individual in the population is not necessarily excluded and still has some chance (at a lower probability) of being selected.

The most commonly employed selection method in GP is perhaps *tournament selection*. In tournament selection, a number of individuals are sampled at random from the population. The number of individuals is determined by the tournament size. These individuals are compared with each other and the best of them is chosen to be a parent. For some genetic operators (e.g. crossover operator which will be introduced in the following sub-section), where two parents are needed, the tournament selection is performed for twice. In the selection process, the tournament size, which

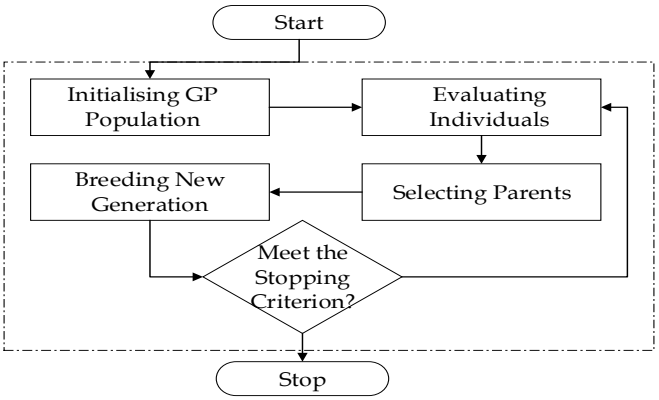


Figure 2.3: The Evolutionary Process in GP.

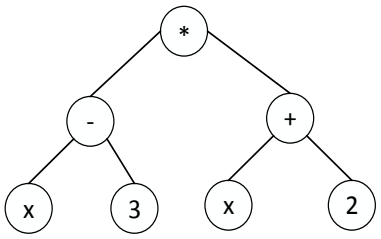
determines the number of candidate individuals to be sampled and compared, leads to different selection pressure. While a large tournament size causes high selection pressure, a small tournament size generally causes a low pressure.

2.4.5 Genetic Operators

Evolution proceeds by transforming parent to offspring by means of genetic operators. The three principal GP genetic operators are: *Crossover*, *Mutation* and *Reproduction*.

For *crossover*, two parents are selected based on the selection mechanism and a random subtree is selected in each parent. By swapping the two subtrees, two chair trees are generated. Traditional tree-based crossover do not need to consider the position information of the crossover points where the two subtree root at. Apart from this traditional crossover, various crossover operators have been proposed, such as the *homologous crossovers* [183], which preserve the position of genetic materials.

Mutation only operates on one parent. A random subtree of the parent is selected and replaced by a new subtree following the constraint of GP setting. Mutation used to be considered not as important as crossover. In



The program ***P* = *(x-3)*(x+2)***

Given a Training Set Consisting of **3** instance:
({4,5},{8,10},{12,15})
(X=4, 8, 12 and Y=5, 10, 15)

The semantic of the program is a **3** dimension vector:
{6, 50,126 }
where **6=(4-3)*(4+2),**
50=(8-3)*(8+2),
126=(12-3)*(12+2).

Figure 2.4: The Example of Calculating Semantics in a Regression Model

the early dates, Koza [124] did not use mutation and wished to show that mutation is not necessary for GP. However, later research found that mutation is also important for GP [47]. Various mutations have been proposed for GP [194, 227].

Reproduction is straightforward. It operates by placing the copy of a selected individual into the population. The commonly used version of reproduction is *elitism* where a number of the highest fitness individuals are copied into the next generation.

GP is a flexible methodology, allowing vigorous co-application of all kinds of strategies and meta-strategies, as long as the cycle (as shown in the inner loop of Figure 2.3) between solutions generation, evaluation ,selection, and variation is maintained [234].

2.5 Semantic GP

2.5.1 Semantics and Semantic GP

Semantic Genetic Programming (SGP) [20, 126] is a recently developed variant of GP that incorporates the semantic information of GP solutions. In GP, *semantics* refers to a description of what the GP solution does [126]. As this variant of GP will be used in this thesis, we briefly describe the basics in this section.

The formal definition of semantics in GP is domain-dependent. In symbolic regression, the definition of semantics is as follows [126]:

Definition 2. *The semantics of a program F is defined as a vector S. The elements of S are the corresponding outputs of the program with respect to a set of n fitness cases X, i.e. $S(F) = \{F(X_1), F(X_2), ..., F(X_n)\}$.*

Figure 2.4 shows how to calculate the semantics given the program and instances. Semantics can be calculated for any subprogram of a given program to describe its behaviour more thoroughly than just by its outputs, since a part of a program is also a valid program itself. The legitimate assumption under incorporating semantics into GP is that taking the detailed behavioural information of solutions into account can increase the effectiveness of GP. Another advantage of incorporating semantics into GP is that it is essentially free to obtain the semantics of a GP solution, since each solution has to run on training instances to access its fitness and the semantics is a side-effect of fitness measure.

2.5.2 Geometric Semantic GP

Geometric semantic GP (GSGP) [126, 156], as one particular branch of SGP, has recently been attractive. While SGP uses the semantics as a guideline for the evolutionary process and evolves toward a program with a satisfied evaluation, GSGP aims to manipulate the semantics directly and has the target of generating a program with (approximated) optimal semantics.

GSGP considers the semantics of a program as a point in an n (n is the number of instances) dimensional space. The semantics of all the candidate solutions in GSGP form a semantic space. In the semantic space, the evaluation of any point is the distance from the target semantics, i.e. the target outputs. Therefore, the surface of the semantic space takes different conic forms according to the distance metrics. More importantly, this conic space is unimodal, i.e. the minimum error can only be obtained at

the target point, and no plateaus exists. Searching in such an unimodal space is easy and promising *in principle*. However, it is not that easy *in practise*, since the program space instead of the semantic space is the space being searched. The move (swapping or replacing) on the programs does not correspond to the desired move in the semantic space.

2.5.3 Theoretical Framework in GSGP

The geometry of the semantics space is attractive for enhancing GP. However, searching directly in the semantic space is difficult. Therefore, GSGP provides a formal *theoretical framework* for designing geometric search operators [156]. The framework defines the desired semantic properties of the offspring generated by the geometric semantic operators. Specifically, the *geometry requirements* in the semantic space are 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).

Theoretical framework in GSGP [156] is defined as follows:

Definition 3. *Geometric Semantic Crossover (geometric crossover): Given two parent individuals* P_1 *and* P_2*, a geometric crossover generates offspring* O_j $(j\in1,2)$ *having semantics* $\vec{O_j}(j\in1,2)$ *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 4. *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.,* $\|\vec{O}-\vec{P}\|\leq r$

Based on this framework, various geometric semantic operators are developed to fulfil the semantic requirements/control in the literature, which will be reviewed later.

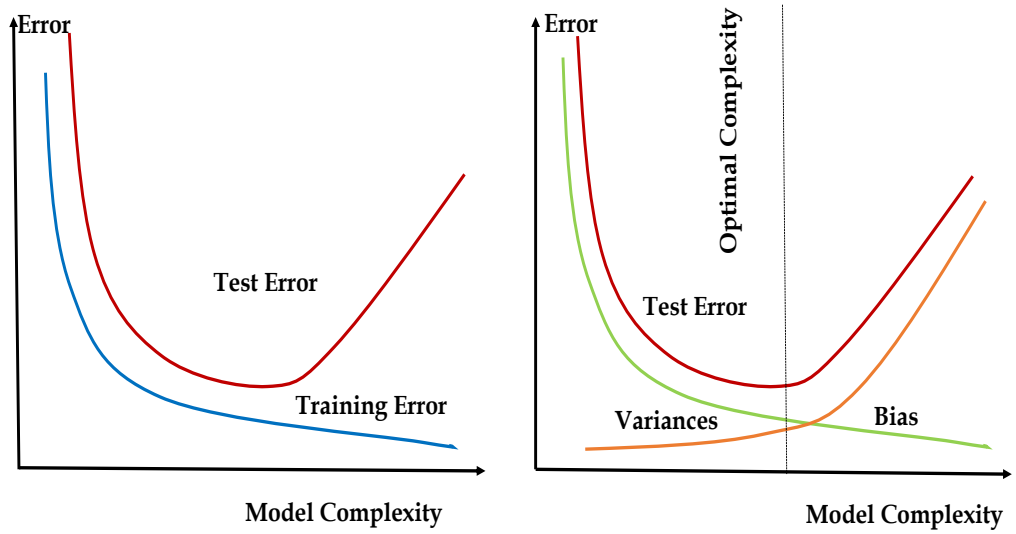


Figure 2.5: Bias-Variance Trade-off.

2.6 Elements of Learning Theory Related to Generalisation

2.6.1 Bias-Variance Decomposition

A widely accepted statement in machine learning is that simpler models are more likely to generalise well. A more accurate description is that to get an optimal representation of the underlying pattern of the given data, and hence to achieve a good generalisation on unseen data, it is necessary to obtain a trade-off between achieving an impressive training accuracy and obtaining a reasonably smooth model. The theoretical tool of bias-variance decomposition, i.e. a decomposition of learning error into bias and variance terms, provides insight into this trade-off. This tool comes from statistical learning theory and is well-known in machine learning [25, 69, 86, 92].

The bias error is a measure of the difference between the expected prediction of the model and the target output value. Due to randomness in the

underlying data sets, the learnt models will have a range of predictions. Bias measures how far these models' predictions are from the correct values. The variance error is taken as the variability of a model prediction for a given data point. The variance measures how much the predictions for a given point vary between different realisations of the model.

Figure 2.5 illustrates this bias-variance trade-off, where increasing model complexity has the effect of reducing bias in the model while increasing variance. There is an optimal "sweet spot" where bias and variance errors are balanced and relatively low, and hence generalisation error (in terms of bias and variance components) is minimised.

A lower variance error indicates that the models are less sensitive to the training data, thus can potentially generalise well on unseen data.

2.6.2 Vapnik-Chervonenkis Dimension and Structural Risk Minimisation

Statistical learning theory [231], particularly probably approximately correct (PAC) [221], defines Vapnik-Chervonenkis dimension (VC-dimension), which is a general measure for model complexity [230]. The original definition of VC-dimension is for indicator functions $\{I(X, \alpha)\}$, where X are the input vectors, α is a set of parameters and the outputs of $\{I(X, \alpha)\}$ take the values of 0 or 1. The VC-dimension h is the maximal number of input vectors X_1, X_2, $\cdots$, X_h that can be *shattered* by $\{I(X, \alpha)\}$ [232], i.e. no matter what the class labels are (in 2^h possible values), with proper parameters α, $\{I(X, \alpha)\}$ always can perfectly separate these vectors into two classes.

Later, this definition was extended for real-value functions $\{R(X, \alpha)\}$, where $A \leq \{R(X, \alpha)\} \leq B$. The VC-dimension of a set of real-value functions $\{R(X, \alpha)\}$ is defined as the VC-dimension of its corresponding indicator functions $\{I(R(X, \alpha) - \beta)\}$ [231], where $\beta \in (A, B)$. In statistical learning theory, the VC-dimension plays an important role in various up-

per bounds of generalisation error [83]. Structural risk minimisation (SRM) [231] defines one of these generalisation bounds.

SRM provides a powerful framework to estimate the generalisation performance of models. The practical forms of SRM are different for classification and regression, which are taken from [56, 231] and shown in following: SRM for classification [231] is:

$$R_{exp}(h) \leq R_{emp}(h) + \sqrt{\frac{h(\ln(2n/h)+1) - \ln(n/4)}{n}} \tag{2.7}$$

and SRM for regression [56] is:

$$R_{exp}(h) \leq R_{emp}(h) \left(1 - \sqrt{h/n(1 - \ln(h/n)) + \frac{\ln n}{2n}}\right)_{+}^{-1} \tag{2.8}$$

where h is the VC-dimension value and n is the number of training instances.

As shown in the definitions, SRM defines an upper bound of the generalisation error considering both the *empirical risk/error* and the *confidence interval*. The confidence interval estimates a difference between the empirical risk/error and the expected risk/error, and it is determined by the size of the training set and the VC-dimension value of the models. Since the size of training sets is usually fixed, the confidence interval is determined purely by the VC-dimension. Thus, the generalisation bound relying on the confidence interval is also named VC generalisation bound or VC bound [83].

The essence of the SRM principle is to minimise the generalisation upper bound. Therefore, the learning process under SRM, which tries to select the models having a good trade-off between the empirical error and VC-dimension (i.e. model complexity), can potentially lead to models with better generalisation ability.

2.7 Related Work

2.7.1 Improving GP for Symbolic Regression

GP is able to automatically create various programs. It does not require for the predefined shape and size of solutions. It has a strong expressive power. All these properties make GP the most suitable approach to symbolic regression. On the other hand, symbolic regression was one of the earliest applications of GP [124]. During the past several decades, GP has been remarkably successful in solving symbolic regression problems [38, 133, 105, 116, 122].

Although GP based symbolic regression has matured significantly in the last few years, there are a number of issues which require further research when applying GP to symbolic regression such as *premature convergence*, *limited scalability*, *generalisation*, *computational intensiveness*. In the past several decades, researchers aim to address these issues along with discovery of quality models. They proposed various approaches to these issues. Later we will have a brief review of these approaches.

A number of techniques have been proposed to make GP a more reliable and attractive approach to symbolic regression. A full catalogue of these approaches would be beyond the scope of this thesis. Here, some prominent ones are reviewed, particularly those related to this thesis.

Keijzer [116] proposed two relatively minor but promising modifications to GP based symbolic regression, *interval arithmetic* and *linear scaling*, which aimed to improve the predictive performance and reliability of the induced models. In that work, the commonly used protected operators were argued to have severe shortcomings in the vicinity of mathematical singularities. Thus, instead of the protected operators, interval arithmetic was used to ensure the induced models do not contain any undefined behaviour in their output ranges. Meanwhile, to make GP more effective, linear scaling was used prior to calculating the error measure. The scaling took the form of a simple linear regression that is to find the optimal

slope and intercept of the model against the target. The linear scaling calculates constants that otherwise need to be found during GP evolutionary process. This enables GP to concentrate on the search of optimal models with desired shape. A series of experiments were conducted on a number of regression benchmarks. A significant improvement on the training set was reported. However, linear scaling is not without apparent limitations. In one example of applying it to real-world applications [222], the findings suggested that the application of linear scaling may lead to overfitting. But it needs to be pointed out that the study led to this conclusion was based on experiments of only 20 independent runs.

Pennachin et al. [180] proposed a system similar to Keijzer [116], but used *affine arithmetic*, a more refined interval method that generates tighter bounds for solutions. In that work, interval arithmetic was argued to be too wide to be useful to eliminate the solutions with outside the desired range output. The results showed integrating affine arithmetic with the implementation of standard GP can significantly improve the effectiveness of GP and the generalisation ability of induced models.

Reducing Evaluation Cost

Since model evaluation is often a time-consuming process, various efforts have been made to reduce evaluation cost. These evaluation relaxation schemes include caching subtree, and partial fitness evaluation.

Keijzer [117] presented a number of subtree caching mechanisms where a cache of subtrees and their evaluations was maintained. By looking up the subtree evaluated previously, it can avoid the reevaluation of commonly occurring subtrees. Thus, the runtime of a GP system can be rapidly reduced.

Majeed et al. [145] applied the subtree caching mechanism [117] to the implementation of the context-aware crossover operator. They tested this approach on symbolic regression problems and confirmed that the use of a cache improves the performance of GP for symbolic regression by

dramatically reducing the number of node evaluations.

When calculating the fitness value of the candidate models, generally GP uses the whole training set for every individual in every generation. However, researchers have reported the improvement of performance when partial fitness evaluation is used.

Smits et al. [207] applied the selection schemes for partial fitness evaluation to symbolic regression problems. They applied three cases of partial fitness evaluation to three types of symbolic regression problems. These three cases are: constant subset size with constant population size, increasing subset size with constant population size and increasing subset size with decreasing population size. Based on the experimental results, they claimed that even when the subset size is down to 40% of the original size, GP can still produce models that were comparable with the counterparts obtained with the exhaustive evaluations at a lower computational cost. Later, a number of different selection schemes were proposed in a number of contributions [99, 100].

Counteracting Premature Convergence

Premature convergence occurs when the candidate models get stuck in a local optimal and the GP population rapidly converges to a very concentrated set of behaviours and consists of very few clusters of individuals that express the same phenotype. Moreover, no improvement in performance can be achieved in the successive generations. Since premature convergence poses such a threat to the performance of GP and is a well-known open issue in GP, a number of works have emphasised maintaining the population diversity during the evolution process as a means to counteract this issue [34, 35, 37, 74, 105, 148, 193]. These methods range from enhancing diversity on the genotype/structure of GP solutions [34, 148, 193], the phenotype/semantics of solutions [105, 167], to using niching techniques [37].

Ryan et al. [193] presented the pygmy algorithm implementing disas-

sortative mating. The pygmy algorithm requires the maintenance of two lists of individuals using two different fitness functions. The fitness function of the individuals in the first list is simply their performance. Individuals that were qualified to enter the first list are given a second chance by a slight modification of the fitness function to include a weighting of length. When choosing parents for breeding, individuals from these two lists are analogous to differing genders with the intention that the offspring receives the good attributes of each. The results showed that pygmy algorithm can improve convergence speed while evolving better solutions.

Ekárt et al. [74] extended fitness sharing to GP based symbolic regression by defining a distance function that reflected the structural difference of candidate trees. Fitness sharing regards fitness as a shared resource of the population and requires similar solutions to share the fitness values. By using fitness sharing, the population does not converge as a whole, but converges as several different niches. In that work, they defined a new metric that reflects the structural difference of the trees and applied the metric for applying fitness sharing. The results showed that the approach could obtain compact solutions as well as maintain the population diversity but no improvement on accuracy of models has been obtained. The main shortcomings of fitness sharing are that the computation of the shared fitness for the whole population in every generation might be very time consuming and the difficulty in determining the niche radius.

Nguyen et al. [167] modified the implementation of fitness sharing in order to speed up its execution. More important, they investigated the effect of two difference types of metric: semantic and syntactic distance in fitness sharing. Experimental results of that work showed that while fitness sharing with the semantic metric often remarkably improves the performance of GP, fitness sharing with the syntactic metric can hardly improve the performance. At the same time, when applying fitness sharing with the semantic metric to GP, the computational cost in terms of time was almost the same of standard GP and much faster than its syn-

tactic counterpart. However, the change of semantic diversity or syntactic diversity in the population during the evolution process has not been reported.

Gustafson et al. [105] proposed a simple improvement to GP which forced the genetic operators to always use parents with different fitness values. The core of that work was rooted on the fact that the probability of no change in solution fitness value increased with the similarity of solutions. By forcing the mating of solutions with different fitness values, a significant performance improvement was shown in the training phase.

Burke et al. [34] investigated six different measures of diversity in GP for symbolic regression including *genotype, phenotype, entropy diversity, preudo-isomorphs, edit distances 1 and 2*. They conducted a series of experiments to find the relationship between these measures and fitnesses. They found that compared to other problem domains of GP, symbolic regression has the weakest correlation between any measure of diversity and fitness overall. However, entropy and edit distance diversity showed comparative stronger correlation with fitness.

Burks et al. [36, 37] proposed an effective genetic diversity technique which is called the genetic maker diversity algorithm to enhance the diversity of the GP population while maintaining a good search ability. Their method relies on tree fragments as genetic makers and prevents the population from converging to a single structure by taking genetic marker density as one objective of GP as well as the regular fitness value. The proposed method was tested on various tasks including symbolic regression and has shown to perform significantly better than standard GP in terms of learning performance and maintaining a more sustainable search in GP.

2.7.2 Improving Generalisation of GP for Symbolic Regression

For GP, as an important learning algorithm in machine learning, a good generalisation ability is desired. Despite the importance of generalisation for any learning algorithm, the generalisation of GP did not receive the deserved attention for quite a long time in the past, let alone GP for symbolic regression. In the past, symbolic regression had been token as an optimisation issue and all the available data were used for training. Before Kushchu published his work on study the generalisation ability of GP [130], there was little work on the research on generalisation of GP for symbolic regression. However, in the past several years, an increasing number of studies have been devoted to enhance the generalisation ability of GP. These studies can be mainly classified into two categories, controlling bloat [45, 46, 88, 206] and tackling overfitting [42, 51, 54, 97, 100, 112, 170].

Controlling Bloat

Bloat is one of the most well-known problems in GP. It is defined as rapid code growth without corresponding improvement in fitness. It was one of the main areas on GP. Methods controlling bloat to improve the generalisation ability of GP solutions are in light of theories such as the *Occam's razor* [27] and the *Minimum Description Length* [18]. Based on these theories, researchers had a common agreement that bloat and overfitting are two related phenomenons. Also it is widely accepted that shorter solutions can generalise better. However, in recent years contributions found that overfitting occur even when bloat is eliminated and vice versa [41, 67, 143].

In [67, 143], the Tarpeian method, which can successfully control bloat, has shown to be unable to significantly improve generalisation performance of GP solution. In [41], a bloat-free technique named operator equalisation can not have much positive effect on controlling overfitting. Therefore, it is suggested in these contributions that bloat and overfitting

may be two separate phenomenons and should be tackled by separate methods.

Avoiding Overfitting

Validation Set to Detect and Avoid Overfitting: One of the commonly used approaches to detecting overfitting is to use a validation set.

Gagné et al. [88] investigated the effect of validation sets. In their work, instead of selecting the best one solution, the best n solutions on the training set were selected. Among these solutions, the best solution on the validation set was returned as the final solution. The effort of the validation set on improving the generalisation of GP was compared with that of a lexicographic parsimony pressure. It was found that the validation set brings a little benefit while the parsimony pressure has a negative effort on the generalisation. However, more benefits were achieved when using the combination of these two methods.

Smits et al. [208] extended the use of a validation set. They maintained an archive of Pareto optimal solutions in terms of validation fitness and model size. The advantage of this method is that it is possible to select a satisfactory solution *a posteriori*. The key issue of the validation method is that its effectiveness highly depends on the instances in the validation set. These instances should contain a representative sample of all possible instances.

Evolving towards Smoother Models: Much recent work counteracts overfitting in GP by introducing various strategies to evolve towards simpler/smoother models.

Chan et al. [46] incorporated a statistical method, backward elimination (BE), into GP to eliminate insignificant terms in polynomial models. In their work, the parameters of the evolved polynomial models were determined by BE, which eliminates the insignificant terms one-by-one. As a result polynomial models, which have a fewer number of terms but only

significant terms, can be produced. The BE-GP was shown to be effective on reducing overfitting.

Haeri et al. [107] proposed variance-based layered learning GP, which decomposes the evolutionary process into several hierarchical layers. From lower to higher, these layers were trained using different training sets, from less to more complex. The complexity of the training sets is measured by the variance of the output values. This measure is also applied to evaluate the complexity of candidate models. The experimental results showed that the layered GP enhanced the generalisation ability of GP while reducing the model complexity.

Mousavi et al. [157] presented a multi-objective GP (MOGP) method to enhance the generalisation by controlling the first order derivative of GP models. In MOGP, in addition to pursuing a lower training error, the RMSE on the first order derivative of the candidate model (i.e. measuring the difference from the first order derivative of the target model), which measures the model complexity, is considered to be the other objective. The experimental results showed that MOGP had better generalisation gain than standard GP.

Bias-Variance Decomposition in GP: Typically the variance error in GP is estimated by Bootstrap methods [73]. Various GP methods incorporating bootstrap techniques have been proposed [3, 77, 81]. On these methods, the GP population is trained on a list of bootstrap samples. The variance error is estimated according to the error on the bootstrap sample and individuals with a lower variance error are selected.

Fitzgerald et al. [77] claimed that introducing bootstrap into GP has the effect on improving both generalisation and learning performance, as well as producing more impact solutions. However, the effect of the estimation by bootstrap might decrease when the number of instances is relatively small due to the overlap of training instances and bootstrap instances.

Kowaliw et al. [123] investigated the bias-variance decomposition in

linear GP and analysed factors that are important in obtaining this trade-off, e.g. program size, the initialization of the population and the function set. In their work, the bias and variance error of GP were approximated by the averaged response of 50 independent GP runs. They found that the variance of GP is largely due to the random seeds and the three examined factors are all related to the trade-off. A reasonable adjustment of each of these values led to significant improvements in obtaining the trade-off. Moreover, the change of the function set has the largest gain on the trade-off and the larger function set was consistently preferred. They found that increase the size of the function set does not necessary increase the variance error. This is against the finding in [78], i.e. a simple function set is more likely to evolve solutions with a smaller variance error.

Restricting Model Complexity in GP: In the past, many researchers in the GP community considered smaller solutions to be simple and able to generalise better. Rosca et al. [190] investigated generality versus size in GP. They concluded that smaller programs tended to generalise better but that methods which can control the effective size are more likely to be beneficial in promoting generalisation than approaches which apply a general size penalty. The results indicated that small, generalisable solutions might be difficult to find when a simple size penalty was applied. More recently, researchers found that, instead of shorter solutions, behaviourally/functionally simpler solutions are more likely to generalise better [78, 217, 224, 233]. The idea of quantitatively studying the relationship between generalisation ability and model complexity have been approached by several works.

Vladislavleva et al. [233] proposed two measures of model complexity: one is genotype and the other one is phenotype. While the genotype one measures the complexity of models by counting the number of nodes of a GP tree, the phenotype one which is named *order of nonlinearity* measures the functional complexity of models by calculating the degree of the *Chebyshev polynomial approximation* of it. In that work, these two kinds

of complexity measures were used as independent objectives separately in two multi-objective GP approaches. The results showed that the two methods can control bloat and overfitting simultaneously. The drawback of the order of nonlinearity lies in the difficulty of calculating the degree of the Chebyshev polynomial approximation of models.

Vanneschi et al. [224] proposed a measure of functional complexity of solution inspired by the concept of *curvature* [39]. They simplified the calculation of the curvature and gave a definition to measure the complexity of a partial feature and the complexity of a model is the average of all the partial complexities on all the dimensions of the feature space.

Castelli et al.[42] improved the measure of curvature by proposing another complexity measure, called *graph based complexity* (GBC) measure, where the curvature of solution can be measured by counting a number V, which refers to the number of pairs of close training data point X_i and X_j for which the corresponding output value of the model y_i and y_j are very different. More specifically, given the distance metric for calculating the distance between two instances X_i and X_j and a predefined constant value δ, when this distance is within the ball of radius δ centred on a given instance X_i, the two instances are considered to be close. Otherwise, they are far away from each other. The same measure is used to determine whether their corresponding outputs are far or close. Then the complexity of a model is express by the ratio of V over the total number of close instances without considering the outputs. They measured the GBC of the best-of-generation models and showed that GBC has a positive correlation with the generalisation performance. They have also shown that incorporating the number V into the fitness function is able to improve the generalisation of GP in some cases.

Fitzgerald et al. [78] investigated the effect of operator complexity together with a control on solution size on the generalisation ability of GP. In their study, GP employed a combination of various function sets with different levels of complexity and various limitations on the depth. They

found that a simple function set was more likely to evolve solutions with a similar training and generalisation performance while a complex function set was prone to guide the evolutionary process to overfitting. Another important finding is that solutions which generalise well may tend to be small and simple but the evolution of these solutions may be more successful in an environment which facilitates medium to large programs.

There are a number of model complexity measures. However, not many of them are shown to related to the generalisation of GP and the mathematical foundation of these methods is not solid enough. A more comprehensive investigation on the relationship between model complexity and overfitting or the underlying relationship between model complexity and generalisation error needs to be investigated.

2.7.3 Structural Risk Minimisation on Enhancing Generalisation

Cherkassky et al. [57] investigated the effect of SRM on controlling the model complexity of linear models. In their work, a comparison between SRM and various model selection methods such as final prediction error [8], Bayesian Information Criterion [23], Shibatas model selector [204] and generalized cross-validation [63], was conducted. It showed the superiority of SRM on selecting models with a better generalisation performance and smaller complexity. However, their finding was limited to linear regression models.

Implementation of the SRM Principle in Learning Algorithms

Two kinds of approaches have been found to implement SRM directly into learning algorithms.

The first approach is to keep the empirical error fixed and minimise the confidence interval. The design of support vector machines (SVMs) [232] follows this rule. Grounding on the VC theory, SVMs have been proposed

and developed over decades [29, 232]. The original SVMs were extended to support vector regression (SVR) for regression tasks [232]. SVR maps data into a high-dimensional input space through some nonlinear mapping, and its kernel functions and parameters are selected to minimise the VC generalisation bound. Via regularisation operators, the kernel function in SVR 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 aims to keep the confidence interval fixed and try to minimise the empirical error. This strategy is widely used in neural networks [87, 202]. 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 is to find the weights to minimise 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 trade-off between underfitting and overfitting. A lot of research has been conducted to estimate a more accurate VC bound for neural networks [131, 163, 198].

Implementing SRM in GP

Amil et al. [13] presented a theoretical analysis of GP from the perspective of statistical learning theory and highlighted the advantage of a parsimonious fitness using VC-theory. However, the practical implementation of SRM into 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 trade-off 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.

[28] and [154] are the only work that can be found before our initial work [51]. In their work, SRM was introduced as a new fitness function

to GP for symbolic regression. The VC-dimension of the evolved models was measured by a simplified estimator, which counted the number of non-scalar nodes (i.e. nodes that are not being operated by the functions $\{+,-\}$) in a GP tree. They compared GP equipped with SRM with GP methods incorporated with two classical statistical model selection strategies, Akaike Information Criterion [8] and Bayesian Information Criterion [23]. The comparison demonstrated the advantage of SRM in enhancing the generalisation performance of GP. However, the relationship between the number of non-scalar nodes and the VC-dimension of the model needs further investigation.

In summary, the conceptual contributions and practical significance of VC-dimension and SRM are not yet fully appreciated. Furthermore, compared with a rough approximation, measuring the VC-dimension of the evolved models through a well-designed *experimental*[1] method is more reliable and desired.

2.7.4 Feature Selection Approaches

Categories of Feature Selection Approaches

The search strategy and the evaluation criterion are two core factors in designing a feature selection approach. Feature subsets are produced according to the search strategy, while the produced subsets are evaluated and compared under the evaluation criterion.

Based on evaluation criteria, feature selection approaches are generally classified into three categories: filter methods, wrapper methods and embedded methods [106, 242]. Filter methods [179, 243] select a subset of features based on kinds of criteria such as mutual information. Wrapper methods [6, 103, 120] use a learning algorithm as a black-box and select subsets of features based on the performance of the learning algorithm.

[1]The word "experimental" was used to emphasise that the method is not a theoretical estimation.

Embedded methods [104, 169, 241] incorporate the feature selection process within the learning process. Not many techniques can perform embedded feature selection. Decision tree [187] is one of these typical techniques, and among current EC technique, only GP and learning classifier systems (LCSs) have this ability [158, 160]. Some work treats embedded methods as a kind of wrapper methods. The main difference among these methods is that wrapper and embedded methods include a learning algorithm in the feature subset evaluation step. Filter methods usually are independent of any learning algorithm, thus they are more general and less computationally expensive. However, the performance of the selected features on a learning algorithm, which is ignored by filter methods, often can bring notable benefits to wrapper methods and make them perform much better than filter methods.

We group feature selection methods into non-EC techniques and EC techniques, which cover a wide range of ideas. It is impossible to cover all of these varying approaches. In the following two sub-sections, the machine learning approaches for feature selection, which are closely related to this thesis, are highlighted.

Non-EC Techniques for Feature Selection

A kind of feature selection method in this category is inspired from decision tree inducing algorithms such as ID3 [185], C4.5 [187], C5.0 [186], classification and regression trees (CART) [32]. Decision trees are hierarchical structures where each internal node n implements a decision function $fn(x)$, and each leaf corresponds to a region for regression (a class for classification). Feature selection approaches based on these decision trees are straightforward. They aim to obtain a feature ranking according to the feature/variable importance score. In these decision trees, the variable importance is computed in two ways. The first one is calculating the percentage of training examples falling into all the terminal nodes after the split of the variable. However, it is biased to features in the early

split nodes. The other one tries to avoid this bias by taking the percentage of splits that a variable is used into consideration as well when assigning importance score.

Sugumaran et al. [212] investigated the use of decision trees (constructed by C4.5) to identify the important features for SVM for classification. It was shown that the decision tree is able to identify important features, i.e. the values of these features with minimum variation within a class and maximum variation between classes, and SVM can perform well when fed with the selected features.

Cho et al. [58] presented the feature selection ability of decision trees using two kinds of tree models: C4.5 and CART. In their work, the feature selection ability of these two kinds of decision trees was compared with principle component analysis (PCA) [236] and variable selection proposed in [33] on improving the classification accuracy of the multilayer perceptron (MLP) and the fuzzy ARTMAP networks[238]. The decision tree has a superior feature selection ability since the best classification accuracy in both classification methods was achieved based on the features selected by CART.

Regarding some slightly more complex techniques, ensembles of decision trees play an important role in feature selection. The random forest (RF) method [31] is an ensemble learning algorithm which constructs a forest of decision trees and provides solutions for classification and regression tasks. Further, RF also provides two different feature importance measures, which can be utilised for feature selection. One measure is *gini importance*, which is derived from the training of decision trees in the forests, and the other one — *permutation importance*, which is motivated by the concept of permutation in statistics.

When constructing the decision trees, RF performs an implicit feature selection and use a subset of features to build the trees, which leads to superior performance on high-dimensional data. The outcome of the implicit feature selection is named Gini importance in RF. The value of Gini

importance provides a ranking of features. In RF, at each node in a decision tree, the best split is obtained from a set of randomly selected features according to their Gini impurity. Adding up the Gini decreases for each feature over all trees in the forest gives the Gini importance.

The permutation importance is related to another randomness in RF, which is in the examples/instances. Each tree in the forest is constructed using a different bootstrap sample from the original whole set of examples. Usually, about one third of the examples are left out of the bootstrap and not used for constructing the tree. These examples are called out-of-bag (OOB) examples. The concept of permutation importance of the features is measured over the out-of-bag (OOB) examples. For each variable in the decision tree, the values of the variable in the OOB examples are permuted (randomly rearranged within the OOB examples). Then each permuted example is passed down the tree. The total increase of the regression error is defined to be the variable importance. The bigger value means the variable is more important.

Many previous studies on feature selection using Gini importance [101, 151] and permutation importance [159, 203] have shown better learning and prediction performance.

EC Techniques for Feature Selection

EC techniques have been used for feature selection in recent years. Such methods including genetic algorithms (GAs), particle swarm optimisation (PSO) and GP. EC techniques do not require domain knowledge and make no assumption of the search space. Moreover, EC techniques are population-based, which generally produce multiple solutions in one run. All these characteristics make EC techniques more attractive than traditional methods for feature selection in many cases.

GAs are the very first EC technique that has been applied to feature selection [242]. One of the very early work was published in 1989 [205]. The representation of a binary string in GA is a natural way to represent the

selection of features, where 1 means the corresponding feature is selected and 0 refers to not being selected. A number of enhancements have been proposed to GA to improve its feature selection ability [93, 136, 244].

PSO [119] has also shown to be a promising technique for feature selection. There has been rapid development on PSO for feature selection. Similar to GAs, PSO has an advantage for feature selection, i.e. a straightforward representation. In PSO, a bit string is typically used for the representation of a particle. For binary PSO, the bit string uses binary numbers, where 1 means the feature is selected and 0 means not. In continuous PSO, the bit string uses real-values. The selection of a feature is determined by a threshold. Compared with other EC-techniques, PSO has a velocity term that facilitates a faster convergence, and spend less computational cost. All these advantages make it a preferable technique for feature selection. However, PSO is not free from challenges. A number of new PSO methods have been proposed to enhance its performance on feature selection [59, 239, 164, 216].

However, GAs and PSO has a common limitation, i.e. it is difficult for the representation of GAs and PSO to scale well on problems with thousands or more features due to a huge search space. GP has a potential to handle large-scale feature selection since the representation does not have to include the selection information of all features. Moreover, many real-world problems contain a large number of features but a small number of instances (e.g. gene selection). Feature selection on such data is a challenge not only in machine learning but also in traditional domains such as statistics. GP might solve this challenge since it is able to handle data with a relatively small number of instances [9].

As the features appearing in the evolved individual can be treated as a set of selected features, GP is considered to have the built-in feature selection ability. The built-in ability of GP in detecting important features by exploring the feature space has made it a valuable method for feature selection. A number of different GP-based feature selection methods have

been proposed in the literature [161, 196, 7, 109]. However, compared with GAs and PSO, there is a much smaller number of works on GP for feature selection.

Neshatian et al. [161] developed a Pareto GP for feature selection in classification tasks. They designed a function to measure the relevance of subsets of features. A Pareto front archive was maintained, which consists of non-dominated subsets of features having a low cardinality (i.e. the number of features the subset contains) and high relevance. They also adopted methods to avoid bloat and overfitting to allow GP to explore large subsets of features. The experimental results showed that the feature selection method can improve the classification accuracy while decreasing the complexity of the evolved classifiers. However, the proposed method might have some limitations when the cardinality of desired best subset of features is high.

Muni et al. [196] proposed a GP-based feature selection method to address the skewed/unbalanced high-dimensional classification tasks, which combined multiple most commonly used feature selection metrics. The results indicate that the method can bring dimension reduction as well as increase the classification accuracy.

Moore et al. [155] introduced a nonlinear gene-gene (feature-feature) interactions measure based on information entropy into their Pareto GP system for genetic analysis of diseases. The many-objective GP system uses three objectives (classification accuracy, model size and the interaction measure) to guide the search towards models including features that are risk factors for the disease. The GP system is claimed to be able to find accurate models despite the size and complexity of the feature space.

Nag et al. [160] proposed a many-objective GP method for simultaneous designing of classifiers and feature selection. Their new GP method minimised three objectives, which are false positives, false negatives, and the number of terminal nodes in the GP trees. In this many-objective GP method, several filter based feature selection approaches are used in dif-

ferent stages of the evolutionary process. They defined a concept of fitness and unfitness of features, which is based on an index to measure the discrimination ability of each feature. This index is according to the Pearson's correlation between the values of the feature and their ideal values (i.e. a vector with values of the labels). Then during mutation, roulette wheel selection was performed on the unfitness to remove a feature, and on the fitness to insert a feature. In addition, during the process of obtaining the fitness and unfitness of the features, several thresholds are set to filter the features with a low discrimination ability. Compared with a bi-objective GP method which does not use any feature selection method and traditional classification algorithms (e.g. Naive Bayes, C4.5, lazy learning algorithms) with various feature selection methods, their method has a much better classification accuracy on a majority of the examined datasets.

Mei et al. [150] proposed a niching-GP feature selection framework for designing job-shop scheduling rules. In this framework, the niching techniques, more specifically — the clearing method [181], is employed to adjust the fitness of GP solutions. This aims to reduce poor solutions in density areas and promotes population convergence to different local optima. At the end of the evolutionary process, the best solutions in each niche are collected. These best solutions are treated as the source of important features. A quantitative measure of the relevance of a feature is obtained according to its specific contribution to one specific best solution and the fitness of this solution. Features are selected according to this value. A comparison between the test performance of the dispatching rules evolved by GP using the selected features, the entire set of features, and the best-so-far features found in previous research on the examined data has been conducted. The effectiveness of their proposed feature selection method has been confirmed.

We recently proposed a method namely *genetic programming with feature selection* (GPWFS) in [50]. GPWFS is a two-stage feature selection method for high-dimensional symbolic regression. It splits the evolution-

ary process of GP into two phases by a parameter G_f. The major task of the first phase is feature selection. On each generation of this phase, all the distinct features appearing in the top β percent individuals are collected, since these features are considered to be candidates for important features. At the end of the first phase, a set of potentially important features F_c is formed. The second phase is the standard evolutionary process on a population of reinitialised individuals. On the first generation of the second phase, GPWFS reinitialises the population by keeping the top β percent individuals while replacing the rest. The replacement will take the form of an equal number of randomly generated individuals using a new terminal set formed by the set of selected features F_c. The effectiveness of GPWFS on enhancing the generalisation of GP was investigated and confirmed in [50]. However, GPWFS needs to tune two key parameters G_f and β, which are problem dependent and sensitive to the parameter settings of GP, such as the population size and the total number of generations. Moreover, GPWFS treats all the features appearing on the best models the same regardless of their position. This might limits the accuracy of the feature selection process. More details can be seen in [50].

In summary, much work has been devoted to GP-based feature selection to improve the classification performance. However, for regression tasks, especially for high-dimensional regression, more attention is deserved. There is no existing work in this field (before this thesis) and feature selection is desired to improve the regression performance and generalisation of GP when the dimensionality is high.

2.7.5 Geometric Semantic GP Methods

Semantic GP Methods

SGP methods generally use semantic information to design or guide genetic operators in GP, however there are various ways to utilise semantics in GP.

Nguyen et al. [165], which is a very early work in SGP for symbolic regression, developed a new crossover operator with semantic-awareness, which is called semantics aware crossover (SAC). SAC was based on checking for semantic equivalence of subtrees and attempted to increase the semantic diversity. To determine the semantic equivalence of two subtrees, they are evaluated on a random set of instances sampled from the domain. If the output of the two subtrees on this set of sampled instances are close enough (subject to a parameter — semantic sensitivity), then they are considered to be semantically equivalent and allowed to swap. Although the idea of SAC is novel, when compared with standard crossover, it has limited improvement on some test problems.

Later, Nguyen et al. [220] proposed another semantic crossover operator named semantic similarity based crossover (SSC) based on SAC. Different from SAC that checks the semantic equivalence, this new semantic crossover selected subtrees for crossover by checking semantic similarity and aimed to improve semantic locality. Compared with standard crossover and SAC, SSC has been shown to be superior on both the training and test performance. The idea of SSC was then extended to mutation and led to a semantic similarity based mutation (SSM) [166, 219], which has much better performance than standard mutation. Further analysis showed that the fitness landscapes induced by SSM on two well-studied measures were significantly smoother than those of standard mutation on a number of symbolic regression tasks.

Implementation Algorithms of GSGP

The implementation of geometric operators can be grouped into two categories, i.e the *exact* geometric operators [156] and the *approximated* geometric operators [126, 128, 178]. 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 pro-

grams directly, *guarantee* the geometry of the offspring in the semantic space. An implementation method of the geometric crossover and mutation was proposed in [156] to guarantee the geometry in semantic spaces. 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 a 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 [223], 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 interpret.

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 behaviour, approximating it seems to be more sensible. Based on this assumption, various approximated geometric operators have been developed. The approximated geometric operators are working in the genotype space and *approximating* behaviours geometrically in the corresponding semantic space. The approx-

imating geometric operators such as locally geometric semantic crossover (LGX) [128], approximately geometric semantic crossover (AGX) [127] and Random Desired Operator (RDO)(mutation) [178] do not lead to overgrown offspring, since they generally rely on various mechanisms to approximately satisfy the semantic requirements.

Krawiec et al. [128] proposed LGX that approximates the geometric recombination of parents at the level of homologously chosen subprograms. More specifically, given two parents $p1$ and $p2$, LGX calculates a syntactic common region of them, which is performed by simultaneously descending both parent trees from their roots until reaching nodes of different arities. Then, LGX uniformly selects two homologous nodes n_1 and n_2 in the common region of the two parents. A desired semantics of the subprograms that approximate the geometry of child programs is calculated by the mid-point in the segment of the two subprograms. Finally, a library of programs is searched for a new subprogram that minimises the semantic distance to the desired semantics. The child programs are generated by replacing the old subprograms that are rooted at the selected homologous nodes with the new subprograms.

Krawiec et al. [127] proposed an approximate geometric crossover (AGX). In AGX, the desired semantics of the offspring is defined to be the midpoint on the segment of the two parents. Semantic backpropagation (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 updated 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. [178] further developed SB and proposed a geometric mutation operator named Random Desired Operator (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 on improving the generalisation of GP.

Nguyen et al.[168] proposed a subtree semantic geometric crossover (SSGC), which relies on subtree semantic similarity to approximate the geometric property. In their method, given two parents and a random generated number r, if r is smaller than a predefined threshold, then SSGC performs. Otherwise, standard crossover executes. When performing SSGC, the selection of crossover point is according to the metric of selecting the subprogram having the most similar semantics with that of the parent program. The number of candidate subprograms is determined by a predefined parameter. Two child programs are then generated by a convex combination of the subprograms. They compared the new geometric crossover with the exact geometric crossover, AGX and RDO on a set of

symbolic regression tasks and the superiority of SSGC over the exact geometric crossover on both the training and test performance has been reported. Furthermore, this superior performance was achieved with much less computational time compared to AGX and RDO. The major limitation of SSGX lies in tuning the two key parameters determining the number of candidate subprograms and the rate to perform SSGX. The performance of SSGX highly depends on these two parameters.

Semantic-awareness in Other Fundamental Components of GP

Recently, researchers began to explore the effect of introducing semantic-awareness in different fundamental parts of the evolutionary process to enhance the performance of GP. These methods mainly cover the population initialisation [21, 176] and the selection process [89].

Beadle et al. [21] proposed a semantic driven initialisation (SDI) method to generate a population of individuals with a high diversity and to produce effective starting programs. The results have shown that, compared with traditional initialisation methods, SDI is more effective.

Pawlak et al. [176] developed a new geometric semantic initialisation method, which generates a population of individuals to make sure the convex hull spanning these individuals covers the target semantics. Their experiments showed that compared with Ramped Half-and-Half, their new initialisation method is more likely to guarantee the success of GSGP in fitting the target semantics.

Galvan-Lopez et al. [89] introduced a two-mate selection operator called semantic tournament selection (STS). Given a first parent selected by standard tournament selection, STS samples the population t times and selects the best of the t sampled individuals that have different semantics from that of the first parent. If none of the sampled individuals has different semantics, a random one is returned. The main advantage of STS is to discourage semantic duplicates, and hence to maintain a high semantic diversity. The performance of GP employed STS is statistically better than

that of standard GP.

However, Szubert et al. [213] assumed that in some cases maintaining a high diversity in semantics is conflict with the target of approximating the target semantic, since methods promoting behavioural diversity tend to increase distance between behaviours while the fitness function typically rewards minimising distance to the target behaviour. Consequently, promoting diversity can result in spreading individuals over the semantic space and slowing down the convergence of the search process. They defined two metrics to measure the semantic diversity of the population, which are an Euclidean semantic metric and an angular semantic metric. The first one measures the novelty of a program as a mean Euclidean distance between its semantics and semantics of its k nearest neighbours in the semantic space, while the later measures the novelty of a program by a mean angle between its residual vector and residual vectors of its k ($k = 15$) nearest neighbours. Multiobjective GSGP methods, which have one objective of minimising the Euclidean distance from the target semantics and the other objective of maximising the semantic diversity, have been examined. They claimed that maximising Euclidean diversity scatters the semantics of the population away from the target semantics, hence leads to a worse learning and generalisation performance. But the angular semantic metric does not conflict with the fitness and can improve the performance of GP on both the learning and generalisation performance. However, these two metrics are only tested on locally geometric crossover, but no other geometric operator has been involved.

Generalisation in GSGP

Not much work has been devoted to investigate the generalisation of GSGP. However, in this small number of studies, introducing semantic-awareness into GP methods has been shown to have a positive effect on promoting its generalisation [98, 49, 52, 218].

Uy et al. [218, 220] proposed new semantic-aware crossover operators,

which imposed different kinds of requirement 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 generalisation of GP.

Gonçalves et al. [98] compared the generalisation ability of GSGP methods using different settings, i.e. GSGP methods employing sole geometric crossover, geometric mutation, and bounded geometric mutation [223]. They claimed that geometric crossover contributed little to improve the generalisation ability, while geometric mutation had a good effect on promoting the generalisation. They also claimed the benefit of GSGP to the generalisation ability is brought by the bounded geometric mutation, which produced a small variation to the offspring.

In summary, different from traditional methods that improve the generalisation of GP by evolving structurally simpler programs, GSGP methods can potentially enhance the generalisation of GP by controlling the semantics of the programs during the evolutionary process. In GSGP, the semantic-awareness can potentially drive GP search in a smoother fitness space and the geometry of the semantic space makes GP search more effectively, both of which are important for a better generalisation gain in GP.

2.8 Summary

This chapter reviewed the main concepts and learning tasks in machine learning including generalisation. Symbolic regression, which is the focus of this thesis, was introduced by comparing with classical regression. Then a brief introduction of evolutionary computation was presented. A detailed description of all the components of GP was presented. We also discussed the strength and challenges in GP for symbolic regression in detail.

To highlight the related work, this chapter also discussed some recently proposed GP approaches for symbolic regression. Although generalisation has been recognised as an open issue in GP just for several years [173], an increasing number of works have been done to enhance generalisation of GP solutions. In this chapter we also discussed the mainstream of these methods. The limitation of the existing works that form the motivation of this thesis were also discussed. Summaries of these limitations are as follows:

- Feature selection is desired for high-dimensional data in order to reduce the impact of the curse of dimensionality and the risk of overfitting. However, most of the research is for classification. There is not much research on feature selection to improve the performance of GP for symbolic regression.

- The relationship between overfitting and solution complexity deserves further research. Reducing the complexity of GP solutions is a widely used method to control overfitting, but the measures of model complexity are lack of solid theoretical foundation. Meanwhile estimating the generalisation error is seldom considered to be an approach to controlling overfitting in GP.

- Methods to enhance the ability of GP on generalisation focus mainly on counteract overfitting. In some cases, where there is no overfitting, these methods might loose their advantages on improving the generalisation of GP. This issue has rarely been considered yet.

The following chapters of this thesis will show how we can employ GP to tackle these issues.

Chapter 3

GP with Feature Selection for Symbolic Regression

3.1 Introduction

Feature selection in symbolic regression (SR) is a process of identifying a subset of relevant features (input variables) that are necessary to describe the output variable(s). When learning from high-dimensional data, feature selection is desired. Much work has been devoted to feature selection [158, 160, 161]. However, most of the existing work is for classification tasks. GP for symbolic regression (GPSR) seldom considers feature selection, and it is even rare when GP tackles high-dimensional symbolic regression tasks. The underlying reason is that GP has a kind of built-in feature selection ability when exploring the feature space to create symbolic regression models. However, the built-in feature selection ability of GP is typically not strong enough for high-dimensional regression tasks. Moreover, when learning an unknown model for SR in a high-dimensional feature space, GP runs a high risk of overfitting, which leads to a poor generalisation capability. Feature selection, as a data preprocessing method, can remove noise and irrelevant features. It has the potential to reduce the risk of overfitting, thus promotes the generalisation ability of GP. How-

ever, not much work on feature selection to improve the generalisation of GP has been proposed for high-dimensional regression to date. This chapter aims to fill this gap by utilising the built-in feature selection ability of GP in a better way.

3.1.1 Chapter Goals

The goal of this chapter is to propose a new feature selection method to improve the generalisation ability of GPSR, particularly when learning from high-dimensional data. More specifically, this chapter indicates four research objectives as follows:

- whether and how feature selection can influence the learning ability of GP regarding the training performance,
- whether and how feature selection can enhance the generalisation ability of GP for high-dimensional regression tasks,
- whether the new feature selection method can select the truly relevant features for high-dimensional symbolic regression, and
- whether the new method can outperform GPSR methods with common feature selection schemes.

3.1.2 Chapter Organisation

The reminder of the chapter is organised as follows. The second section describes the new feature selection method. The third section describes the experiment settings. The results and analysis are presented in the fourth section. It is followed by the final section, which summarises this chapter.

3.2 The Proposed Method

This chapter proposes a new feature selection method, which is named *genetic programming with permutation importance* (GPPI). It is based on GPWFS, which is our previously proposed method in [50], but it is superior to GPWFS in several aspects.

GPWFS [50] is a two-stage GP method for high-dimensional symbolic regression, which is equipped with an embedded feature selection approach. It splits the evolutionary process of GP into two phases according to the number of generations. The major task of the first phase (i.e. the first G_f generations of the evolutionary process) is feature selection. On each generation of this phase, all the distinct features appearing in the top β percent individuals are collected, since these features are considered to be candidates of important features. At the end of the first phase, a set of potentially important features F_c is formed. The second phase (i.e. from the $(G_f + 1)$ generation) is the standard evolutionary process on a population of reinitialised individuals. On the first generation of the second phase, GPWFS reinitialises the population by keeping the top β percent of individuals while replacing the rest. The replacement takes the form of an equal number of randomly generated individuals using a new terminal set formed by the set of selected features. The effectiveness of GPWFS on enhancing the generalisation of GP was investigated and confirmed in [50]. However, GPWFS needs to tune two key parameters G_f and β, which are problem dependent and sensitive to the parameter settings of GP, such as the population size and the total number of generations. More details can be seen in Appendix A.

We intend to overcome these potential limitations of GPWFS and further develop a feature selection method for GP of high-dimensional SR. The new feature selection method — GPPI and GPWFS share the same assumption that GP can explore the search space to detect important features automatically. We assume that features which appear in highly fit GP indi-

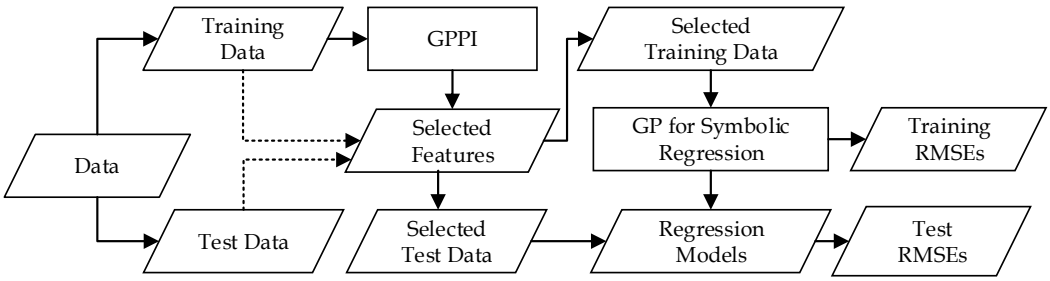


Figure 3.1: Data Flow Diagram for GP-GPPI.

viduals are more likely to be relevant to the output feature/variable, even though not all these features are relevant. Thus, they possibly form a candidate set of important features for feature selection. However, GPPI has a significantly different way to determine the importance of the features from GPWFS. Moreover, while GPWFS is a GP method for regression with an embedded feature selection phase, GPPI is a feature selection method for preprocessing the data for GPSR.

3.2.1 The Overall Structure

Figure 3.1 shows the data flow diagram for the new GPSR system named *GP-GPPI*, which introduces a feature selection component *GPPI*. The training process in GP-GPPI consists of two sequential phases. Firstly, GPPI is applied to the training data to select a subset of important features. Then standard GP evolves regression models on the selected training data with only the selected features. The key component of the new GPSR system is GPPI. GPPI improves GPWFS in two aspects. The first is the definition of the *good* individuals, which are the source of important features. GPWFS treats certain top individuals with the highest fitness values from generations in one GP run as the *good* individuals. Instead, GPPI collects the best-of-run individuals from a number of GP runs. Compared with their counterpart in GPWFS, the *good* individuals in GPPI are sufficiently evolved and contain important features by utilising the natural feature

selection ability of GP. The second aspect lies in the determination of important features. GPWFS collects all the distinct features presented in the good individuals as the relatively important features. GPPI computes a quantitative importance value for each distinct feature. Feature selection are based on the metric of importance values. The details of GPPI will be presented in Section 3.2.3 and 3.2.4.

3.2.2 Fitness Function

In this work, the fitness function in both the feature selection process and the GP for SR training process is *Normalised Root Mean Square Error* (NRMSE), which evaluates the performance of individuals for feature selection and regression. The definition of $NRMSE$ is given in Equation (3.1).

$$NRMSE = \frac{RMSE}{Y_{max} - Y_{min}} \tag{3.1}$$

where the term $(Y_{max} - Y_{min})$ is the range of the target variable and $RMSE$ is the root mean square error. The definition of $RMSE$ is shown in Equation (3.2).

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(f(X_i) - Y_i)^2} \tag{3.2}$$

where N is the number of instances, $f(X_i)$ is the output of the model and Y_i is the target output.

3.2.3 Permutation Feature Importance

As mentioned above, not all the features appearing in the highly fit individuals are important. Thus, the crucial component of the feature selection method is a quantitative measure of feature importance, which can tell the difference between the appearing features. Feature importance can be defined as the correlation between the feature and the output variable, or the extent to which the feature can contribute to reducing the error between the outputs of the model and the target values.

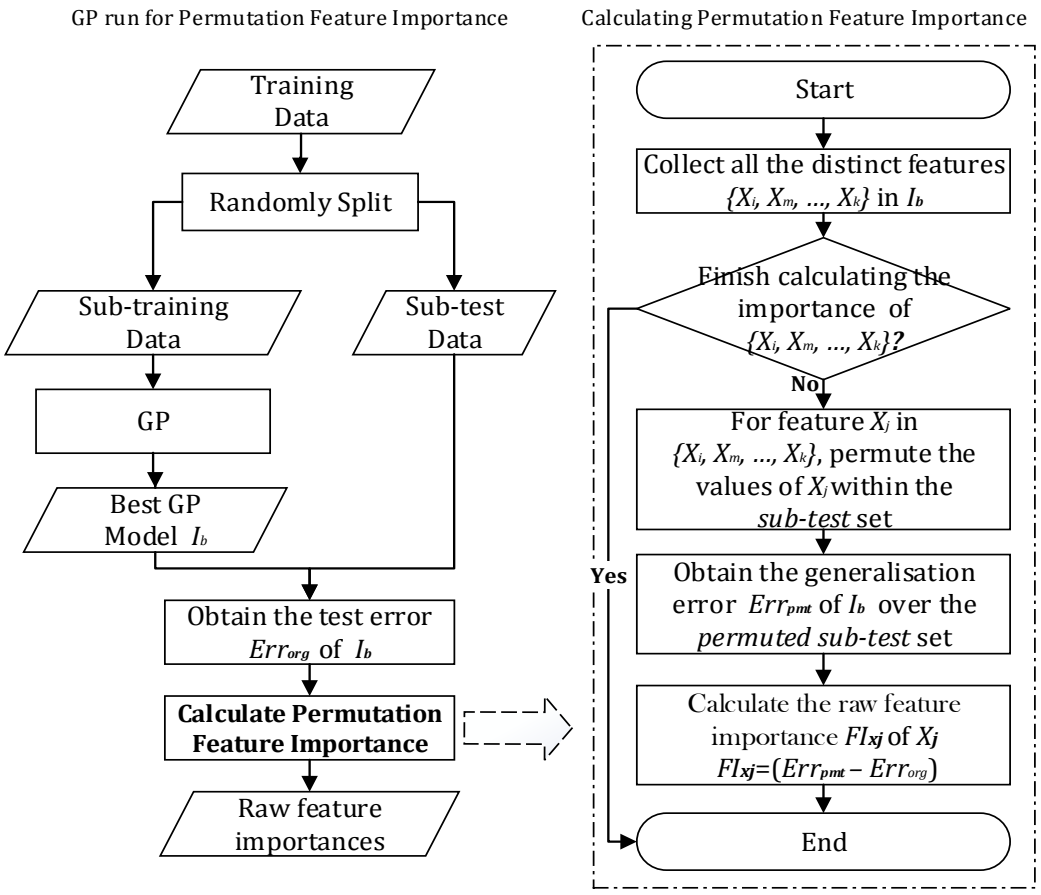


Figure 3.2: GP for Permutation Importance (GPPI).

Permutation feature/variable importance in random forests (RF) is a widely used score to measure the importance of features [48]. Permuting a feature refers to rearranging the values of the feature within the dataset randomly. For example, if the values of a feature are denoted as {4,7,9}, the permutation of the feature can take a random form among {4,9,7}, {7,4,9}, {7,9,4}, {9,7,4}, {9,4,7}. The rationale behind permutation importance is that important features should have a higher influence on the performance of models, i.e. for regression problems, permuting a more important feature will lead to a higher regression error. Based on this hypothesis, we measure the feature importance in GP for SR based on permutation. Note

that Dick et al. [68] presented the very first work for introducing the idea of permutation into GP. [68] aims to examine the data quality of a specific dataset. This involves using the permutation method to measure the usefulness/importances of features. Different from [68], GPPI is a *feature selection* method based on the importance of features, which is obtained by permutation method.

Figure 3.2 presents the main process of GPPI. While the left part of the figure shows the process of calculating permutation feature importance in one GP run, the right part describes the process to obtain the permutation importance of one feature. The whole process is defined as:

1. Randomly split the training data into a sub-training set and a sub-test set.

2. Carry out a GP search on the sub-training set and get the best-of-run model I_b.

3. Compute the generalisation error of I_b on the sub-test set, which is referred to $Err_{org}(I_b)$.

4. Collect the distinct features in I_b, which form the candidate feature set $X = \{X_i, X_m, ..., X_k\}$.

5. Generate a permuted sub-test set for each X_j in X by permuting its values within the sub-test set.

6. Calculate the test error of I_b on the permuted sub-test set, i.e. $Err_{pmt}(I_b)$.

7. Measure the raw feature importance of X_j by the distance between $Err_{org}(I_b)$ and $Err_{pmt}(I_b)$, i.e.

$$FI_{raw}(X_j) = Err_{pmt}(I_b) - Err_{org}(I_b) \tag{3.3}$$

Steps 5, 6 and 7 need to be performed for each distinct feature in the best-of-run individual I_b. It is important to note that the importance values

Algorithm 1: Permutation Importance of Features

Input : The number of GP runs n, the total number of features m, an empty array for feature importances $FIs[n][m]$;
Output: all feature importances $FIs[m]$;
for $g := 1$ *to* n **do** *GP run loop*
  Randomly split the training set into a sub-training set (70%) D_{ta} and a sub-teset set (30%) D_{te};
  Run GP on D_{ta};
  Select the individual I_b with $min(Err_{ta})$ on D_{ta};
  Calculate the Err_{org} of I_b over D_{te};
  Collect all the distinct features $\{X_i, ..., X_k\}$ in I_b;
  for $j := 1$ *to* k **do** *Raw Importance loop*
    Shuffle/permute the values of X_j ($\in \{X_1, X_2, ..., X_k\}$) within D_{te}, to form the permuted sub-test set D_{pte};
    Compute the Err_{pmt} of I_b over D_{pte};
    Calculate the raw importance value $FI_{raw}(X_j) = Err_{pmt}(I_b) - Err_{org}(I_b)$;
    Put $FI_{raw}(X_j)$ into FIs, i.e. $FIs[g][j] = FI_{raw}(X_j)$;
  end
end
for $f := 1$ *to* m **do** *Scaled Importance loop*
  Obtain the mean raw importance of X_f by $\overline{FI_{raw}(X_f)} = \sum_{i=1}^{n} FIs[i][f]/n$;
  Obtain the standard deviation of X_f by $\delta(X_f) = \sqrt{\sum_{i=1}^{n} \left(FIs[i][f] - \overline{FI_{raw}(X_f)}\right)^2 / n}$;
  Calculate the scaled importance value of X_f by $FI_{sca}(X_f) = \overline{FI_{raw}(X_f)} / \left(\frac{\delta}{\sqrt{n}}\right)$;
end

of features absent from I_b are defined to be 0. The whole process repeats n ($n \geq 30$) times on the best-of-run individuals collected from n independent GP runs on the given training data, respectively. The condition $n \geq 30$ is to reduce the bias of random seed used in GP runs on the importance values. The pseudo-code of this procedure is shown in Algorithm 1.

In order to make the feature selection according to the importance values to be flexible and problem-independent, the final importance of a fea-

ture is defined as the scaled importance, which is the average raw feature importance normalised by the standard error of raw feature importance. It is given as:

$$FI_{sca}(X_j) = \frac{\overline{FI_{raw}(X_j)}}{\frac{\delta}{\sqrt{n}}} \tag{3.4}$$

where n is the number of GP runs, $\overline{FI_{raw}(X_j)}$ is the average value of the raw feature importances in n GP runs, δ is the standard deviation, and $\frac{\delta}{\sqrt{n}}$ is the standard error.

3.2.4 Feature Selection according to Permutation Feature Importance

In GPPI, features with a positive value of $FI_{sca}(X_j)$ are selected. The positive importance value indicates that these features have a positive effect in reducing the regression error, and they are potentially more important than their counterparts with negative importance values. The main advantage of this selection metric is *problem-independence*, which is especially suitable for regression tasks without any domain knowledge. This metric does not depend on the problem domain and the specific number of features. The feature selection criterion, which selects features with positive importance values, is expected to help remove noisy and irrelevant features effectively while keeping the relevant features.

Instead of employing the whole set of features, the regression training process in GP-GPPI uses the set of positive features selected by GPPI. A population of programs are initialised and evolved using these positive features. Thus, GP-GPPI is expected to reduce the risk of overfitting and potentially improve the generalisation performance of GP for high-dimensional SR.

3.3 Experiment Design

To demonstrate the feature selection ability of GPPI and investigate its effectiveness on promoting the generalisation ability of GPSR, sets of experiments have been conducted. A comparison has been made between our new GPSR system GP-GPPI and GPSR with several various feature selection methods.

3.3.1 Benchmark Methods

A comparison between GP-GPPI and five benchmark methods, which employ feature selection for GPSR, has been conducted in this chapter. The first two benchmark methods are *GP-Random Forests* (GP-RF) and *GP-C5.0 decision trees* (GP-C5.0). The other three methods are our previous method GPWFS and two variants of GPWFS. The details of the five benchmark methods are as follows:

- *GP-Random Forests (GP-RF)* is a GPSR method using features selected by random forest (RF). The permutation feature/variable importance values in RF are obtained from 30 runs using 30 different seed numbers. This setting can reduce the influence of the random seed on the importance of features. Bootstrap samples are exposed to construct the trees, and the out-of-bag samples, which are not used for training the forest, are used to calculate the permutation importance.

- *GP-C5.0 decision trees (GP-C5.0)* is a GPSR method which employs C5.0 for feature selection. The feature importances in C5.0 are also obtained from 30 runs using the same sub-training sets as GPPI. When calculating the importance of a feature, the metric, which considers the percentage of splits that the feature makes, is employed.

- *GPWFS* is a GPSR method for simultaneous feature selection and regression. A brief description of GPWFS has been given in Section 3.2. For more details, readers are referred to [50].

- *GPWFS1* is GPWFS using a different setting. GPWFS1 differs from GPWFS in the number of generations for the two stages. The first stage of GPWFS1 has the same number of generations that GPPI uses for feature selection. The number of generations in the second stage is the same as that GP-GPPI used for symbolic regression. GPWFS1 is examined to make the comparison between GP-GPPI and GPWFS using the same number of generations.

- *GPWFS2* is a variant of GPWFS. It splits GP for feature selection and GP for SR into two separate stages. At the end of the feature selection stage, GPWFS2 collects all the distinct features appearing in the best-of-run individuals of 30 GP runs. Then the regression will perform on these selected features. Actually, GPWFS2 differs from GP-GPPI only in lacking the permutation method to determine the importance of features.

In addition to these feature selection methods, standard GP, which uses the whole set of features as input and performs built-in feature selection, is also used for comparison. However, it is used as a baseline for comparison. The main focus of this work is on the comparison among GP with various feature selection methods. In each GP for regression method, 100 independent GP runs have been conducted. All the GP methods are implemented under the GP framework of the evolutionary computation framework written in Java (ECJ) [142]. RF and C5.0 for feature selection are implemented under the R packages, which are "randomForest" [137] for RF and "C50" [129] for C5.0.

For a more comprehensive comparison, we also compare GP-GPPI with two statistical regression methods, least absolute shrinkage and selection operator (LASSO) [215] and RF for regression, which were considered to be the most commonly used methods for regression. LASSO performs feature selection by employing an ℓ_1 penalty to shrink some coefficients in the regression model to be 0. In this way, LASSO can effectively enhance

the prediction performance of the regression model. The two methods are implemented under R packages "glmnet" [84] and "randomForest" [137] with default settings. Furthermore, we investigate the influence of the computation load on GP-GPPI, and conduct a comparison between GP and GP-GPPI under the same computation time.

3.3.2 Parameters

The parameters for all GP methods are summarised in Table 3.1. The number of generations for GPWFS1 is 100 (50 generations for the first stage and the other 50 generations for the second stage). For GPWFS, the two key parameters, which are the number of generations G_f to decide the splitting point of the two phases and β to define the percentage of top individuals, are tuned using three different values, respectively. Since the total number of generations is 50, G_f is properly set to $25, 30, 35$. β takes the values of $5\%, 10\%, 15\%$. Thus, 9 $(3 * 3)$ different settings of GPWFS have been conducted. For GPWFS1, the value of G_f is fixed (G_f=50). The value of β in GPWFS1 is tuned among $5\%, 10\%, 15\%$. Thus, 3 different settings of GPWFS1 have been conducted.

In each run of RF for feature selection, a forest of 500 trees is built. The number of randomly chosen candidate features for each node is defined to be $m/3$, where m is the total number of features in the dataset. The values are recommended for regression [31]. The same settings are used in RF for regression. In C5.0, the parameter "metric" is set to "splits", which means the percentage of splits associated with each feature will take a part in calculating the feature importance. Other parameters use the default values. The feature selection criterion in C5.0 and RF is the same as GPPI, which is selecting features with positive importance values. As mentioned earlier, we assume these features have the potential to reduce the regression error. Thus, they are more important and should be selected. Compared with only selecting the top features, it is expected to reduce the risk of

Table 3.1: Parameter Settings

Parameters	Values	Parameters	Values
Population Size	512	Generations	50 (100 for GPWFS1)
Crossover Rate	0.9	Mutation Rate	0.1
Elitism Rate	0.01	Maximum Tree Depth	10
Initialisation	Ramped-Half&Half	Initialisation Depth	2-6
Function Set	$+$, $0-$, $*$, $Inv(\frac{1}{x})$, sqrt	Terminal Set	Features (Selected Features) , Random Constant $\in [-1.0, 1.0)$
Fitness Function	NRMSE		
GPWFS (GPWFS1) — G_f	25, 30, 35 (50 for GPWFS1)	— β	5%, 10%, 15%

missing some important features (particularly when the top features are redundant).

We have also investigated the influence of computation load on the results of GP-GPPI. This is conducted in forms of comparing GP and GP-GPPI under the same computation time and performing GP-GPPI in a higher computation load than the setting in Table 3.1. In this set of experiments, the population size of GP-GPPI is increased to 1024, while other parameters are the same as shown in Table 3.1. Standard GP has a population of 2048 and will be terminated when it uses the same computation time as GP-GPPI (the time including both feature selection and regression).

3.3.3 Datasets

In this chapter, GP methods with feature selection are tested on six high-dimensional regression datasets. While two of the datasets are synthetic data, the other four are real-world high-dimensional data. A high level of noise and the known relevant features make the two synthetic datasets particularly suitable for examining the ability of a feature selection method [211]. The functions of the two synthetic datasets are shown in Table 3.2. F_1 is the famous Newton's Law of gravitation, where g is the gravitational constant with the value of $6.67408E-11$. F_2 was taken from [116]. The sampling strategies for the training data and the test data are also shown

Table 3.2: Two Synthetic Functions

Functions	Training Samples	Test Sample	Noise
$F_1 = -g\frac{X_1 X_2}{X_3^2}$	70 points $X_1, X_2 = rnd[0, 1]$ $X_3 = rnd[1, 2]$	30 points $X_1, X_2 = rnd[0, 1]$ $X_3 = rnd[1, 2]$	50 input variables $= rnd[0, 1]$
$F_2 = \frac{30X_1 X_3}{(X-10)X_2^2}$	1000 points $X_1, X_3 = rnd(-1, 1)$ $X_2 = rnd(1, 2)$	10000 points $X_1, X_3 = rnd(-1, 1)$ $X_2 = rnd(1, 2)$	

in Table 3.2. The noise, which was added to each dataset deliberately [211], consists of 50 input variables with random values in the range $[0, 1]$. The purpose of adding noise is to check whether the feature selection methods can eliminate noise and select the truly relevant features.

The four real-world regression datasets are taken from UCI Machine Learning Repository [138] and previous literature on the generalisation of GP for SR [224, 226]. They are high-dimensional regression datasets with hundreds to thousands of features. Feature selection is more desired for these datasets than their counterparts with a smaller number of features. The first dataset LD50 is about the pharmacokinetics, the task of which is to predict the value of a pharmacokinetics parameter — the median lethal dose (represented as LD50). It has been used in much recent work on the generalisation of GP [223, 224, 226]. The second dataset is the Diffuse Large-B-Cell Lymphoma (represented as DLBCL), which was collected by Rosenwald et al. [191]. The task is to predict the survival time of patients who have diffuse large-B-cell lymphoma and received chemotherapy. The remaining two datasets are taken from UCI [138]. They are about communities and crimes within the United States, the Communities and Crime unnormalised dataset (CCUN) and the Communities and Crime normalised dataset (CCN). Both of them are to predict the per capita crimes. We discarded the instances which have missing values, so the number of instances in CCUN and CCN used in this work is smaller than in the original data.

Table 3.3: Benchmark Problems

Name	# Features	#Total Instances	#Training Instances	#Test Instances
F_1	53	100	70	30
F_2	53	11000	1000	10000
LD50	626	234	163	71
DLBCL	7399	240	180	60
CCUN	124	1994	1395	599
CCN	122	1994	1395	599

3.3.4 The Training Sets and the Test Sets

In this work, each dataset is split into a training set and a test set to investigate the generalisation performance of the evolved models in GP. The numbers of features, training instances and test instances of the six datasets (including two synthetic datasets) are shown in Table A.2. Four of the six datasets (expect DLBCL and F_2) are split with 70% of instances randomly selected from the training sets and the other 30% instances form the test sets. This is a widely accepted way of splitting the dataset in machine learning [98, 226]. The training set and test set of DLBCL are provided [191]. The numbers of training data points and test data points are given for F_2 [116].

It is important to note that, during the feature selection process, the data used by all the feature selection methods is *only* the training sets. The test set of each task is kept to be unseen during the feature selection and the model training process. Therefore, a *fair* comparison on the effects of the feature selection methods on the generalisation of GP has been conducted. In each GP for feature selection run, each training set is further split, where 70% randomly selected instances forms the sub-training set and the other 30% forms the sub-test set to obtain the feature importance. The same sub-training sets are used in $C5.0$. RF uses the whole training set by bootstrapping a number of samples (i.e. sub-training set) for constructing the trees and obtaining the permutation importance of features on the out-of-the-bag samples (i.e. sub-test set).

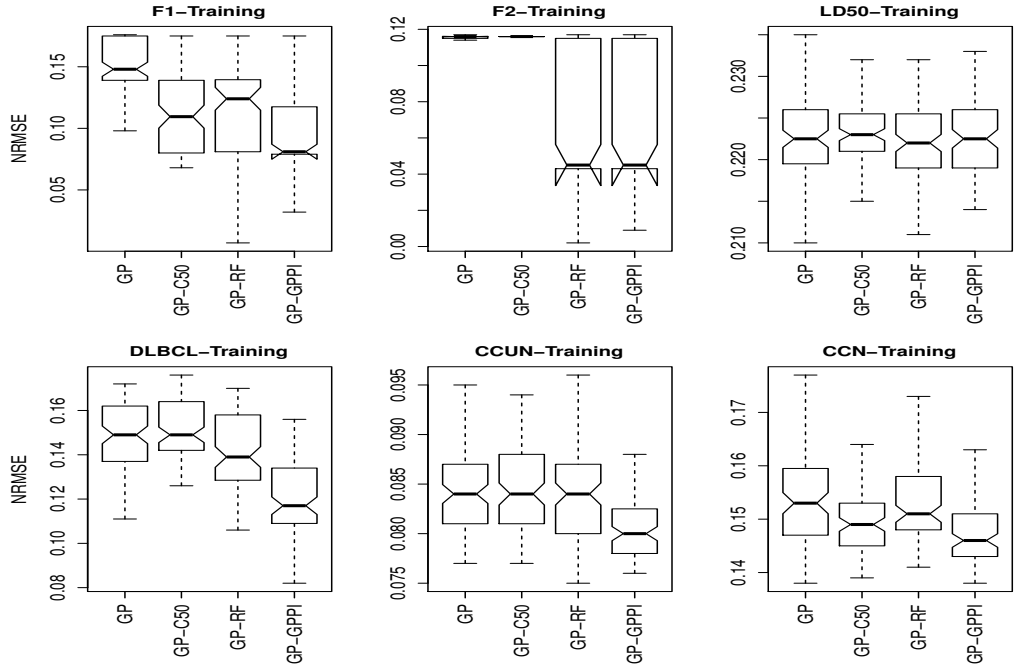


Figure 3.3: Distribution of **Training** NRMSEs of the 100 best-of-run individuals.

3.4 Result Analysis

3.4.1 Comparing GP-GPPI with GP-C5.0 and GP-RF

A comparison between GP-C5.0, GP-RF and GP-GPPI is presented in this section, which focuses mainly on the influence of the three feature selection methods (C5.0, RF and GPPI) to the learning ability and generalisation of GP for SR. Standard GP is used as a baseline for comparison.

Figure 3.3 shows the distribution of training NRMSEs of the 100 best individuals from the 100 GP for regression runs on the six training sets, while Figure 3.5 presents the distribution of their corresponding test NRMSEs. Each boxplot consists of four whiskered boxes for the four GP methods, respectively (On F_2, the notched boxplots in GP-RF and GP-GPPI look different from others, because the first quartile is too close to the median

of the results).

Figure 3.4 and Figure 3.6 show the evolution plots. On every generation, the lowest NRMSEs obtained by the best-of-run individuals on the training set are recorded, and the corresponding test NRMSEs of these best individuals are also obtained (the test errors serve to examine the evolution of generalisation, but are never taken into account during the evolutionary process). For the 100 independent GP runs, the 100 lowest training NRMSEs and the corresponding test NRMSEs are collected on every generation. The evolution plots are drawn using the median values of these 100 NRMSEs. Since the median value is suggested to be more robust to outliers [98], it is preferred over the mean value in this work. The Wilcoxon test, which is one of non-parametric statistical significance tests, is conducted to compare the 100 training NRMSEs and test NRMSEs of the 100 best-of-run models. Two sets of Wilcoxon tests with a significance level of 0.05 have been performed. The first set is between GP-GPPI and the other three methods, and the second is between GP and GP-RF and GP-C5.0, i.e. GP with GP-RF and GP with GP-C5.0.

Results on the Training Sets — Learning Ability

To investigate the effect of the feature selection methods on the learning ability of GP for high-dimensional regression tasks, the regression performance regarding the training NRMSEs is reported here.

As shown in the training boxplots in Figure 3.3, the difference between the median of NRMSEs in GP-GPPI and the other three methods are large on four of the six datasets, i.e. F_1, DLBCL, CCUN and CCN. On these four datasets, the boxes of GP-GPPI and other three methods overlap but not the median values. It indicates GP-GPPI has much better training performance than the other three methods on these datasets. On F_2, the boxes of GP-GPPI and GP-RF overlap with medians. However, there is no overlap between these two methods and the other two methods, i.e. GP and C5.0. This means the training performance in GP-GPPI is comparable to

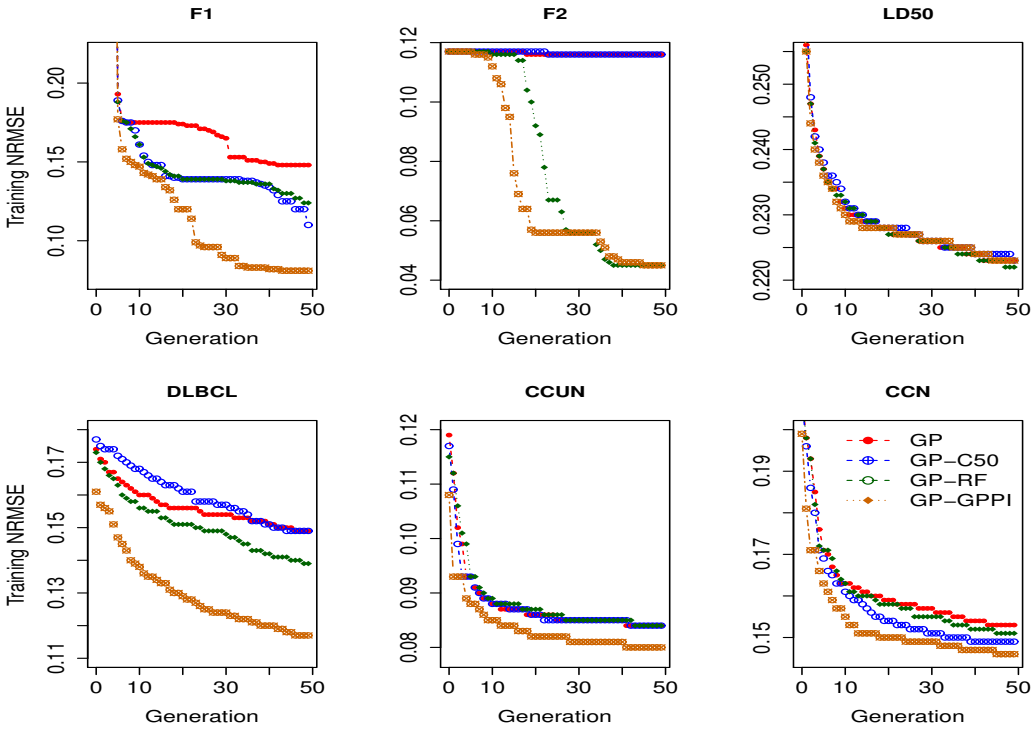


Figure 3.4: The **Training** Error Evolution Plots.

that in GP-RF, which is much better than their counterparts in GP and GP-C5.0 on F_2. On LD50, there is no obvious difference between the four methods. According to the statistical significance tests, on F_1, DLBCL, CCUN and CCN, GP-GPPI has the best training performance among the four methods. On F_2, GP-GPPI has no significant difference from GP-RF on the training set. However, both are significantly better than the other two methods (GP and GP-C5.0). On LD50, there is no significant difference between the training performance of all the methods.

Figure 3.4 shows more details of the evolutionary training process of the four GP methods. GP-GPPI generally achieves better training performance than the other three methods over generations on four out of the six datasets except for on F_2 and LD50. On these four training sets, GP-GPPI outperforms the other three methods on the first several generations.

The difference on the NRMSEs between GP-GPPI and the other methods increases over generations. On F_2, GP-GPPI and GP-RF are significantly superior to GP and GP-C5.0. The median training errors in GP and GP-C5.0 only decrease slightly during the training process, which indicates the difficulty in the two GP methods to learn on this noisy dataset.

It is clear that on most of the datasets, feature selection can promote the learning ability of GP. An intuitive reason is that the reduction of feature space shrinks the search space of GP as well as decreases the production of programs manipulating the irrelevant features. Thus the evolutionary process is more likely to be guided towards the better models. Better feature selection methods can shrink the search space of GP to be much smaller but more effective since they can discard more irrelevant features while keeping important features. Thus less effort is needed for GP to converge to (near) optimal models. It also explains the pattern that the difference on NRMSEs between GP-GPPI and the other three methods is increasing over generations, and why GP-GPPI has a distinguished advantage over the other methods.

Results on the Test Sets — Generalisation Ability

Figure 3.5 shows the distribution of generalisation errors of the 100 best-of-run individuals. The overall trend is similar to the training set, i.e. GP-GPPI outperforms the other methods. Particularly on LD50, GP-GPPI has much lower median NRMSEs than the other methods on the test set. The test evolution plots in Figure 3.6 clearly show that GP-GPPI generally has the best generalisation performance among all the methods, i.e. the lowest test NRMSE over generations. Based on the results of the statistical significance tests, GP-GPPI achieves significantly better generalisation gain than the other three methods on five of the six datasets, except for F_2. On F_2, GP-GPPI has slightly higher test error than GP-RF, but not significantly. They both have a significantly lower NRMSE than C5.0 and standard GP on the test set of F_2.

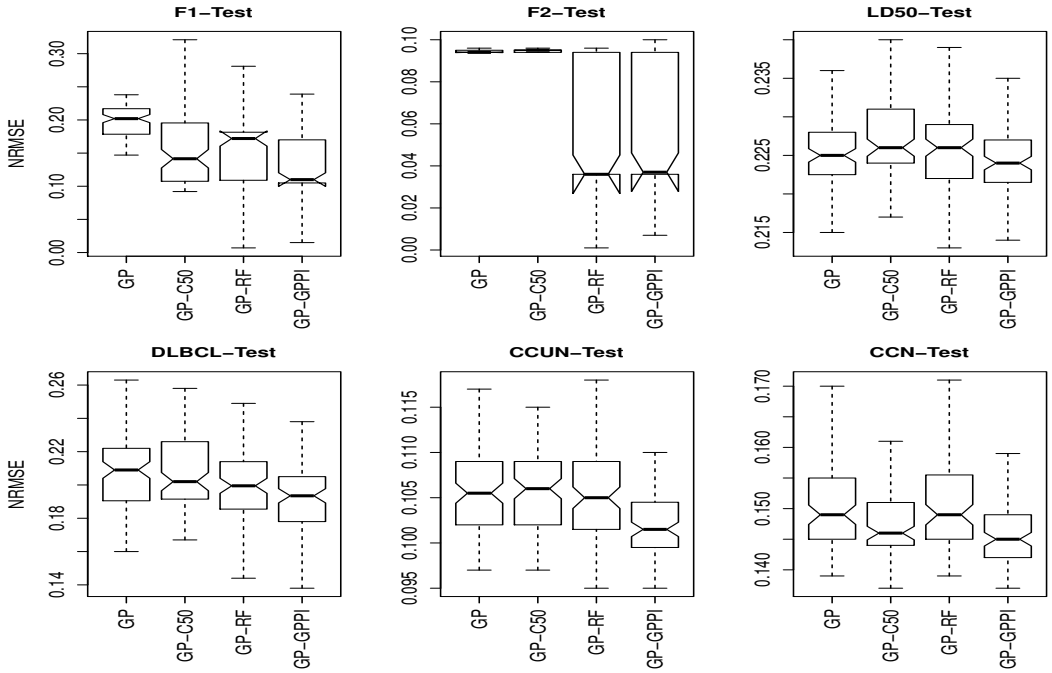


Figure 3.5: Distribution of the Corresponding **Test** NRMSEs.

On the two synthetic datasets F_1 and F_2, which contain the same number of relevant features and noisy features, the generalisation ability of the four methods show different patterns. The intuitive reason might be that the target function of $F_1(F_1 = -g\frac{x_1x_2}{x_3^2})$ is simpler than F_2 $(F_2 = \frac{30x_1x_3}{(x_1-10)x_2^2})$. All the three feature selection methods can have generalisation gain for GP on F_1. On F_2, which has a more complex target function, the test errors of GP and GP-C5.0 do not reduce over generations and even increase slightly over the final several generations, while GP-RF and GP-GPPI can generalise well. On F_2, applying C5.0 for feature selection does not enhance but rather decrease the generalisation of GP. GP-C5.0 has significantly larger NRMSEs than GP on the test set. One of the possible reasons is that it did not keep all the relevant features (in fact, it excluded the $2nd$ feature), although it discarded a number of irrelevant features.

From the generalisation performance on the two synthetic datasets, it

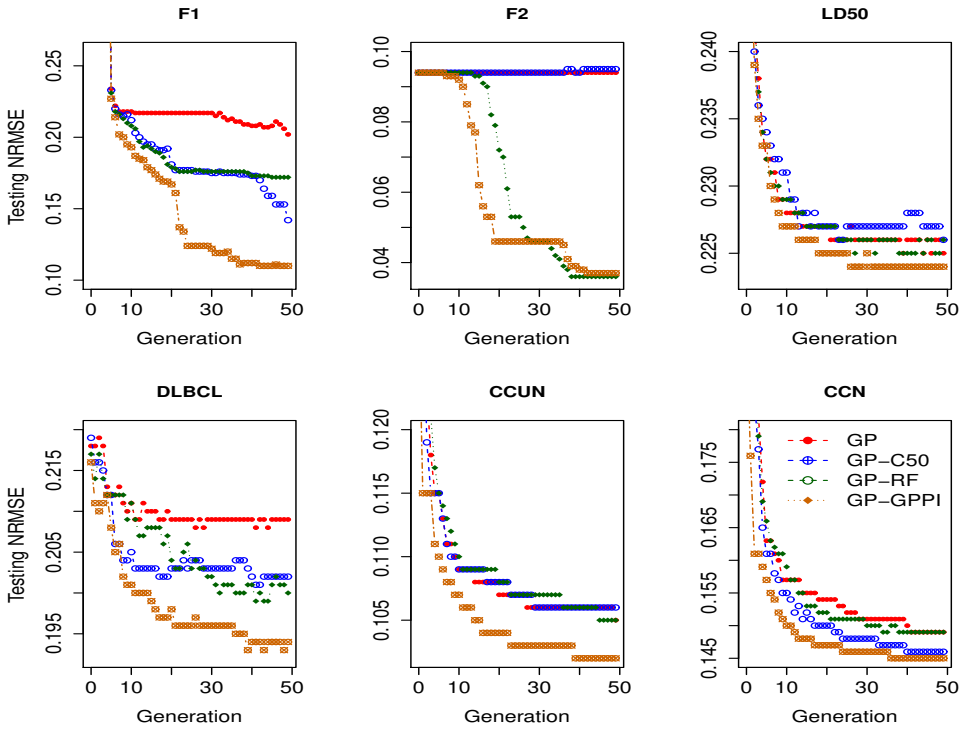


Figure 3.6: The **Testing** Error Evolution Plots.

can be observed that in GP-RF and GP-GPPI, where the feature selection methods can reduce the noise that was deliberately added while keeping the relevant features, the evolved models have a higher probability of including the truly relevant features. They are more accurate in expressing the true relationship between the input variables and the target variables, thus can definitely have better generalisation performance.

On the four real-world datasets, GP-GPPI achieves the best generalisation performance among the four methods, which is confirmed by the Wilcoxon test. While on DLBCL, GP-GPPI has notable generalisation gain over the other three methods, on the other three tasks, GP-GPPI still outperforms the other methods. On LD50 and DLBCL, while GP-RF has slightly but not significantly better generalisation performance than GP, GP-C5.0 has significantly higher test NRMSEs than GP on LD50 and slightly

better generalisation gain than GP on DLBCL. On CCUN and CCN, GP-RF can not improve the generalisation performance of GP to a significant level. GP-C5.0 achieves a significant generalisation gain on CCN.

In summary, GPPI can enhance the generalisation of GP more effectively because it can discard more noisy/irrelevant features than other feature selection methods (feature selection results will be presented in more detail in Section 3.5), so that GP is more likely to construct models using the relevant features. Moreover, the GP models with a continuous property learn more than the stepwise function of the decision trees, thus leading to better performance of GPPI in detecting important features. This could be a major reason that GP-GPPI is superior to the other GPSR methods on generalisation performance.

3.4.2 Comparisons between GP-GPPI and Variants of GPWFS

As mentioned in Section 3.2, the major difference between feature selection in GP-GPPI and GPWFS is the way to decide the importance of features. GPPI has an additional component, which is the permutation feature importance. To investigate the effect of the permutation method on identifying the truly important features from the potentially relevant features, the comparison between GP-GPPI and variants of GPWFS is necessary.

Figure 3.7 and Figure 3.8 show the evolution plots of the median training and test NRMSEs of the 100 best-of-generation individuals obtained on the training data in GP, GPWFS, GPWFS1, GPWFS2 and GP-GPPI. Here, the best settings among the 9 settings (combination of three different settings of G_f and β) of GPWFS and 3 settings (i.e. three different setting of β) of GPWFS1, which lead to the best generalisation performance in these two methods, are chosen to report and compare with other methods.

As shown in Figure 3.7, compared with the three variants of GPWFS, GP-GPPI has better training performance on five of the six datasets ex-

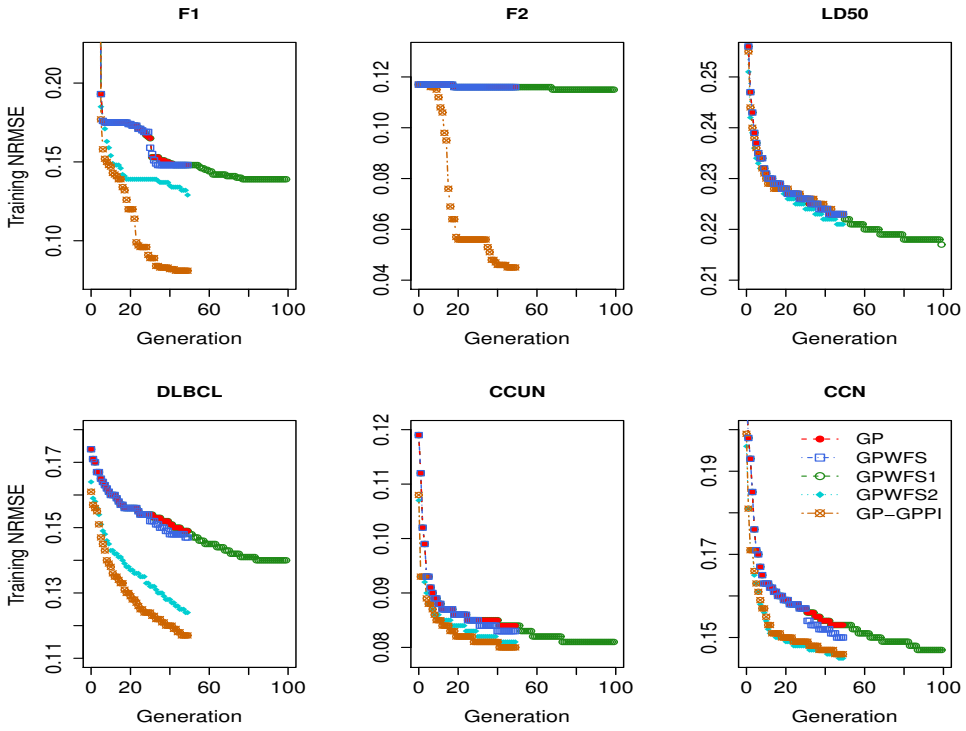


Figure 3.7: GP-GPPI and Variants of GPWFS — The **Training** Error Evolution Plots (GPWFS1 is deliberately set to run 100 generations).

cept for LD50. On the two synthetic datasets F_1 and F_2, GP-GPPI has a much better learning performance than all the three variants of GPWFS, shown as the much lower training errors over generations. The advantage of GP-GPPI over the variants of GPWFS are all significant on F_1 and F_2. On DLBCL, GP-GPPI achieves significantly smaller training errors than GPWFS and GPWFS1, and slightly better training performance than GP-WFS2. On CCUN and CCN, GP-GPPI has slightly but not significantly better learning performance than GPWFS1 and GPWFS2. However, it still significantly outperforms GPWFS on these two training sets. On LD50, GP-GPPI achieves a comparable training performance with GPWFS and GPWFS2, which are significantly worse than GPWFS1 (the plot in green colour in Figure 3.7).

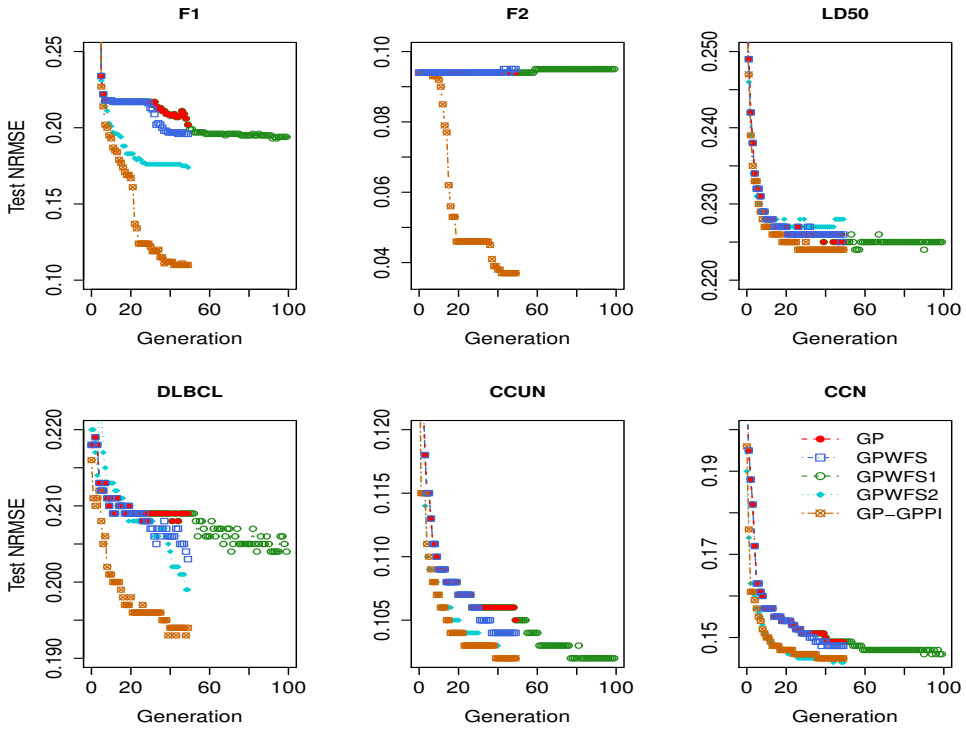


Figure 3.8: GPPI and Variants of GPWFS — The Corresponding **Testing** Error Evolution Plots (GPWFS1 is deliberately set to run 100 generations).

From Figure 3.8, it can be observed that GP-GPPI outperforms the three variants of GPWFS in achieving better generalisation performance on all the six datasets. On F_1, F_2 and DLBCL, GP-GPPI has achieved a dramatic generalisation gain than the three variants of GPWFS. On LD50, GP-GPPI also obtains significantly better generalisation performance than the three variants of GPWFS. On CCUN and CCN, GP-GPPI has significantly smaller test errors than GPWFS. Compared with the generalisation performance of GPWFS1, GP-GPPI obtained comparable results on CCUN and significantly better results on CCN. The comparison between GP-GPPI and GPWFS2 is different from GPWFS1. While GP-GPPI has significantly smaller test errors than GPWFS2 on CCUN, their generalisation performance is similar on CCN.

In summary, GP-GPPI is superior to the variants of GPWFS in both the learning performance and the generalisation ability on most of the datasets. On the two synthetic datasets, which contain a large number of noisy features, the advantage of GP-GPPI over the variants of GPWFS is more obvious than on the four real-world datasets. GP-GPPI outperforms GPWFS, which might be because GPPI collects features from a number of best-of-run GP individuals in different GP runs, which are more sufficiently evolved than those in GPWFS. Features appearing in these individuals are intuitively more reliable than those selected from the best-of-generation individuals in GPWFS. However, the major contribution is owed to the effectiveness of permutation importance, which helps to identify the real important features in GPPI. The comparison between GP-GPPI and GPWFS2 confirms the contribution of the permutation importance method.

3.5 Further Analysis

This section presents a further analysis of the feature selection results and the regression models evolved by the GP methods. It is expected to provide a good way to understand how GPPI advances the other feature selection methods in promoting the generalisation of GPSR.

3.5.1 Feature Selection Results

To further analyse the feature selection ability of GPPI, it is necessary to compare the selected features from different methods. Here, we focus mainly on the comparison between RF and GPPI, since they generally have the best performance on benchmark problems and share the same mechanism(i.e. permutation) to evaluate the importance of features. As the main difference between the two methods lies in searching for features and building the trees, the comparison is to demonstrate whether the feature selection ability of GP is superior to the search ability of RF.

Regarding the importance values of features, a positive importance value is formed by the increased regression error when the feature is shuffled. Since a new sub-test error is obtained from a completely random feature, it should be higher than the initial error on the sub-test sets. A positive value indicates that the feature can contribute to reducing the regression error. The bigger the value is, the more important the feature is to the response variable. In highly correlated datasets, the importance value of a feature in a single tree (in GP or RF) can be small since another feature might have duplicate information. However, obtaining the feature importance over a group of trees can reduce the limitation. A negative importance value is obtained by the reduced new regression error on the permuted sub-test set. It indicates that the completely random feature works even better than the original feature. Thus, the feature is probably not predictive enough to the response value, i.e. it is very likely not important. The zero value means the feature does not appear in the GP trees or the decision trees in most cases.

Feature Selection Results on the Synthetic Datasets

The feature importance values in GPPI and RF on F_1 and F_2 are shown in Figure 3.9. It can be seen that for F_1, both methods have a relatively small number of positive features, which indicates that both can dramatically reduce the number of features. Moreover, in the two methods, the top three important features are consistent with all the truly relevant features. In RF, all the top three features have much higher feature importance from the other features, although it has a larger number of positive features. On the right part which is for F_2, the most obvious pattern is the same as F_1. Both GPPI and RF can find the truly relevant features, and most of these features can have much higher importance values than other features. Compared with F_1, both methods have a much higher number of negative features on F_2. The existence of these features indicates permutation importance can identify the noise which is not predictive to the

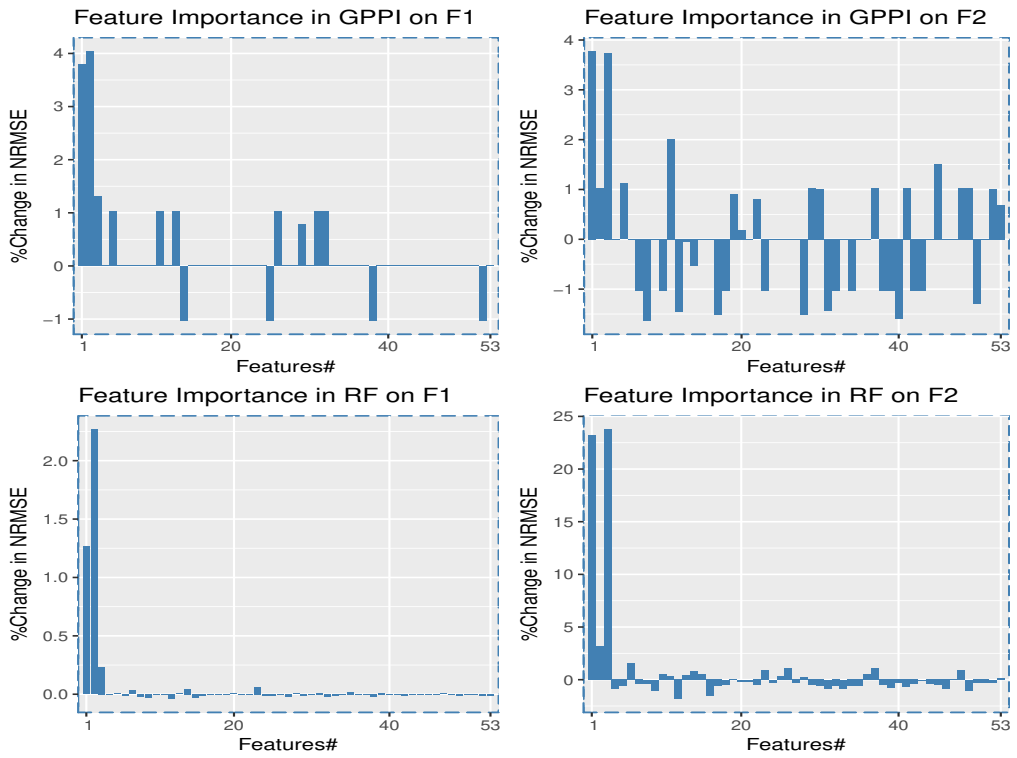


Figure 3.9: Feature Selection Results on the Two Synthetic Datasets.

response variable. As mentioned earlier, F_2 is harder than F_1. This might cause a decrease in the search performance of GP and RF for the important features. Permutation importance contributes more to identify the irrelevant features in this case. It is clear that for the two tasks, both GPPI and RF can have good feature selection ability on discarding a large number of noisy features while keeping the most important features. The discarding of noise takes the forms of not being used/included by the *good* individuals and assigning a negative importance value to them.

Feature Selection Results on the Real-World Datasets

For the real-world datasets, which have higher dimensionality than the synthetic datasets, the overall pattern is different. The detailed importance

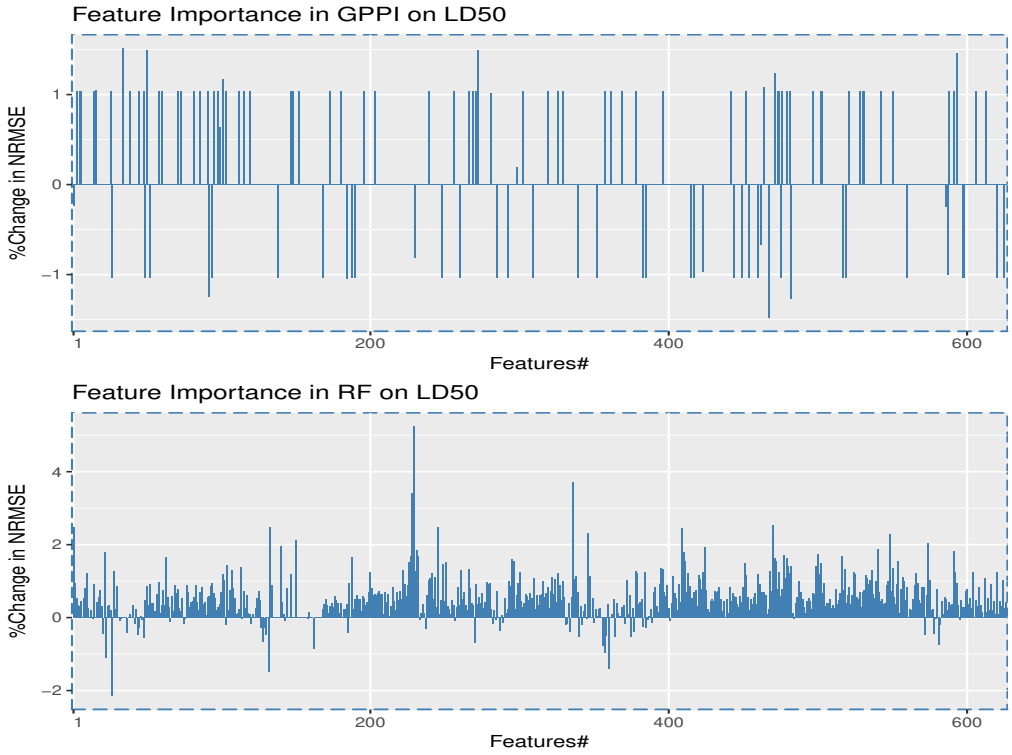


Figure 3.10: Feature Selection Results on LD50.

values of LD50 and DLBCL are presented since they have extremely higher numbers of features and are (much) harder than the other two datasets (CCUN and CCN).

Figure 3.10 shows the importance values of features in LD50. In GPPI, the number of features included in the evolved models is much lower, and the permutation of features helps identify more negative features than RF. Thus the number of features with a negative importance value is much higher in GPPI (around 30% of the total features) than in RF, and a much smaller number of positive features in GPPI than RF. Figure 3.11 shows the feature importance results on DLBCL. Compared with the total number of features (which is 7399), both GPPI and RF can discard a large number of noisy/irrelevant features. With regards to the number of features included in models, the two methods are quite different. GPPI selects a

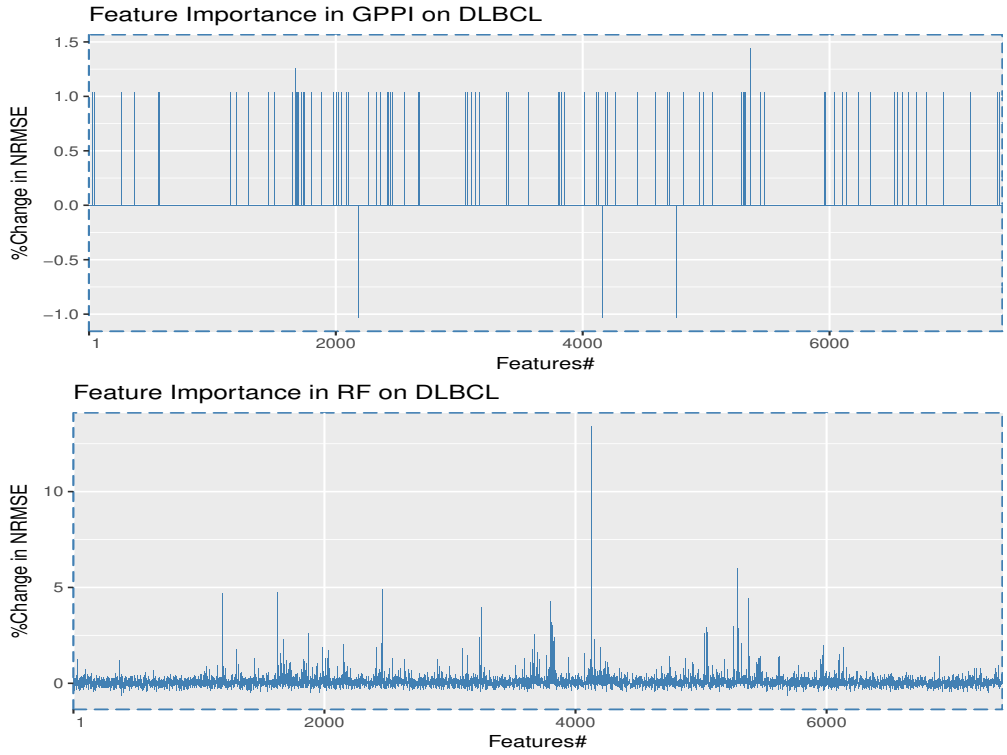


Figure 3.11: Feature Selection Results on DLBCL.

much smaller number of features than RF. Different from LD50, where GPPI detects a large number of negative features, on DLBCL most of the features included in the *good* individuals of GPPI are positive. However, many of the features appearing in the trees in RF have negative importance values. Considering the number of positive features, GPPI is still much smaller than RF. The overall pattern on the other two datasets (CCUN and CCN) is similar with LD50 and DLBCL, that is, RF selects many more features than GPPI. This is partially due to the larger number of trees in RF, but the major reason is that due to the randomly restricted feature selection scheme in RF, more redundant features can appear to be important in RF, which would be considered redundant and eliminated by GP. This is confirmed by the much larger number of positive features in RF than GPPI on LD50, where many pairwise features are highly correlated regarding Pear-

Table 3.4: Top 10 Important Features on Real-World Datasets.

Dataset	Method	Top 10 Features
LD50	RF	228, 335, 227, 469, 244, 0, 132, 408, 345, 547
	GPPI	33, 271, 49, 592, 470, 100, 463, 15, 2, 146
DLBCL	RF	4130, 5289, 2458, 1629, 1187, 5378, 5292, 3798, 3251, 5291
	GPPI	5357, 1674, 2441, 3400, 6134, 48, 262, 566, 573, 1197
CCUN	RF	43, 49, 2, 42, 67, 48, 14, 39, 123, 1
	GPPI	49, 37, 40, 26, 70, 39, 4, 16, 48, 67
CCN	RF	50, 41, 68, 2, 32, 40, 27, 71, 30, 69
	GPPI	44, 50, 3, 43, 15, 68, 2, 40, 49, 41, 71

son correlation (the correlation plots are shown in Appendix B). Overall, GPPI is better in selecting fewer important features when tackling high-dimensional real-world datasets.

Since the number of selected features is large on all of the four datasets in both methods, we further analyse the results by listing the most important features. The top 10 important features selected by the two methods on the four real-world datasets are shown in Table 3.4 in a descending order of importance. It can be observed that the top features selected by the two methods are totally inconsistent on LD50 and DLBCL. The small ratio of the number of instances over the number of features is a major reason. Another reason is the existence of redundant features. In contrast, the two methods selected many consistent features among the top ten important ones on the other two datasets. On CCN, six out of the ten are consistent. On CCUN, the number is four. Compared with LD50 and DLBCL, the ratios of the number of instances over features are much higher on CCN and CCUN. In summary, for the tasks that have a high dimensionality and a large number of training samples, such as CCUN and CCN, GPPI and RF can select many consistent features. While for tasks which have a much higher number of features than the number of instances, like LD50 and DLBCL, feature selection becomes much harder, and many features have similar importance to the response variable. Thus it is difficult to select the

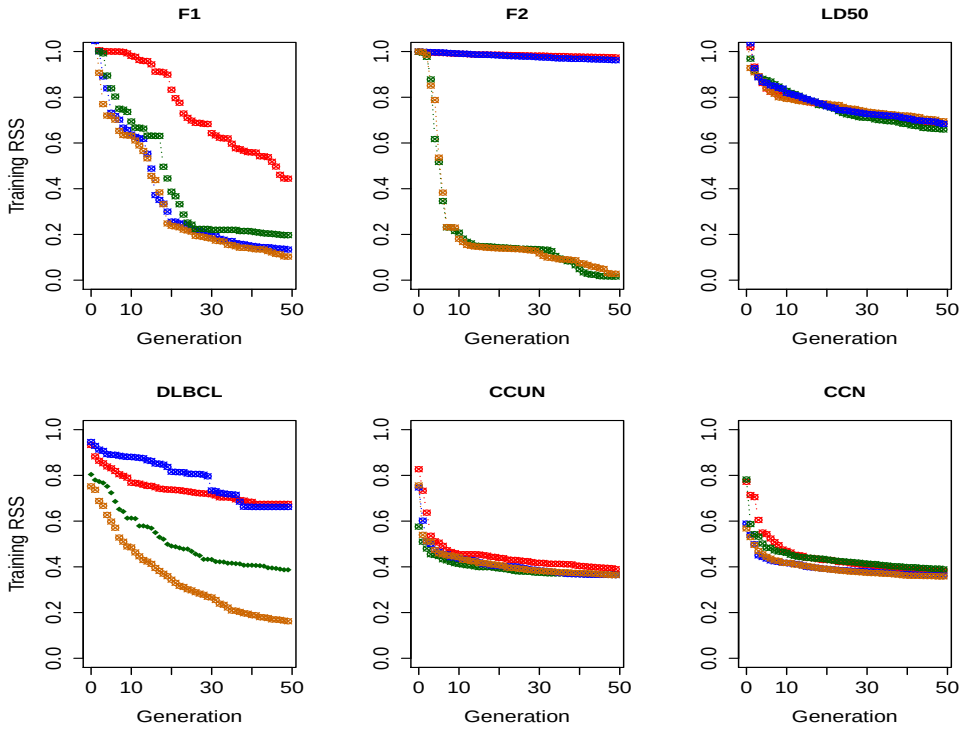


Figure 3.12: The **Training RSS** Evolution Plots.

same features between the two methods.

3.5.2 Analysis of Evolved Models

A further analysis was taken on examining the models evolved by GP using features selected by the different methods. We randomly picked three evolved regression models from the 100 GP runs on the synthetic dataset F_1, where the target function is known, and all the feature selection methods performed well. The examples of the evolved models are shown in Table 3.5. The mathematically simplified forms of the models are also presented in Table 3.5 for an easier analysis of the models. In these three runs, regarding the relevant features in the target function $F_1 = -g\frac{X_1X_2}{X_3^2}$ ($g = 6.67408E{-}11$), it can be observed that GP-GPPI can include all the im-

Table 3.5: Examples of GP Individuals on $F_1 = -g\frac{X_1X_2}{X_3^2}$ ($g = 6.67408E-11$)

Method	Example of Best-of-Run Individual	Simplified Individual
GP	(*(*-0.33(*(*(*-0.60 0.099)(0-X_2))(Inv X_3)))(*-0.052(*(sqrt(sqrt X_{14}))(*(*(0-(*0.10(0- -0.0054)))(*0.047(0- -0.0054)))(*-0.29(0- (Inv X_3)))))))	$\frac{4.83E-11X_2\sqrt{\sqrt{X_{14}}}}{X_3^2}$
GP-C5.0	(*0.083(*(*0.10(*(0-(*-0.015 X_2))(*(*(*0.10(0-(*-0.015 X_2)))(* X_1-0.039))-0.97)))-0.0054))	$-4.449E-11X_1X_2$
GP-RF	(*(*(0-0.36)(*(*(0-0.36)(*0.083(*(*(*-0.15-0.018)(+0.83(*X_{23} X_{31})))(*-0.034-0.015))))-0.15))(*-0.15-0.018))	$-1.049E-11(0.83+X_{23}X_{31})$
GPWFS	(*(*(0-X_2)(*(*(*(*-0.60 0.099) (0- X_2))((Inv X_3))(Inv X_4))) (* -0.052 (* (sqrt X_{14})(*(*(0-(* 0.11(0–0.0054)))(* 0.047 (0–0.0054)))(* -0.29 -0.60)))))	$\frac{7.366E-11X_2^2\sqrt{X_{14}}}{X_3^2}$
GP-GPPI	(*(*0.90(*(*-0.052(*(*(0-0.032)(Inv(+X_3 0.74)))(Inv(+(+X_3 0.74)0.74))))(*-0.29-0.60)))(*(*(*(* -0.015(*(*(*-0.29 0.099) 0.054)(Inv(+X_3 0.74)))X_1))(Inv-0.78))(0-(0-0.055)))X_2))	$\frac{7.9104E-9X_1X_2}{(X_3+0.74)^2(X_3+1.48)}$
GP	(* (* -0.170.073)(* (* (* (* -0.097 0.56)(+ (* (* X_{50}-0.44) (* 0.56 -8.30E-4))-6.86E-4))(+ (* 0.0065 (0-X_1)) -6.86E-4))(* 0.0065 (0-X_1))))	$4.381E-6X_1(2.045E-4X_{50}-6.86E-4)(0.0065X_1+6.86E-4)$
GP-C5.0	(* (* (0- X_2) (0-0.092)) (*(* (0- (+ (* (sqrt 0.39) (sqrt0.39))(0- (+ 0.45 X_1)))-8.30E-4)(* (Inv (0- (Inv(* (* 0.01-0.82) 0.002))))(* (sqrt 0.39) (* -0.033-0.82)))))	$2.116E-11X_2(X_1+0.06)$
GP-RF	(*7.16E-4 (0-(* (* (* X_1(*(* X_2 -0.76) -8.30E-4))(* 0.010 -0.76))(* 0.010-0.76))))	$-2.608E-11X_1X_2$
GPWFS	(*(0-(*-0.170.073))(*(*(*(*-0.097 0.56)(0-(+ (*0.0065(0-X_1))-6.86E-4)))(0- X_1))(* 0.0065(+(*(*X_{50} -0.44)(+(*(*X_{50} -0.44)(*0.56 -8.31E-4))-6.86E-4))-6.86E-4))))	$4.381E-6(0.0065X_1+6.86E-4)(2.12E-4X_{50}^2-6.86E-4)$
GP-GPPI	(*(*(*(+ -0.23 0.86)(*X_1 0.0022))(Inv(+X_3(*(Inv (Inv(+ X_3 X_3))(* 0.074 -0.046)))))(* (* (sqrt (*(Inv (+ X_3 (*-0.046 (+ -0.230.86))))(Inv(+X_3 X_3))))(*(*(0- 0.0065)-0.44)(*0.074(*-0.046 0.0065)))))(Inv(Inv(Inv (Inv X_2))))))	$\frac{8.711E-11X_1X_2}{X_3\sqrt{2X_3^2-0.05796X_3}}$
GP	(*(*(*(*(0–0.086)(*-0.018(0-(0–0.086))))(*(0–0.086)(*(*(*(0-0.0051)(0- -0.086))(0-(+-0.46(0–0.018))))(0-0.54))))0.0051)(Inv -0.90))	$6.527E-12$
GP-C5.0	(*(sqrt(+(sqrt(sqrt X_2))X_2))(*(*(*(*(0-0.0051)(0-0.0051))(*(sqrt X_2)0.0046))(-0.080(0-(*(sqrt X_2)0.0046))))-0.28))	$-1.03E-10X_2\sqrt{X_2+\sqrt{\sqrt{X_2}}}$
GP-RF	(*(*(*(*(*0.19 0.0039)(*(+(+(*0.980.34)X_1)(* 0.900.28))(* 0.98 0.0039)))(0- 0.0051))(*(0-(*(+(+(* 0.980.0039) X_1) X_{47}) (* 0.980.0039))) X_2))(0-(+(*(*(+ X_2 X_5)(* (* -0.014 (* 0.19 0.0039))X_2)) (0- (+ X_2 X_5))) (+ -0.014(* 0.98 0.28)))))	$5.413E-11X_2(X_1+0.5852)(X_1+X_{47}+0.0038)$
GPWFS	(*(*(*(*(sqrt (0–0.24))(* (* (* (0- 0.0051) (0–0.086))(0-(+ -0.46 0.11)))(0- 0.54))) (0-(0- 0.0051)))0.0051)(*(+ -0.46 0.11)(0- -0.018)))	$7.04E-12$
GP-GPPI	(*(Inv(0-(*X_3 X_3)))(*(*(0–0.012)(*(0–0.012)(*-0.023(0-(*0.0320.95))))))(*(* (+ (* -0.014 X_1)(* -0.014 X_1))(* 0.057 X_2))(+(+(Inv(0-(Inv 0.33)))(*-0.014(*-0.023(0-(Inv0.33))))))(*(*-0.014 X_1)(*-0.023(Inv(*-0.014 X_1))))))))	$\frac{5.669E-10X_1X_2}{X_3^2}$

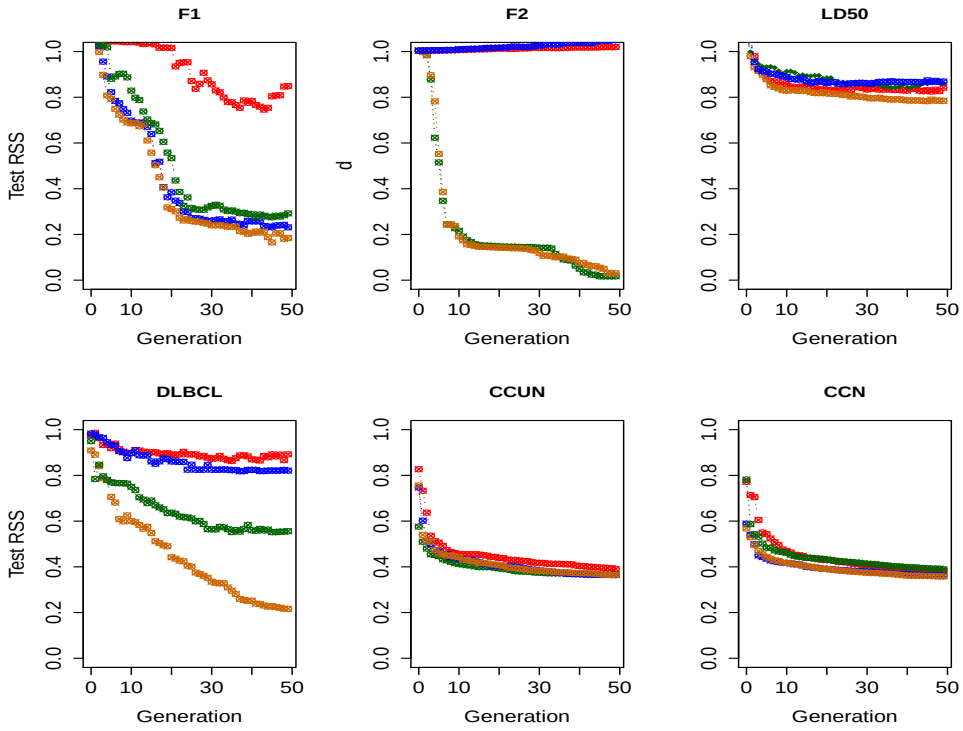


Figure 3.13: The **Test RSS** Evolution Plots.

portant features, and uses only the relevant features to construct the models. On the other hand, the other three GPSR methods and standard GP either include some noisy features or can not include all the relevant features. Concerning the shape of the models, it is also easy to find that models in GP-GPPI have a much closer/similar shape with the target model than those in the other four methods (in the 69th run, GPPI almost found the "true" model).

3.5.3 A Direct Interpretation of Model Accuracy

To better interpret the model accuracy, we employ a new fitness function, i.e. relative sum fo squared error (RSS) to compare the models evolved by GP, GP-C50, GP-RF and GP-GPPI. The definition of RSS is shown in

Table 3.6: Comparisons between LASSO, Random Forest (RF), GP, and GP-GPPI

Benchmark	Method	Training NRMSE (Median±MAD)	Test NRMSE (Median±MAD)	Significance Test (with GP-GPPI) (training, test)
F_1	LASSO	0.17	0.22	$(-, -)$
	RF	0.055±0.0013	0.16±0.0017	$(-, -)$
	GP	0.012±0.016	0.095±0.03	$(+, -)$
	GP-GPPI	0.037±0.043	0.049±0.064	
F_2	LASSO	0.11	0.09	$(-, -)$
	RF	0.040±4.20E-4	0.078±5.61E-4	$(-, -)$
	GP	0.002±2.97E-3	0.005±4.45E-3	$(=, =)$
	GP-GPPI	0.005±4.45E-3	0.004±2.97E-3	
LD50	LASSO	0.04	0.68	$(+, -)$
	RF	0.097±7.61E-4	0.23±0.0013	$(+, -)$
	GP	0.19±0.009	0.25±0.026	$(+, -)$
	GP-GPPI	0.21±4.45E-3	0.21±4.45E-3	
DLBCL	LASSO	0.18	0.22	$(-, -)$
	RF	0.058±7.77E-4	0.13±0.0014	$(+, -)$
	GP	0.088±0.012	0.182±0.032	$(-, -)$
	GP-GPPI	0.081±0.012	0.11±0.019	
CCUN	LASSO	0.13	0.15	$(-, -)$
	RF	0.030±1.18E-4	0.098±2.25E-4	$(+, =)$
	GP	0.073±1.48E-3	0.099±2.22E-3	$(+, =)$
	GP-GPPI	0.076±1.48E-3	0.097±2.97E-3	
CCN	LASSO	0.21	0.23	$(-, -)$
	RF	0.054±1.77E-4	0.141±3.44E-4	$(+, -)$
	GP	0.133±2.97E-3	0.143±2.97E-3	$(+, -)$
	GP-GPPI	0.139±2.22E-3	0.139±2.97E-3	

Equation (3.5)

$$RSS = \frac{\sum_{i=1}^{N}(\hat{Y}_i - Y_i)^2}{\sum_{i=1}^{N}(\bar{Y} - Y_i)^2} \tag{3.5}$$

where $\hat{Y}_i$ is the ith predicted output, Y_i is the ith response value, $\bar{Y}$ is the mean of the target values, N is the number of instances.

RSS allows a direct comparison between the results of different datasets. Meanwhile, the RSS error could be interpreted by comparing the evolved model against a baseline model, i.e. using the mean of the responses as the predict value. When RSS > 1, it means the evolved model has worse

performance than the baseline model, RSS < 1 means a better model and RSS $= 1$ indicates similar performance to the baseline model.

The evolutionary plots on the training RSS and the test RSS are shown in Figure 3.12 and Figure 3.13, respectively. They are drawn using the median RSS values obtained by the 100 best-of-generation individuals.

As shown in the two figures, the overall pattern is similar to that using NRMSE. Among the four methods, GP-GPPI is still the winner on both the learning and generalisation performance on most of the datasets. Meanwhile, models evolved by the four GP methods all have a good accuracy on F1, CCUN and CCN, where the RSS values are much smaller than 1. While on the other three datasets, GP has a higher RSS than the other three GP methods. On F2 and DLBCL, the selected features in GP-RF and GP-GPPI bring an impressive improvement and decrease a lot on both the training and the test RSS errors, while features selected by GP-C50 do not improve the performance .On LD50, the three feature selection methods do not bring an obvious benefit.

3.5.4 Further Comparisons

Table 3.6 shows the results of GP and GP-GPPI under the same computation time. The results of LASSO and RF for regression are also shown in the TABLE. For methods with stochastic results (i.e. RF, GP and GP-GPPI), the median value and the mean absolute deviation (MAD) of the 100 lowest training errors and their corresponding test errors are presented. Median and MAD are claimed to be more robust to the outliers [134]. For LASSO, only one unique result is reported. The Wilcoxon test has been conducted on the paired training errors and the test errors of the 100 best-of-run individuals in GP-GPPI and the two methods (i.e., GP-GPPI vs. GP, GP-GPPI vs. RF). The statistical significance test, Z-test, is used to test the significant difference between GP-GPPI (with a group of 100 results) and LASSO (one result). While "$-$" means GP-GPPI performs significantly

better than the compared method, "+" indicates GP-GPPI is significantly worse, and "=" stands for no significant difference.

It is clear that GP-GPPI achieves much smaller NRMSEs than LASSO on both the training sets and the test sets on most of the benchmark problems except for LD50. On LD50, GP-GPPI has a significantly worse training performance than LASSO. However, it achieves a much better generalisation performance on LD50, which is also significant. On these synthetic datasets generated by the nonlinear target functions with much noise and real-world high-dimensional regression tasks, LASSO does not generalise well. Compared with RF for regression, GP-GPPI obtains much smaller training errors and test errors on the two synthetic datasets, both of which are significant. On the four real-world datasets, GP-GPPI has significantly larger training errors than RF. However, the generalisation performance of GP-GPPI is better than RF on all the four test sets. While on CCUN, the generalisation error of GP-GPPI is slightly better than RF, on the other three test sets, GP-GPPI outperforms RF in a significant way. In general, GP-GPPI has a better generalisation ability than RF on the benchmark problems.

When comparing GP-GPPI with GP (GP doubles the population size of GP-GPPI) under the same computation time, the results show that on most of the benchmark problems, the training errors of GP-GPPI are larger than RF, but it has much smaller test errors than GP. On DLBCL which has a large number of features (7399) and a small number of instances, GP-GPPI outperforms GP on both the training set and the test set, which indicates that the generalisation improvement brought by GPPI to GP is not because of more computation effort. Moreover, the regression performance of GP-GPPI in this set of experiments is generally better than the original setting, which means that better regression performance can be expected when increasing the computation load properly. However, it does not indicate that more computation load can always achieve better generalisation, because of the risk of overfitting.

3.6 Chapter Summary

In this chapter, a new feature selection method GPPI for GP for high-dimensional symbolic regression is proposed. GPPI collects features appearing in a number of best-of-run GP individuals, and obtains the importance of features using a permutation measure. Feature selection is based on the importance values. The feature selection results show that compared with RF, which is effective in finding important features along with the presence of redundant features, GPPI is more effective in identifying the truly relevant features. The regression results of GP employing various feature selection methods show that GPPI not only outperforms RF and C5.0 in improving the learning performance of GP, but also gains much more benefits on the generalisation of GP. GP-GPPI also advances different variants of GPWFS in enhancing both the learning performance and generalisation ability of GP. Further analysis of the evolved models indicates that GP-GPPI is superior to the other methods on evolving models including only the truly relevant features. Generally, these models are more comprehensible.

This chapter confirms that our new feature selection method has an impressive benefit on enhancing the generalisation of GP for high-dimensional symbolic regression tasks. However, it is not a silver bullet for all the overfitting issues in GP for SR, since overfitting does not occur only when training models on high-dimensional regression data. When learning regression models on datasets with a relatively small number of features but an insufficient number of instances, GP is also prone to overfitting. In this scenario, the feature selection method proposed in this chapter might have limited effect on increasing the generalisation of GP. Next chapter will seek a solution to improve the generalisation of GP for SR in this scenario.

Chapter 4

Structural Risk Minimisation in GP

4.1 Introduction

As mentioned in previous chapters, generalisation is one of the most important performance criteria for learning algorithms in machine learning, since it reflects their prediction performance on unseen data. Chapter 3 developed a feature selection method to improve the generalisation performance of GP for symbolic regression, but low generalisation may happen in the situation of learning from datasets with a relatively small number of features yet an insufficient number of instances. This chapter develops a new method to address the generalisation issue in this scenario.

As a learning algorithm, GP has a major goal, which is to find a model that can minimise the expected/prediction error $Err = E[L(Y, f(X))]$. The generalisation error $Err_T = E[L(Y, f(X))|_T]$ measures the prediction error of the learnt 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 expected error Err is taken with respect to the joint distribution $P(X, Y)$, which is typically unknown in most real-world learning tasks. Thus, to minimise the expected error Err, many learning algorithms rely on the empirical risk minimisation principle [83]. This principle consists of computing the errors of a set of candidate models over the training set, and then selecting the one that obtains the minimum training error among the set of models. The empirical/training error is considered to be a good indicator of the expected test error in general. However, in many cases, this indicator does not work well, particularly when the number of training samples is too small to represent the real distribution of $P(X, Y)$ and/or over-complex models have been learnt. In these scenarios, an accurate estimation of the expected generalisation error is more reliable.

During the evolutionary process, the training error consistently decreases along with the increase of model complexity, but the generalisation error typically increases. While over-simple models generally have high training and test errors, over-complex models have lower training error but even higher generalisation errors than simple models. There is an (near) optimal model complexity in between (over-complex and over-simple) that might minimise the generalisation error. Moreover, previous research has confirmed that the model complexity directly influences its generalisation ability [224, 233]. 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 [55, 228, 233]. A key issue in obtaining the optimal model complexity is how to measure the model complexity.

4.1.1 Chapter Goals

This chapter aims to develop a new GP approach to enhance the generalisation of GP for symbolic regression. This will be accomplished by in-

troducing an generalisation estimating framework — structural risk minimisation (SRM), which uses an empirical measurement of the complexity of candidate solutions, into GP to develop a SRM-driven GP approach. Specifically, this chapter has the research objectives as follows:

- whether and how the proposed SRM-driven GP approach improves the training performance of GP,
- whether SRM-driven GP can lead to a significant generalisation gain over standard GP and GP with other generalisation estimation methods such as a bootstrap method,
- how SRM-driven GP influences the complexity and behaviour of the evolved models, and
- whether SRM with a non-uniform setting can outperform its counterpart with a uniform setting in improving the training and generalisation performance of GP.

4.1.2 Organisation

The remaining sections of this chapter are organised as follows. The second section proposes the new GP methods, which implement the generalisation estimating framework, and describes the procedure of obtaining the model complexity in detail. In the third section, the details of the experiment design are presented, including the benchmark problems, the benchmark methods for comparison, and the parameter settings. The fourth section provides an analysis on the learning and generalisation performance, as well as an analysis on the evolved models. Finally, the fifth section draws a summary of this chapter.

4.2 The Proposed Methods

This section firstly presents Vapnik-Chervonenkis dimension and structural risk minimisation, then describes the proposed SRM-driven GP approach including two specific methods, *genetic programming with structural risk minimisation* (GPSRM) and *genetic programming with optimised structural risk minimisation* (GPOPSRM).

4.2.1 Vapnik-Chervonenkis Dimension and Structural Risk Minimisation

In statistical learning theory [231], probably approximately correct (PAC) [221] defines a general measure for the complexity of a learning machine, which is the Vapnik-Chervonenkis dimension (VC-dimension) [230]. The original definition of the VC-dimension is for a set of indicator functions $\{I(X, \alpha)\}$, where X are the input vectors, α is a set of parameters and the outputs of $\{I(X, \alpha)\}$ take the values of 0 or 1. The VC-dimension h of functions $\{I(X, \alpha)\}$ is the maximal number of input vectors X_1, X_2, $\cdots$, X_h that can be *shattered* by $\{I(X, \alpha)\}$ [232]. In other words, with proper α, $\{I(X, \alpha)\}$ can always perfectly separate these vectors into two classes in all the 2^h possible ways. Later, 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 the corresponding indicator functions $\{I(R(X, \alpha) - \beta)\}$ [231], where $\beta \in (A, B)$.

After the proposal of VC-dimension, various assessments of the *expected generalisation risk (i.e. expected test error)* have been developed [56]. Structural risk minimisation (SRM) [231] is one of these approaches, which provides a powerful framework to estimate the generalisation ability of prediction models. SRM defines an upper bound of the generalisation 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 VC-dimension. For a fixed size training set, the confidence interval is determined purely by the VC-dimension. Thus, the generalisation bound relying on the confidence interval is also named the VC generalisation bound or VC bound [83]. The learning process under SRM, which tries to select the models having a good trade-off between the empirical error and VC-dimension (i.e. model complexity), has the potential to lead to models with better generalisation ability. In [56, 231], a practical form of VC generalisation bound for regression problems is proposed. It is defined as:

$$R_{exp}(h) \leq R_{emp}(h) \left(1 - \sqrt{p - p\ln p + \frac{\ln n}{2n}}\right)_{+}^{-1} \tag{4.1}$$

where $R_{exp}(h)$ is the expected test risk, $R_{emp}(h)$ stands for the empirical risk of the model, $\left(1 - \sqrt{p - p\ln p + \frac{\ln n}{2n}}\right)_{+}^{-1}$ represents the confidence interval ("+" denotes the positive part of $1 - \sqrt{p - p\ln p + \frac{\ln n}{2n}}$). 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 generalisation bound $R_{exp}(h)$. Let a set of k regression models be evaluated by SRM. These models form a nested sequence with increasing estimated generalisation errors like $R_{exp1} < R_{exp2} < \cdots < R_{expk}$. 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 generalise well on unseen data.

Despite its solid theoretical foundation and the ability to assess the expected test error, SRM is seldom considered in GP for symbolic regression. The underlying reason is the great difficulty in measuring the VC-dimension of evolved models. It is challenging to obtain a tight theoretical estimation of the VC-dimension for nonlinear models. Vapnik et al. [228] developed an *experimental*[1] method to measure the VC-dimension

[1]The word "experimental" was used in the original paper [228] to emphasise that the

of a learning machine for classification. This chapter aims to extend it to GP for symbolic regression and develops two methods named genetic programming with structural risk minimisation (GPSRM) and genetic programming with optimised structural risk minimisation (GPOPSRM).

4.2.2 GPSRM: Measuring the VC-Dimension using Uniform Setting

In GP for symbolic regression, obtaining a small empirical/training error does not guarantee a good generalisation 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, the evolutionary process is prone to overfitting and the empirical error is not a good indicator of the generalisation performance. Instead, an accurate estimation of the expected generalisation error of the evolved models is more reliable. Guided by the estimated generalisation error, the evolutionary process is expect to move towards models having a good generalisation ability.

When adopting SRM in GP, the crucial and most difficult aspect is to obtain the VC-dimension of the evolved models. Note that, the fruitfulness of SRM in promoting the generalisation of GP depends greatly on the precision 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 [232]. However, this complexity index is not appropriate for nonlinear models [232]. 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. There is no existing work measuring the VC-dimension of the evolved models in GP experimentally, which could be more accurate

method is not a theoretical estimation.

and reliable than any simple theoretical estimation. Different from the existing methods [28, 154] 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. This *experimental* method was proposed in [228] to measure the VC-dimension of a learning machine for classification, which is suitable for both linear and non-linear models. We decided to extend this method to regression models in GP.

We introduce SRM into GP to propose a GP approach named *genetic programming with structural risk minimisation* (GPSRM). GPSRM employs SRM as the fitness function, and intends to achieve a good trade-off between the accurate approximation on the training data and the lower complexity of these models. The evaluation process in GPSRM is shown in Figure 4.1. An example of this procedure is shown on Figure fig:exampleVCDim
.

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 $I(f(X, \alpha) - \beta)$ [231], 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 step 2 to step 4 in Figure 4.1, the VC-dimension of the model is measured. The method to measure VC-dimension is described as follows.

Main Idea

Vapnik et al. [229] derived a criterion to decide whether a learning process is consistent. Being consistent means that the maximum deviation between the empirical error and the expected error does not exceed a small value ε. Specifically, for the indicator function $I(X, \alpha)$ with a VC-

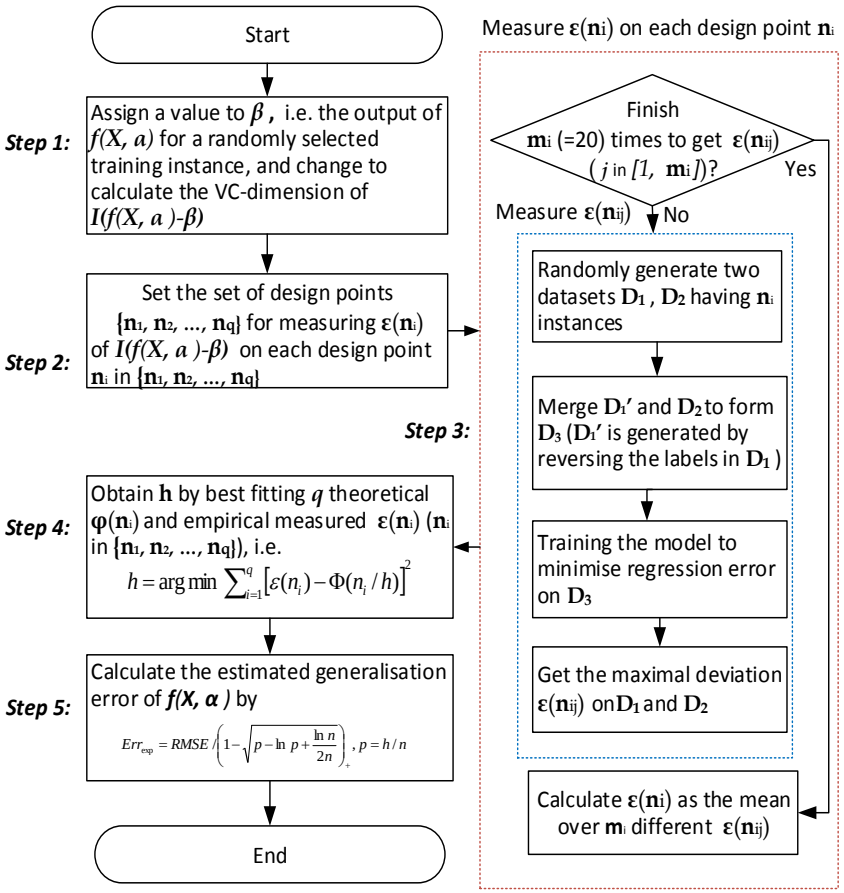


Figure 4.1: The evaluation process in GPSRM for each individual $f(X, \alpha)$.

dimension h, to decide whether the learning process is consistent, Vapnik et al. [229] defined the bound as follows:

$$P\{sup[R_{exp} - R_{emp}] > \varepsilon\} < min\left\{1, \exp\left[\left(C_1 \frac{\ln 2n/h + 1}{n + h} - C_2\varepsilon^2\right) n\right]\right\} \quad (4.2)$$

where C_1 and C_2 are two constants, $C_1 <= 1$ 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: for any given constant δ, a number n_l exists so that when the number of training instances n is larger than n_l ($n > n_l$), the

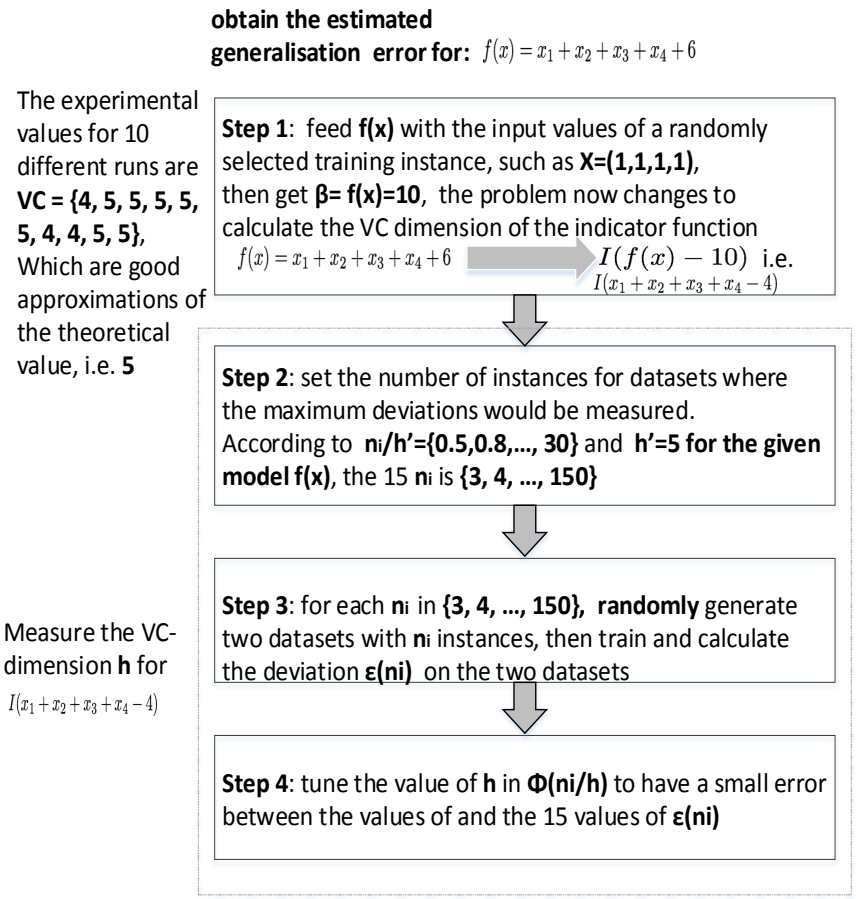


Figure 4.2: An example on how to measure the VC-dimension.

following inequality holds.

$$P\left\{sup\left[R_{exp} - R_{emp}\right] > \varepsilon\right\} < \exp\left[\left(-(C_2 - \delta)\varepsilon^2\right) n\right] \tag{4.3}$$

Later, researchers improved the value of the constant to $C_2 = 2$ [66]. In this case, it was found that for a large n, the bound is close to the value given by the Kolmogorov-Smirnov law [121], which defines the distribution law of the maximum deviation 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 a sufficiently large number of instances,

the equality

$$P\left\{sup\left[R_{exp}-R_{emp}\right]>\varepsilon\right\}=\exp\left(-2\varepsilon^{2}\right)n-2\sum_{k=2}^{\infty}(-1)^{k}\exp\left(-2\varepsilon kn\right) \tag{4.4}$$

holds. Note that compared with the value of the first term, the value of the second term is very small. The closeness of the above bound and the value of the Kolmogorov-Smirnov law indicate that the bound is tight and close enough to the exact value. Based on this observation, [228] assumed that it is possible to find a value for C_1 to make the bound tight for both small and large numbers of instances. In this scenario, the function $\Phi(\frac{n}{h})$, which defines the expected maximum deviation between the error rates, is independent of the conditional distribution $P(Y|X)$. The definition of $\Phi(\frac{n}{h})$ is:

$$\Phi(\frac{n}{h})=E\left\{sup\left[R_{exp}-R_{emp}\right]\right\} \tag{4.5}$$

Based on the assumption that it is possible to derive $\Phi(\frac{n}{h})$, the idea of the *experimental* method to measure the VC-dimension was proposed in [228]. Suppose that $\Phi(\frac{n}{h})$ can be derived successfully, and 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 fit between the theoretical values given by $\Phi(\frac{n}{h})$ and the maximum deviations obtained *experimentally*. Figure 4.3 visualises this process. As the figure shows, given a list of $\epsilon(n_i)$ (i.e. the orange dots, the values of which are obtained from the *experimental* method), the VC-dimension value is obtained by getting the parameter h of the curve $\Phi(n/h)$ to make it best fit into 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 [228]. Note that in practice, it is impossible to obtain an 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

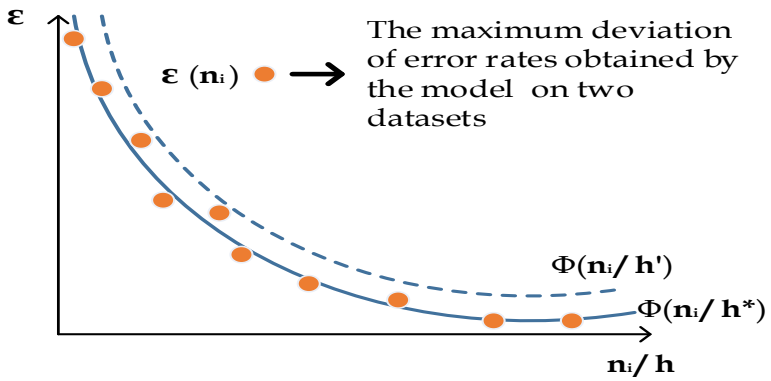


Figure 4.3: h is measured by searching for a better fitting (from h' to h^*) between $\Phi(\frac{n}{h})$ and the *experimental* maximum deviation.

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 $\Phi(\frac{n}{h})$ and the process of obtaining the *experimental* values of the maximum difference between the errors will be presented in detail in the following sections.

Theoretical Formula of the Maximum Deviation

To estimate a bound on the expectation of the maximum deviation between errors, we first need to formulate this maximum deviation. For a set of indicator functions $I(X, \alpha)$ with a VC-dimension h, given a set of samples $Z^{2n} = X_1, Y_1, X_2, Y_2, \ldots, X_{2n}, Y_{2n}$ 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:

$$Pe_i(Z^{2n}, \alpha) = \frac{1}{n}\left(\sum_{j=1}^{n} |Y_j - I(X_j, \alpha)|\right), \quad i \in (1, 2) \tag{4.6}$$

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

$$\epsilon(n) = sup\left[Pe_1(Z^{2n}, \alpha) - Pe_2(Z^{2n}, \alpha)\right] \tag{4.7}$$

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

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

$$E\{\epsilon(n)\} \leq \begin{cases} 1 & \text{if } \frac{n}{h} <= 0.5 \\ C_1 \frac{\ln(2n/h)+1}{n/h} & \text{if } 0.5 < \frac{n}{h} <= 8 \\ C_2 \sqrt{\frac{\ln(2n/h)+1}{n/h}} & \text{if } \frac{n}{h} > 8 \end{cases} \tag{4.8}$$

where C_1 and C_2 are two constants. It is possible to find values for C_1 and C_2 that make the bound tightly hold. The two values 0.5 and 8 are to distinguish datasets with large and small n/h values [228]. Using a continuous approximation, the right side of the bound can be defined as

$$\Phi(\frac{n}{h}) = \begin{cases} 1 & \text{if } \frac{n}{h} <= 0.5 \\ a \frac{\ln(2\frac{n}{h})+1}{\frac{n}{h}-k} \left(\sqrt{1 + \frac{b(\frac{n}{h}-k)}{\ln(2\frac{n}{h})+1}} + 1 \right) & \text{otherwise.} \end{cases} \tag{4.9}$$

where the parameters a and b determine that $\Phi(\frac{n}{h})$ can cover the region of large ($\frac{n}{h} > 8$) and small ($0.5 < \frac{n}{h} <= 8$) values of n/h. The values of a and b are obtained by fitting Equation (C.4) 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 is found that $a = 0.16$ and $b = 1.2$ in [228]. Then, according to $\Phi(0.5) = 1$, it is easy to get $k = 0.14928$.

Experimental Measure of the Maximum Deviation

Step 3 in Figure 4.1 shows the procedure for 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 *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 Figure 4.1 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, the error rate needs to be maximised on the first dataset, while minimising 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 work. Therefore, we change the task from calculating the VC-dimension of $f(X, \alpha)$ to obtaining 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 2.

As shown in Algorithm 2, 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, which is the number of distinct features in $f(X, \alpha)$. 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 minimise its MSE on dataset D_3 ($D_3 = D_1^{'} \cup D_2$), thus $I(f(X, \alpha) - \beta)$ can (approximately) obtain 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 Equations (4.6) and (C.2).

Algorithm 2: 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 [231], 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 minimise its MSE on D_3 using mini-batch gradient descent, then calculate its error rates on D_1 and D_2 according to Equations (4.6) and (C.2) .

    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 $\epsilon(n_i)$

In Step 3 of the above procedure, *mini-batch gradient descent* [53] is used to train the coefficients in the model to obtain the minimal MSE on D_3. Minimising the error of the model on D_3 is equivalent to getting the maximum deviation $\epsilon(n)$ on D_1 and D_2 (since $D_3 = D_1^{'} \cup D_2$). 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 value. Step 1 to Step 4 should be repeated for m_i times independently ($m_i = 20$ is

suggested in [228], which is considered to be large enough to make the average value represent 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 $0.5 \leq \frac{n_i}{h'} \leq 32$ (it is the recommended setting in [228]), where 0.5 is the starting point of the definition of $\Phi(\frac{n}{h})$ shown in Equation (C.4) and 32 is to make sure that the range of $\frac{n_i}{h'}$ is big enough for various n_i. h' is an initial guess of the VC-dimension of the model. In this study, h' is set to the number of distinct nodes in the GP model. A larger q means more design points, which would lead to a more accurate fit between $\epsilon(n)$ and $\Phi(\frac{n}{h})$ and a tighter VC bound. However, more design points also lead to more computational cost. Thus, to achieve a good balance, the trial 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, that is $m_1 = m_2 = \cdots = m_q = 20$, which is recommended in [228].

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 $\Phi(\frac{n}{h})$ according to $h = \arg\min \sum_{i=1}^{q} [\epsilon(n_i) - \Phi(n_i/h)]^2$, i.e. choosing an approximate h to minimise the error (such as the MSE used in this chapter) between the set of $\epsilon(n)$ and $\Phi(\frac{n}{h})$.

Despite the potential generalisation improvement of GPSRM over standard GP, it still has some drawbacks. One major limitation of GPSRM is the *uniform setting*[2] in the method that measures the VC-dimension of the

[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

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. Accordingly, it limits the effect of SRM on improving the generalisation of GP. To address this problem, a more precise and reliable setting is needed.

4.2.3 GPOPSRM: Measuring the VC-Dimension using Optimised Setting

In applied statistics, there is an important research topic of experimental design, which aims to construct the optimised design for experiments. In previous work [115, 135], the experimental design is iteratively improved by exchanging the design points according to the optimality criteria. In [201], this idea is introduced into the process of measuring the VC-dimension of linear models and shown its effectiveness.

Since we aim to make further improvement on measuring the maximum deviation by employing a better setting, and also motivated by the idea of constructing an optimised design and the promising results achieved in [201], we further propose an improved method to measure the VC-dimension of evolved models in GP to increase the accuracy of the estimated generalisation errors. The proposed method is named *genetic programming with optimised structural risk minimisation* (GPOPSRM).

The idea of optimised experiment design from applied statistics [115, 135] is employed in this study. The process of searching for a better setting

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.

starts from a well-designed setting, then repeats a process of constructing neighbour settings, which have better performance than the current setting, until an "optimal" setting is achieved (i.e. no any better neighbouring setting is available) or the stopping 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 optimisation process is to minimise the error between a set of $\epsilon(n)$ and $\Phi(n, h)$. Thus the evaluation criterion for a setting is set as:

$$MSE = \sum_{i=1}^{q} \sum_{j=1}^{m_i} \left(\epsilon\left(n_{i,j}\right) - \Phi(n_i, h^*)\right)^2 / (m_i * q) \tag{4.10}$$

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 work). 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 $\Phi(n, h)$.

To improve the setting, we need to adjust m_i for each design point appropriately. To find a better setting, the neighbours of current setting are obtained and evaluated. A neighbouring setting can be reached by decreasing the number of experiments by one on the worst design point while running the experiments one more time on the best design point. The quality of a design point n_i is measured by its contribution (which is defined as $\epsilon(n_i) - \Phi(n_i, h)$) to the overall MSE in Equation (4.10), the smaller the better. The procedure of measuring VC-dimension using the optimised setting is presented as follows:

1. Measure the $\epsilon(n_i)$ and $\Phi(n_i)$ on q design points, each for m_i times. Here, $n_i \in \{n_1, n_2, ..., n_q\}$, $m_i = 20$ and $q = 15$ (i.e. the process starts from the uniform setting).

2. Calculate the VC-dimension h^* by finding the best fit among all the various $\epsilon(n_{i,j})$ and $\phi(n_i/h)$. Here each $\epsilon(n_{i,j})$ 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 $\{n_1, n_2, ..., n_q\}$ are potentially different now.

3. Calculate the MSE according to Equation (4.10).

4. Rank the design points $n_1, n_2, ..., n_q$ according to theirs contribution $CB(n_i)$ to MSE, $CB(n_i) = (MSE(\text{remove } n_i) - MSE)/n_i$ (the contribution is normalised by the number of instances).

5. Construct a neighbouring 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 the one that yields to lower MSE^* is found).

7. Repeat Steps 2 to 6 until no design point has a 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.

8. Obtain the VC-dimension value h that can create (near) the best fit between the set of $\epsilon(n_i)$ and the theoretical values of $\Phi(n_i)$.

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

For GPSRM and GPOPSRM, note that it is only necessary to measure the VC-dimension for a number of top/best individuals ranked according to their empirical errors (i.e. RMSE). 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 all the evolved models is expensive. Therefore, we define a parameter γ in GPSRM and GPOPSRM, so that only the best 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 study). The setting is to make the evolutionary process focus on the comparison of estimated generalisation error between the best γ percent of individuals and make the method more efficient.

4.2.4 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 methods, i.e. GPSRM and GPOPSRM, the solutions are evaluated by the estimated generalisation error given by SRM. Assuming the VC generalisation error bound is tight, the fitness function is defined as:

$$Err_{exp} = \frac{RMSE}{\left(1 - \sqrt{p - p\ln p + \frac{\ln n}{2n}}\right)_+} \tag{4.11}$$

where

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

is the error of the models on the training data, $\left(1 - \sqrt{p - p\ln p + \frac{\ln n}{2n}}\right)_+^{-1}$ is the confidence interval between the empirical/training error and the esti-

mated generalisation 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 then causes a larger confidence interval $\left(1 - \sqrt{p - p\ln p + \frac{\ln n}{2n}}\right)_+^{-1}$. Consequently, given the same/similar values of $RMSE$, a higher h will lead to a larger generalisation 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 pattern in the training set and difficult to generalise well on unseen data. By assigning a higher estimated generalisation error to those models and decreasing the probability of selecting them for breeding, our new GP methods are expected to eliminate or decrease the trend of overfitting thus generalise well.

4.3 Experiment Design

To investigate the generalisation ability of GPSRM and 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.

4.3.1 Benchmark Problems

Due to the lack of benchmarks (datasets) specially designed for testing the generalisation ability of GP for symbolic regression, 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 [233, 116, 226]. These benchmark

Table 4.1: 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
$f_1 = e^{-x}x^3 cosx sinx(cosx sin^2 x - 1)$	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])
$f_3 = 6sinx_1 cosx_2$	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])
$f_5 = x_1x_2 + sin((x_1-1)(x_2-1))$	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 4.2: Real-World Problems

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

problems have been shown to be prone to overfitting, therefore generalisation 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 4.1. The first four functions are taken from [233]. Despite the low dimensionality, they are claimed to be difficult regression tasks. The other four problems are from [116]. 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.

We also test the methods on two high-dimensional real-world regression datasets, which are shown in Table 4.2. The first dataset is from the field of pharmacokinetics [14]. 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 [223, 224, 226] to investigate the generalisation 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 (represent as DLBCL) dataset [191]. The task is to predict the survival time of patients who have diffuse large-B-cell lymphoma and received chemotherapy. In DLBCL, the training set and the test set are provided.

4.3.2 Benchmark Algorithms for Comparison

To further investigate and confirm the effect of SRM on estimating the generalisation performance, comparisons between GPSRM, GPOPSRM and the following two GP methods have been conducted:

- Standard GP, which is a baseline for comparison.
- GP with 0.632 Bootstrap (BGP). We would like to compare the proposed approach with GP methods that also take the estimation of generalisation error into account during the evolutionary process. In [51], GPSRM was compared with a state-of-the-art method namely Bias/Variance Error Decomposition (BVGP) [3]. In BVGP, the generalisation error of a GP model is assessed by two aspects, the bias error and the variance error. The 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 in [51] show BVGP generally has worse generalisation performance than GPSRM on the examined benchmark problems (therefore, the results of BVGP are not included in this thesis). The underlying reason might be that, due to the potential overlap of instances in bootstrap datasets and the original training set, standard bootstrap does not provide a good estimation of variance error [83]. Therefore, in this work we decide to compare with an improved version of BVGP employing the 0.632 bootstrap [72], which is named following the

fact that the probability of an instance appearing in a bootstrap set is 0.632. Under 0.632 bootstrap, the definition of the estimation of generalisation error as follows:

$$R_{est} = .368 * R_{emp} + .632 * V_{err} \tag{4.12}$$

$$V_{err} = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{|C^{-i}|}\sum_{b\in C^{-i}}\left(E\left(D^{*b}\right)\right) \tag{4.13}$$

where R_{est} is the estimated generalisation error, R_{emp} is the empirical error, and V_{err} is the variance error. V_{err} is estimated by the leave-one-out bootstrap, which is different from the traditional bootstrap. Leave-one-out bootstrap is a smoothed version of cross validation. As shown in Equation (4.13), n is the number of training instances. C^{-i} refers to the set of bootstrap samples which do not contain the training instance i, and $|C^{-i}|$ is the number of such bootstrap samples. $E(D^{*b})$ is the error of each bootstrap set 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. In this work, we set $B = 200$, which is a recommended setting in [83]. The 0.632 bootstrap is shown to overcome the shortcoming of standard bootstrap and have a better ability in estimating the variance error. Thus, we decide to compare GPOPSRM with GP employing this improved version of bootstrap, namely BGP.

These four GP methods use different indicators for the generalisation 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 generalisation of GP. All the examined GP methods are implemented under the ECJ GP framework [142].

4.3.3 Parameter Settings

The parameter settings for the four GP methods (GP, BGP, GPSRM and GPOPSRM) can be found in Table 4.3. Following the settings in [116, 233],

Table 4.3: 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 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 experiments, the parameter γ is set to 20%, which is sufficiently large for not missing individuals that have potentially good generalisation 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 generalisation performance of the four methods.

4.4 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 generalisation performance in more detail, the evolutionary plots drawing the median test

RMSE of the 100 best individuals on every generation are provided. Further analyses on the model size and model behaviour are also presented.

The Wilcoxon test, which is a non-parametric statistical significance test, is conducted to compare the 100 best training RMSEs and the corresponding test RMSEs. The significance level is 0.05. 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 vs. BGP and GP vs. GPSRM).

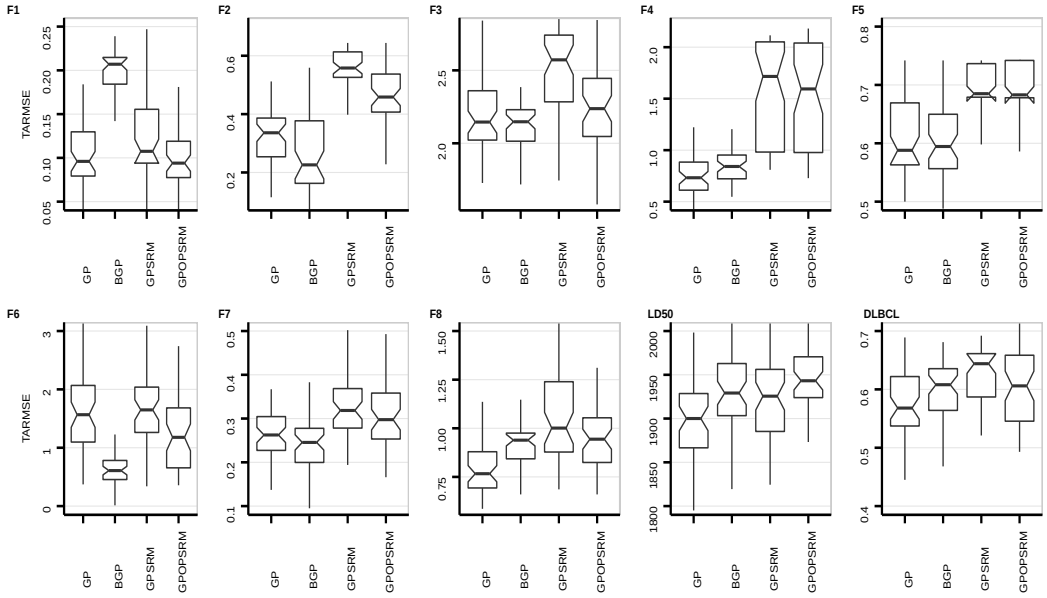


Figure 4.4: Distribution of RMSE of the 100 Best-of-run Individuals on the **Training** Sets.

4.4.1 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 boxplots in Figure 4.4 and Figure 4.5, respectively. The overall pattern is that the two SRM-driven GP methods (GPSRM and GPOPSRM) generally have worse learning perfor-

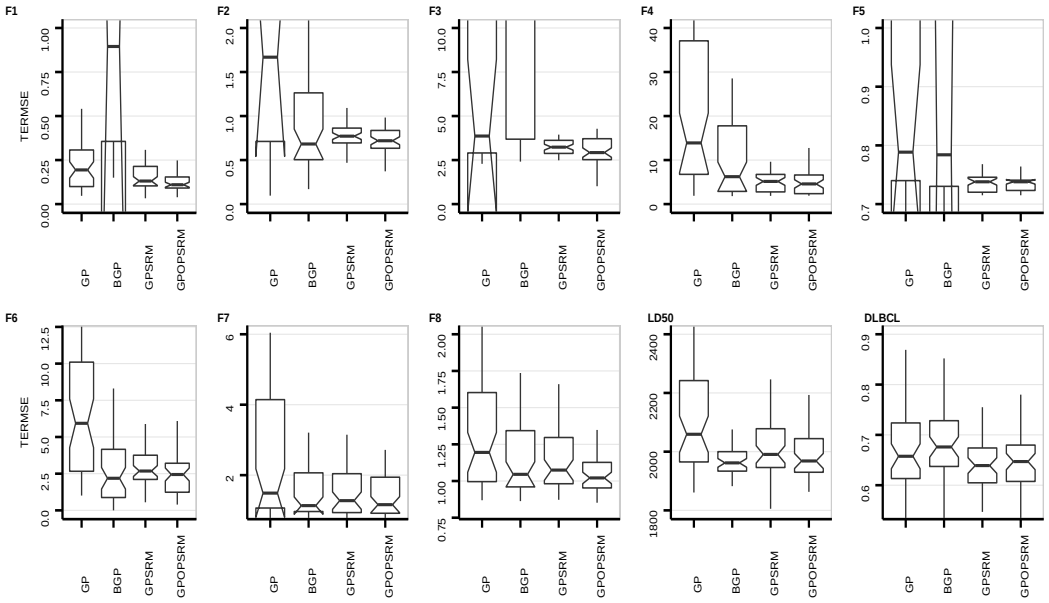


Figure 4.5: Distribution of the Corresponding **Test** RMSE of the 100 Best-of-run Individuals.

mance (shown in Figure 4.4) but much better generalisation performance (demonstrated in Figure 4.5) than standard GP and BGP on the examined datasets.

Learning Performance on Training Sets

As shown in Figure 4.4, on most of the ten training sets, both of the two SRM-driven GP methods have a worse training performance than standard GP. On seven of the ten training sets, GPOPSRM has much higher training RMSEs than GP (except for f_1, f_3 and f_6). The training advantage of standard GP over GPOPSRM on these seven datasets is significant. While GPOPSRM has a better training performance (in median) than GP on f_1 and f_6, and a slightly larger training error on f_3, 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 its counterpart in SRM. This is confirmed by the fact that on three of these training sets (f_2, f_6 and f_7), BGP has better training performance than GP.

Generalisation Performance

Compared with the training performance, we are more interested in the generalisation performance, which is a more important criterion for the success of the learnt model. The overall pattern is very clear in Figure 4.5. Both GPSRM and GPOPSRM have significantly better generalisation performance, i.e. a much smaller RMSE, 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 Figure 4.5, on most of the test sets, GPOPSRM has much lower median values than GP, which indicates that GPOPSRM has much bet-

ter generalisation 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 generalisation 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 generalisation 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 generalisation 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 it is difficult for bootstrap to provide a good estimation of the generalisation performance. Compared with BGP on the two real-world datasets with a larger number of training instances, GPOPSRM has significantly better generalisation 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 vs. 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 generalisation 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 generalisation performance than GPSRM on most 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 significant level. The advantage of GPOPSRM over GPSRM is due to the non-uniform setting for measuring VC-dimension of evolved models, which is the major difference between the two methods.

4.4.2 Evolution of Generalisation Performance

Since the capability of generalisation is the focus of this chapter, we will examine the generalisation performance in more detail. The evolutionary plots on the test sets are drawn using the median corresponding test RMSE over the 100 best-of-generation models. On every generation, the generalisation performance of the best-of-generation model on the test sets is recorded, but the test sets *never* take any part in the evolutionary process.

It can be observed that overfitting occurs in GP in most cases. GP has an increasing generalisation 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. 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 small number of training instances and/or the smaller ratio of the number of instances over features in the training sets. BGP, which relies on the bootstrap of the training instances to estimate the variance error, fails to generalise 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 gener-

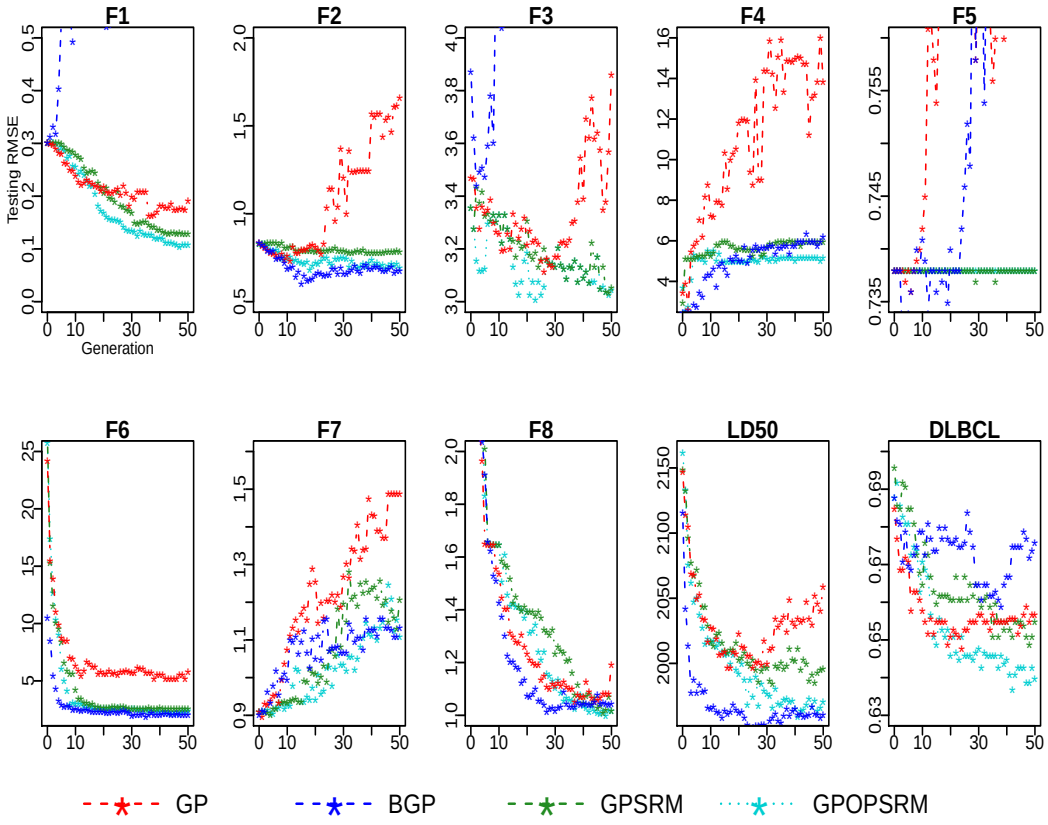


Figure 4.6: Evolution Plots of the Median RMSE of the Best Individual on Every Generation on the **Test** Sets.

alise 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 generalisation errors significantly, but still overfit. On the other test sets, where GP does not overfit or overfits slightly, the two SRM-driven GP methods generalise very well. The pattern of generalisation errors in the evolutionary process confirms the advantage of SRM principle over the empirical risk minimisation principle. In other words, guiding the evolutionary process by the estimated generalisation error typically leads to a better generalisation 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 generalises better than GPSRM. The improvement on the generalisation performance is brought by the non-uniform setting in GPOPSRM, which outperforms the uniform setting in GPSRM in two aspects. Firstly, the optimised setting can reduce the random variability of the measured VC-dimension by removing the relatively large MSE as defined in Equation (4.10). Secondly, compared with the uniform setting, the non-uniform 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 accurate VC-dimension of evolved models, therefore will achieve a better generalisation estimation. The advantage of GPOPSRM confirms the expectation that a better estimation of the VC-dimension of the evolved models can lead to a lower generalisation error.

4.4.3 Further Analysis

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

Structural Level

To examine how SRM influences the model complexity in GP, we draw the evolutionary plots to show the relationship between RMSEs and VC-dimensions in both GP and the SRM-driven GP. 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 the fitness value, i.e. the estimated generalisation 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 totally different from each other

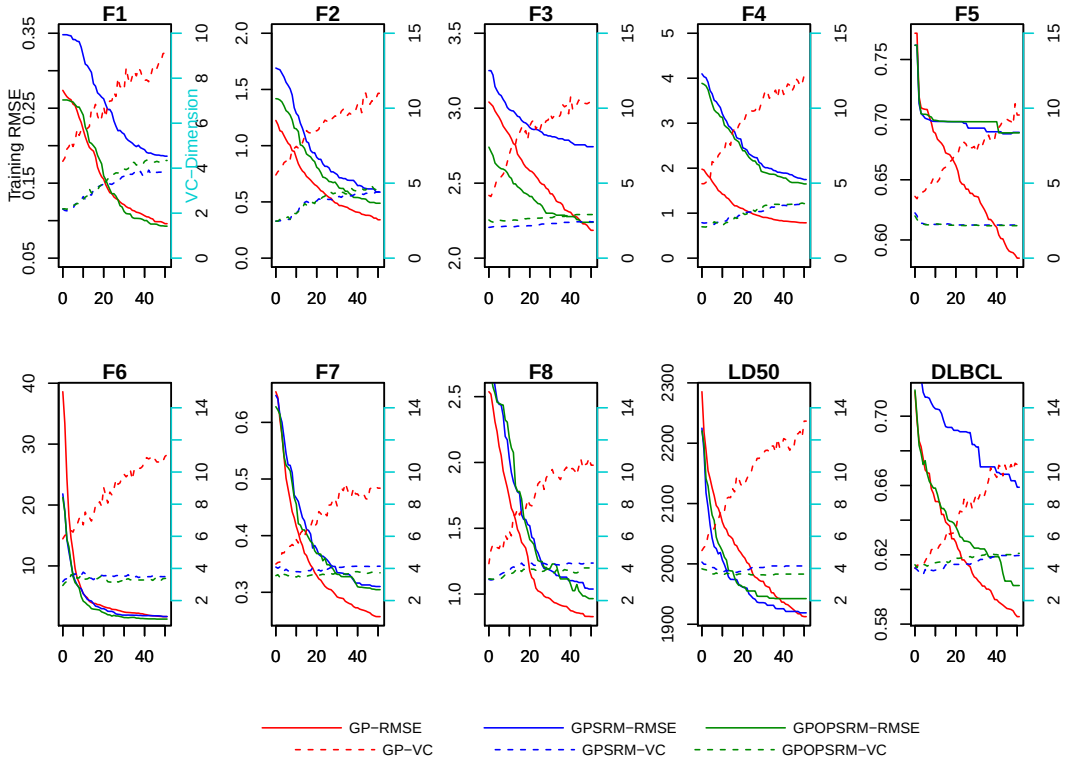


Figure 4.7: Evolutionary Plots on RMSEs and VC-dimensions.

from the first generation (e.g. on $f1$, $f2$, $f3$, $f4$). For standard GP, the VC-dimension of the best solution is measured and recorded, but it never plays a role in the GP evolutionary process.

As Figure 4.7 shows, the overall pattern is that the VC-dimension of the best solutions increases along with the generation, while 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 $F4$ and $F6$). 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 when comparing the VC-dimensions of GPSRM and GPOPSRM, the VC-dimensions in these two methods are slightly different. In some cases, this small difference brings a big difference in RMSE (e.g. on $f1$, $f2$, $f3$, 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 that there might exist a certain threshold for the model complexity. Under this threshold, the increase in the model complexity brings notable benefits for the training performance. When the complexity is above the threshold, it is difficult to improve the performance by increasing the model complexity. We will investigate this in future work.

Behavioural Level

On the behavioural 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 (from the 100 groups) of the best-of-run models on f_6, where the four algorithms all have good generalisation performance. They are displayed in the *Evolved Model* column of Table 4.4. To make the behaviour of the evolved models more obvious, we present the mathematically simplified form of these models. The original evolved models confirm that SRM-driven GP methods can evolve more compact models with a simpler structure. The behaviour 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 generalise 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 behavioural similarity between these models and the target models is higher than their counterparts in BGP and GPSRM.

Computational Cost for Measuring SRM

The computation cost in both GPSRM and GPOPSRM is much higher than standard GP, usually more than 10 times higher. Here, we summarise the additional computational effort needed when introducing SRM into GP under the uniform setting. Using the non-uniform setting needs more effort.

The major cost in SRM-driven GP methods is spent on measuring the VC-dimension of the solutions experimentally. To measure the VC-dimension of programs, when using the uniform setting, SRM-driven GP needs $102*(3000+600)$ additional evaluations than standard GP on every generation. 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 VC-dimension of a model is obtained by fitting the theoretical maximum deviations to the experimental values on 15 different design points (i.e. 15 different numbers of instances). On each design point, the deviation between the errors of the model on 2 datasets (with the same number of instances) is calculated 20 independent times. To approximate the maximum deviation, the mini-batch gradient descent is employed to train the model to gain the largest error on one dataset while achieving the smallest error on the other dataset. On one combined dataset (i.e. a combination of the two datasets), a model is trained for 10 steps by gradient descent. On each step, the model is evaluated on a number of in-

Table 4.4: 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
GP	(+(+(0-(sqrt(*(0-(sqrt(*(*-4.604x_2)x_1)))x_1))))(*(sqrt(+(% 1.0 (*(sqrt(+-4.604(* -10.18 x_1)))(*(*(% 1.0 x_2) x_1) (+ x_2 x_1))))(* -10.182 x_1)))(*(*(*(*0.343 x_1)(% 1.0 x_2)) x_1)(0- (+ (0- x_2)(* -4.604 x_2))))))(* 0.343 (* (* 0.343 x_1(% 1.0 x_2)) x_1)))	$1.92x_1^2 * \left(\frac{1}{\left(x_1 + \frac{x_1^2}{x_2}\right) * (-10.18x_1 + 4.6)^{1/2}} - 10.18x_1 \right)^{1/2} + \frac{0.117x_1^2}{x_2} - (-x_1 - (-4.6x_1x_2)^{1/2})^{1/2}$
BGP	(+(+(*(0–0.332)(sqrt(0–0.332)))-0.95)(+(*(+(sqrt(0-x_2))(* -0.978 x_1))(+(* x_1 x_1)x_1))(+(+(+(+(*(*x_1 x_1)(* x_1 x_1))(* -0.978 x_1))(sqrt x_2))(0- x_2))(sqrt x_2))))	$x_1^4 - 0.978x_1^3 - 0.978x_1^2 - 0.978x_1 + 2\sqrt{x_2} - x_2 + 0.758$
GPSRM	(+ (0- (* (* x_1 x_1) x_1))(+ (sqrt (+ (* (0-(+x_2 x_2))(sqrt (+ (0- (+ (+ x_2 x_2) x_2))(* (0- (+ x_2 x_2))(* x_1 x_1)))))(+ x_2 x_2)))(* (* x_1 x_1)(* x_1 x_1))))	$x_1^4 - x_1^3 + \sqrt{2x_2\left(1 - \sqrt{x_2\left(-2x_1^2 - 3\right)}\right)}$
GPOPSRM	(+ (sqrt (+ (sqrt (* (0- (+x_2 (+ x_2 x_2)))(+ (*(* (+x_2 x_2) x_2) (* x_2 (+ x_2 x_2)))(sqrt (* x_1 x_1))))) (0- (sqrt(0- (* (+ x_2 x_2) (+ x_2x_2))))))) (* (+ (0- x_1)(* x_1 x_1))(* x_1 x_1)))	$x_1^4 - x_1^3 + \sqrt{\sqrt{-3x_2(x_1 + 6(x_2)^2)}}$
GP	(+(+(0-x_2)(*(0-(* x_1 x_1)) (+ (0- (*x_1 x_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	(+ (* (* x_1 x_1)(sqrt(+ (+ (+ (* (* x_1 x_1)(* x_1 x_1))(*(% 1.0 x_1) (+ (% 1.0 x_1) x_2)))(*(% 1.0 x_1) (0- x_1)))(* (% 1.0 x_1) (+x_1 x_2))))))(*(0- x_1)(+(sqrt(+(*(*x_1 x_1)(* x_1 x_1))(*(% 1.0 x_1)(+ x_1 x_2))))(sqrt (% 1.0(+ (+ (0–0.456)(+(% 1.0 0.191)(+ x_1 x_2)))(+ x_2(+(* x_1 x_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- (*(sqrt x_1) x_2))(* (* x_1 x_1)(0- x_1)))(+ (* (* x_1 x_1) (* x_1 x_1)) (+ (sqrt(* (* (sqrt(* (sqrt x_1) x_2)) x_2) x_2))(+ (* (*(+ (0- (*(sqrt x_1) x_2)) (* (* x_1 x_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 (* (* (* (0–0.192)(0- -0.192)) x_2)(* (0- -0.192)(0–0.192)))) x_2)x_2) (+ (*x_1 (*(*x_1 x_1) x_1))(0- (* (* x_1 x_1)x_1))))	$x_1^4 - x_1^3 + 0.036x_2^{5/2}$

stances (at most 50 instances) to get the gradient. Therefore, to obtain the maximum deviations of one model, $15 * 20 * (10 + 2)$ additional evaluations are performed independently. While the $3000(= 15 * 20 * 10)$ evaluations are performed on the subsets, the $600(= 15 * 20 * 2)$ evaluations are performed on the whole datasets. These evaluation cost scales with the size of the models, since the number of instances $_{\{n_1,n_2,...,n_{15}\}}$ in these datasets are determined by the number of coefficients p in the program ($\{n_1, n_2, ..., n_{15}\} = \{0.5, 0.8, 1.0, 1.2, 2, 2.5, 3, 3.5, 5, 6.5, 8, 10, 15, 20, 30\} * p$).

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.

Although at a considerable computational cost, the experimental estimation of the VC-dimension provides a practical solution to measure the model complexity. More importantly, the positive effect of SRM-driven GP on the generalisation of GP confirms that it is feasible to utilise SRM to achieve a proper trade-off between model accuracy and complexity. This trade-off has been desired but is lack of solution in GP community for long. This is perhaps a very first work to demonstrate that the theory on VC-dimension is practically useful for GP to improve its generalisation performance.

4.5 Chapter Summary

This chapter proposed new GP approaches including two methods GP-SRM and GPOPSRM by incorporating structural risk minimisation (SRM) into GP to improve the generalisation ability of GP for symbolic regression. SRM has solid theoretical foundation and is able to provide tight generalisation error bound for models. This study extended the *experimental* method to measure the VC-dimension for regression models and made 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 an impressive generalisation gain over standard GP on all the ten examined datasets. Moreover, the better generalisation performance in SRM-driven GP than BGP confirms the advantage of SRM as a framework to estimate generalisation error. This study also conducts a comparison between GPOPSRM and GPSRM, which are SRM-driven GP methods using non-uniform and uniform settings, respectively. The results confirm that GPOPSRM outperforms GPSRM in both the training performance and the generalisation ability on most of the examined problems. Further analyses of the evolved models show that GPOPSRM not only evolves more compact models but also approximates the behaviour of the target functions better than the other methods.

Although taking more computational efforts, the proposed SRM-drive GP methods represent perhaps the first approach showing that the VC-dimension is practically useful to measure the complexity of the evolved programs for estimating the generalisation of GP for symbolic regression.

In this chapter and the previous one (Chapter 3), the proposed methods eliminate/reduce overfitting thus improve the generalisation of GP. However, overfitting is not the only reason for poor generalisation in GP. In scenarios when the generalisation performance of GP is poor but no overfitting occurs, the proposed methods might not work. Meanwhile, the proposed methods in Chapters 3 and 4, which are including feature selection to decrease/eliminate the possibility of incorporating irrelevant/noise features into the evolved models and restricting the model complexity, seek for a better generalisation of GP mainly from the structural aspect. Another very important and promising aspect — the behavioural information of the evolved models is worth exploring further. Therefore, the next chapter will propose a new method to improve the generalisation of GP, which relies on utilising the detailed behavioural information of solutions instead of putting effect on avoiding overfitting.

Chapter 5

Angle-driven Geometric Semantic GP

5.1 Introduction

Geometric semantic genetic programming (GSGP) [126, 156] has recently been attractive. The geometric semantic operators in GSGP aim to manipulate the semantics of GP solutions/programs. These operators generally make a bounded semantic impact and generate child programs with similar behaviour to their parents. These properties are shown to be highly related to a notable generalisation improvement in GP. Different from the previous two chapters (Chapters 3 and 4), where new developments were made based on the canonical form of GP to avoid overfitting, thus improve its generalisation, this chapter aims to further explore the geometry of the semantic space of GP to gain a better generalisation ability, particularly when the generalisation of the learnt models is poor but no overfitting occurs. In this case, the difference between the training and test errors might become bigger or remain the same, which is shown in Figure 2.1 (in Chapter 2 Page 19).

5.1.1 Advances and Limitations of GSGP

Semantic Genetic Programming (SGP) [126, 20] is a relatively new variant of GP that considers the semantic information during the evolutionary process. In GP, *semantics* refers to a description of what the GP solution does [126]. The widely adopted definition of semantics in symbolic regression is that the semantics of a program F is a vector V, in which the number of elements is determined by the number of instances n, while the values of these n elements are the corresponding outputs of F over the n instances X, i.e. $V(F) = \{F(X_1), F(X_2), ..., F(X_n)\}$. And the n target values Y form the target semantics $T = \{Y_1, Y_2, ..., Y_n\}$. More detail on how to calculate the semantics of a program can be found in Chapter 2 (Page 32).

GSGP [126, 156] is a particular branch of SGP, where the semantics of a program is considered as a point in an n (n is the number of instances) dimensional space. The semantics of all the candidate solutions in GSGP form a semantic space. In the semantic space, the evaluation of any point is the distance from the target semantics, i.e. the target outputs. Therefore, the surface of the semantic space takes various conic forms according to the distance metrics. More importantly, this conic space is unimodal, i.e. the minimum error can only be obtained at the target point, and no plateaus exists. Searching in such an unimodal space is easy and promising *in principle*. Therefore, GSGP provides a formal *theoretical framework* for designing geometric search operators [156]. The framework (the detailed definition is given in Chapter 2 Page 34) defines the desired semantic properties of the offspring generated by the geometric semantic operators.

Guided by the theoretical framework in GSGP, a number of *implementation algorithms* have been proposed for formalising the geometric properties and various forms of geometric semantic operators (geometric operators for short) are presented. In Chapter 2 (Page 58), we have a detailed review on the exact GSGP [156] and the approximated GSGP methods including locally geometric semantic crossover (LGX) [128], approximately geometric semantic crossover (AGX) [127] and Random Desired Operator

(RDO)(mutation) [178]. A good generalisation ability has shown in the existing geometric operators [98, 223], which was justified by their geometric properties. The geometric crossover performs a more constructive reproduction, i.e. more likely to generate child programs with better fitness and generalisation ability than their parents. More importantly, the geometric operators generally bring a small semantic variation to the new generations. This helps GP preserve a high semantic locality, which correlates to the smoothness of the fitness landscape and in turn influences the performance of GP search [125]. High semantic locality does not guarantee to find good solutions in GP. But it can help GP converge to good solutions with better learning performance and generalisation ability by improving the exploitation ability of GP.

Despite the promising performance of GSGP, there exist a number of major limitations in the existing GSGP methods. The potential *ineffectiveness* of geometric semantic crossover should be noted. The geometric crossover can produce child programs that outperform both parents only when the target semantics are between the semantics of the two parents. This ineffectiveness holds 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, which 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 bound. Too large mutation steps might lead to a decrease on the training error but an increase on the test error when overfitting appears, while too small mutation steps decrease the exploration/search ability of GP, thus might lead to underfitting and poor generalisation performance.

5.1.2 Chapter Goals

To address the above issues, the overall goal of this chapter is to develop new geometric operators for obtaining a greater generalisation improve-

ment in GSGP for symbolic regression than the existing geometric operators. The new geometric operators will be designed to further utilise the geometric properties of the semantic space and incorporate angle-awareness. Specifically, this work has three objectives as follows:

- Whether and how the proposed geometric operators can improve the learning performance of GSGP for symbolic regression over the state-of-the-art GSGP methods, and the canonical GP method,
- Whether and how the proposed geometric operators can further improve the generalisation ability of GP for symbolic regression over the state-of-the-art GSGP and canonical GP methods (particularly when no overfitting occurs in the canonical GP), and
- Whether and how the proposed geometric operators influence the size and interpretability of the evolved programs in GP.

5.1.3 Chapter Organisation

The remainder of this chapter is organised as follows. The second section describes the proposed angle-driven GSGP method. The third section describes the design of the experiments. The results and discussion are presented in the fourth section. The fifth section provides a summary of this chapter.

5.2 The Proposed Method

The theoretical framework in GSGP has shown to be positive in enhancing the generalisation of GP. However, the geometric operators defined by the framework still have the potential limitation, which restrict their effectiveness on gaining better generalisation ability. This work attempts to further explore the geometry of these search operators to make them more constructive. To this end, we incorporate angle-awareness into GSGP and

make the angle-awareness as a main force to drive the evolutionary process of GSGP. The proposed GSGP method is thus named *angle-driven geometric semantic GP* (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. The following subsections will describe the five key components of ADGSGP, which are the angle-distance, measurement, the angle-awareness driven selection, crossover and mutation.

5.2.1 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 arccos of the dot product of their normalised vectors. The definition is given as follows:

$$\gamma = \arccos\left(\frac{\vec{V}_1}{\|\vec{V}_1\|} \cdot \frac{\vec{V}_2}{\|\vec{V}_2\|}\right) \tag{5.1}$$

where the normalised vector/semantics is

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

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 normalised vectors with the normalised 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:

$$\gamma_r = \arccos\left(\frac{(\vec{V}_3 - \vec{V}_1)}{\|\vec{V}_3 - \vec{V}_1\|} \cdot \frac{(\vec{V}_3 - \vec{V}_2)}{\|\vec{V}_3 - \vec{V}_2\|}\right) \tag{5.2}$$

where the normalised relative semantics is

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

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

5.2.2 Angle-Driven Selection

We introduce an angle-awareness mating scheme to geometric crossover in [49] (More details on angle-awareness mating scheme can be seen in Appendix D). 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 the limitation of angle-awareness mating scheme and further utilise the angle-awareness for selecting better parents, a new selection operator named *angle-driven selection (ADS)* is proposed to select pairs of parents for geometric crossover in ADGSGP. 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 can potentially bring several benefits to the evolutionary process. First, they help 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 [49]. Compared with the angle-aware mating scheme in [49], 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 [49] selects the satisfied parent pairs from the winners of the tournament selection. The pseudo code of ADS is shown in Algorithm 3.

5.2.3 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 been considered to improve the effectiveness of the geometric crossover by introducing the geometry of the target semantics.

Algorithm 3: Angle-Driven Selection

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 (5.2);
    if $\gamma_r > ta$ **then**
      $p_2 = cp_2$, $f = true$;
      Stop the loop;
    else
      Put cp_2 into cl;
    end
  end
  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;
  end
  Put the selected pair p_1 and p_2 into l
end
return l;

A better geometric crossover can produce offspring that is not only highly correlated to the parent semantics but also effective in approximating the target semantics.

To this end, a new crossover operator named *perpendicular crossover (PC)* is proposed. PC imposes a more precise semantic requirement than exact geometric crossover. In this way, it aims to drive the search converging to the target semantics much faster. Given two parent individuals,

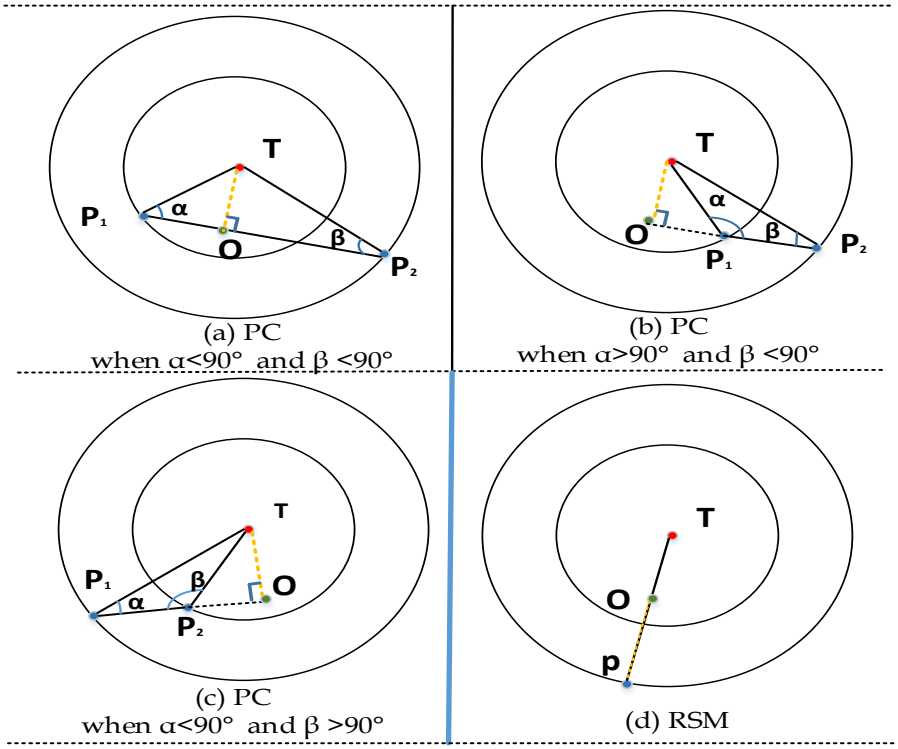


Figure 5.1: Offspring generated by PC (a-c) and RSM (d).

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 Figure 5.1 (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 $\vec{T}$, $\vec{P}_1$, $\vec{P}_2$, according to Equation (5.2), it is straightforward to obtain the values of these two angles.

To obtain the semantics of the offspring $\vec{O}$, it needs 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 Figure 5.1(a-c), according to the values of α and β, there are three possible positions of $\vec{O}$. In the

first case (as shown in Figure 5.1 (a)), when α and β are both smaller than 90 degrees, the offspring $\vec{O}$ (represented by the green point in the figure) stands in the segment of $(\vec{P_1} - \vec{P_2})$. In the other two cases where either α or β is larger than 90 degrees, 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 Equations (5.3) - (5.5) (corresponding to Figure 5.1 (a-c)) as follows.

- when $\vec{O}$ stands on the segment between $\vec{P_1}$ and $\vec{P_2}$, i.e. $\alpha < 90$ and $\beta < 90$

$$\vec{O} = \vec{P_1} + \frac{\|\vec{P_1} - \vec{T}\| \cdot cos(\alpha)}{\|\vec{P_1} - \vec{P_2}\|} \cdot (\vec{P_1} - \vec{P_2}) \tag{5.3}$$

- when $\vec{O}$ is outside the segment on the $\vec{P_1}$ side, i.e. $\alpha > 90$

$$\vec{O} = \vec{P_1} - \frac{\|\vec{P_1} - \vec{T}\| \cdot cos(180 - \alpha)}{\|\vec{P_1} - \vec{P_2}\|} \cdot (\vec{P_1} - \vec{P_2}) \tag{5.4}$$

- when $\vec{O}$ is outside on the $\vec{P_2}$ side, i.e. $\beta > 90$

$$\vec{O} = \vec{P_2} + \frac{\|\vec{P_2} - \vec{T}\| \cdot cos(180 - \beta)}{\|\vec{P_1} - \vec{P_2}\|} \cdot (\vec{P_1} - \vec{P_2}) \tag{5.5}$$

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 is not worse than the worse parent, but PC guarantees that the offspring program is better than both of its parents.

Algorithm 4: 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 $\vec{O}$;

5.2.4 Random Segment Mutation

Inspired by RDO (Chapter 2 Page 61) and to have a better control of the semantic variation induced by geometric mutation, we consider to utilise the target semantics. We propose an angle-driven geometric mutation operator named *random segment mutation* (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 (Figure 5.1 (d)).

The pseudo code of the procedure to obtain the desired semantics in RSM is shown in Algorithm 4. Given a parent $\vec{P}$, RSM firstly 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 $\vec{O}$. Given the semantics of the parent $\vec{P}$ and the target $\vec{T}$, the desired semantics of $\vec{O}$ is calculated according to Equation (5.6), which has the same principle as Equations (5.3) - (5.5).

$$\vec{O} = \vec{P} + k \cdot (\vec{T} - \vec{P}), k \in (0, 1) \tag{5.6}$$

This requirement on the semantics of the offspring follows the theoretical requirement in geometric mutation and meanwhile makes the angle between the relative semantics of $(\vec{O} - \vec{P})$ and $(\vec{T} - \vec{O})$ be 180 degrees. 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. On the other hand, compared with RDO,

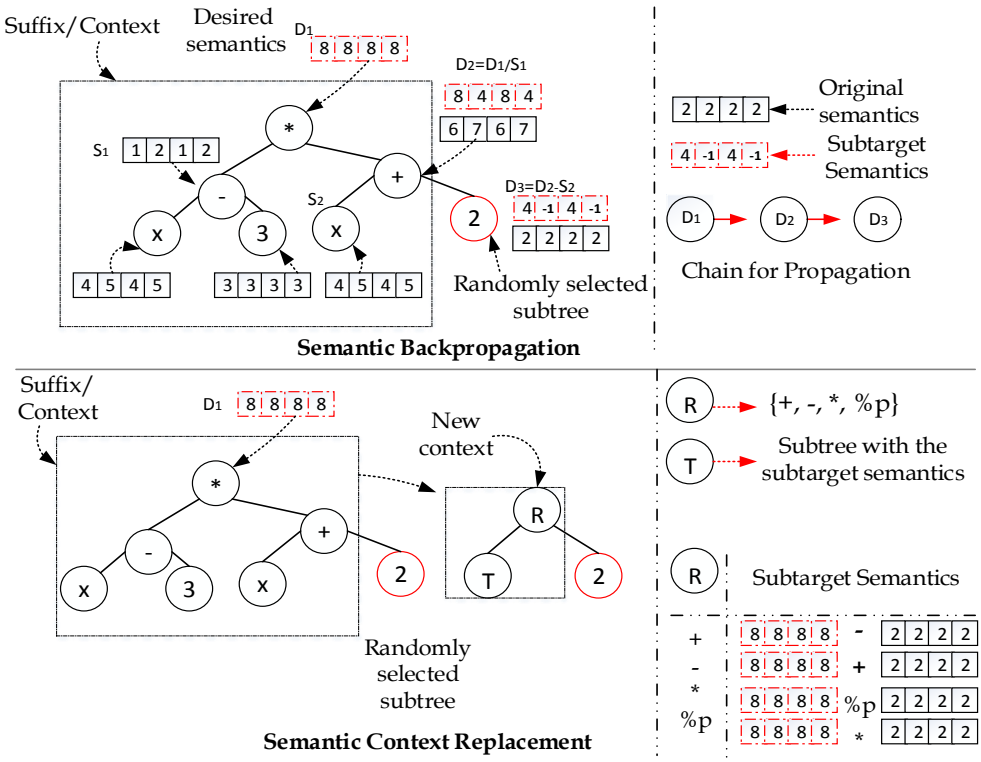


Figure 5.2: Semantic Backpropagation and Semantic Context Replacement.

which only considers the unique target semantics to be the desired semantics for all the offspring, RSM requires varying semantics for the offspring, which is more likely to maintain the semantic diversity and increase the exploration ability of GSGP.

5.2.5 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 [52], semantic backpropagation (SB) is employed, which aims to achieve the desired semantics. In SB, a parent tree is split into two parts, i.e. a suffix and a prefix by a randomly selected

Algorithm 5: Semantic Context Replacement

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 binary operator from the function set to be the root node R, i.e. select one operator among $\{+, -, *, \%protected\}$;

Calculate the subtarget semantics for the remain 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 (5.1), and put it into AL;

end

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 (5.7)

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), and obtains the subtarget semantics by propagation backward through these complex trees, which is still very time consuming. Meanwhile, over-complex models are often prone to overfit the training set and do not generalise well on the unseen data. All these limitations prevent SB from being applied to problems with a lager semantic space.

To avoid the potential limitations of SB, we further propose a new method named *semantic context replacement (SCR)* to fulfil the semantic requirement. The procedures of fulfilling the desired semantics in SB and SCR are shown in Figure 5.2. 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 a GP individual to its fitness, which has been confirmed by previous research [144]. 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 and less complex to calculate the desired semantics for the context. As shown in Figure 5.2, 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 $\%p$ 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.

The pseudo code of SCR is shown in Algorithm 5. Different from SB, which searches for a tree that has the closest distance with the subtarget semantics from the semantic library, SCR considers the angle-distance between the semantics of candidate trees (according to Equation (5.1)) and the subtarget semantics. SCR selects the tree TR having the smallest relative angle-distance. In this chapter, a 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 hypothesise that the subtree with the shortest angle-distance to the desired semantics is most likely to have the desired structure. Then performing a linear scaling will help find better coefficients to the selected subtree so that it can fit the subtarget semantics better. Compared with obtaining the subtree which has the closest distance with the subtarget semantics in SB, SCR is expected to provide a better context for the child

programs that will not only fit the target semantics better but also be potentially resistant to overfitting. Therefore, after finding TR, a linear scaling [116] 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 [116], the values of a and b are defined as follows:

$$b = \frac{\sum_{i=1}^{n}[(t_i - \bar{t})(ct_i - \overline{ct})]}{\sum_{i=1}^{n}[(ct_i - \overline{ct})^2]}, \quad a = \bar{t} - b \cdot \overline{ct} \tag{5.7}$$

where t_i is the value of the subtarget semantics in the ith dimension, $\bar{t}$ 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.

5.3 Experiment Design

To investigate and confirm the effectiveness of the proposed ADGSGP method, 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:

- GSGP refers to the exact GSGP method which uses the exact geometric crossover and geometric mutation. The improved implementation of GSGP [223] is used in the experiments.
- AGSGP stands for the approximate GSGP method [178] which employs two state-of-the-art geometric operators: AGX and RDO. Both of these two operators are based on SB.
- Standard GP, which employs standard crossover and mutation, is also used as a baseline for comparison.

All the four GP methods are implemented under the GP framework provided by Distributed Evolutionary Algorithms in Python (DEAP)[82].

Table 5.1: Synthetic Datasets.

The training samples are drawn regularly 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, 10000)

Table 5.2: 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	7399	240	160	80

5.3.1 Benchmark Problems

Following 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 four synthetic datasets given in [147]. The target functions and sampling strategies are shown in Table 5.1. Furthermore, we are also interested in testing the proposed method on real-world datasets, which have not been widely used in research on GSGP. As shown in Table 5.2, compared with the synthetic datasets, these real-world datasets have a much larger number of features/variables and instances, which means larger semantic spaces, and typically contain noise. Implementing the geometry of the geometric operators and approximating target semantics in these larger semantic spaces are clearly more difficult. The first three datasets are taken from UCI [138]. The Housing Data Set (BHouse), which concerns housing values in suburbs of Boston. The Concrete Com-

pressive Strength Data Set (Concrete) is to model the concrete compressive strength. The third one is the (red) Wine Quality Data Set (Wine), the task of which is to predict the quality of the wine. The fourth dataset is the Diffuse Large-B-Cell Lymphoma (DLBCL)[191], which has been used in previous chapters. The training set and the test set are given in DLBCL, while each of the other three datasets is split randomly with 70% of the data for training and the rest 30% for test in each GP run, which is the same as previous chapters. The splitting is different in each independent GP run, but is the same for the four GP methods in one run.

5.3.2 Parameter Settings

The parameter settings for all the four GP methods are summarised in Table D.2. Most of these parameters are common settings in GP and GSGP [124, 156]. 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 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. Root mean square error (RMSE) is adopted as the fitness function. For all the four GP methods, at every generation, the RMSE of the best-of-generation model on the training set and its corresponding test error are recorded. Each GP method has been conducted for 50 independent runs on each problem.

5.4 Results and Discussions

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 generalisation ability and the size of the evolved programs

Table 5.3: 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

in the GP methods. The computational cost will also be shown. The main comparison is conducted between GSGP methods. The non-parametric statistical significance test — Wilcoxon test, with a significance level of 0.05, is conducted 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.

5.4.1 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 Figure 5.3. Each method has two boxes on each problem. Red and blue boxes are for the training set and the test set, respectively. Note that on BHouse and Wine the interquartile range of the box for the test error (in blue) of AGSGP is too large to display. For a better comparison, they were cut off. Table 5.4 presents the results of the two sets of statistical significance

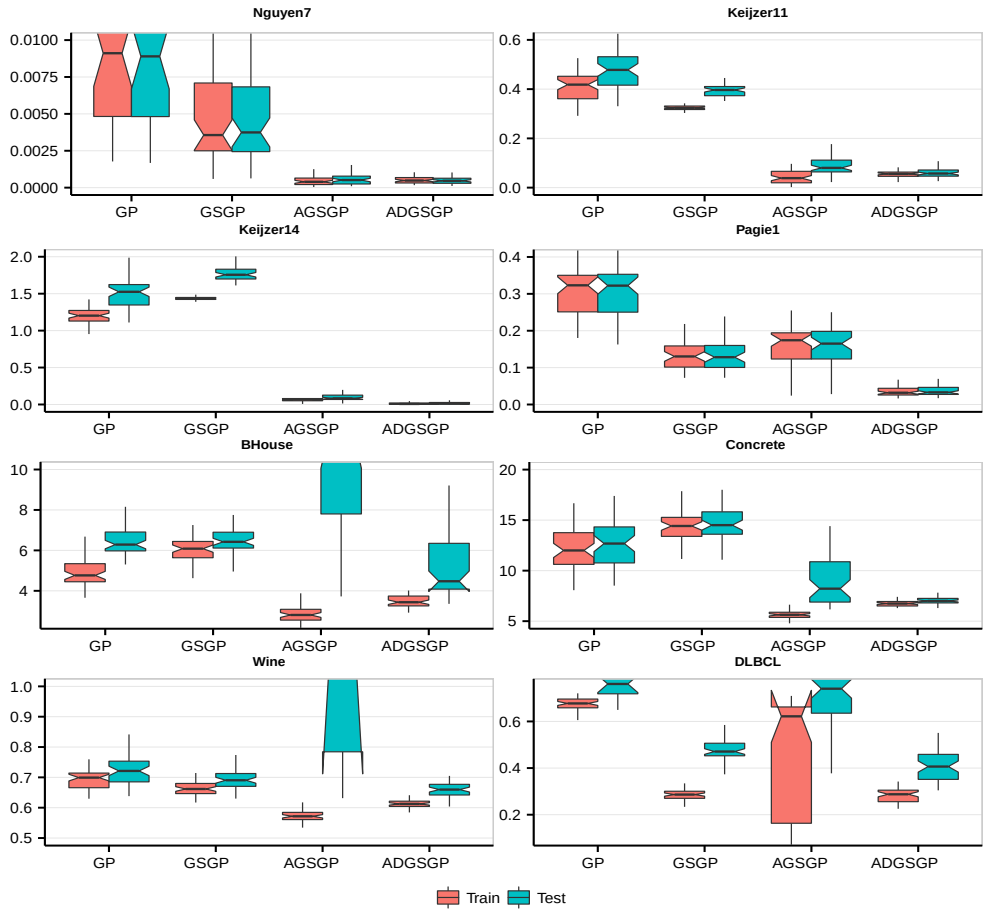


Figure 5.3: Distribution of the Training errors and the Corresponding Test Errors of the Best-of-run Models.

tests. While "−" indicates that ADGSGP (or GP) performs significantly better than the compared method, "+" means that ADGSGP (or GP) is significantly worse, and "=" means no significant difference can be found.

On three of the four synthetic datasets, except for Keijzer14, the three GSGP methods generally have much lower RMSEs than standard GP on both the training sets and the test sets. On Keijzer14, the exact GSGP method produces significantly higher training and test RMSEs than standard GP. The other two GSGP methods significantly outperform standard GP in terms of both the learning ability and generalisation performance.

Table 5.4: Results of Statistical Significance Tests.

Datasets	GP(training, test)			ADGSGP(training, test)		
	GSGP	AGSGP	ADGSGP	GSGP	AGSGP	GP
Nguyen-7	$(+,+)$	$(+,+)$	$(+,+)$	$(-,-)$	$(=,=)$	$(-,-)$
Keijzer-11	$(+,+)$	$(+,+)$	$(+,+)$	$(-,-)$	$(=,-)$	$(-,-)$
Keijzer-14	$(-,-)$	$(+,+)$	$(+,+)$	$(-,-)$	$(-,-)$	$(-,-)$
Pagie1	$(+,+)$	$(+,+)$	$(+,+)$	$(-,-)$	$(-,-)$	$(-,-)$
BHouse	$(-,=)$	$(+,-)$	$(+,+)$	$(-,-)$	$(+,-)$	$(-,-)$
Concrete	$(-,-)$	$(+,+)$	$(+,+)$	$(-,-)$	$(+,-)$	$(-,-)$
Wine	$(+,+)$	$(+,-)$	$(+,+)$	$(-,-)$	$(+,-)$	$(-,-)$
DLBCL	$(+,+)$	$(+,=)$	$(+,+)$	$(=,-)$	$(-,-)$	$(-,-)$

On all the synthetic datasets, ADGSGP significantly outperforms exact GSGP on both the learning and generalisation performance. ADGSGP achieves significantly smaller training errors on Pagie1 and Keijzer14 than AGSGP, and no significant difference on their training performances was found on Nguyen7 and Keijzer11. More importantly, ADGSGP has lower test errors than AGSGP on all the synthetic datasets, where on three of the four test sets (except for Nguyen-7) the advantage is significant. Meanwhile, ADGSGP has more robust performance than the other two GSGP methods in both the learning and generalisation 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 not very similar to that on the synthetic datasets. The superiority of GSGP methods over standard GP is not as obvious as on the synthetic datasets, particularly in GSGP and AGSGP. GSGP performs worse than standard GP on Concrete regarding both the train and the 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 the training

sets and the test sets. On the four real-world problems, AGSGP obtains significantly better training performance than standard GP. However, it has a significantly worse generalisation performance than GP on two test sets (BHouse and Wine), while having comparable generalisation performance with GP on DLBCL, and only generalising 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 generalisation 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 generalisation 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 generalises the best on *all* the real-world datasets among *all* the four GP methods.

To sum up, the overall pattern is that the newly proposed method ADGSGP has comparable training performance to AGSGP, and much better than standard GP and GSGP. More importantly, regarding the generalisation ability, ADGSGP is undoubtedly the winner among the four GP methods. It performs significantly better than all the other three GP methods in all cases except for one case (Nguyen-7), where it performs similar to AGSGP.

5.4.2 Learning performance

To have a closer view on the training performance, here the evolutionary training plots are presented (as shown in Figure 5.4) and analysed in

detail. 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 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 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 towards 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). These datasets have a much higher number of training instances than the other 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

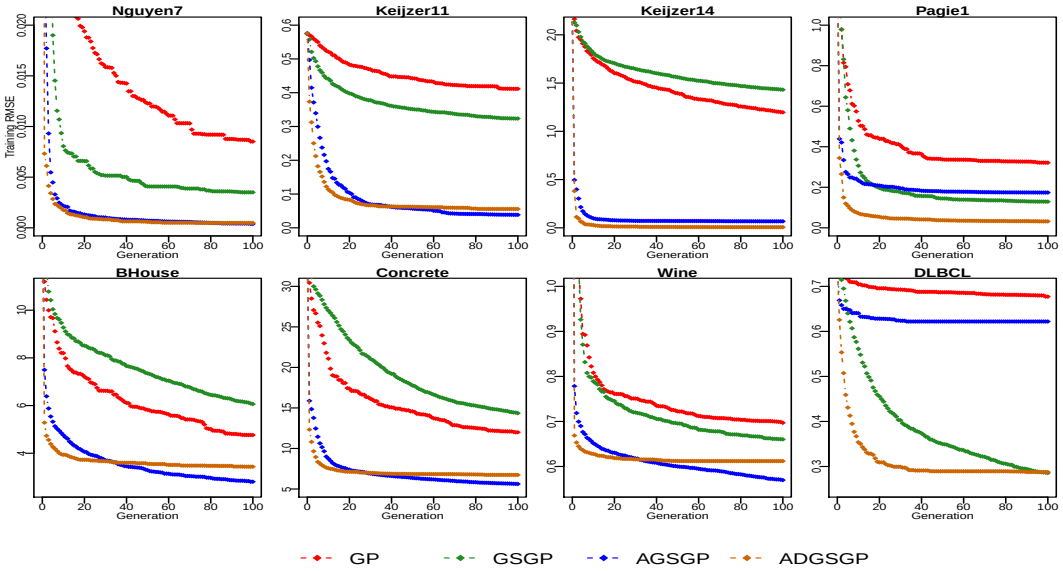


Figure 5.4: Evolutionary Plots on the **Training** Median RMSEs of the Best-of-Generation Models.

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 summarise, 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 BHouse, Concrete and Keijzer14. How to utilise 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 semantics much faster and more accurately than GP and exact GSGP.

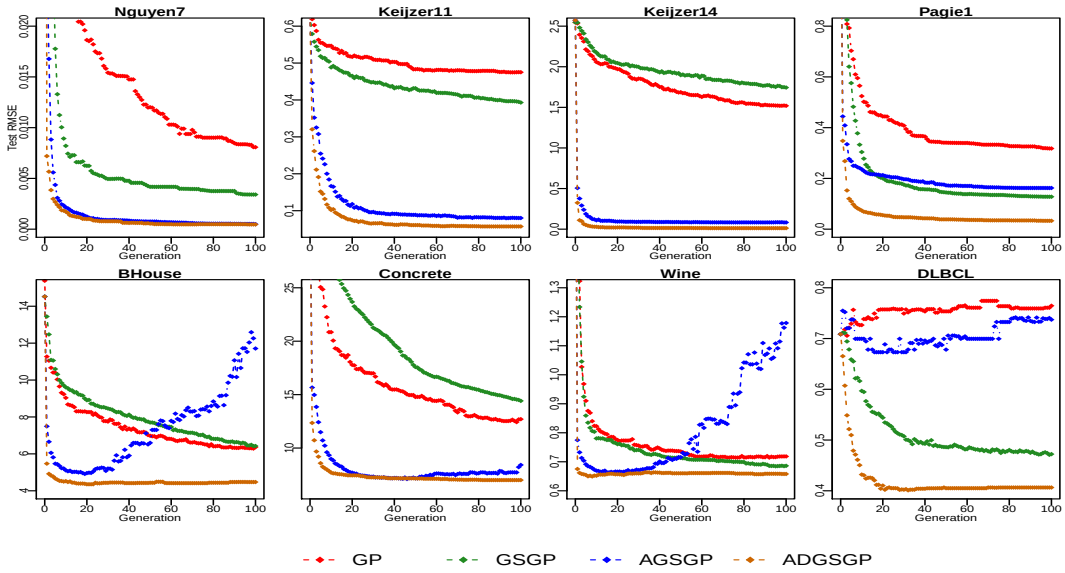


Figure 5.5: Evolutionary Plots on the Corresponding **Test** RMSEs of the Best-of-Generation Models.

5.4.3 Generalisation Performance

Compared with the training performance, we are more interested in the generalisation ability of the GP methods. Figure 5.5 shows the evolutionary plots on the median generalisation errors of the best-of-generation models on the test sets. As shown in Figure 5.5, GSGP continues the advantage over standard GP on five of the eight test sets. While on BHouse and Concrete, where GSGP has worse training performance, it also generalises worse than standard GP. On BHouse, along with the increase of the generations, the difference between the generalisation performance of GSGP and standard GP has decreased. At the end of the evolutionary process, they have almost the same generalisation 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 generalise well on these test sets. Meanwhile, ADGSGP generalises the best on four test sets. It generalises significantly better than GP and GSGP on all these

test sets, and significantly better than AGSGP on three 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 generalise 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 generalise 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 the parent. Along with the evolutionary process, when both AGX in AGSGP and PC in ADGSGP have difficulties to bring further progress to the search process, RSM brings a smaller variation to the offspring than RDO, so it has the ability to limit the deterioration of the generalisation error. In other words, the less greed of RSM approximating the target semantics drives ADGSGP to generalise well and resist overfitting. This can be shown by the fact that the stage where AGSGP advances ADGSGP on the training set is right at the time when overfitting happens in AGSGP. Another possible reason is from SCR in our new method to fulfil the semantic requirement. Compared with SB in AGSGP, SCR induces simpler programs that are more likely to generalise better on the unseen data. This will be confirmed later in the analysis of the evolved programs in Section 5.5.

In general, on most of the test sets, the three GSGP methods generalise much better than standard GP. It confirms that the geometric properties of the geometric operators can hold on the unseen data, i.e. the generalisation error committed by the child program might be bounded by the worst (or best for ADGSGP) parents. It is extremely the case for the synthetic datasets, where the problems are 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 with that on the training sets.

On the other hand, the geometric operators in GSGP may also bring negative variations to GP individuals, which might increase the generalisation 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. In this way, RSM can make the deterioration (increase) of the generalisation error to be slower and consequently lead to a better generalisation in ADGSGP.

5.4.4 Comparisons of Program Size and Computational Time

Table D.3 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 in Table D.3 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 limit on the maximum depth 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 [71], which performs a

Table 5.5: 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)

black-box regression. In AGSGP, the size of the evolved programs is 3 to 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 3 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 affordable. More importantly, the program simplification methods [247, 237] 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 generally generalise better than their counterparts in standard GP on unseen data. This conflicts with the widely accepted theories, such as the Minimum Description Length principle [189] and Occam's razor [214], which claim that complex programs are difficult to generalise well. 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 consid-

ers these oversize programs as ensembles of programs [98]. 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 generalisation error can be reduced/eliminated by their counterparts in the programs that generalise well.

Considering the comparison of the computational cost between standard GP, AGSGP and ADGSGP, AGSGP and ADGSGP have much more expensive computational cost than standard GP. The maintenance and the search procedure for the semantic library in AGSGP and ADGSGP take 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.

5.4.5 Analysis of Evolved Models

A further examination of the models evolved by the three GP methods is conducted (since the models evolved by GSGP are too large to show and analyse, they are omitted from the analysis). We randomly pick two examples of the evolved models from the 50 GP runs on Keijzer14, where the target function is known, and both AGSGP and ADGSGP achieve good learning and generalisation 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 5.6, respectively.

Table 5.6: 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))))))))))	$7.99/(2+x^2+y^2)*(124.49/(2+x^2+y^2)-0.027)*(1/(0.0175+0.004/(x^2+y^2)))$

Continued on next page

Table 5.6 – continued from previous page

Method	Evolved Model	Simplified Model
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)))

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), *(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))))))))))))))))	$8.11/(2.01 + x^2 + y^2)$

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 behaviour of the evolved models, is even more apparent. In the two examples, the functions/models in ADGSGP not only have a higher smoothness than their counterparts in standard GP and AGSGP, but also

are 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 coefficients 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 generalise very well.

5.5 Further Analysis

To further investigate the effect of the three proposed angle-driven operators, PC with standard tournament selection (PC-SS), PC with the proposed ADS (PC), and RSM are tested 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. The evolutionary plots of these five methods on the training and test RMSEs are shown in Figure 5.6 and 5.7, respectively

As shown in Figure 5.6, 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

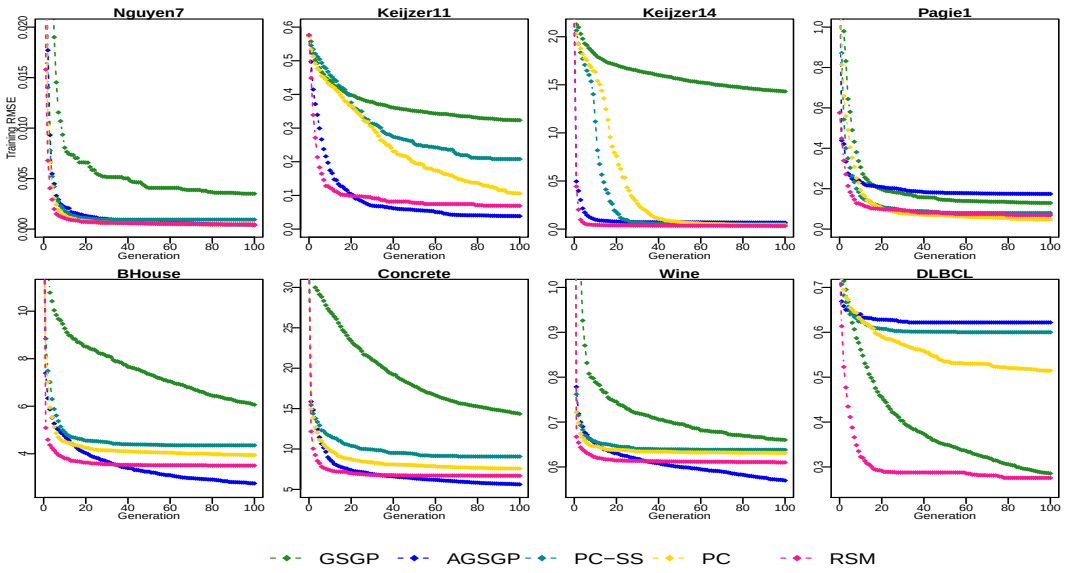


Figure 5.6: Evolutionary Plots on the **Training** RMSEs of the Best-of-Generation Models in GSGP, AGSGP, and GSGP with three operators solely.

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.

The evolutionary plots on the generalisation performance in Figure 5.7 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 generalisation performance than GSGP and AGSGP. On Keijzer11, AGSGP loses the advantage on the generalisation performance. PC has similar generalisation performance to AGSGP, while RSM has the best generalisation 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 generalise 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.

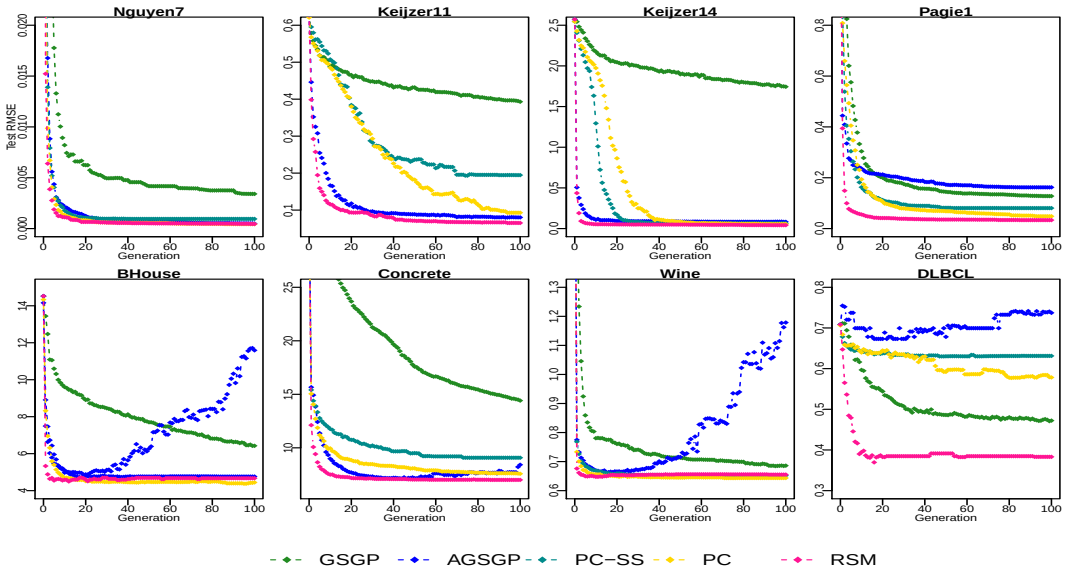


Figure 5.7: Evolutionary Plots on the **Test** RMSEs of the Best-of-Generation Models in GSGP, AGSGP, and GSGP with three operators solely.

In summary, GSGP with any of the three proposed operators solely can outperform GSGP on most of the examined datasets on both the learning performance and the generalisation ability. When compared with AGSGP, they have comparable learning ability but much better generalisation performance. PC and RSM have comparable training and generalisation 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 Figures 5.4 to 5.7 (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 the training and generalisation performance of GP.

5.6 Chapter Summary

The goal of exploring the geometric properties of geometric operators to obtain a greater generalisation gain in GP for symbolic regression has been achieved by developing an angle-awareness driven crossover, mutation and selection. The proposed angle-driven geometric operators gain an impressive generalisation 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 towards the target semantics. These new geometric properties offer a productive leverage for approximating the target semantics in each operation.

A comprehensive comparison among ADGSGP, exact GSGP and approximate GSGP (AGSGP) has been conducted. 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 generalisation ability. However, the worse learning and generalisation 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 utilise the semantic information is the key factor influencing on the performance of GP. The angle-aware geometric operators remove 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 semantic context replacement consistently produces much simpler/smaller programs, which are more likely to contain the right structure than the

state-of-the-art GSGP methods. All these characteristics make the solutions more resistant to overfitting and generalise better on unseen data.

Another interesting finding is that the models in the GSGP methods are generally generalise well but overcomplex, which conflicts with the common claim that complex programs are difficult to generalise well. The better generalisation 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 generalise well on unseen data.

The chapter addresses the generalisation issue in GP for symbolic regression when the learnt model has poor generalisation performance but no overfitting occurs. On the datasets used in the experiments, the canonical GP does not overfit the training data. The distance between its generalisation error and training error becomes larger or keeps the same. As expected, GSGP can not only help the evolutionary/learning process to be more effective but also generalise better on unseen data. This confirms that incorporating semantics and geometry (angle-awareness in geometric space) into GP is a good alternative approach for improving the generalisation of GP, which can cover the scenario when approaches focus on counteracting overfitting (e.g. approaches proposed in Chapters 3 and 4) do not work.

Chapter 6

Conclusions and Future Work

This chapter provides conclusions for each of the research objectives of this thesis and presents main findings from each individual chapter, and then highlights potential research areas for future work.

This thesis focuses on genetic programming (GP) for symbolic regression tasks. The overall goal was to investigate and enhance the generalisation ability of GP for symbolic regression by developing new GP approaches which can evolve models exhibiting an impressively good generalisation ability for symbolic regression. The goal was successfully achieved by proposing three new GP approaches to selecting highly important features, measuring the true complexity and expected errors of the evolved regression models, and incorporating semantics and geometric measures into GP to guide the evolutionary process. The proposed methods were examined and compared with state-of-the-art methods on a range of symbolic regression tasks of varying difficulty. The results have shown clearly that our proposed GP methods achieve a significantly higher generalisation gain than standard GP and state-of-the-art GP variants.

6.1 Contributions

This thesis has achieved the following research objectives,

- Proposes a new feature selection method to GP for high-dimensional symbolic regression. The proposed GP based feature selection method utilises GP's build-in feature selection ability and incorporates the permutation importance to score features. By selecting features with a positive permutation importance value, which are assumed to have a positive contribution on reducing the regression errors, the proposed feature selection method can successfully discard irrelevant or noisy features while effectively identifying the truly relevant features. Moreover, it leads to a significantly smaller feature space. GP for symbolic regression on this relatively smaller feature space is able to reduce/eliminate overfitting effectively and achieves notably better generalisation performance than on the whole set of features, since the evolved models are prone to contain truly relevant features while having a lower chance to incorporate the noisy/irrelevant features. The proposed method also outperforms two commonly used feature selection methods, which use random forests and C5.0 for feature selection. Compared with these two feature selection methods, the proposed method generally selects a smaller number of features while keeping the truly relevant features. Further comparisons on the regression performance of GP using the features selected by the examined feature selection methods confirm the superiority of the propose method on promoting the generalisation of GP and evolving more compact models containing only relevant features.

- Introduces VC-dimension and structural risk minimisation (SRM) to GP for estimating the generalisation error of the evolved models during the evolutionary process to improve their performance on unseen data. This represents the first method that implements SRM by extending an experimental method to measure the VC-dimension for regression models and makes SRM available by the experimental method for a mixture of linear and nonlinear regression models in GP in the literature. In the proposed method, SRM provides

a reliable prediction on the generalisation performance of GP solutions by taking both the training error and the confidence interval based on the VC-dimension value into consideration. GP then selects solutions with a lower estimated generalisation error to survive and breed. The SRM based fitness function drives the evolutionary process towards a good trade-off between the training accuracy and the model complexity, which makes GP more resistant to overfitting, and hence can enhance the generalisation of GP. Moreover, the proposed method has been shown to be more effective than the existing generalisation estimating methods for GP in detecting overfitting and evolving much simpler models containing useful building blocks.

- Proposes a novel GSGP approach with angle-awareness for symbolic regression. Three new angle-awareness driven geometric operators, i.e. angle-awareness driven selection, crossover and mutation, have been developed to effectively utilise the geometric properties in the semantic space to search for models with promising performance. The new geometric properties of the three new geometric operators offer a more productive leverage for approximating the target semantics in each operation. The new angle-aware selection operator and geometric crossover operators increase the chance of effective breeding and eliminate the potential ineffectiveness in the existing geometric crossover operators, while the new geometric mutation operator brings a smaller variation to the parent programs and is less greedy in approximating the target semantics. The new method to fulfil the semantic requirement also utilises angle-awareness, which leads to a much smaller model with satisfied semantics. All these characteristics make the proposed GSGP method advance standard GP and two state-of-the-art GSGP methods dramatically on learning better and faster on training data and gaining a better generalise performance on unseen test data.

6.2 Main Conclusions

Overall, this thesis finds that the generalisation of GP for symbolic regression, which is still an open issue in GP community, can be effectively addressed by performing feature selection, measuring program complexity with VC-dimension and developing effective search operators (geometric semantic operators).

This section presents the main conclusions for the three research objectives drawn from the three contribution chapters (Chapter 3 to Chapter 5).

6.2.1 Feature Selection in GP for High-dimensional Symbolic Regression

A new feature selection approach to GP for high-dimensional symbolic regression is proposed in Chapter 3.

Feature Selection to Improve the Generalisation of GP

It is found that feature selection can significantly influence the learning and generalisation of GP when tackling high-dimensional symbolic regression tasks.

A good feature selection method adopts effective search strategies and appropriate feature evaluation criteria. Feature selection in this method identifies the relevant features and discards irrelevant/noise features, which can dramatically reduce the size while improving the quality of the feature space. The reduction of feature space shrinks the search space of GP meanwhile decreases the production of solutions with irrelevant features. Thus, with a higher chance, the evolutionary process is guided towards solutions with relevant features. Thus, GP needs less effort to converge to (near) optimal regression models, particularly when tackling the regression data with a substantially high dimensionality.

Permutation to Further Evaluate features selected by GP

This thesis confirms that GP is able to automatically explore the search space to detect relevant features, which is considered as a built-in feature selection ability. Based on this built-in ability, a two-stage GP for symbolic regression method, where the first stage selects features appearing in the best evolved models and the second stage uses the selected features for symbolic regression, can have significant improvement on the learning and generalisation performance over standard GP using the same computational effort. Comparisons between GP-based feature selection (collecting features appearing in the evolved models) with and without permutation on enhancing the generalisation of GP confirms the effectiveness of permutation importance on further utilising the build-in feature selection ability. Measuring the importance of features with the proposed permutation based method can help identify the truly important features and further shrinks the search space of GP for symbolic regression. This is particularly useful when the data is high-dimensional and contains many noisy features.

GP-based Feature Selection Method

The proposed GP-based feature selection method outperforms the decision-tree based feature selection methods (both C5.0 and random forests) on selecting a smaller set of features while keeping the truly relevant features. It is difficult for C5.0 to identify the relevant features in some cases, and it might discard import/relevant features. Meanwhile, due to the randomness and constructing multiple decision trees, the ensembles in random forests contain redundant features. GP for symbolic regression based on the features selected by the new method outperforms the features selected by C5.0 or random forests on both the learning performance and generalisation gains. The superiority of the proposed feature selection method indicates that GP models with a continuous property learn more than the

stepwise function of decision trees for regression, thus leading to a better ability in detecting important features.

6.2.2 Introducing Structural Risk Minimisation into GP

Chapter 4 proposes a SRM-driven GP method introducing a generalisation estimation based fitness function to guide the evolutionary process towards a trade-off between the training accuracy and the model complexity.

Training Accuracy, Model Complexity and Generalisation Performance

It is found that the training accuracy and model complexity (which is measured by VC-Dimension) are generally in conflict with each other. During the evolutionary process, the training error consistently decreases along with the increase of the model complexity. Moreover, due to this conflict, when introducing an underlying objective of restricting the model complexity into GP, its training performance can become worse than that produced by GP without restriction on model complexity, particularly when over-complex models with smaller training errors and smoother models with larger training errors are competing in the GP population.

This thesis finds that a good trade-off between the training error and the model complexity can lead to significantly better generalisation performance. SRM is able to guide GP to well handle this trade-off. SRM-driven GP evolves models with a lower complexity, which have a lower interval between the training and generalisation errors. When learning from a relatively small number of training instances, models generated by SRM-driven GP can generalise well on unseen data while models evolved by standard GP suffer from the overfitting issue.

Indicators of Generalisation Performance

It is found that the structural risk is more reliable than the empirical risk and the variance error as an indicator of generalisation performance. The empirical risk/training error is not a good indicator of the generalisation performance in many scenarios, particularly when the number of instances in the training set is small or the learning process overfits the training data. The structural risk/error, which considers both the empirical risk and the confidence interval, usually guides the evolutionary process to be free from overfitting. Moreover, SRM makes GP less greedy on chasing a lower training error and enhances the exploration of GP. Hence the structural risk is more reliable than the empirical risk.

This thesis also confirms the advantage of SRM over bootstrap on estimating the generalisation ability of GP solutions, particularly when the number of training instances is small. In this case, it is difficult for Bootstrap to provide a good estimation of the generalisation performance since the bootstrap sets and the training set have a high potential to overlap with each other, i.e. usually have instances in common. BGP, which relies on extracting information from the training set during the evolutionary process, often loses the advantage in this case. SRM, which does not rely on the information from the training set to estimate the generalisation error, is able to provide reliable estimation in this case. Due to this advantage, structural risk is superior to variance error given by bootstrap methods as an indicator of generalisation performance.

Estimation of VC-Dimension

This thesis finds that a better estimation ability on the VC-dimension of evolved models can lead to a lower generalisation error. For measuring the VC-dimension of GP solutions experimentally, compared with the method under uniform setting, the method under optimised non-uniform setting brings improvement on decreasing the difference between the theoretical

and the experimental maximum deviation of errors and reducing the random variability of the measured VC-dimension, which can obtain a more accurate VC-dimension of the evolved models, and accordingly leads to a tighter generalisation bound. SRM-driven GP under the uniform setting can obtain much better generalisation performance than under uniform setting in many cases.

6.2.3 Geometric Semantic GP

A novel angle-awareness driven GSGP method is proposed in Chapter 5. The angle-awareness is introduced into geometric operators along with a new method to fulfil the semantic requirements. Angle-awareness driven GSGP achieves promising improvement on both the learning and generalisation performance.

Generalisation of GSGP

This thesis finds that with semantic-awareness, GSGP methods can have impressively better learning and generalisation performance than the canonical form of GP. However, introducing the semantics into the evolutionary process does not always bring benefits. The way to utilise the semantic information has a high influence on the performance of GP. When the number of instances becomes larger, which leads to a high-dimensional semantic space, the ineffectiveness in exact GSGP and the overfitting trend in approximated GSGP become more obvious and lead to a worse generalisation performance. Introducing angle-awareness into the evolutionary process to further explore the geometry properties of geometric operators is able to increase the chance of effective breeding and reduce/eliminate the ineffectiveness. Moreover, angle-awareness driven GSGP is less greedy in approximating the target semantics and brings a smaller variation to the parent programs which make it more resistant to the trend of overfitting, and hence obtaining an impressive generalisation gain.

Model Complexity in GSGP

This thesis has an interesting finding that the evolved models in GSGP generally generalise well but over-complex. A further examination of the evolved models (including the mathematically simplified form of these models) in GSGP methods and standard GP confirms that, compared with standard GP, GSGP methods generally evolve much more complex models. However, these over-complex models generally have significantly better generalisation performance than their simpler counterparts in standard GP. This somehow conflicts with the well-known theories, such as the Minimum Description Length principle [189] and Occam's razor [214], which claim that it is difficult for complex programs to generalise well. A possible explanation is that not only the size/complexity of the programs matters, but also the way how they are generated is also important for the generalisation ability. These theories might not hold among models that are produced in different manners.

It is also found that with angle-awareness, our new GSGP approach can produce smooth models, which are much smoother than those in the existing GSGP methods. In some cases, they are even smoother than models in standard GP. These simpler/smoother models are able to generate important building blocks in the target models and behave much more similar to the target functions than in the ones evolved by the existing GSGP and standard GP.

6.3 Future Work

During the study we identified several areas of research that particularly deserve further investigation. Below we enumerate the most important ones.

6.3.1 Feature Selection in GP for High-dimensional Symbolic Regression

A number of interesting directions on feature selection in GP for high-dimensional symbolic regression deserve further investigation in the near future.

Permutation for Identifying Important Features

The ability of permutation based feature selection to identify the truly relevant features is only confirmed on two synthetic datasets (since only the two datasets have known relevant features available) in this thesis. We would like to find more real-world datasets, in which the truly relevant features are known in advance, to further investigate the feature selection ability of the proposed permutation based method on various types of problems.

Permutation Based Feature Selection for Regression Algorithms

We also intend to investigate whether the proposed feature selection method is able to promote the performance of other regression algorithms, such as random forests for classical regression/classification. It is also worth exploring the effect of feature selection based on Gini importance in random forests [31] to enhance the prediction ability of regression/classification trees.

Feature Selection Based on Permutation Importance of a Subset of Features

In this thesis, the permutation importance is measured for each single feature and a feature is selected if its importance value is above a predefined threshold. However, the interaction between features is also important but challenging in feature selection. In the proposed feature selection method,

interactions between features is considered, since the permutation importance is measured on the increase of regression error committed by the evolved model containing a subset of features. However, it performs in an implicit way. It is deserved to explore the potential of permutation importance and some other measures on a subset of features. It involves explicitly considering the interaction between features. Accordingly, a new feature selection method can be developed in near future.

6.3.2 Structural Risk Minimisation in GP

Improving the Efficiency of SRM-Driven GP

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 to speed up SRM-driven GP. We plan to develop a context-aware mechanism to estimate the number of solutions to evaluate for VC- dimension. The new mechanism is expected to save effort on the unnecessary measure of VC-dimension while maintain the effectiveness of SRM in GP.

Model Size and Model Complexity

This thesis touches the difference between model size and complexity, but has not analysed the difference in a very deep level, which is very challenging. We will conduct a comprehensive comparison between SRM-driven GP and some bloat control methods, such as the parsimony method on the model size and model complexity.

Implementing SRM in Other Machine Learning Algorithms

We focus mainly on the benefit of SRM with the experimentally measured VC-dimension of GP individuals in this work. However, SRM should be effective for more domains with the demand of measuring the model com-

plexity (e.g. decision trees). This study opens the door for further investigations of SRM and VC-dimension in other related learning machines.

6.3.3 Angle-awareness Driven GSGP

Several interesting directions on geometric semantic GP for symbolic regression deserve further investigation.

Semantic Library

Since the major expense of the proposed GSGP method is in maintaining and searching in the semantic library, introducing improvement to this aspect is a sensible way to enhance the efficiency of the GSGP method. When searching for the desired subtrees from the semantic library, instead of exhaustive search used in the proposed GSGP method, a heuristic search method will be explored to improve the efficiency while not decreasing the effectiveness of this process.

Selection Operator

The proposed selection operator is based on an indirect manner, i.e. select candidate parents by trial and error, the angle-distance between candidate parents is obtained until the one has big enough is found. 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.

6.3.4 Transfer Learning in GP for Symbolic Regression

This thesis focuses on improving the generalisation of GP for symbolic regression, which has an underlying assumption that the distribution of future data will be the same as that of the training data. However, in many

real-world applications, this assumption does not hold. Moreover, the labelled data of the target domain might be difficult collect and even after being collected, the data may be easily outdated. Thus, it is desired to make the best use of the related data, i.e. data in the source domain. The model/solution trained in the source domain is expected to adapt well to different but similar target domains. *Transfer learning* [140, 174], which allows the domains, tasks and distributions used in the training set and the test set to be different, is desired in these scenarios. Although transfer learning has received considerable attention from the machine learning community, research on transfer learning in GP for symbolic regression is far from enough. We plan to investigate and develop novel GP based transfer learning methods to solve various real-world problems.

Bibliography

[1] ABRAHAM, A., GUO, H., AND LIU, H. *Swarm intelligence: foundations, perspectives and applications*. Springer, 2006.

[2] AFZAL, W., AND TORKAR, R. On the application of genetic programming for software engineering predictive modeling: A systematic review. *Expert Systems with Applications 38*, 9 (2011), 11984–11997.

[3] AGAPITOS, A., BRABAZON, A., AND O'NEILL, M. Controlling overfitting in symbolic regression based on a bias/variance error decomposition. In *Parallel Problem Solving from Nature-PPSN XII*. Springer, 2012, pp. 438–447.

[4] AGGARWAL, V., AND O'REILLY, U.-M. Design of posynomial models for MOSFETs: symbolic regression using genetic algorithms. In *Genetic Programming Theory and Practice IV*. Springer, 2007, pp. 219–236.

[5] AGRAWAL, R., IMIELIŃSKI, T., AND SWAMI, A. Mining association rules between sets of items in large databases. In *ACM SIGMOD Record* (1993), vol. 22, ACM, pp. 207–216.

[6] AHMAD, F., ISA, N. A. M., HUSSAIN, Z., OSMAN, M. K., AND SULAIMAN, S. N. A GA-based feature selection and parameter optimization of an ann in diagnosing breast cancer. *Pattern Analysis and Applications 18*, 4 (2015), 861–870.

[7] AHMED, S., ZHANG, M., AND PENG, L. Enhanced feature selection for biomarker discovery in LC-MS data using GP. In *IEEE Congress on Evolutionary Computation (CEC)* (2013), pp. 584–591.

[8] AKAIKE, H. Statistical predictor identification. *Annals of the Institute of Statistical Mathematics 22*, 1 (1970), 203–217.

[9] AL-SAHAF, H., ZHANG, M., AND JOHNSTON, M. Binary image classification: A genetic programming approach to the problem of limited training instances. *Evolutionary Computation 24*, 1 (2016), 143–182.

[10] ALBINATI, J., PAPPA, G. L., OTERO, F. E., AND OLIVEIRA, L. O. V. The Effect of Distinct Geometric Semantic Crossover Operators in Regression Problems. In *Genetic Programming*. Springer, 2015, pp. 3–15.

[11] ALPAYDIN, E. *Introduction to machine learning*. MIT press, 2014.

[12] AMARI, S.-I., AND WU, S. Improving support vector machine classifiers by modifying kernel functions. *Neural Networks 12*, 6 (1999), 783–789.

[13] AMIL, N. M., BREDECHE, N., GAGNÉ, C., GELLY, S., SCHOENAUER, M., AND TEYTAUD, O. A statistical learning perspective of genetic programming. In *EuroGP* (2009), Springer, pp. 327–338.

[14] ARCHETTI, F., LANZENI, S., MESSINA, E., AND VANNESCHI, L. Genetic programming for computational pharmacokinetics in drug discovery and development. *Genetic Programming and Evolvable Machines 8*, 4 (2007), 413–432.

[15] BABOVIC, V., AND KEIJZER, M. Genetic programming as a model induction engine. *Journal of Hydroinformatics 2* (2000), 35–60.

[16] BÄCK, T., FOGEL, D. B., AND MICHALEWICZ, Z. *Evolutionary computation 1: Basic algorithms and operators*, vol. 1. CRC press, 2000.

[17] BANZHAF, W., NORDIN, P., KELLER, R. E., AND FRANCONE, F. D. Genetic Programming—An Introduction: On the Automatic Evolution of Computer Programs and Its Applications, dpunkt. verlag and Morgan Kaufmann Publishers. *Inc., San Francisco, California* (1998).

[18] BARRON, A., RISSANEN, J., AND YU, B. The minimum description length principle in coding and modeling. *Information Theory, IEEE Transactions on 44*, 6 (1998), 2743–2760.

[19] BATES, D. M., AND WATTS, D. G. *Nonlinear regression: iterative estimation and linear approximations*. Wiley Online Library, 1988.

[20] BEADLE, L., AND JOHNSON, C. G. Semantically driven crossover in genetic programming. In *IEEE World Congress on Computational Intelligence* (2008), pp. 111–116.

[21] BEADLE, L., AND JOHNSON, C. G. Semantic analysis of program initialisation in genetic programming. *Genetic Programming and Evolvable Machines 10*, 3 (2009), 307–337.

[22] BELLMAN, R. E. *Dynamic Programming*. Dover Publications, Incorporated, 2003.

[23] BEMARDO, J. M., AND SMITH, A. Bayesian theory. *Chichester: John-Wiley and Sons, Ltd* (1994).

[24] BEYER, H.-G., AND SCHWEFEL, H.-P. Evolution strategies–A comprehensive introduction. *Natural computing 1*, 1 (2002), 3–52.

[25] BISHOP, C. M., ET AL. *Pattern recognition and machine learning*, vol. 4. springer New York, 2006.

[26] BLUM, C., AND LI, X. *Swarm intelligence in optimization*. Springer, 2008.

[27] BLUMER, A., EHRENFEUCHT, A., HAUSSLER, D., AND WARMUTH, M. K. Occam's razor. *Information processing letters 24*, 6 (1987), 377–380.

[28] BORGES, C. E., ALONSO, C. L., AND MONTAÑA, J. L. Model selection in genetic programming. In *Proceedings of the 12th annual conference on genetic and evolutionary computation* (2010), ACM, pp. 985–986.

[29] BOSER, B. E., GUYON, I. M., AND VAPNIK, V. N. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory* (1992), ACM, pp. 144–152.

[30] BRAMEIER, M., AND BANZHAF, W. A comparison of linear genetic programming and neural networks in medical data mining. *Evolutionary Computation, IEEE Transactions on 5*, 1 (2001), 17–26.

[31] BREIMAN, L. Random forests. *Machine learning 45*, 1 (2001), 5–32.

[32] BREIMAN, L., FRIEDMAN, J., STONE, C. J., AND OLSHEN, R. A. *Classification and regression trees*. CRC press, 1984.

[33] BREZMES, J., CABRÉ, P., ROJO, S., LLOBET, E., VILANOVA, X., AND CORREIG, X. Discrimination between different samples of olive oil using variable selection techniques and modified fuzzy artmap neural networks. *IEEE Sensors Journal 5*, 3 (2005), 463–470.

[34] BURKE, E. K., GUSTAFSON, S., AND KENDALL, G. Diversity in genetic programming: An analysis of measures and correlation with fitness. *Evolutionary Computation, IEEE Transactions on 8*, 1 (2004), 47–62.

[35] BURKE, R., GUSTAFSON, S. M., AND KENDALL, G. A Survey and Analysis of Diversity Measures in Genetic Programming. In *Proceedings of the 17th Annual Conference on Genetic and Evolutionary Computation Conference (GECCO)* (2002), vol. 2, pp. 716–723.

[36] BURKS, A. R., AND PUNCH, W. F. An efficient structural diversity technique for genetic programming. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation* (2015), ACM, pp. 991–998.

[37] BURKS, A. R., AND PUNCH, W. F. An analysis of the genetic marker diversity algorithm for genetic programming. *Genetic Programming and Evolvable Machines* (2017), 1–33.

[38] CAI, W., PACHECO-VEGA, A., SEN, M., AND YANG, K. Heat transfer correlations by symbolic regression. *International Journal of Heat and Mass Transfer 49*, 23 (2006), 4352–4359.

[39] CASEY, J. Exploiting Curvature. *Wiesbaden, Germany: Vieweg 7*, 51 (1996), 176.

[40] CASTELLI, M. *Measures and methods for robust genetic programming*. PhD thesis, Università degli Studi di Milano-Bicocca, 2012.

[41] CASTELLI, M., MANZONI, L., SILVA, S., AND VANNESCHI, L. A comparison of the generalization ability of different genetic programming frameworks. In *IEEE Congress on Evolutionary Computation (CEC)* (2010), IEEE, pp. 1–8.

[42] CASTELLI, M., MANZONI, L., SILVA, S., AND VANNESCHI, L. A quantitative study of learning and generalization in genetic programming. In *Genetic Programming*. Springer, 2011, pp. 25–36.

[43] CASTELLI, M., VANNESCHI, L., AND SILVA, S. Prediction of high performance concrete strength using genetic programming with ge-

ometric semantic genetic operators. *Expert Systems with Applications 40*, 17 (2013), 6856–6862.

[44] CASTILLO, F., KORDON, A., SWEENEY, J., AND ZIRK, W. Using genetic programming in industrial statistical model building. In *Genetic programming theory and practice II*. Springer, 2005, pp. 31–48.

[45] CAVARETTA, M. J., AND CHELLAPILLA, K. Data mining using genetic programming: The implications of parsimony on generalization error. In *IEEE Congress on Evolutionary Computation (CEC)* (1999), vol. 2, IEEE, pp. 1330–1337.

[46] CHAN, K. Y., KWONG, C., DILLON, T. S., AND TSIM, Y. Reducing overfitting in manufacturing process modeling using a backward elimination based genetic programming. *Applied Soft Computing 11*, 2 (2011), 1648–1656.

[47] CHELLAPILLA, K. Evolutionary programming with tree mutations: Evolving computer programs without crossover. *Genetic Programming* (1997), 431–438.

[48] CHEN, L., CHU, C., HUANG, T., KONG, X., AND CAI, Y.-D. Prediction and analysis of cell-penetrating peptides using pseudo-amino acid composition and random forest models. *Amino acids 47*, 7 (2015), 1485–1493.

[49] CHEN, Q., XUE, B., MEI, Y., AND ZHANG, M. Geometric semantic crossover with an angle-aware mating scheme in genetic programming for symbolic regression. In *European Conference on Genetic Programming* (2017), Springer, pp. 229–245.

[50] CHEN, Q., XUE, B., NIU, B., AND ZHANG, M. Improving generalisation of genetic programming for high-dimensional symbolic regression with feature selection. In *IEEE Congress on Evolutionary Computation (CEC)* (2016), pp. 3793–3800.

[51] CHEN, Q., XUE, B., SHANG, L., AND ZHANG, M. Improving generalisation of genetic programming for symbolic regression with structural risk minimisation. In *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference* (2016), ACM, pp. 709–716.

[52] CHEN, Q., XUE, B., SHANG, L., AND ZHANG, M. New geometric semantic operators in genetic programming:perpendicular crossover and random segment mutation. In *Proceedings of the 2017 on Genetic and Evolutionary Computation Conference* (2017), ACM, pp. 223–224.

[53] CHEN, Q., XUE, B., AND ZHANG, M. Generalisation and domain adaptation in GP with gradient descent for symbolic regression. In *IEEE Congress on Evolutionary Computation (CEC)* (2015), pp. 1137–1144.

[54] CHEN, Q., ZHANG, M., AND XUE, B. Feature selection to improve generalization of genetic programming for high-dimensional symbolic regression. *IEEE Transactions on Evolutionary Computation 21*, 5 (2017), 792–806. doi:10.1109/TEVC.2017.2683489.

[55] CHERKASSKY, V., AND MA, Y. Comparison of model selection for regression. *Neural computation 15*, 7 (2003), 1691–1714.

[56] CHERKASSKY, V., AND MULIER, F. M. *Learning from data: concepts, theory, and methods*. John Wiley & Sons, 2007.

[57] CHERKASSKY, V., SHAO, X., MULIER, F. M., AND VAPNIK, V. N. Model complexity control for regression using vc generalization bounds. *IEEE transactions on Neural Networks 10*, 5 (1999), 1075–1089.

[58] CHO, J. H., AND KURUP, P. U. Decision tree approach for classification and dimensionality reduction of electronic nose data. *Sensors and Actuators B: Chemical 160*, 1 (2011), 542–548.

[59] CHUANG, L.-Y., CHANG, H.-W., TU, C.-J., AND YANG, C.-H. Improved binary pso for feature selection using gene expression data. *Computational Biology and Chemistry 32*, 1 (2008), 29–38.

[60] COHN, D., ATLAS, L., AND LADNER, R. Improving generalization with active learning. *Machine learning 15*, 2 (1994), 201–221.

[61] CORTES, C., AND VAPNIK, V. Support-vector networks. *Machine learning 20*, 3 (1995), 273–297.

[62] COSTELLOE, D., AND RYAN, C. On improving generalisation in genetic programming. In *Genetic Programming*. Springer, 2009, pp. 61–72.

[63] CRAVEN, P., AND WAHBA, G. Smoothing noisy data with spline functions. *Numerische mathematik 31*, 4 (1978), 377–403.

[64] DASH, M., AND LIU, H. Feature selection for classification. *Intelligent data analysis 1*, 3 (1997), 131–156.

[65] DAVIDSON, J., SAVIC, D. A., AND WALTERS, G. A. Symbolic and numerical regression: Experiments and applications. *Information Sciences 150*, 1 (2003), 95–117.

[66] DEVROYE, L. Bounds for the uniform deviation of empirical measures. *Journal of Multivariate Analysis 12*, 1 (1982), 72–79.

[67] DICK, G. Bloat and Generalisation in Symbolic Regression. In *Simulated Evolution and Learning*. Springer, 2014, pp. 491–502.

[68] DICK, G., RIMONI, A. P., AND WHIGHAM, P. A. A re-examination of the use of genetic programming on the oral bioavailability problem. In *Proceedings of the 17th Annual Conference on Genetic and Evolutionary Computation Conference (GECCO)* (2015), ACM, pp. 1015–1022.

[69] DOMINGOS, P. A unified bias-variance decomposition for zero-one and squared loss. *AAAI/IAAI 2000* (2000), 564–569.

[70] DORIGO, M., AND BIRATTARI, M. Ant colony optimization. In *Encyclopedia of Machine Learning*. Springer, 2010, pp. 36–39.

[71] DRUCKER, H., BURGES, C. J., KAUFMAN, L., SMOLA, A., VAPNIK, V., ET AL. Support vector regression machines. *Advances in neural information processing systems 9* (1997), 155–161.

[72] EFRON, B., AND TIBSHIRANI, R. Improvements on cross-validation: the 632+ bootstrap method. *Journal of the American Statistical Association 92*, 438 (1997), 548–560.

[73] EFRON, B., AND TIBSHIRANI, R. J. *An introduction to the bootstrap*. CRC press, 1994.

[74] EKÁRT, A., AND NÉMETH, S. Z. A metric for genetic programs and fitness sharing. In *Proceedings of the 3rd European conference on Genetic Programming (EuroGP)* (2000), Springer, pp. 259–270.

[75] FARIS, H., SHETA, A., AND ÖZNERGIZ, E. Modelling hot rolling manufacturing process using soft computing techniques. *International Journal of Computer Integrated Manufacturing 26*, 8 (2013), 762–771.

[76] FERREIRA, C., AND GEPSOFT, U. What is Gene Expression Programming, 2008.

[77] FITZGERALD, J., AZAD, R., AND RYAN, C. A bootstrapping approach to reduce over-fitting in genetic programming. In *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation Conference (GECCO)* (2013), pp. 1113–1120.

[78] FITZGERALD, J., AND RYAN, C. On size, complexity and generalisation error in gp. In *Proceedings of the 16th Annual Conference on Genetic*

and Evolutionary Computation Conference (GECCO) (2014), pp. 903–910.

[79] FOGEL, D. B. *Evolutionary computation: toward a new philosophy of machine intelligence*, vol. 1. John Wiley & Sons, 2006.

[80] FOGEL, L. J. *Intelligence through simulated evolution: forty years of evolutionary programming*. John Wiley & Sons, Inc., 1999.

[81] FOLINO, G., PIZZUTI, C., AND SPEZZANO, G. Ensemble techniques for parallel genetic programming based classifiers. In *Proceedings of the 6th European conference on Genetic Programming (EuroGP)* (2003), Springer, pp. 59–69.

[82] FORTIN, F.-A., RAINVILLE, F.-M. D., GARDNER, M.-A., PARIZEAU, M., AND GAGNÉ, C. DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research 13*, Jul (2012), 2171–2175.

[83] FRIEDMAN, J., HASTIE, T., AND TIBSHIRANI, R. *The elements of statistical learning*, vol. 1. Springer series in statistics Springer, Berlin, 2001.

[84] FRIEDMAN, J., HASTIE, T., AND TIBSHIRANI, R. Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software 33*, 1 (2010), 1.

[85] FRIEDMAN, J. H. Multivariate adaptive regression splines. *The annals of statistics* (1991), 1–67.

[86] FRIEDMAN, J. H. On bias, variance, 0/1loss, and the curse-of-dimensionality. *Data mining and knowledge discovery 1*, 1 (1997), 55–77.

[87] FUNAHASHI, K.-I. On the approximate realization of continuous mappings by neural networks. *Neural networks 2*, 3 (1989), 183–192.

[88] GAGNÉ, C., SCHOENAUER, M., PARIZEAU, M., AND TOMASSINI, M. Genetic programming, validation sets, and parsimony pressure. In *Proceedings of the 9th European conference on Genetic Programming (EuroGP)* (2006), Springer, pp. 109–120.

[89] GALVAN-LOPEZ, E., CODY-KENNY, B., TRUJILLO, L., AND KATTAN, A. Using semantics in the selection mechanism in genetic programming: a simple method for promoting semantic diversity. In *IEEE Congress on Evolutionary Computation (CEC)* (2013), IEEE, pp. 2972–2979.

[90] GANDOMI, A. H., AND ALAVI, A. H. A new multi-gene genetic programming approach to nonlinear system modeling. Part I: materials and structural engineering problems. *Neural Computing and Applications 21*, 1 (2012), 171–187.

[91] GEMAN, S., BIENENSTOCK, E., AND DOURSAT, R. Neural networks and the bias/variance dilemma. *Neural Computation 4*, 1 (1992), 1–58.

[92] GEMAN, S., BIENENSTOCK, E., AND DOURSAT, R. Neural networks and the bias/variance dilemma. *Neural Networks 4*, 1 (2008).

[93] GHEYAS, I. A., AND SMITH, L. S. Feature subset selection in large dimensionality domains. *Pattern recognition 43*, 1 (2010), 5–13.

[94] GIUSTOLISI, O., AND SAVIC, D. A symbolic data-driven technique based on evolutionary polynomial regression. *Journal of Hydroinformatics 8*, 3 (2006), 207–222.

[95] GOLBERG, D. E. Genetic algorithms in search, optimization, and machine learning. *Addion wesley 1989* (1989).

[96] GOLDBERG, D. E., AND HOLLAND, J. H. Genetic algorithms and machine learning. *Machine learning 3*, 2 (1988), 95–99.

[97] GONÇALVES, I., AND SILVA, S. Experiments on controlling overfitting in genetic programming. In *15th Portuguese conference on artificial intelligence (EPIA 2011)* (2011), pp. 978–989.

[98] GONÇALVES, I., SILVA, S., AND FONSECA, C. M. On the generalization ability of geometric semantic genetic programming. In *Genetic Programming*. Springer, 2015, pp. 41–52.

[99] GONALVES, I., AND SILVA, S. *Balancing learning and overfitting in genetic programming with interleaved sampling of training data*. Springer, 2013.

[100] GONALVES, I., SILVA, S., MELO, J. B., AND CARREIRAS, J. M. Random sampling technique for overfitting control in genetic programming. In *Genetic Programming*. Springer, 2012, pp. 218–229.

[101] GRANITTO, P. M., FURLANELLO, C., BIASIOLI, F., AND GASPERI, F. Recursive feature elimination with random forest for PTR-MS analysis of agroindustrial products. *Chemometrics and Intelligent Laboratory Systems 83*, 2 (2006), 83–90.

[102] GREFENSTETTE, J. J., AND FITZPATRICK, J. M. Genetic search with approximate function evaluations. In *Proceedings of an International Conference on Genetic Algorithms and Their Applications* (1985), pp. 112–120.

[103] GU, S., CHENG, R., AND JIN, Y. Feature selection for high-dimensional classification using a competitive swarm optimizer. *Soft Computing* (2016), 1–12.

[104] GUO, G., AND DYER, C. R. Simultaneous feature selection and classifier training via linear programming: A case study for face expression recognition. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition(CVPR)* (2003), pp. 346–352.

[105] GUSTAFSON, S., BURKE, E. K., AND KRASNOGOR, N. On improving genetic programming for symbolic regression. In *Evolutionary Computation, 2005. The 2005 IEEE Congress on* (2005), vol. 1, IEEE, pp. 912–919.

[106] GUYON, I., AND ELISSEEFF, A. An introduction to variable and feature selection. *The Journal of Machine Learning Research 3* (2003), 1157–1182.

[107] HAERI, M. A., EBADZADEH, M. M., AND FOLINO, G. Improving GP generalization: a variance-based layered learning approach. *Genetic Programming and Evolvable Machines 16*, 1 (2015), 27–55.

[108] HARMAN, M., JIA, Y., KRINKE, J., LANGDON, W., PETKE, J., AND ZHANG, Y. Search based software engineering for software product line engineering: a survey and directions for future work. In *Proceedings of the 18th International Software Product Line Conference-Volume 1* (2014), ACM, pp. 5–18.

[109] HARVEY, D. Y., AND TODD, M. D. Automated feature design for numeric sequence classification by genetic programming. *IEEE Transactions on Evolutionary Computation 19*, 4 (2015), 474–489.

[110] HASTIE, T., TIBSHIRANI, R., FRIEDMAN, J., HASTIE, T., FRIEDMAN, J., AND TIBSHIRANI, R. *The elements of statistical learning*, vol. 2. Springer, 2009.

[111] HOLLAND, J. H. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992.

[112] HOOPER, D. C., AND FLANN, N. S. Improving the accuracy and robustness of genetic programming through expression simplification. In *Proceedings of the 1st annual conference on genetic programming* (1996), MIT Press, pp. 428–428.

[113] JAIN, A. K., DUIN, R. P. W., AND MAO, J. Statistical pattern recognition: A review. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 22*, 1 (2000), 4–37.

[114] JOHNSON, C. G. Artificial immune system programming for symbolic regression. In *Genetic Programming*. Springer, 2003, pp. 345–353.

[115] JOHNSON, M. E., AND NACHTSHEIM, C. J. Some guidelines for constructing exact d-optimal designs on convex design spaces. *Technometrics 25*, 3 (1983), 271–277.

[116] KEIJZER, M. Improving symbolic regression with interval arithmetic and linear scaling. In *Proceedings of the 6th European conference on Genetic Programming (EuroGP)* (2003), pp. 70–82.

[117] KEIJZER, M. Alternatives in subtree caching for genetic programming. In *Genetic Programming*. Springer, 2004, pp. 328–337.

[118] KENNEDY, J., KENNEDY, J. F., AND EBERHART, R. C. *Swarm intelligence*. Morgan Kaufmann, 2001.

[119] KENNEDY, R. J. and eberhart, particle swarm optimization. In *Proceedings of IEEE International Conference on Neural Networks IV, pages* (1995), vol. 1000.

[120] KOHAVI, R., AND JOHN, G. H. Wrappers for feature subset selection. *Artificial intelligence 97*, 1 (1997), 273–324.

[121] KOLMOGOROV, A. Sulla determinazione empirica di una lgge di distribuzione. *Inst. Ital. Attuari, Giorn. 4* (1933), 83–91.

[122] KOTANCHEK, M., SMITS, G., AND VLADISLAVLEVA, E. Pursuing the Pareto paradigm: tournaments, algorithm variations and ordinal optimization. In *Genetic Programming Theory and Practice IV*. Springer, 2007, pp. 167–185.

[123] KOWALIW, T., AND DOURSAT, R. Bias-variance decomposition in genetic programming. *Open Mathematics 14*, 1 (2016), 62–80.

[124] KOZA, J. R. *Genetic programming: on the programming of computers by means of natural selection*. The MIT Press, 1992.

[125] KRAWIEC, K. Learnable embeddings of program spaces. In *European Conference on Genetic Programming* (2011), Springer, pp. 166–177.

[126] KRAWIEC, K., AND LICHOCKI, P. Approximating geometric crossover in semantic space. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation* (2009), ACM, pp. 987–994.

[127] KRAWIEC, K., AND PAWLAK, T. Approximating geometric crossover by semantic backpropagation. In *Proceedings of the 15th annual conference on Genetic and evolutionary computation* (2013), ACM, pp. 941–948.

[128] KRAWIEC, K., AND PAWLAK, T. Locally geometric semantic crossover: a study on the roles of semantics and homology in recombination operators. *Genetic Programming and Evolvable Machines 14*, 1 (2013), 31–63.

[129] KUHN, M., WESTON, S., COULTER, N., AND QUINLAN, R. C50: C5. 0 decision trees and rule-based models. *R package version 0.1. 0-21, URL http://CRAN. R-project. org/package C 50* (2014).

[130] KUSHCHU, I. Genetic programming and evolutionary generalization. *Evolutionary Computation, IEEE Transactions on 6*, 5 (2002), 431–442.

[131] LAPPAS, G. Estimating the size of neural networks from the number of available training data. In *International Conference on Artificial Neural Networks* (2007), Springer, pp. 68–77.

[132] LEE, Y.-S., AND TONG, L.-I. Forecasting energy consumption using a grey model improved by incorporating genetic programming. *Energy conversion and Management 52*, 1 (2011), 147–152.

[133] LEW, T., SPENCER, A., SCARPA, F., WORDEN, K., RUTHERFORD, A., AND HEMEZ, F. Identification of response surface models using genetic programming. *Mechanical Systems and Signal Processing 20*, 8 (2006), 1819–1831.

[134] LEYS, C., LEY, C., KLEIN, O., BERNARD, P., AND LICATA, L. Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median. *Journal of Experimental Social Psychology 49*, 4 (2013), 764–766.

[135] LI, W. W., AND JEFF WU, C. Columnwise-pairwise algorithms with applications to the construction of supersaturated designs. *Technometrics 39*, 2 (1997), 171–179.

[136] LI, Y., ZHANG, S., AND ZENG, X. Research of multi-population agent genetic algorithm for feature selection. *Expert Systems with Applications 36*, 9 (2009), 11570–11581.

[137] LIAW, A., AND WIENER, M. Classification and regression by randomforest. *R news 2*, 3 (2002), 18–22.

[138] LICHMAN, M. UCI machine learning repository, 2013.

[139] LIU, H., AND YU, L. Toward integrating feature selection algorithms for classification and clustering. *Knowledge and Data Engineering, IEEE Transactions on 17*, 4 (2005), 491–502.

[140] LU, J., BEHBOOD, V., HAO, P., ZUO, H., XUE, S., AND ZHANG, G. Transfer learning using computational intelligence: a survey. *Knowledge-Based Systems 80* (2015), 14–23.

[141] LUKE, S., AND PANAIT, L. A comparison of bloat control methods for genetic programming. *Evolutionary Computation 14*, 3 (2006), 309–344.

[142] LUKE, S., PANAIT, L., BALAN, G., PAUS, S., SKOLICKI, Z., POPOVICI, E., SULLIVAN, K., HARRISON, J., BASSETT, J., HUBLEY, R., ET AL. A java-based evolutionary computation research system. *Online (March 2004) http://cs. gmu. edu/~ eclab/projects/ecj* (2004).

[143] MAHLER, S., ROBILLIARD, D., AND FONLUPT, C. Tarpeian bloat control and generalization accuracy. In *Genetic Programming*. Springer, 2005, pp. 203–214.

[144] MAJEED, H., AND RYAN, C. A less destructive, context-aware crossover operator for gp. *Genetic Programming* (2006), 36–48.

[145] MAJEED, H., AND RYAN, C. On the constructiveness of context-aware crossover. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation* (2007), ACM, pp. 1659–1666.

[146] MCCONAGHY, T. Ffx: Fast, scalable, deterministic symbolic regression technology. In *Genetic Programming Theory and Practice IX*. Springer, 2011, pp. 235–260.

[147] MCDERMOTT, J., WHITE, D. R., LUKE, S., MANZONI, L., CASTELLI, M., VANNESCHI, L., JASKOWSKI, W., KRAWIEC, K., HARPER, R., DE JONG, K., ET AL. Genetic programming needs better benchmarks. In *Proceedings of the 14th annual conference on Genetic and evolutionary computation* (2012), ACM, pp. 791–798.

[148] MCPHEE, N. F., AND HOPPER, N. J. Analysis of genetic diversity through population history. In *Proceedings of the Genetic and Evolutionary Computation Conference* (1999), vol. 2, Citeseer, pp. 1112–1120.

[149] MCPHEE, N. F., OHS, B., AND HUTCHISON, T. Semantic building blocks in genetic programming. In *European Conference on Genetic Programming* (2008), Springer, pp. 134–145.

[150] MEI, Y., NGUYEN, S., XUE, B., AND ZHANG, M. An efficient feature selection algorithm for evolving job shop scheduling rules with genetic programming. *IEEE Transactions on Emerging Topics in Computational Intelligence 1*, 5 (2017), 339–353.

[151] MENZE, B. H., PETRICH, W., AND HAMPRECHT, F. A. Multivariate feature selection and hierarchical classification for infrared spectroscopy: serum-based detection of bovine spongiform encephalopathy. *Analytical and bioanalytical chemistry 387*, 5 (2007), 1801–1807.

[152] MITCHELL, T. M. Machine learning. 1997. *Burr Ridge, IL: McGraw Hill 45* (1997).

[153] MOHRI, M., ROSTAMIZADEH, A., AND TALWALKAR, A. *Foundations of machine learning*. MIT press, 2012.

[154] MONTAÑA, J. L., ALONSO, C. L., BORGES, C. E., AND DE LA DEHESA, J. Penalty functions for genetic programming algorithms. In *Computational Science and Its Applications-ICCSA 2011*. Springer, 2011, pp. 550–562.

[155] MOORE, J. H., HILL, D. P., SAYKIN, A., AND SHEN, L. Exploring interestingness in a computational evolution system for the genome-wide genetic analysis of alzheimers disease. In *Genetic Programming Theory and Practice XI*. Springer, 2014, pp. 31–45.

[156] MORAGLIO, A., KRAWIEC, K., AND JOHNSON, C. G. Geometric semantic genetic programming. In *Parallel Problem Solving from Nature-PPSN XII*. Springer, 2012, pp. 21–31.

[157] MOUSAVI ASTARABADI, S. S., AND EBADZADEH, M. M. Avoiding overfitting in symbolic regression using the first order derivative of GP trees. In *Proceedings of the Companion Publication of the 2015 on Genetic and Evolutionary Computation Conference* (2015), ACM, pp. 1441–1442.

[158] MUNI, D. P., PAL, N. R., AND DAS, J. Genetic programming for simultaneous feature selection and classifier design. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 36*, 1 (2006), 106–117.

[159] MURSALIN, M., ZHANG, Y., CHEN, Y., AND CHAWLA, N. V. Automated epileptic seizure detection using improved correlation-based feature selection with random forest classifier. *Neurocomputing 241* (2017), 204–214.

[160] NAG, K., AND PAL, N. R. A multiobjective genetic programming-based ensemble for simultaneous feature selection and classification. *IEEE Transactions on Cybernetics 46*, 2 (2016), 499–510.

[161] NESHATIAN, K., AND ZHANG, M. Pareto front feature selection: using genetic programming to explore feature space. In *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation Conference (GECCO)* (2009), pp. 1027–1034.

[162] NETER, J., WASSERMAN, W., AND KUTNER, M. H. Applied linear regression models.

[163] NEUNER, H. Design of artificial neural networks for change-point detection. In *The 1st International Workshop on the Quality of Geodetic Observation and Monitoring Systems (QuGOMS'11)* (2015), Springer, pp. 139–144.

[164] NGUYEN, H. B., XUE, B., AND ZHANG, M. A subset similarity guided method for multi-objective feature selection. In *Aus-*

tralasian Conference on Artificial Life and Computational Intelligence (2016), Springer, pp. 298–310.

[165] NGUYEN, Q. U., NGUYEN, X. H., AND O'NEILL, M. Semantic aware crossover for genetic programming: the case for real-valued function regression. In *Genetic Programming*. Springer, 2009, pp. 292–302.

[166] NGUYEN, Q. U., NGUYEN, X. H., AND O'NEILL, M. Examining the landscape of semantic similarity based mutation. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation* (2011), ACM, pp. 1363–1370.

[167] NGUYEN, Q. U., NGUYEN, X. H., O'NEILL, M., AND AGAPITOS, A. An investigation of fitness sharing with semantic and syntactic distance metrics. In *Genetic Programming*. Springer, 2012, pp. 109–120.

[168] NGUYEN, Q. U., PHAM, T. A., NGUYEN, X. H., AND MCDERMOTT, J. Subtree semantic geometric crossover for genetic programming. *Genetic Programming and Evolvable Machines 17*, 1 (2016), 25–53.

[169] NIE, F., HUANG, H., CAI, X., AND DING, C. H. Efficient and robust feature selection via joint 2, 1-norms minimization. In *Advances in neural information processing systems* (2010), pp. 1813–1821.

[170] NIKOLAEV, N. Y., AND IBA, H. Regularization approach to inductive genetic programming. *Evolutionary Computation, IEEE Transactions on 5*, 4 (2001), 359–375.

[171] OLIVEIRA, L. O. V., OTERO, F. E., AND PAPPA, G. L. A dispersion operator for geometric semantic genetic programming. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, pp. 773–780.

[172] OLTEAN, M., AND GROSAN, C. Evolving Digital Circuits using Multi Expression Programming. In *Evolvable Hardware* (2004), pp. 87–90.

[173] O'NEILL, M., VANNESCHI, L., GUSTAFSON, S., AND BANZHAF, W. Open issues in genetic programming. *Genetic Programming and Evolvable Machines 11*, 3-4 (2010), 339–363.

[174] PAN, S. J., AND YANG, Q. A survey on transfer learning. *Knowledge and Data Engineering, IEEE Transactions on 22*, 10 (2010), 1345–1359.

[175] PARROTT, D., LI, X., AND CIESIELSKI, V. Multi-objective techniques in genetic programming for evolving classifiers. In *Evolutionary Computation, 2005. The 2005 IEEE Congress on* (2005), vol. 2, IEEE, pp. 1141–1148.

[176] PAWLAK, T. P., AND KRAWIEC, K. Semantic geometric initialization. In *European Conference on Genetic Programming* (2016), Springer, pp. 261–277.

[177] PAWLAK, T. P., WIELOCH, B., AND KRAWIEC, K. Review and comparative analysis of geometric semantic crossovers. *Genetic Programming and Evolvable Machines 16*, 3 (2015), 351–386.

[178] PAWLAK, T. P., WIELOCH, B., AND KRAWIEC, K. Semantic backpropagation for designing search operators in genetic programming. *IEEE Transactions on Evolutionary Computation 19*, 3 (2015), 326–340.

[179] PENG, H., LONG, F., AND DING, C. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Transactions on Pattern Analysis and Machine Intelligence 27*, 8 (2005), 1226–1238.

[180] PENNACHIN, C. L., LOOKS, M., AND DE VASCONCELOS, J. A. Robust symbolic regression with affine arithmetic. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation* (2010), ACM, pp. 917–924.

[181] PÉTROWSKI, A. A clearing procedure as a niching method for genetic algorithms. In *Evolutionary Computation, 1996., Proceedings of IEEE International Conference on* (1996), IEEE, pp. 798–803.

[182] PIATETSKY-SHAPIRO, G. Discovery, analysis, and presentation of strong rules. *Knowledge discovery in databases* (1991), 229–238.

[183] POLI, R., AND LANGDON, W. B. On the search properties of different crossover operators in genetic programming. *Genetic Programming* (1998), 293–301.

[184] POLI, R., LANGDON, W. B., MCPHEE, N. F., AND KOZA, J. R. *A field guide to genetic programming*. Lulu. com, 2008.

[185] QUINLAN, J. R. Induction of decision trees. *Machine learning 1*, 1 (1986), 81–106.

[186] QUINLAN, R. Data mining tools See5 and C5.0.

[187] QUINLAN, R. J. C4.5: Programs for machine learning.

[188] REDMOND, M., AND BAVEJA, A. A data-driven software tool for enabling cooperative information sharing among police departments. *European Journal of Operational Research 141*, 3 (2002), 660–678.

[189] RISSANEN, J. Modeling by shortest data description. *Automatica 14*, 5 (1978), 465–471.

[190] ROSCA, J. P. Generality versus size in genetic programming. In *Proceedings of the 1st annual conference on genetic programming* (1996), MIT Press, pp. 381–387.

[191] ROSENWALD, A., WRIGHT, G., CHAN, W. C., CONNORS, J. M., CAMPO, E., FISHER, R. I., GASCOYNE, R. D., MULLER-HERMELINK, H. K., SMELAND, E. B., GILTNANE, J. M., ET AL. The use of molecular profiling to predict survival after chemotherapy for diffuse large-b-cell lymphoma. *New England Journal of Medicine 346*, 25 (2002), 1937–1947.

[192] RUSSELL, S. J., AND NORVIG, P. *Artificial Intelligence: A Modern Approach*, 2 ed. Pearson Education, 2003.

[193] RYAN, C. Pygmies and civil servants. In *Advances in Genetic Programming* (1994), MIT Press, pp. 243–263.

[194] RYAN, C., AND KEIJZER, M. An analysis of diversity of constants of genetic programming. In *EuroGP* (2003), Springer, pp. 404–413.

[195] SAMUEL, A. L. Ai, where it has been and where it is going. In *IJCAI* (1983), pp. 1152–1157.

[196] SANDIN, I., ANDRADE, G., VIEGAS, F., MADEIRA, D., ROCHA, L., SALLES, T., AND GONÇALVES, M. Aggressive and effective feature selection using genetic programming. In *IEEE Congress on Evolutionary Computation (CEC)* (2012), pp. 1–8.

[197] SARIDEMIR, M. Genetic programming approach for prediction of compressive strength of concretes containing rice husk ash. *Construction and Building Materials 24*, 10 (2010), 1911–1919.

[198] SCHMITT, M. Radial basis function neural networks have superlinear VC dimension. In *International Conference on Computational Learning Theory* (2001), Springer, pp. 14–30.

[199] SEBER, G. A., AND LEE, A. J. *Linear regression analysis*, vol. 936. John Wiley & Sons, 2012.

[200] SEMENKIN, E., AND SEMENKINA, M. Empirical study of self-configuring genetic programming algorithm performance and behaviour. In *IOP Conference Series: Materials Science and Engineering* (2015), vol. 70, IOP Publishing, p. 012004.

[201] SHAO, X., CHERKASSKY, V., AND LI, W. Measuring the VC-dimension using optimized experimental design. *Neural computation 12*, 8 (2000), 1969–1986.

[202] SHAWE-TAYLOR, J., BARTLETT, P. L., WILLIAMSON, R. C., AND ANTHONY, M. Structural risk minimization over data-dependent hierarchies. *IEEE transactions on Information Theory 44*, 5 (1998), 1926–1940.

[203] SHEN, K.-Q., ONG, C.-J., LI, X.-P., HUI, Z., AND WILDER-SMITH, E. P. A feature selection method for multilevel mental fatigue eeg classification. *IEEE Transactions on Biomedical Engineering 54*, 7 (2007), 1231–1237.

[204] SHIBATA, R. An optimal selection of regression variables. *Biometrika 68*, 1 (1981), 45–54.

[205] SIEDLECKI, W., AND SKLANSKY, J. A note on genetic algorithms for large-scale feature selection. *Pattern recognition letters 10*, 5 (1989), 335–347.

[206] SILVA, S., AND COSTA, E. Dynamic limits for bloat control in genetic programming and a review of past and current bloat theories. *Genetic Programming and Evolvable Machines 10*, 2 (2009), 141–179.

[207] SMITS, G., AND VLADISLAVLEVA, E. Ordinal pareto genetic programming. In *IEEE Congress on Evolutionary Computation (CEC)* (2006), IEEE, pp. 3114–3120.

[208] SMITS, G. F., AND KOTANCHEK, M. Pareto-front exploitation in symbolic regression. In *Genetic programming theory and practice II*. Springer, 2005, pp. 283–299.

[209] SMOLA, A. J., ET AL. Regression estimation with support vector learning machines. *Master's thesis, Technische Universit at M unchen* (1996).

[210] SMOLA, A. J., AND SCHÖLKOPF, B. A tutorial on support vector regression. *Statistics and computing 14*, 3 (2004), 199–222.

[211] STIJVEN, S., MINNEBO, W., AND VLADISLAVLEVA, K. Separating the wheat from the chaff: on feature selection and feature importance in regression random forests and symbolic regression. In *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation Conference (GECCO)* (2011), pp. 623–630.

[212] SUGUMARAN, V., MURALIDHARAN, V., AND RAMACHANDRAN, K. Feature selection using decision tree and classification through proximal support vector machine for fault diagnostics of roller bearing. *Mechanical systems and signal processing 21*, 2 (2007), 930–942.

[213] SZUBERT, M., KODALI, A., GANGULY, S., DAS, K., AND BONGARD, J. C. Reducing antagonism between behavioral diversity and fitness in semantic genetic programming. In *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference* (2016), ACM, pp. 797–804.

[214] THORBURN, W. M. The myth of Occam's Razor. *Mind 27*, 107 (1918), 345–353.

[215] TIBSHIRANI, R. Regression shrinkage and selection via the LASSO. *Journal of the Royal Statistical Society. Series B (Methodological)* (1996), 267–288.

[216] TRAN, B., XUE, B., AND ZHANG, M. A new representation in PSO for discretization-based feature selection. *IEEE Transactions on Cybernetics PP*, 99 (2017), 1–14. doi:10.1109/TCYB.2017.2714145.

[217] TRUJILLO, L., SILVA, S., LEGRAND, P., AND VANNESCHI, L. An empirical study of functional complexity as an indicator of overfitting in genetic programming. In *Genetic Programming*. Springer, 2011, pp. 262–273.

[218] UY, N. Q., HIEN, N. T., HOAI, N. X., AND O'NEILL, M. Improving the generalisation ability of genetic programming with semantic similarity based crossover. In *Genetic Programming*. Springer, 2010, pp. 184–195.

[219] UY, N. Q., HOAI, N. X., AND O'NEILL, M. Semantics based mutation in genetic programming: The case for real-valued symbolic regression. In *15th international conference on soft computing, Mendel* (2009), vol. 9, pp. 73–91.

[220] UY, N. Q., HOAI, N. X., O'NEILL, M., MCKAY, R. I., AND GALVÁN-LÓPEZ, E. Semantically-based crossover in genetic programming: application to real-valued symbolic regression. *Genetic Programming and Evolvable Machines 12*, 2 (2011), 91–119.

[221] VALIANT, L. G. A theory of the learnable. *Communications of the ACM 27*, 11 (1984), 1134–1142.

[222] VALIGIANI, G., FONLUPT, C., AND COLLET, P. Analysis of GP improvement techniques over the real-world inverse problem of ocean color. In *Genetic Programming*. Springer, 2004, pp. 174–186.

[223] VANNESCHI, L., CASTELLI, M., MANZONI, L., AND SILVA, S. A new implementation of geometric semantic GP and its application to problems in pharmacokinetics. In *European Conference on Genetic Programming* (2013), Springer, pp. 205–216.

[224] VANNESCHI, L., CASTELLI, M., AND SILVA, S. Measuring bloat, overfitting and functional complexity in genetic programming. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation* (2010), ACM, pp. 877–884.

[225] VANNESCHI, L., CASTELLI, M., AND SILVA, S. A survey of semantic methods in genetic programming. *Genetic Programming and Evolvable Machines 15*, 2 (2014), 195–214.

[226] VANNESCHI, L., SILVA, S., CASTELLI, M., AND MANZONI, L. Geometric semantic genetic programming for real life applications. In *Genetic Programming Theory and Practice XI*. Springer, 2014, pp. 191–209.

[227] VANNESCHI, L., TOMASSINI, M., COLLARD, P., AND CLERGUE, M. Fitness distance correlation in structural mutation genetic programming. In *EuroGP* (2003), vol. 2610, Springer, pp. 455–464.

[228] VAPNIK, V., LEVIN, E., AND LECUN, Y. Measuring the VC-Dimension of a learning machine. *Neural Computation 6*, 5 (1994), 851–876.

[229] VAPNIK, V. N., AND CHERVONENKIS, A. Y. On uniform convergence of the frequencies of events to their probabilities. *Teoriya Veroyatnostei i ee Primeneniya 16*, 2 (1971), 264–279.

[230] VAPNIK, V. N., AND KOTZ, S. *Estimation of dependences based on empirical data*, vol. 40. Springer-verlag New York, 1982.

[231] VAPNIK, V. N., AND VAPNIK, V. *Statistical learning theory*, vol. 1. Wiley New York, 1998.

[232] VLADIMIR, V. N., AND VAPNIK, V. The nature of statistical learning theory, 1995.

[233] VLADISLAVLEVA, E. J., SMITS, G. F., AND DEN HERTOG, D. Order of nonlinearity as a complexity measure for models generated by symbolic regression via pareto genetic programming. *Evolutionary Computation, IEEE Transactions on 13*, 2 (2009), 333–349.

[234] VLADISLAVLEVA, E. Y. *Model-based problem solving through symbolic regression via pareto genetic programming*. CentER, Tilburg University, 2008.

[235] WHIGHAM, P. A., ET AL. Grammatically-based genetic programming. In *Proceedings of the workshop on genetic programming: from theory to real-world applications* (1995), vol. 16, pp. 33–41.

[236] WOLD, S., ESBENSEN, K., AND GELADI, P. Principal component analysis. *Chemometrics and intelligent laboratory systems 2*, 1-3 (1987), 37–52.

[237] WONG, P., AND ZHANG, M. Algebraic simplification of GP programs during evolution. In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation Conference companion(GECCO)* (2006), ACM, pp. 927–934.

[238] XU, Z., SHI, X., WANG, L., LUO, J., ZHONG, C.-J., AND LU, S. Pattern recognition for sensor array signals using fuzzy artmap. *Sensors and Actuators B: Chemical 141*, 2 (2009), 458–464.

[239] XUE, B., ZHANG, M., AND BROWNE, W. N. Single feature ranking and binary particle swarm optimisation based feature subset ranking for feature selection. In *Proceedings of the Thirty-fifth Australasian Computer Science Conference-Volume 122* (2012), Australian Computer Society, Inc., pp. 27–36.

[240] XUE, B., ZHANG, M., AND BROWNE, W. N. Particle swarm optimization for feature selection in classification: a multi-objective approach. *IEEE transactions on cybernetics 43*, 6 (2013), 1656–1671.

[241] XUE, B., ZHANG, M., AND BROWNE, W. N. Particle swarm optimization for feature selection in classification: A multi-objective approach. *IEEE Transactions on Cybernetics 43*, 6 (2013), 1656–1671.

[242] XUE, B., ZHANG, M., BROWNE, W. N., AND YAO, X. A survey on evolutionary computation approaches to feature selection. *IEEE Transactions on Evolutionary Computation 20*, 4 (2016), 606–626.

[243] YU, L., AND LIU, H. Feature selection for high-dimensional data: A fast correlation-based filter solution. In *ICML* (2003), vol. 3, pp. 856–863.

[244] ZAMALLOA, M., BORDEL, G., RODRÍGUEZ, L. J., AND PEÑAGARIKANO, M. Feature selection based on genetic algorithms for speaker recognition. In *Speaker and Language Recognition Workshop, 2006. IEEE Odyssey 2006: The* (2006), IEEE, pp. 1–8.

[245] ZELINKA, I., OPLATKOVA, Z., AND NOLLE, L. Analytic programming–Symbolic regression by means of arbitrary evolutionary algorithms. *Int. J. of Simulation, Systems, Science and Technology 6*, 9 (2005), 44–56.

[246] ZHANG, B.-T., AND MHLENBEIN, H. Balancing accuracy and parsimony in genetic programming. *Evolutionary Computation 3*, 1 (1995), 17–38.

[247] ZHANG, M., WONG, P., AND QIAN, D. Online program simplification in genetic programming. In *Asia-Pacific Conference on Simulated Evolution and Learning* (2006), Springer, pp. 592–600.

Appendix A

Improving Generalisation of Genetic Programming for High-Dimensional Symbolic Regression with Feature Selection

[This is a short version of the paper:
Qi Chen, Bing Xue, Mengjie Zhang. "Improving Generalisation of Genetic Programming for High-Dimensional Symbolic Regression with Feature Selection". *Proceedings of 2016 IEEE World Congress on Computational Intelligence/ IEEE Congress on Evolutionary Computation (WCCI 2016 /CEC2016)*. Vancouver, Canada. 24-29 July, 2016. pp. 3793-3800.]

A.1 Introduction

Recent years, the dimensionality of real-world data is becoming increasingly higher as the data collection techniques evolve. The properties of high-dimensional data can affect the ability of learning algorithms to extract useful information from original data. Furthermore, many issues

arise when learning from high-dimensional data, for example, the curse of dimensionality [22], risk of overfitting, and high computational cost. In addition, learning from all these features does not necessarily perform well, since noise is more likely to accumulate during the observation of too many features. Feature selection is a process of identifying relevant features that are necessary to describe the output variables. When learning from high-dimensional data, feature selection is desired. A plethora of contributions have already been devoted to feature selection. However, most of them are developed for classification. When using Genetic Programming (GP) [124] for symbolic regression (SR), feature selection is seldom considered, and no research on feature selection has been reported for high-dimensional symbolic regression tasks.

Generalisation measures the performance of learned models on unseen data. Learning algorithms with good generalisation capability can produce models with similar performance on unseen data as obtained from the training data. While generalisation has been considered as an important aspect in many fields in machine learning for a long time [91, 210, 12, 60], it has not received enough attention as it deserves in GP for symbolic regression. Prior to Kushchu's work on generalisation of GP in 2002 [130], most of the contributions on GP based symbolic regression did not investigate the performance on the unseen test sets. Since then, growing attention has been devoted to promote the generalisation of GP [218, 42, 98]. However, contributions to the generalisation in GP have not developed near enough compared with the fast development of GP for symbolic regression. Thus, generalisation remains an open issue on GP [173].

A.1.1 Goals

This work aims to develop a new feature selection method in GP for high-dimensional symbolic regression problems and investigate whether GP with the new feature selection method can enhance its generalisation capa-

bility. Specifically, this work will investigate the following research questions:

- whether the new feature selection method can increase the learning ability of GP on training data,
- whether the new feature selection method can promote the generalisation performance of GP on unseen data, and
- how the feature selection method influences the size and number of features of the models evolved by GP.

A.2 The Proposed Method

The new feature selection method proposed in this work is based on the fact that GP can explore the search space to detect important/informative features automatically. We assumed that relevant features must present in high fitness individuals even though not all the features present in these individuals are relevant. Features present in highly fitted individuals can be a candidate subset for selecting good features to construct desired models.

Base on this assumption, we propose a new method which is named *genetic programming with feature selection* (GPWFS) to improve the natural feature selection ability of GP by introducing a very simple mechanism. The flow chart for describing GPWFS is shown in Fig. A.1.

In GPWFS, the population of candidate solutions will be virtually divided into two groups based on their fitness values: the *good* individuals and the *not good enough* individuals. The split of individuals depends on the value of a parameter α ($\alpha \in (0, 1)$), which refers to the proportion of population to be considered to be *good*. For example, when the parameter α is set to be 0.1, it means GPWFS will treat the top 10% of the individuals in the population to be *good* and the rest 90% are *not good enough*

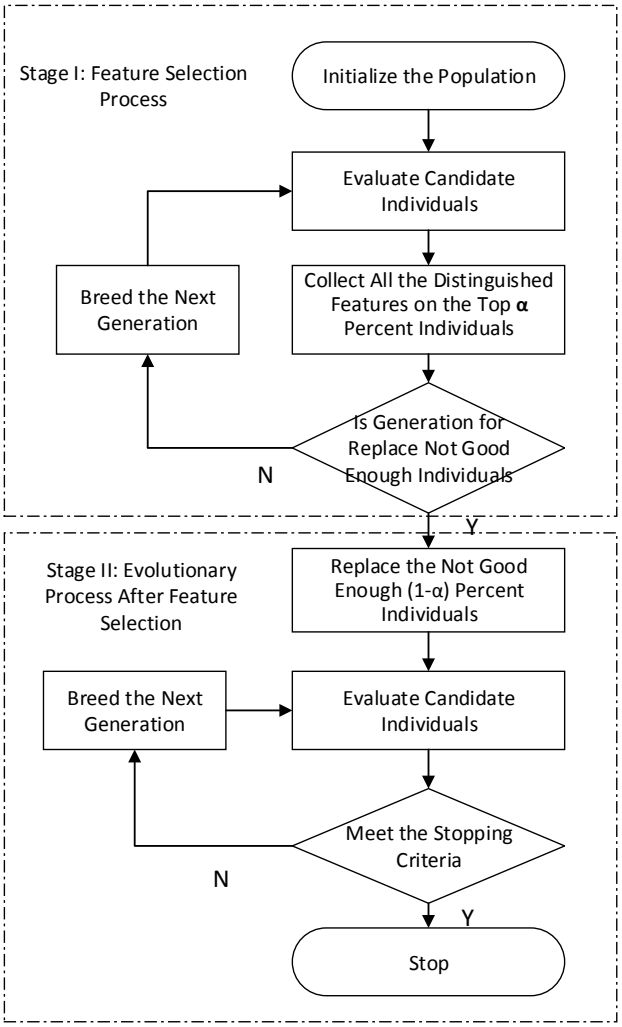


Figure A.1: Flow Chart of GPWFS.

individuals. While α should neither be too small, in which case too many individuals in the population will be treated as *not good enough* ones, and potentially good features will be missed since they have a higher probability to appear in *not good enough* individuals, nor be too large, which will lead to an excessive number of selected features thus degrades the

effectiveness and efficiency of feature selection.

In addition, GPWFS splits the whole evolutionary process into two stages. The first stage is the feature selection process. During this stage, on every generation, all the distinguished features present in the *good* individuals are collected. At the end of the feature selection process, a set of selected features F_c is formed, which is a subset of original features. The second stage is a standard GP evolutionary process with an additional component. The component is that, at the start generation of the second stage, the population of GP is reinitialised by replacing the *not good enough* individuals with an equal number of randomly initialised individuals, while keeping the *good* individuals. The features used to reinitialise the new individuals are all from F_c instead of the original features. Then a standard GP evolutionary process will perform on the reinitialised population generation by generation until the stopping criterion is met.

A parameter G_f defines how to split the evolutionary process into two stages. The first G_f generations belong to the feature selection process and the rest generations are used for evolutionary process after feature selection. The value of G_f relies on the maximum number of generations. It should be neither too small nor too large. When G_f is too small, the number of generations will not be enough for GP to find good features by searching the feature space. When G_f is too close to the end generation, the number of generations for evolving the newly reinitialised population is too small, and there will not be enough time for GP to converge to optimal (near optimal) solutions. A good trade-off of the two stages of GPWFS relies on the setting of G_f.

A.3 Experiment Design

This work uses standard GP as a baseline for comparison. Both GPWFS and standard GP were tested on six real-world high-dimensional regression datasets.

A.3.1 Evaluation Measure — Fitness Function

The performance of models evolved by GPWFS and standard GP are evaluated by the *Normalised Root Mean Square Error* (NRMSE) on both the training set and the test set. The fitness function is shown in Equation (A.1).

$$NRMSE = \frac{RMSE}{Y_{max} - Y_{min}} \tag{A.1}$$

where $Y_{max} - Y_{min}$ is the range of the target outputs and $RMSE$ is the root mean square error and is denoted in Equation (A.2).

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(f(X_i) - Y_i)^2} \tag{A.2}$$

where N is the number of instances, $f(X_i)$ is the output(s) of the candidate models and Y_i is the target output(s).

A.3.2 Parameters

The parameters for GPWFS and standard GP are summarised in Table D.2.

GPWFS has two key parameters: the parameter α which refers to the proportion of *good* individuals in the candidate population and the parameter G_f which refers to the number of generations to split the two stages of GPWFS.

α is set to be 0.05 since the size of population (which is 512) is relatively large, top 5% individuals of the population (the number of which is 25) are enough for represent the *good* individuals and contain enough good features. The parameter G_f can not be either too small or too large. Since the maximum number of generations in this work is 100, the reasonable values for G_f are 40, 50 and 60. The results of a few trails show that setting G_f to 60 is a good choice to achieve a good trade-off between the two stages of GPWFS.

Table A.1: Parameters for GP

parameter	Values
Population Size	512
Generations	100
Crossover Rate	0.9
Mutation Rate	0.1
Elitism Rate	0.01
Maximum Tree Depth	17
Initialisation	Ramped-Half&Half
Minimum Initialisation Depth	2
Maximum Initialisation Depth	6
Function Set	$+, -, *, \%$protected
Terminal Set	Features, Random Constant $\in [-1.0, 1.0)$
Fitness Function	NRMSE
Generation for Feature Selection — G_f	60
Percentage of Top Individuals — α	5%

A.3.3 Test Problems

Since GP does not have any benchmark dataset designed for the research of generalisation specifically, this work took six real-world datasets from previous contributions on generalisation of GP for symbolic regression [14, 226, 224] and UCI [138]. These six datasets are all have a large dimensionality of the feature space for regression tasks and feature selection for the datasets seems to be desired. The number of features and instances of the six datasets are shown in TABLE A.2.

The first two datasets tackle problems in the field of pharmacokinetics. The task of these two datasets is to predict the value of two different pharmacokinetics parameters. While the first dataset is to predict the human oral bioavailability (represent as %F), the second one is to predict the median lethal dose (LD50). These two datasets have been used in many recent works [14, 226, 224, 223, 68] (For %F, we understand from [68], some features in this dataset are redundant/useless. We would like to investigate whether the proposed GP method can automatically select relevant features for regression, so we include this dataset in this work). For more information of these two datasets, readers are referred to [14, 68].

Table A.2: Benchmark Problems

Name	# Features	#Total Instances	#Training Instances	#Test Instances
%F	241	359	251	108
LD50	626	234	163	71
DLBCL	7399	240	180	60
CCUN	124	1994	1395	599
CCN	122	1994	1395	599
RLCT	384	53500	37450	16050

The third dataset is the Diffuse Large-B-Cell Lymphoma (DLBCL), which was collected from Rosenwald et al. [191]. The task of this dataset is to predict the survival time of patients who have diffuse large-B-cell lymphoma and received chemotherapy.

The rest three datasets are taken from UCI [138]. Two of the datasets are about communities and crime within the United States. They are the Communities and Crime unnormalised dataset (CCUN) and the Communities and Crime normalised dataset (CCN). These two dataset were used in [188] and both of them are to predict the per capita crimes. As shown in TABLE A.2, the number of instances of the two datasets used in this work is different from the original data. The values are based on the discard of the instances which have missing values of features and the output variable. The last dataset is the Relative location of CT slices on axial axis (RLCT). The task of RLCT is to predict the relative location of the CT slice on the axial axis of the human body.

A.3.4 Training sets and Test sets

In this work, each dataset is split into a training set and a test set in order to investigate the generalisation performance of the evolved model of standard GP and GPWFS on unseen data. The numbers of instances on the training sets and the test sets are shown in TABLE A.2. Among the six datasets, DLBCL is the only one on which the training set and test set are provided. All the experiments conducted on the other five datasets are

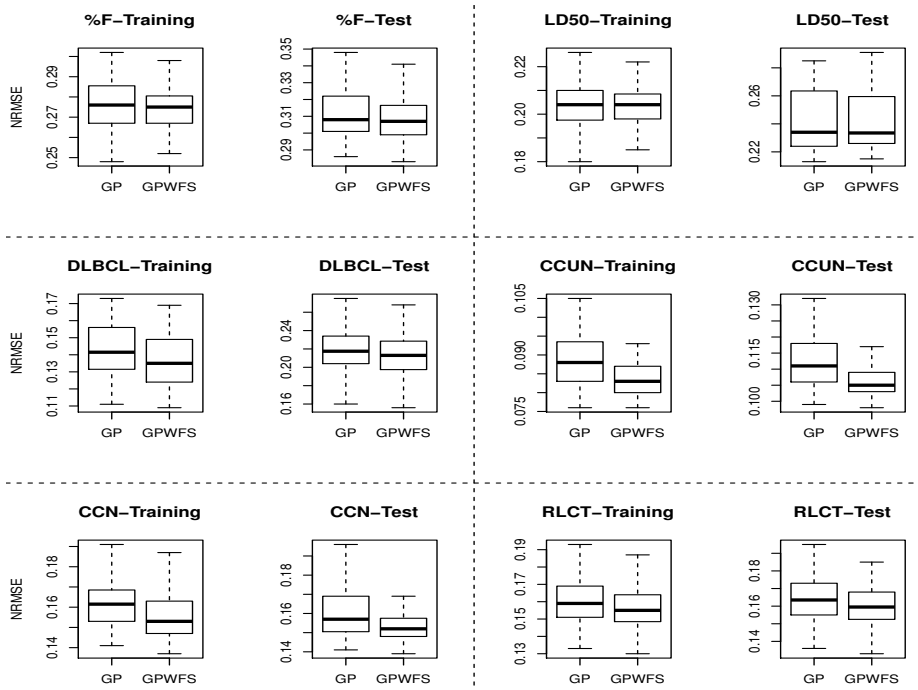


Figure A.2: Distribution of NRMSE of the 100 best-of-runs individuals.

with 70% of instances randomly selected from the dataset for training and the other 30% instances forms the test set, since this is a common way in many previous research [226, 175, 200].

The experiments of each method have been conducted for 100 independent runs on every dataset. Therefore, 1200 (i.e. 2*6*100) experiments have been run for the two methods on six datasets and 2400 (i.e. 1200*2) training and test results are used here to discuss the feature selection performance and generalisation of GPWFS for high-dimensional SR.

A.4 Results and Discussions

The experimental results of GPWFS and standard GP are presented and discussed in this section. The results will be presented in terms of comparisons of NRMSEs on the training sets and the test sets, the size of the

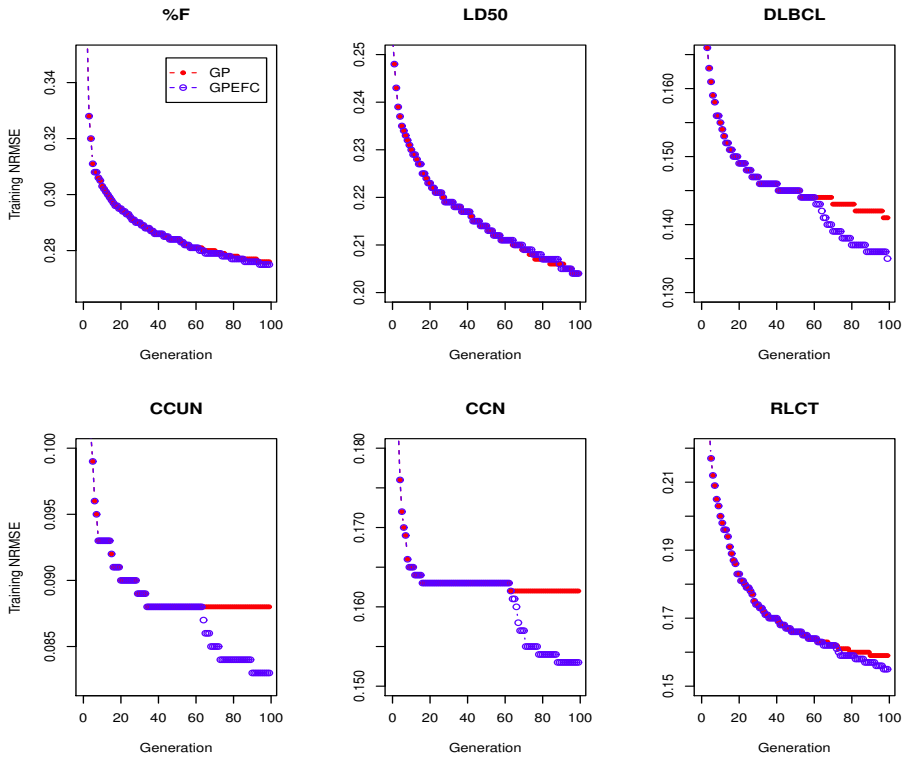


Figure A.3: The evolution plots of the median NRMSE of 100 runs of the best individual on every generation on the *Training Sets*.

100 best-of-run models, and the number of features and distinguished features used to construct these models. The comparison of computational time on the six datasets between the two methods will also be provided.

Fig. D.2 shows the distribution of NRMSEs of the 100 best-of-run individuals on the six datasets. Each dataset has two boxplots, one for the training set and the other for the test set. Each boxplot consists of a pair of whiskered boxes, the first one displays the results of standard GP and the second one shows its counterpart of GPWFS.

The evolution plots of the training sets are shown in Fig. A.3, while Fig. A.4 is for their counterparts on the test sets. Since the median value is suggested to be more robust to outliers [98], it is preferred over the mean value in this work and the evolution plots are drawn using the median

values of the best individuals of 100 runs on every generation.

The Wilcoxon test, which is an non-parametric statistical significant test, is used in this work to compare the NRMSE values of 100 best-of-run programs of standard GP and GPWFS both on the training sets and the test sets. The significance level is 0.05.

A.4.1 Results on the Training Sets

The evolution plots of the median NRMSE of best individuals of 100 runs on training sets on every generation are shown in Fig. A.3. As it shows, GPWFS generally achieves better training performance than standard GP on all the six problems. While GPWFS has slight better training performance on %F, LD50 and RLCT, on the other three datasets (DLBCL, CCUN and CCN), it has a quite dramatic improvement on training performance over standard GP. The statistical test results are that on five of the six datasets, GPWFS can have significant better training performance (except for LD50 which is slightly better but not significant).

The training boxplots in Fig. D.2, which display the distribution of NRMSE of the 100 best-of-run individuals on the training sets, also confirm the advantage of GPWFS on the training sets.

The training results suggest that GPWFS, which is equipped with feature selection and the reinitialisation of a big proportion of individuals in the population using the selected features, could incorporate more useful information in the training set, thus have a positive effect on enhancing the learning/training/optimisation ability of GP. The feature selection process can effectively explore the search space and automatically collect good features from the *good* individuals by GPWFS.

A.4.2 Results on the Test Sets — Generalisation

Fig. A.4 presents the evolution plots of the median NRMSE of 100 best-of-generation individuals on the test sets. It is easy to observe that the overall

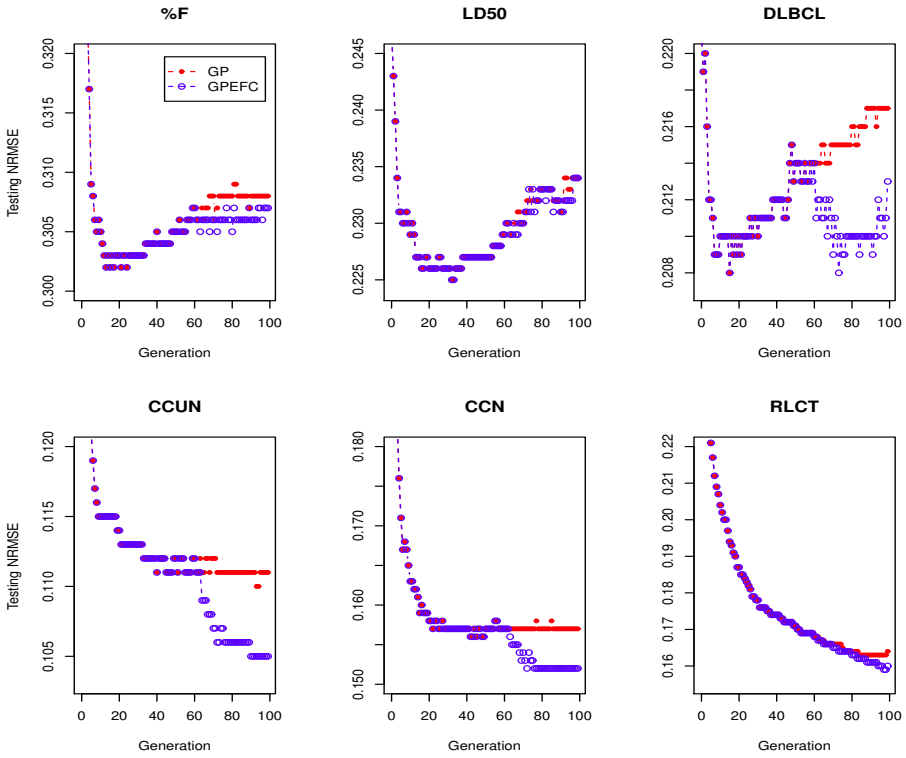


Figure A.4: The evolution plots of the median NRMSE of 100 runs of the best individual on every generation on the *Test Sets*.

pattern on the test sets is very similar to the training sets. GPWFS is still superior to standard GP on all the six datasets. The results of statistical test on the test errors of 100 best-of-run individuals show that, on four of the six datasets, GPWFS has significant better generalisation performance than standard GP, except for %F and LD50 where GPWFS is slightly better but no significant differences have been found.

On the first two datasets (%F and LD50), overfitting happens. The issue of overfitting might be due to the small ratio of instances over the number of features in the two datasets that does not provide enough useful information to the evolutionary process. Although both methods have increasing test errors over generations, GPWFS can still have lower NRMSE on both tasks, i.e., better generalisation ability than standard GP.

On the other four datasets, GPWFS has much better generalisation performance, which differs from the first two benchmarks. On these datasets, the pattern on the test sets is the same as the training set. On the first three datasets — DLBCL, CCUN and CCN, comparing to standard GP, GPWFS has a dramatical generalisation gain, which is significant (p-values of the three benchmarks are: DLBCL=0.008, CCUN=6.15E-10, CCN=2.05E-10). It is notable that, while standard GP has overfitting problem on DLBCL, GPWFS can eliminate overfitting effectively and has a dramatical generalisation gain. On the task of RLCT, GPWFS can also have significant better generalisation ability than standard GP (p-value=5.12E-7), although the distance of NRMSE between the two methods looks much smaller than that on the other three datasets.

On most of the six datasets, GPWFS contributes a positive effect on enhancing the generalisation performance of GP. When overfitting happens (e.g. %F, LD50), the feature selection process tends to select features that is possibly unique to the training set, thus the positive effect of GPWFS on improving generalisation may decrease. In the future, we will find an effective way to avoid this kind of overfitting.

A.4.3 Further Analysis — Result on Program Size, Number of Total Features and Distinguished Features

TABLE A.3 shows the average program size of the best-of-run individuals, the average number of features and the average number of distinguished features for constructing these individuals. It can be observed that the average program size of GPWFS is smaller than GP on all the six datasets. It is obvious that GPWFS can evolve more compact models than GP, which are generally faster in execution and easier to interpret.

As it is shown in TABLE A.3, comparing to the original number of features, there is a dramatic drop in the average number of features on all the six benchmarks on both methods. The trend on the number of features is

Table A.3: Program Size, Number of Features and Distinguished Features

Dataset	#Original Features	Method	#Node (Mean±Std)	#Features (Mean±Std)	#Distinguished Features (Mean±Std)
%F	241	GP	260.02±86.35	55.01±24.76	11.72±4.58
		GPWFS	242.0±70.52	52.77±20.9	13.41±4.77
LD50	626	GP	295.12±79.98	75.37±25.25	16.29±5.27
		GPWFS	286.72±73.75	74.98±24.45	17.86±5.64
DLBCL	7399	GP	160.08±84.89	40.49±22.89	4.16±2.22
		GPWFS	155.12±78.13	38.96±21.46	6.02±2.8
CCUN	124	GP	113.06±85.9	33.74±29.72	3.47±2.33
		GPWFS	108.8±71.45	30.13±22.71	5.1±2.88
CCN	122	GP	105.36±78.33	30.99±29.63	3.5±2.23
		GPWFS	93.76±62.6	26.84±21.37	5.47±2.99
RLCT	384	GP	251.6±76.22	51.4±22.2	11.9±3.87
		GPWFS	235.4±64.36	48.48±17.64	13.51±3.82

the same as the program size. Compared with standard GP, GPWFS uses a smaller number of features to construct the best-of-run individuals.

As already mentioned, the reason for the overfitting on the first two datasets (%F and LD50) might be that the ratio of instances over the number of features is small. From TABLE A.3, the extremely large average program size, the numbers of features and distinguished features of the best individuals on these two datasets, which are much higher than the other four datasets, can also confirm the reason. On the DLBCL, GPWFS can effectively select a small but sufficient number of good features from the very large number of original features (i.e. 7399), which shrinks the search space for the evolutionary process in the second stage, and potentially prevents the training/evolutionary process from being overfitted to the training set.

In terms of distinguished features, it is interesting to note that GPWFS has a slightly larger average number of distinguished features in the evolved models on all the six problems, which is different from the number of features. This phenomenon may be due to the feature selection process, which reduces the number of features while keeping the relevant/informative features. Thus these distinguished relevant/informative

features have more opportunities to be selected into the candidate programs instead of the building blocks used in standard GP. In these building blocks, a feature might be manipulated several times in order to get enough information.

A.4.4 Computational Cost

TABLE A.4 shows the average computational time of the 100 runs in milliseconds on the six datasets. As it shows, GPWFS has a lower computational cost than standard GP which is around 10% lower on all the six datasets. It is not very intuitive. Compared with standard GP, GPWFS needs additional computational cost for feature selection process which including ranking the individuals and collecting features from the *good* individuals. However, the smaller program size of GPWFS (as showing in TABLE A.4) is a reason for the decrease of computational cost. For the smaller models, evaluation cost which is the major cost of evolutionary process will be much smaller than their bigger counterparts. The lower evaluation cost of the smaller programs should be the major reason for the efficiency of GPWFS.

Table A.4: Program Size and Computational Time

Dataset	Method	#Node (Mean±Std)	Time (Millisecond) (Mean±Std)
%F	GP	260.02±86.35	7806.23±2891.83
	GPWFS	242.0±70.52	7339.25±2703.76
LD50	GP	295.12±79.98	7188.16±3124.71
	GPWFS	286.72±73.75	6610.9±2624.44
DLBCL	GP	160.08±84.89	3537.37±2146.43
	GPWFS	155.12±78.13	3221.97±1647.59
CCUN	GP	113.06±85.9	1.76E4±1.44E4
	GPWFS	108.8±71.45	1.51E4±1.09E4
CCN	GP	105.36±78.33	1.65E4±1.22E4
	GPWFS	93.76±62.6	1.41E4±9330.36
RLCT	GP	251.6±76.22	1.81E6±5.22E5
	GPWFS	235.4±64.36	1.67E6±4.53E5

A.5 Conclusions and Future Work

This work proposed a new method — GPWFS, which is a feature selection method to GP, and investigated how it can influence the generalisation of GP for high-dimensional symbolic regression tasks. For this purpose, a series of experiments have been conducted on six real-world symbolic regression datasets with a large dimensionality of the feature space.

The results show that GPWFS can evolve more compact model for all the datasets. It uses more distinguished features to construct the models while keeping the number of features small. Compared with models evolved by standard GP, these compact models can have not only significantly better regression performance on the training sets but also huge generalisation gains on the unseen data. Furthermore, GPWFS generally spend less computational time than standard GP since it can reduce the size of models effectively.

The results of the experiments provided strong evidence for the important role of feature selection in enhancing the generalisation of GP. However, further analysis and explanation of the evolved solutions will be needed. So in the future, we are intending to investigate more detail of the evolved models. Furthermore, we plan to incorporate an overfitting detecting mechanism, for example using a validation set, and/or introducing model complexity measurement to GPWFS. It is expected to improve the generalisation of GP in a more effective way. Last but not least, we also plan to do a more comprehensive comparison between the generalisation ability of GPWFS with some other generalisation approaches in the previous publications for GP, such as validation set [88], semantic based crossover [218], GP with linear scaling and no same mate [62].

Appendix B

Analysis on the Correlation of Features on the Real-world Data in Chapter 3

Previous research shows that permutation importance in random forest performs poorly when features in the dataset are highly correlated. It is worth to have an analysis on how correlated the features are in the four real-world datasets in our work. However, it is difficult to analyse the correlation of a set of features in these high-dimensional regression datasets. Thus, we have analysed the pairwise correlation of features. We obtained the correlation matrices using Pearson correlation measure and visualised them in correlation plots, which are shown in Fig.B.1, where the red colour represents the highly correlated relationship and the green colour means weak correlation between the features. As it shows, the first plot, which shows the correlation values between features in LD50, which contains large areas of red colour. It indicates features in LD50 are highly correlated. On the other three datasets, compared with the large number of available features, the number of highly correlated features is small. In the later three plots, most of the values are showing in green colour, which means that most of the correlation values are around zero.

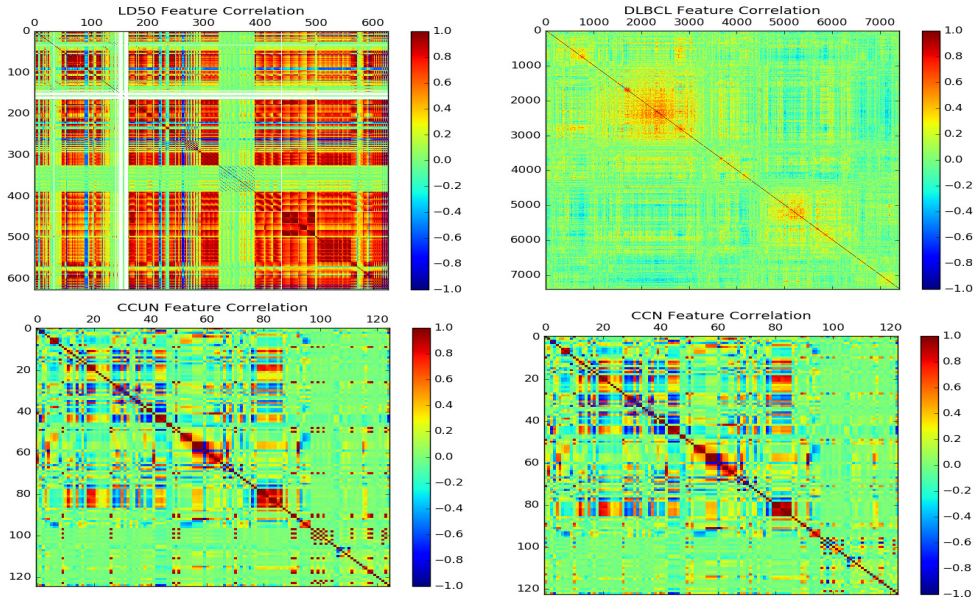


Figure B.1: The Correlation Plots.

No much highly correlated features have been found in any of the other three datasets.

Appendix C

Derivations of the Formula in Chapter 4

C.1 Derivation of the Theoretical Formula of the Maximum deviation

To estimate a bound on the expectation of the maximum deviation between errors (here are the errors obtained on two independently generated paired datasets), 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 $Z^{2n} = X_1, Y_1, X_2, Y_2, \ldots, X_{2n}, Y_{2n}$ 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, $Pe_2(Z^{2n})$ denote the error rate on the other n samples. The error rate is obtained by:

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

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

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

where $\epsilon(n)$ stands for the maximum deviation and sup is the supremum (least upper bound) of the set of derivations. Assume there exist an upper bound for the maximum deviation, and consider the bound of under following three cases [228]:

Case 1. When the number of instances is extremely small ($\frac{n}{h} <= 0.5$), a trivial bound is used, i.e.

$$E\{\epsilon(n)\} \leq 1$$

Case 2. When the number of instances is small ($0.5<\frac{n}{h}<u$, u is small), according to [228], the following conditional expectation of $\epsilon(n)$ valid:

$$E\{\epsilon(n)|C_\delta\} \leq \frac{4}{\delta}\left[\frac{ln(2n/h)+1}{l/h} + \frac{1-P(C_\delta)}{l}\right]$$

where C_δ is the condition denoted by:

$$\frac{Pe_1(Z^{2n},\alpha) - Pe_2(Z^{2n},\alpha)}{[Pe\,(Z^{2n},\alpha)+1/2n]\,[1+1/2n-Pe\,(Z^{2n},\alpha)]} > \delta$$

$P(C_\delta)$ is the probability of C_δ , and

$$Pe\left(Z^{2n},\alpha\right) = \frac{Pe_1(Z^{2n},\alpha)+Pe_2(Z^{2n},\alpha)}{2}$$

When the n/h is not large, usually $\delta \geq 0.5$ and $P(C_\delta)$ approaches 1. In this scenario, according to the conditional expectation, the maximum deviation is bounded as follows:

$$E\{\epsilon(n)\} \leq C_1\frac{ln(2n/h)+1}{n/h}$$

where C_1 is a constant.

Case 3. When there are sufficient instances(n/h is large), according to inequality defined in [230] (P172)

$$P(\epsilon(n)>\theta) < 3\left(\frac{2ne}{h}\right)^h exp\left(-\theta^2 n\right)$$

so that,

$$E\{\epsilon(n)\} < \int_0^w \mathrm{d}\theta + 3\left(\frac{2ne}{h}\right)^h \int_w^\infty \exp\left(-\theta^2 \mathrm{n}\right) \mathrm{d}\theta$$

the right side of the inequality is smaller than

$$w + 3\left(\frac{2ne}{h}\right)^h \int_w^\infty \exp\left(-\theta \mathrm{wn}\right) \mathrm{d}\theta$$

which equals to

$$w + 3\left(\frac{2ne}{h}\right)^h \frac{1}{nw} exp(-wn)$$

For $w = \sqrt{\frac{ln(2n/h)+1}{n/h}}$,

$$E\{\epsilon(n)\} < \sqrt{\frac{ln(2n/h)+1}{n/h}} + \frac{3}{\sqrt{nh[ln(2n/h)+1]}} < C_2\sqrt{\frac{ln(2n/h)+1}{n/h}}$$

where C_2 is a constant.

Then we have the bound as:

$$E\left\{\epsilon(n)\right\} \leq \begin{cases} 1 & \text{if } \frac{n}{h} <= 0.5 \\ C_1 \frac{\ln(2n/h)+1}{n/h} & \text{if } 0.5 < \frac{n}{h} <= u \\ C_2\sqrt{\frac{\ln(2n/h)+1}{n/h}} & \text{if } \frac{n}{h} > u \end{cases} \tag{C.3}$$

Using a continuous approximation, the right side of the bound is represent as

$$\Phi(\frac{n}{h}) = \begin{cases} 1 & \text{if } \frac{n}{h} <= 0.5 \\ a\frac{\ln(2\frac{n}{h})+1}{\frac{n}{h}-k}\left(\sqrt{1+\frac{b\left(\frac{n}{h}-k\right)}{\ln(2\frac{n}{h})+1}}+1\right) & \text{otherwise.} \end{cases} \tag{C.4}$$

where the parameters a and b respectively determine the small and larger regions of n/h , i.e. the role of u in **Case 2** and **Case 3**. The values of a and b are found by fitting Equation (C.4) to the experimental values of the maximum deviation of linear models, the VC-dimension of which are known. Accordingly, it found that $a = 0.16$ and $b = 1.2$ [228]. The third parameter k in Equation (C.4) is chose to keep the continuity at $\Phi(0.5)$, i.e. to make $\Phi(0.5) = 1$.

Inspired by the bound for small n/h (i.e. the bound in **Case 2**), $\Phi(\frac{n}{h})$ has a simpler formula:

$$\Phi^*(\frac{n}{h}) = d\frac{ln(2n/h+1)}{(n/h)+d-0.5}$$

Applying this simpler formula for linear models, it is easy to get $d = 0.39$. In addition, it also found that this formula performs well for a small n/h (up to $n/h = 8$). That is why $u \approx 8$.

Appendix D

Geometric Semantic Crossover with an Angle-aware Mating Scheme in Genetic Programming for Symbolic Regression

[This is a short version of the paper:
Qi Chen, Mengjie Zhang, Bing Xue. "Angle-aware Geometric Semantic Crossover in Genetic Programming for Symbolic Regression". *Proceedings of the 20th European Conference on Genetic Programming (EuroGP 2017)*. Lecture Notes in Computer Science, **Vol. 10196**. Amsterdam, The Netherlands. 18-21 April 2017. pp. 229-245.]

D.1 Introduction

In recent years, semantic genetic programming (GP) [149, 225], which incorporates the semantic knowledge in the evolutionary process to improve the efficacy of search, attracts increasing attention and becomes a hot research topic in GP [124]. One popular form of semantic methods, geomet-

ric semantic GP (GSGP), has been proposed recently [156]. GSGP searches directly in the semantic space of GP individuals. The geometric crossover and mutation operators generate offspring that lies within the bounds defined by the semantics of the parent(s) in the semantic space. The fitness landscape that these geometric operators explore has a conic shape, which contains no local optimal and is easier to search. In previous research, GSGP presents a notable learning gain over standard GP [226, 223]. For the generalisation improvement, GSGP shows some positive effect. However, while the geometric mutation is remarked to be critical in bringing the generalisation benefit, the geometric crossover is criticised to have a weak effect on promoting generalisation for some regression tasks [98]. One possible reason is that of the target output on the test set is beyond the scope of the convex combination of the parents for crossover [171] in the test semantic space. Another possible reason is that crossover might operate on similar parents standing in a compact volume of the semantic space, which leads to generating offspring having duplicate semantics with their parents. In this case, the population has difficulty to converge to the target output, no matter the target semantic is in or out of the covered range. Thus, the offspring produced by the geometric crossover is difficult to generalise well. Therefore, in this work, we are interested in improving the geometric crossover by addressing this issue.

The overall goal of this work is to propose a new angle-aware mating scheme to select for geometric semantic crossover to improve the *generalisation* of GP for symbolic regression. An important property of the geometric semantic crossover operator is that it generates offspring that stands in the segment defined by the two parent points in the semantic space. Therefore, the quality of the offspring is highly dependent on the positions of the two parents in the semantic space. However, such impact of the parents on the effectiveness of geometric semantic crossover has been overlooked. In this paper, we propose a new mating scheme to geometric crossover to make it operats on parents that are not only good

at fitness but also have large *angle* in terms of their relative positions to the target point in the semantic space. Our goal is to study the effect of the newly proposed mating scheme to geometric crossover operator. Specific research objectives are as follows:

- to investigate whether the geometric crossover with angle-awareness can improve the learning performance of GSGP,
- to study whether the geometric crossover with angle-awareness can improve the generalisation ability of GSGP, and
- to investigate how the geometric crossover with angle-awareness influences the computational cost and the program size of the models evolved by GSGP.

D.2 Angle-aware Geometric Semantic Crossover (AGSX)

In this work, tree based GP is employed, and we propose a new angle-aware mating scheme for Geometric Semantic Crossover (AGSX). This section describes the main idea, the detailed process, the characteristics of AGSX, and the fitness function of the GP algorithm.

D.2.1 Main Idea

How the crossover points spread in the semantic space is critical to the performance of GSGP. A better convergence to the target point can be achieved if the convex combinations cover a larger volume when the convex hull is given. AGSX should be applied to the parents that the *target output* is around the intermediate region of their semantics. Given that the semantics of the generated offspring tend to lie in the segment of the semantics of the parents as well, AGSX is expected to generate offspring

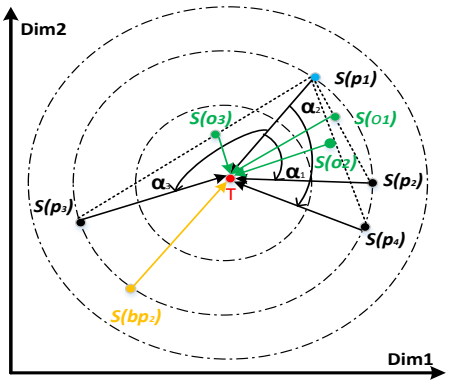


Figure D.1: AGSX in two Dimension Euclidean Semantic Space.

that is close to the target output. To promote the convex combinations to cover a larger volume, the two parents should have a larger distance in the semantics space.

The semantic distance between the parents can be used here, but it often leads to a quick loss of semantic diversity in the population and then results in a premature solution. Therefore, we utilise the angle between the relative semantics of the parents to the target output to measure their distance in the semantic space. Specifically, suppose the target output is $\vec{T}$, and the semantics of the two parents are $\vec{S}_1$ and $\vec{S}_2$, the angle α between the relative semantics of the two parents to the target output is defined as follows:

$$\alpha = \arccos\left(\frac{(\vec{S}_1 - \vec{T}) \cdot (\vec{S}_2 - \vec{T})}{\|\vec{S}_1 - \vec{T}\| \cdot \|\vec{S}_2 - \vec{T}\|}\right) \tag{D.1}$$

where $(\vec{S}_1 - \vec{T}) \cdot (\vec{S}_2 - \vec{T}) = \sum_{i=1}^{n}(s_{1i} - t_i) \cdot (s_{2i} - t_i)$ and $\|\vec{S} - \vec{T}\| = \sqrt{\sum_{i=1}^{n}(s_i - t_i)^2}$. i stands for the ith dimension in the $n-$dimensional semantic space. s_{1i}, s_{2i}, and t_i are the values of S_1, S_2 and $\vec{T}$ in the ith dimension, respectively.

Fig.D.1 illustrates the mechanism of AGSX in a two-dimensional Euclidean space, which can be scaled to any n-dimensional space. Each point represents the semantics of one individual in GP. As shown in the figure, there are four individuals p_1, p_2, p_3 and p_4, which can be selected as the par-

ents of AGSX. Assume p_1 (in blue colour) has been selected as one parent and the mate, i.e. the other parent, needs to be selected from p_2, p_3 and p_4 to perform AGSX. α_1, α_2, and α_3 show the angles in the three pairs of parents, i.e. $\langle p_1, p_2 \rangle$, $\langle p_1, p_4 \rangle$ and $\langle p_1, p_3 \rangle$, respectively. The three green points, $S(o_1)$, $S(o_2)$, and $S(o_3)$, show the three corresponding offspring of the three pairs of parents, and the green lines indicates their distances to the target point. It can be seen from the figure that the pair of parents $\langle p_1, p_3 \rangle$ has a larger angle, i.e. α_3, and the generated offspring $S(o_3)$ is closer to the target output. In the ideal case where the yellow point $S(bp_2)$ is the second parent, the generated offspring is very likely to be the target point. In other words, if the parents have a larger angle between their relative semantics to the target output, the generated offspring tends to be closer to the target output. Therefore, we need to select parents with a large angle in their relative semantics to the target output.

To achieve this, we develop a new mating scheme to select parents with a large angle in their relative semantics to the target output. First, a list of candidate parents called the $WaitingSet$ is generated by repetitively applying a selection operator (e.g. tournament selection) to the current population. The size of $WaitingSet$ is determined by the population size N and the crossover rate R_X, i.e. $|waitingset| = N \cdot R_X$. Then, the parents for each AGSX operation are selected from $WaitingSet$ without replacement so that the angles between the relative semantics of the selected parents can be maximised. The detailed process of AGSX is given in Section D.2.2.

D.2.2 The AGSX Process

The pseudo-code of AGSX is shown in Algorithm 6. The procedure of finding the mate having the largest relative angle for a given parent p_1 is shown in Lines $3 - 18$. The angles are calculated according to Equation (D.1), as shown in Line 6.

Algorithm 6: Pseudo-code of AGSX

Input : $WaitingSet[i_1, i_2, ..., i_m]$ consists of m individuals on which will perform crossover. T is the target semantics point.

Output: The generated offspring

```
while WaitingSet is not empty do
    p1 = is the first individual in WaitingSet;
    remove p1 = from WaitingSet;
    maxangle = 0; /* i.e. the maximum angle that has been found */
    top is an empty list;
    for each individual p in WaitingSet do
        calculate the angle between the relative semantics of S(p1),
        S(p) to T according to Equation (D.1);
        if angle is equal to 180, i.e. p is the optimal mate for p1 then
            top=p;
        else
            if angle is larger than maxangle then
                maxangle = angle;
                top=p;
            else
                if angle is equal to the maxangle then
                    add p to top;
                end
            end
        end
    end
    randomly select an individual, p2, from top;
    perform geometric crossover on p1 and p2;
    remove p1 and p2 from WaitingSet.
end
```

D.2.3 Main Characteristics of AGSX

Compared with GSX, AGSX has three major advantages. Firstly, AGSX employs an angle-aware scheme, which is flexible and independent of the

crossover process itself and can be applied to any form of the geometric semantic operator. Secondly, AGSX operates on distinct individuals in the semantic space. This way, the generated offspring are less likely to be identical with their parents in the semantic space. That is, AGSX can reduce semantic duplicates. Thirdly, by operating on parents with large angles between their relative semantics to the target output, AGSX is more likely to generate offspring that are closer to the target output.

D.2.4 Fitness Function of the algorithm

The Minkowski metric $L_k(X,Y) = \sqrt[k]{\sum_{i=1}^{n} |x_i - y_i|^k}$, which calculates the distance between two points, is used to evaluate the performance of individuals. Typically, two kinds of Minkowski distance between the individual and the target could be used. They are Manhattan distance (L_1 by setting $k = 1$ in $L_k(X,Y)$) and Euclidean distance (L_2). According to previous research [10], Euclidean distance is a good choice and is used in this work. The definition is as follows:

$$D(X,T) = \sqrt{\sum_{i=1}^{n} |x_i - t_i|^2} \tag{D.2}$$

where X is the semantics of the individual and T is the target semantics.

D.3 Experiments Setup

To investigate the effect of AGSX in improving the performance of GP, a GP method implements the angle-awareness into one recent approximate geometric crossover, the locally geometric semantic crossover has been proposed and named GPALGX. A comparison between GPALGX and GP with locally geometric semantic crossover (GPLGX) has been conducted. We have a brief introduction of LGX in previous section. For more details of the GPLGX, readers are referred to [128]. Standard GP is used as

Table D.1: Target Functions and Sampling Strategies.

Benchmark	Target Function	Training	Test
Keijzer1	$0.3xsin(2\pi x)$	20 ponits	1000 ponits
Koza2	$(x^5 - 2x^3 + X)$	x=mesh((-1:0.1:1])	x=Rnd[-1.1,1.1]
Nonic	$\sum_{i=1}^{9} x^i$		
R1	$(x+1)^3/(x^2 - x + 1)$	20 ponits	1000 ponits
R2	$(x^5 - 3x^3 + 1)/(x^2 + 1)$	x=mesh((-2:0.2:2])	x=Rnd[-2.2,2.2]
Mod_quartic	$4x^4 + 3x^3 + 2x^2 + x$		

a baseline for comparison as well. All the compared methods are implemented under the GP framework provided by Distributed Evolutionary Algorithms in Python (DEAP)[82].

D.3.1 Benchmark Problems

Six commonly used symbolic regression problems are used to examine the performance of the three GP methods. The details of the target functions and the sampling strategy of the training data and the test data are shown in Table D.1. The first two problems are the recommended benchmarks in [147]. The middle three are used in [177]. The last one is from [36] which is a modified version of the commonly used Quartic function. These six datasets are used since they have been widely used in recent research on geometric semantic GP [177, 213]. The notation *rnd[a,b]* denotes that 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. Since we are more interested in the generalisation ability of the proposed crossover operator, the test points are drawn from ranges which are slightly wider than that of the training points.

D.3.2 Parameter Settings

The parameter settings can be found in Table D.2. For standard GP, the fitness function is different from that of GPLGX and GPALGX. Since the primary interest of this work is the comparison of the generalisation abil-

Table D.2: Parameter Settings

Parameter	Values	Parameter	Values
Population Size	512	Generations	100
Crossover Rate	0.9	Reproduction Rate	0.1
#Elitism	10	Maximum Tree Depth	17
Initialisation	Ramped-Half&Half	Initial Depth	range(2,6)
Maximum tree depth in Library-M	3	Neighbourhood Number-K	8
Function Set	$+, -, *$, protected $\%, \log, \sin, \cos, \exp$		
Fitness function	Root Mean Squared Error (RMSE) in standard GP Euclidean distance in GPLGX and GPALGX		

ity of the various crossover operators, all the three GP methods only have crossover operators. No mutation operator has taken apart. The values of the two key parameters M and K in implementing LGX, which represent for the maximum depth of the small size tree in the library and the number of the closest neighbouring trees respectively, are following the recommendation in [128].

Overall, the three GP methods are examined on six benchmarks. Each method has 100 independent runs performed on each benchmark problem.

D.4 Results and Discussions

The experiment results of GP, GPLGX and GPALGX are presented and discussed in this section. The results will be presented in terms of comparisons of RMSEs of the 100 best models on the training sets and their corresponding test RMSEs. The fitness values of models in GPLGX and GPALGX are calculated using Euclidean distance. However, for comparison purpose, the Root Mean Squared Error (RMSE) of models are also recorded. The major comparison is presented between GPLGX and GPALGX. Thus, we also compare the angle distribution of GPLGX and GPALGX. The computational time and program size are also discussed. The non-parametric Wilcoxon test is used to evaluate the statistical significance of the difference on the RMSEs on both the training sets and the test

sets. The significance level is set to be 0.05.

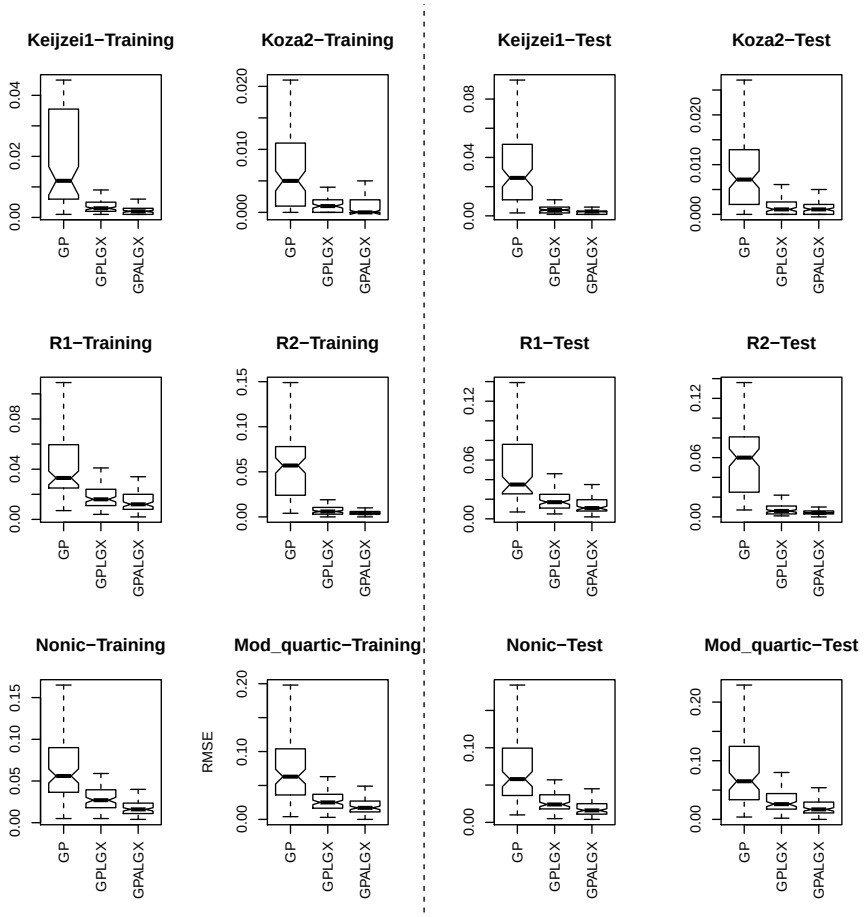


Figure D.2: Distribution of Training RMSE and the Corresponding Test RMSE of the 100 best-of-run individuals.

D.4.1 Overall Results

The results on the six benchmarks are shown in Fig.D.2, which displays the distribution of RMSEs of the 100 best-of-the-run individuals on the training sets and the test sets. As it shows, on all the six benchmarks, GPALGX has the best training performance among the three GP methods. For every benchmark, GPALGX has a better training performance than

GPLGX and GP, by the smaller median value of the 100 best training errors and the much shorter boxplot. This indicates the training performance of GPALGX is superior to the other two methods in a notable and stable way. The results of statistical significance test confirm that the advantage of GPALGX over GPLGX and GP are all significant on the six training sets.

The overall pattern on the test sets is the same as the training set, which is GPALGX achieves the best generalisation performance on all the benchmarks. On each benchmark, the pattern in the distribution of the 100 test errors is also the same as that on the training set. GPALGX has the shortest boxplot which indicates the more consist generalisation error among the 100 runs. GPLGX has a larger distribution than GPALGX, which is still much shorter than standard GP. A significant difference can be found on the six benchmarks between GPALGX, GPLGX and GP, i.e. GPALGX generalises significantly better than GPLGX, while the two geometric methods are significantly superior to GP. The generalisation advance of LGX and ALGX over standard crossover is consistent with the previous research on LGX. In [128], the generalisation gain of LGX has been investigated and confirmed. This generalisation gain has been justified to own to the library generating process which helps reduce the semantic duplicates. The further generalisation gain of ALGX over LGX might lie in the fact that the angle-awareness helps extend the segment connecting each pair of parents for crossover, thus can reduce the semantic duplicates more intensively, and enhance the exploration ability of LGX to find better generalised solutions.

D.4.2 Analysis on the Learning Performance

The evolutionary plots on the training sets are provided in Fig.D.3. To analysis the effect of ALGX on improving the learning performance of GP. These evolutionary plots are drawn using the mean RMSEs of the best-of-generation individuals over the 100 runs.

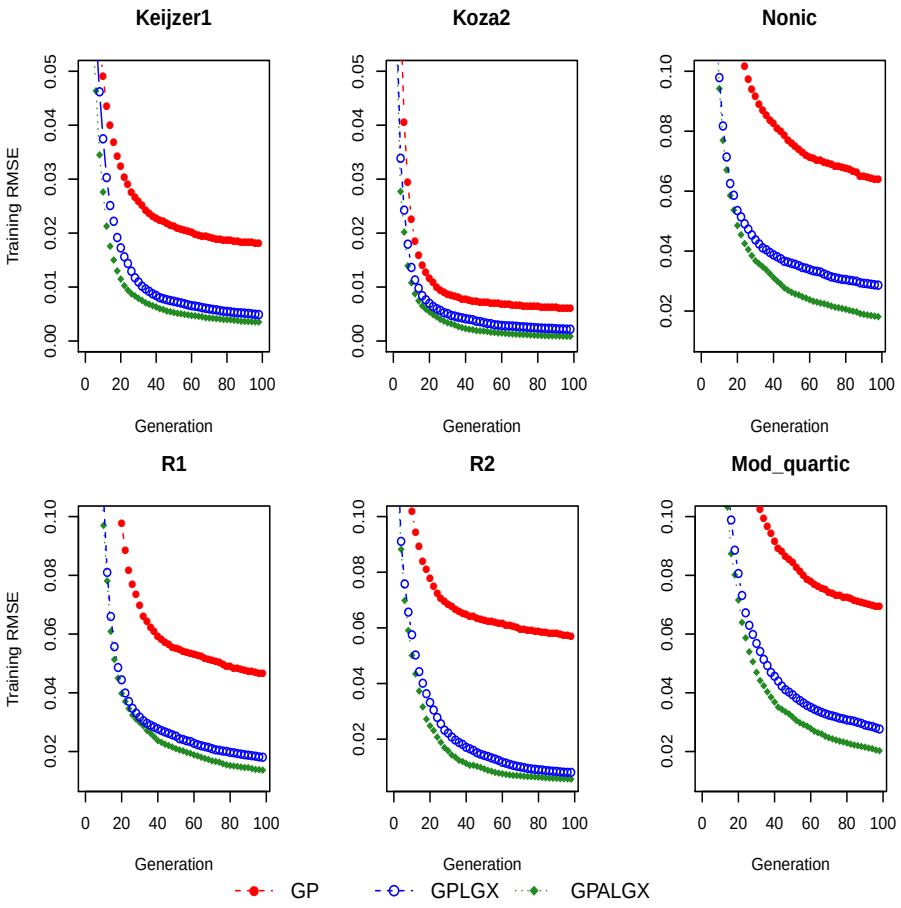


Figure D.3: Evolutionary plot on the training set.

As expected, GP with ALGX achieves the best learning performance. It is superior to the other two GP methods from the early stage of the evolutionary process, which is generally within the first ten generations. The advances of the two geometric GP methods over standard GP on the learning performance confirms that searching in the geometric space is generally much easier, since the semantic space is unimodal and has no local optimal. The comparison between the two geometric GP methods indicates ALGX is able to generate offspring which is much closer to the target point in the semantic space from the very beginning of the searching process. On all the six benchmarks, GPALGX not only has signifi-

cantly smaller training RMSEs but also has higher average fitness gain from generation to generation. On $Koza_2$ and R_2, the two geometric GP methods can find models which are excellent approximations (the RMSE of which is smaller than 0.001), and GPALGX converges to the target semantics much faster than GPLGX. This might be because ALGX performs crossover on individuals having larger angles than GPLX, thus produces offspring closer to the target in the semantic space in an effective way. In this way, it will increase the exploitation ability of LGX and find the target more quickly. For the other four benchmarks, although none of the two geometric GP methods finds the optimal solution, on three of them, the increasingly larger difference between the two methods along with the increase of generations indicates the improvement that ALGX brings is increasing over generations. One of the possible reasons is that, over generations, compared with LGX, ALGX will perform on individuals having smaller relative semantic distance with target output in larger angle pairs, which will generate even better offspring.

D.4.3 Analysis of the Evolution of Generalisation Performance

Compared with the training performance, we are more interested in the generalisation performance of GP with ALGX. Therefore, further analysis on the generalisation ability of GPALGX and a more comprehensive comparison between the generalisation of the three methods is carried out. In Fig.D.4, the evolutionary plots on the test sets are reported along generations for each benchmark on the three GP methods. These plots are based on the mean RMSEs of the corresponding test errors obtained by the best-of-generation models over 100 runs. (On each generation, the test performance of the best-of-generation model obtained from the training data has been recorded, but the test sets never take apart in the evolutionary training process)

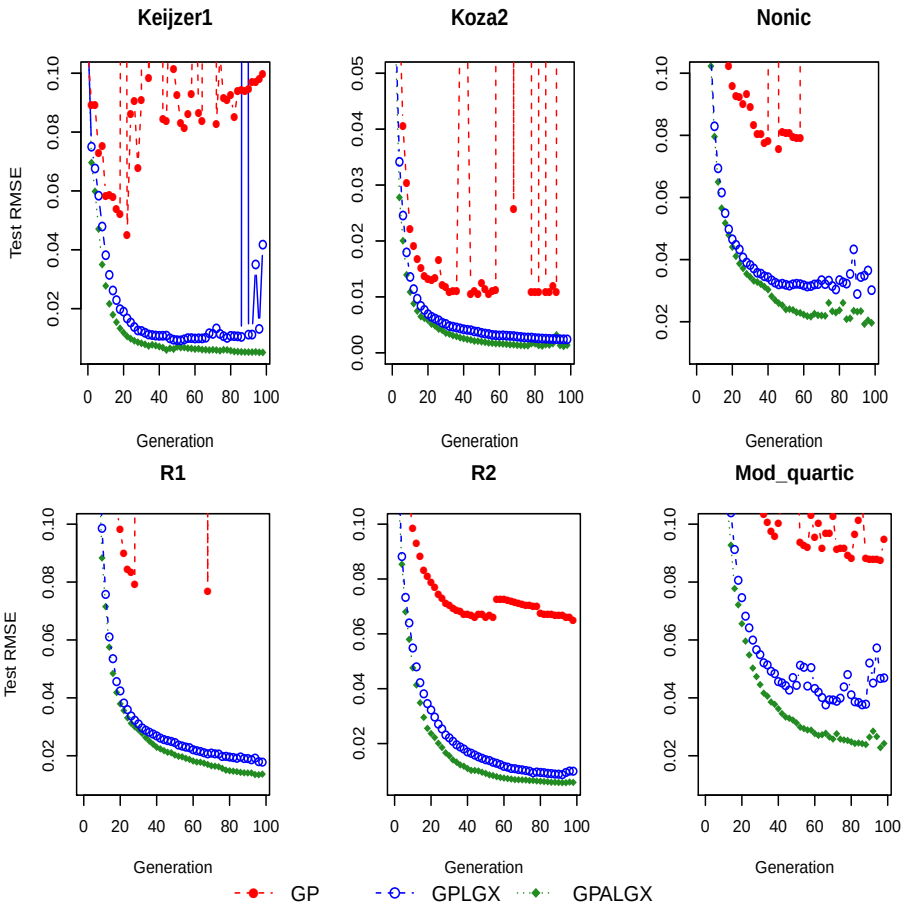


Figure D.4: Evolutionary plot on the test set.

The evolution plots confirm that GPALGX has a better generalisation gain than the other two methods on all the test sets of the considered benchmarks, which is notable. On all the six benchmarks, GPALGX can generalise quite well, while its two counterparts suffer from the overfitting problems on some datasets. On the six problems, GP overfits the training sets. The test RMSEs increase after decreasing over a small number of generations at the beginning. Also, GP generally has a very fluctuate mean RMSE on most test sets. It indicates that training the models on a small number of points (20 points), while testing the models on a larger number of points (1000 points) distributed over a slightly larger region is difficult

for GP. GPLGX can generalise much better than GP but still encounters overfitting problems on three benchmarks, i.e., on Keijzer1, Nonic and Mod_quartic. On these three datasets, GPLGX has an increasing RMSEs on the last ten generations. On other three datasets, GPLGX generalises well. Overall, GPALGX generalises better than GPLGX and GP, shown as obtaining lower generalisation errors and having a smaller difference with its training errors.

The excellent generalisation ability of geometric crossover can be explained by the fact that the geometric properties of this operator are independent of the data to which the individual is exposed. Specifically, the offspring produced by LGX and ALGX lie (approximately) in the segments of parents also hold in the semantic space of the test data. Since this property holds for every set of data, no matter where the test data distributes in, the fitness of the offspring can never be worse than the worse parent. In the population level, this property can not guarantee to improve the test error on every generation for every benchmark (in fact, we can find on the last several generations, LGX has an increasing test error on three benchmarks), but during the process it surely has a high probability of generalisation gain on the test set and only a few times of getting worse generalisation over generations. That is why LGX has the ability to control overfitting and generalise better than the regular crossover.

This interpretation has a direct relationship on why ALGX is less likely to overfitting and generalises better than LGX on the test sets. In other words, ALGX puts more effect on selecting parents which consequently limits the probability of having not good enough parents to crossover, so it can lead to a large number of offspring with better generalisation at the population level. AlGX shares the same benefit with LGX, which is the geometric property leading to offspring never worse than parents on the test set. More importantly, the angle-awareness in ALGX makes the large angle between the parents also holds in the test semantic space. This leads to a higher probability to have a good process on the test data at the pop-

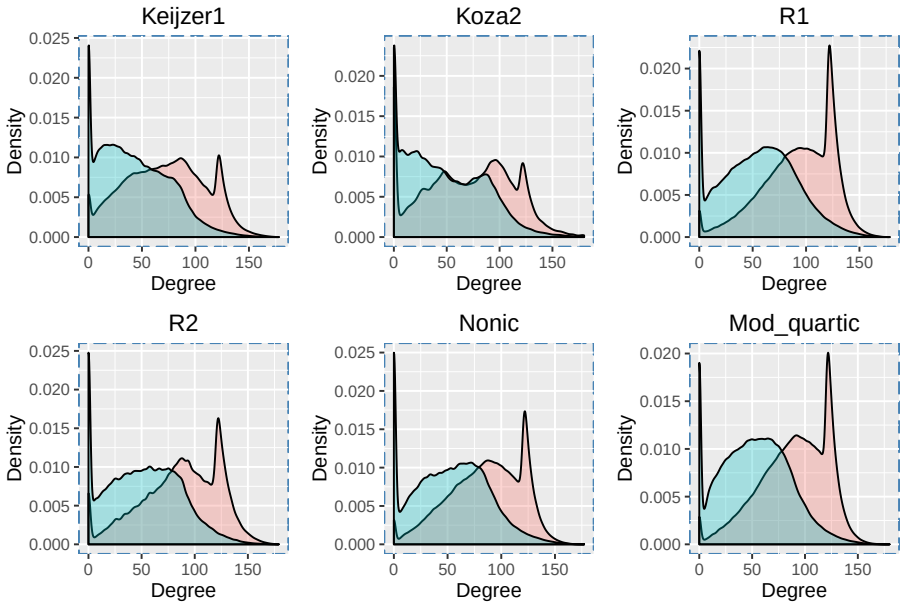


Figure D.5: Distribution of Angles of the Parents for Crossover.

ulation level. The details of the angle distribution will be discussed in the next subsection.

D.4.4 Analysis of the Angles

To investigate and confirm the influence of ALGX to the distribution of angles of the parents, the angles between each pair of parents which performs crossover have been recorded in both GPLGX and GPALGX. In Fig.D.5, the density plots show the distribution of the angles in the two GP methods. The green one is for GPLGX, and the one in orange colour is for GPALGX. The density plots are based on around $2,250,000 (\approx 225 * 100 * 100)$ values of angles in each method. While the x-axis represents the degree of angles, the y-axis is the percentage of the corresponding degree in the $2,250,000$ recorded values.

From Fig.D.5 we can see that the distribution of angles of parents in GPALGX is different from GPLGX in two aspects. On the one hand, it has a much smaller number of angles which are zero degrees. While in GPLGX,

the peak of the distribution is at the zero degrees on all the six datasets, in GPALGX, the angle-awareness can stop the pairs of individuals with zero degrees from performing crossover. The direct consequence of this trend is the elimination of semantic duplicates, and the higher possibility of generating better offspring.

On the other hand, GPALGX has a larger number of larger angles. Most of its angles are over 90 degrees. The peak of the distribution is all around 120 degrees on the six datasets, specifically on the last four datasets. At the first several generations, the larger angles with similar (or the same) vectors will lead to better offspring, which is represented by a shorter vector. At the last several generations, larger angles along with the shorter vectors will lead to a population of even better offspring. This can explain why the distance between the training error and test error of GPLGX and GPALGX increases over generations on most of the benchmarks.

Table D.3: Computational Time and Program Size.

Benchmarks	Method	Time(in second)	Program size (Node)	Significant Test
		Mean±Std	Mean±Std	(on program size)
Keijzer1	GPLGX	523±83.8	90.52±28.72	=
	GPALGX	1400±317	87.74±23.85	
Koza2	GPLGX	560±105	72.66±29.93	+
	GPALGX	1330±232	93.82±24.18	
R1	GPLGX	523±84.5	88.82±27.07	=
	GPALGX	1250±253	92.18±31.71	
R2	GPLGX	524±83.9	89.62±27.41	=
	GPALGX	1250±218	83.8±28.6	
Nonic	GPLGX	571±112	84.5±32.62	+
	GPALGX	1250±212	101.5±39.33	
Mod_quartic	GPLGX	554±105	99.98±38.25	=
	GPALGX	1420±369	105.38±37.29	

D.4.5 Comparison on Computational Time and Program Size

The comparison between the computational cost and program size of the evolved models have been performed between the two geometric methods. Table D.3 shows the computational time in terms of the average training time for one GP run in each benchmark. The average program size represented by the number of nodes in the best_of_run models in each benchmark is also provided. The statistical significance results on the program size are also listed in the table. While "-" means the program size of the evolved model in GPALGX is significantly smaller than GPLGX, "+" indicates the significant larger program size of GPALGX. "=" represents no significant difference can be found.

As shown in Table D.3, on all the six benchmarks, the average computational time for one run in GPALGX is much higher than GPLGX, which is generally around two times as that of GPLGX. This is not surprising since GPALGX needs more effort to identify the most suitable pairs of parents during the crossover process. The longer computational time can be decreased by reducing the population size in GPALGX. Moreover, the computational time for each GP run in both methods is short, which is hundreds to two thousand second. Thus, the additional computational cost of GPALGX is affordable.

In term of the program size, on four benchmarks, i.e., Keijzer1, R1, R2, and Mod_quartic, the two methods have a similar program size and no significant difference has been found. On the other two datasets, ALGX produces offspring which are significantly larger than LGX. However, it is interesting to note that these much more complex models in term of program size still can generalise better than its simpler counterparts on the two test sets, while the simpler model of GPLGX slightly overfits on the Nonic problem.

D.5 Conclusions and Future work

This work proposes an angle-aware mating scheme to select parents for geometric semantic crossover, which employs the angle between the relative semantics of the parents to the target output to choose parents. The proposed ALGX performs on parents having a large angle so that the segment connecting the parents is close to the target output. Thus, ALGX can generate offspring that have better performance. To investigate and confirm the efficiency of the proposed ALGX, we run GP employed ALGX on six widely used symbolic regression benchmark problems and compare its performance with GPLGX and GP. The experimental results confirm that GPALGX has not only better training performance but also significantly better generalisation ability than GPLGX and GP on all the examined benchmarks.

Despite the improvement ALGX brings on performance, it generally is computational more expensive than GPLGX. In the future, we aim to improve the angle detecting process. Instead of using the deterministic method to calculate the angle between two individuals iteratively, we can introduce some heuristic search methods to find the best parent pairs to reduce the computational cost. We also would like to explore a further application of ALGX, for example, to introduce the angle-awareness to other forms of geometric crossover, such as the exact geometric semantic crossover [156] and Approximate geometric crossover [126], to investigate their effectiveness. In addition, this work involves solely crossover and no mutation. The effect of angle-awareness to mutation and using both crossover and mutation are also interesting topics to work on.