Data fitting based on improved genetic programming

Pinchao Meng[1], Weishi Yin[1], Yanzhong Li[2*]

[1] *Department of Applied Mathematics, Changchun University of Science and Technology ,Changchun, China*

[2] *College of mathematics and statistics, Beihua University, Jilin, China*

Received 1 November 2014 , www.cmnt.lv

Abstract

Traditional data fitting techniques usually require estimating basis function and they are specific for different application areas. Based on dynamic characteristics of genetic programming, a two-phase data fitting algorithm is proposed. In this algorithm, genetic programming is used to optimize model structure and Least Square method is applied to estimate parameters. Proposed algorithm is tested for different types of data fitting. Not only can this algorithm be applied in different areas, but also it is of high efficiency and accuracy.

Keywords: Genetic Programming, Least Square Method, Data Fitting.

1 Introduction

Data fitting techniques are often used to examine hidden relationship between large volumes of dataset from experimental analysis. Traditional data fitting technique, e.g., Least Square method, requires theoretical derivation or expert experience to examine the relationship between variables and determine the type of fitting function (linear, logarithmic, polynomial etc.), which is followed by parameter estimation. In practice, it is difficult to correctly determine the model structure, especially when no apparent relationship is available for a large data set. Hence, traditional data fitting techniques have some limitations.The advantage of genetic programming is that, without a specific form of function, one can not only obtain the expression of fitting function, but also prevent local optimization in condition of large initial population size and well defined crossover and mutation rate.

American researcher Koze developed genetic programming (GP) based on genetic algorithm in the beginning of 1990s[1]. In 1992, he published a book entitled "Genetic Programming: on the Programming of Computer by Means of Natural Selection", which introduced the principle and application samples of genetic programming. In 1994, his second book entitled "Genetic Programming II: Automatic Discovery of Reusable Programs" was published, where he provided a way to automatically define functions, and also introduced a new concept -- subroutine. In 1999, he and Forest published another book entitled "Genetic Programming III: Darwinian Invention and Problem Solving". They proposed the concept of architecture-altering operations that controlled subroutines, iteration, recursion and storage. This book mainly introduced application of genetic programming in automated synthesis of analog electrical circuits. From 1990s, genetic programming developed constantly in both scope and depth.

The problem solving process of genetic programming is a self-adaptive nonlinear searching process based on fitness, and its principle is to describe a problem using generalized hierarchical computer programs. These programs can alter the architecture size according to environment and show great representativeness in engineering. It is applied in lots fields with great success, such as auto design, nonlinear function approximation, prediction and modeling, pattern recognition, robotics, structure and coefficient optimization of neural network, machine learning, symbolic expression, music or image generation, and so on. To solve a problem using genetic programming is to search for the programs with best fitness in solution space consists of many feasible programs.

2 Basic principle of Genetic Programming

In genetic programming, adaptive iterations based on a series of operations, including selection, crossover and mutation, on a randomly generated feasible population are progressed to approach the optimal solution.

One feature of genetic programming is that a problem can be described by alterable hierarchical architectures, thus in practical, an individual can be represented by a tree and a population is thus a set of binary trees. Once all the individuals are represented by corresponding trees, reproduction, crossover and mutation operations in

* Corresponding author's Email:lyz@cust.edu.cn

genetic programing can be implemented by copy, delete and query operation on the corresponding tree structure.

2.1 INDIVIDUAL REPRESENTATION

A set of N_f functions can be represented as:

$F = \{f_1, f_2, \cdots, f_{N_f}\}$,

and a set of N_t child nodes as:

$T = \{a_1, a_2, \cdots, a_{N_t}\}$.

Initial population consists of many initial individuals, each of which is a randomly generated symbolic expression and is a possible solution of the problem. Firstly, a function in function set F is selected randomly according to the uniform distribution as the root of algorithm tree. Usually, a root is confined in the function set F to get a hierarchical complicated architecture; otherwise, a degenerative architecture composed of only one terminal will be generated if a root is selected from terminal set T. In general, once a selected function f has a number of Z independent variable, a number of Z lines will be initiated from this node. Then, for each of these Z lines, an element is selected randomly from C, the union of T and F, as the terminal node of this line. If the selected element is a function, repeats the above step; if the selected one from C is a terminal, the tree from this branch node stops growing and takes the terminal as terminal node. Repeats all the steps again and again, from top to down and from left to right, until a complete tree is generated [3,4].

2.2 GENERAYION OF INITIAL POPULATION

An initial population is randomly generated and is comprised of multiple individuals. There are two approaches to generate a random tree.

The full method

For initial individuals generated from the full method, each leave’s depth is equal to pre-determined tree depth. To achieve this, if a node’s depth is less than pre-determined tree depth, then the selection of the node is within the set of functions. If a node’s depth is equal to pre-determined tree depth, then the selection of the node is within the child nodes.

Grow method

Different from the full method, each leave’s depth is less than or equal to pre-determined tree depth for individuals generated with grow method [5]. To achieve this, if a node’s depth is less than pre-determined tree depth, then selection of this node is within the union of the set of functions and child nodes. Alternatively, if a node’s depth is equal to pre-determined tree depth, then the selected node is from the set of child nodes.

Combination of the two methods

In a initial population generated by either full or grow methods, the individuals are similar in the aspect of architecture, for example tree depth. To increase population diversity, a combination of full and grow methods called “ramped half-and-half” can be adopted. First, depth of algorithm tree for each initial individual is determined by a number randomly selected from 2 to maximum depth. The proportion of tree with different depth is:

$$h = \frac{1}{D-1} \times 100\%,$$

where D is the maximum depth. And then, for each depth, half the initial population is constructed using grow and half is constructed using full.

2.3 EVALUAYION OF FITNESS FUNCTION

In GP, fitness is often measured with a group of calculation sample. The distance between calculated value of a symbolic for the testing data and measured value is quantified as the fitness value of the symbolic. For a population, fitness of an individual i for a generation t can be represented as g(i) and calculated in Eqn. (1):

$$g(i) = \sum_{j=1}^{N_C} \left| S(i, j) - C(j) \right| \quad , \tag{1}$$

where $S(i, j)$ denotes the fitness value of individual in term of calculation sample j; N_C is the number of calculation samples and $C(j)$ is the observed value or true value of calculation sample j.

Fitness measure is an important factor that affects efficiency of running genetic programming system. It costs huge computational resources and alters according to problems. So to solve a problem, it is essential to select proper measurement function and sample sets.

2.4 GENETIC OPERATIONS

There are three main operations in GP: selection, crossover and mutation.

2.4.1 Selection operations

Selection is usually based on fitness function [6]. A frequently-used approach is roulette wheel selection, which selects individuals proportional to fitness function. All of other selection methods used in genetic algorithm can be applied in GP.

2.4.2 Crossover operations

In tree crossover [7], random nodes are chosen from both parent trees, and the respective branches are swapped creating two offspring. There is no bias towards choosing internal or terminal nodes as the crossing sites.

2.4.3 Mutation operations

In tree mutation, a random node is chosen from the parent tree and substituted by a new random tree created with the terminals and functions available [8]. This new random tree is created with the Grow initialization method and obeys the size/depth restrictions imposed on the trees created for the initial generation.

2.5 STEP OF ALGORITHM

Step 1*:* Determine control parameters including function set, the set of child nodes, fitness function, population size, the number of iterations, the probability of reproduce and crossover.
Step 2*:* Randomly generate initial population.
Step 3*:* Evaluate each individual's fitness. If stop criteria satisfies, terminate the algorithm. Otherwise, go to Step 4.
Step 4*:* Conduct genetic operations to generate next population:
a) Reproduce individuals of high fitness according to reproduce probability;
b) Conduct crossover operations based on crossover probability;
c) Supplementary operators, e.g., mutation and inversion can be applied depending on specific problems.
Return to ***Step 3.***

Stop criteria could be pre-determined satisfied fitness or maximal given iterations.

3 Genetic Programming-Least Square method (GPLS)

Genetic Programming – Least Square method (GPLS) is proposed to fit data when the dataset is complex and structure of the fitting function is not clear. GP is firstly used to find the structure of the fitting function and Least Square method is applied for parameter optimization.

3.1 STRUCTURE OPTIMIZATION

3.1.1 Tree initialization

The Ramped Half-and-Half method. In the standard procedure, an equal number of individuals are initialized for each depth between 2 and the initial tree depth value. For each depth level considered, half of the individuals are initialized using the Full method, and the other half using the Grow method. The population of trees resulting from this initialization method is very diverse, with balanced and unbalanced trees of several different depths. Node set is comprised of variables and random numbers.

3.1.2 Validating new individuals

After a new individual is produced by any of the genetic operators, it must be validated in terms of depth/size before being considered as a candidate for the new population. We set a filter function. This filter function measures the fitness of an individual that is deeper than the dynamic maximum allowed depth: if the individual is better than the best so far, the dynamic depth is increased and the new individual is accepted; otherwise it is rejected. The filter does nothing if the individual is not deeper than the limit.

3.1.3 Operator probabilities

In runtime, we present an automatic adaptation procedure for the genetic operator probabilities of occurrence. The performance for each genetic operator is calculated by summing the credits of all individuals created by that operator, and dividing the sum by the number of individuals created by that operator. Each operator probability value is then adapted to reflect its performance [9]. A percentage of the probability value is replaced by a value proportional to the operator's performance. Operators that have been performing well see their probability values increased; operators that have been producing individuals worse than the population from which they were born see their probability values decreased. Operators that haven't been able to produce any children since the last adaptation will receive a substantial increase of probability, as if their performance was twice as good as the performance of the best operator.

3.2 PARAMETER OPTIMIZATION

GP aims to search the function space of structured problems. Unfortunately, accuracy of available GP algorithms cannot meet the requirements. With known structure, Least Square method can be used to optimize parameters to find the best function that fits the data, by minimizing square error.

The solution from the algorithm described above is represented as $P(w;x)$, where $w=(w_1,w_2,\cdots,w_J)$ is a set of parameters to be optimized. An illustrated solution for GP is shown in the tree-like structure in Figure 1.

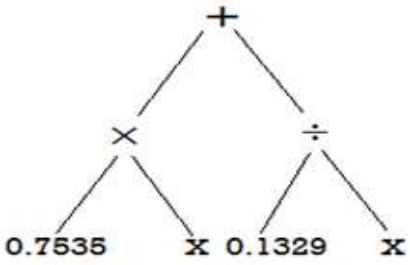

FIGURE 1 The tree-like structure

$P(w;x) = P(w_1, w_2; x) = w_1 x + w_2 / x$, $w_1 = 0.7535$, $w_2 = 0.1329$.

The original optimization problem can be converted to the following:

$$\min_w \sum_{j=1}^{N_C} (P(w;x_j) - C(j))^2 . \quad (2)$$

Least Square method can be used to find the optimal solution.

3.3 STEPS OF ALGORITHM

Step1*:* Determine control parameters including function set, the set of child nodes, fitness function, population size, the number of iterations, the probability of reproduce and crossover.

Step2*:* Randomly generate initial population.

Step3*:* Evaluate each individual's fitness. If stop criteria satisfies, terminate the algorithm. Otherwise, go to ***Step 4.***

Step 4*:* Conduct genetic operations to generate next population:

a) Reproduce individuals of high fitness according to reproduce probability;

b) Conduct crossover operations based on crossover probability;

c) Supplementary operators, e.g., mutation and inversion can be applied depending on specific problems. Return to Step 3.

Stop criteria could be maximal given iterations.

Step5*:* Chose better results given by above algorithm and optimize arguments.

TABLE 1 Measuring data 1

x_i	-0.4	-0.3	-0.2	-0.1	0	0.1	0.2	0.3	0.4	0.5
y_{1i}	0.76	0.79	0.85	0.91	0.99	1.11	1.24	1.39	1.56	1.75
y_{2i}	1.55	1.39	1.25	1.11	1	0.91	0.84	0.79	0.76	0.75
y_{3i}	-0.24	-0.21	-0.15	-0.09	0	0.11	0.24	0.39	0.57	0.74

4.1.1 Results Comparison

Least Square method is applied to fitting a second order polynomial and 3rd degree polynomial for the three datasets listed in Table 1. Comparison among the three algorithms is listed in Table 2.

TABLE 2 Results 1

	Algorithm	Error
1	Second Order Polynomial	0.00433
	Third Order Polynomial	0.0043
	GP	0.1314
	GPLS	0.00434
2	Second Order Polynomial	0.0040
	Third Order Polynomial	0.0030
	GP	0.0142
	GPLS	0.0045
3	Second Order Polynomial	0.0051
	Third Order Polynomial	0.0050
	GP	0.0055
	GPLS	0.0054

4 Numerical experiments

To test the efficiency of the proposed algorithm, matlab is applied to realizing the algorithm and the comparison to GP and Least Square method. The parameters are set as follows: population size N= 50; maximum iteration max_gen=17; function set $F = \{+, -, \times, \div, \sin, \cos, \exp, \log, \mathrm{sqrt}\}$, node set $T = \{x, rand\}$.

4.1 EXPERIMENT 1

Three algorithms, including GP, Least Square method and GPLS, are tested for three datasets with strong patterns, as shown in Table 1.

Average error, calcuated in Eqn. (3), is used to evaluate different genetic programmings.

$$e = (\sum_{j=1}^{N_C} \left| f(x_j) - y_j \right|^2 / N_C)^{1/2} \quad (3)$$

Results analysis

As shown in Figures 2-4, convergence of GP is the fastest. Although it usually converges in several iterations and finds the optimal structure, its accuracy is relatively low. Genetic operators in GP evolution are shown in Figures 5-7. Figures 8-10 present population diversity in the process of GP evolution. Overall, it is found that individual difference is smaller with GP evolution. The changes in accuracy and complexity during GP evolution are shown in Figures 11-13. It can be seen that accuracy is improved at the cost of complexity increasing.

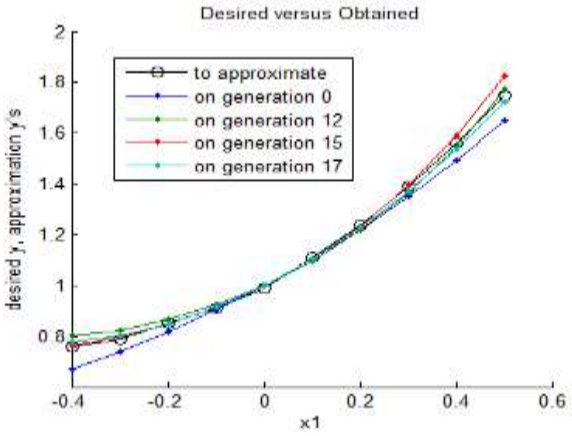

FIGURE 2 Desired versus obtained of the first dataset

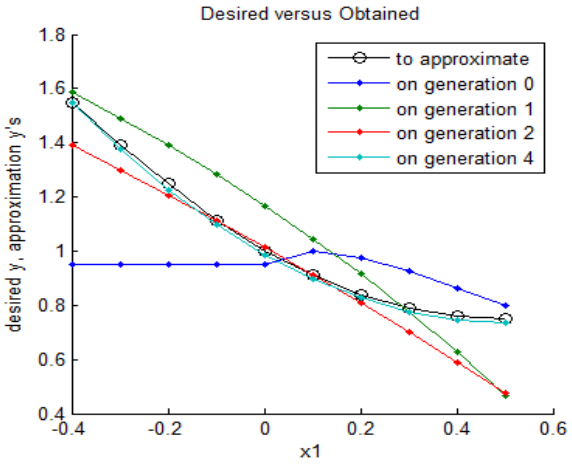

FIGURE 3 Desired versus obtained of the second dataset

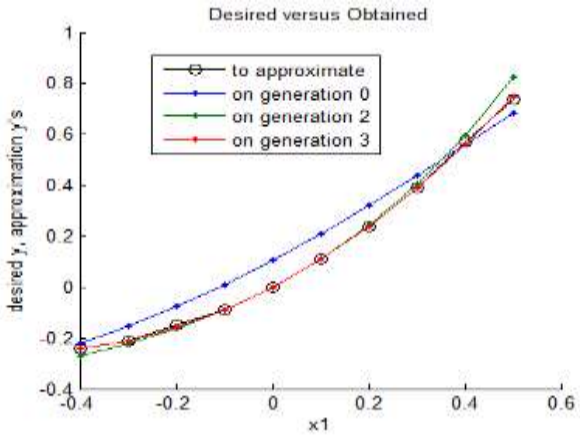

FIGURE 4 Desired versus obtained of the third dataset

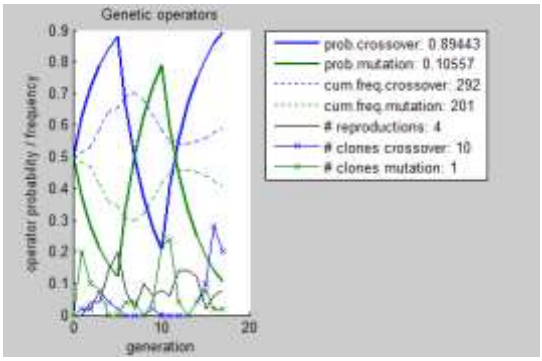

FIGURE 5 Genetic operators of the first dataset

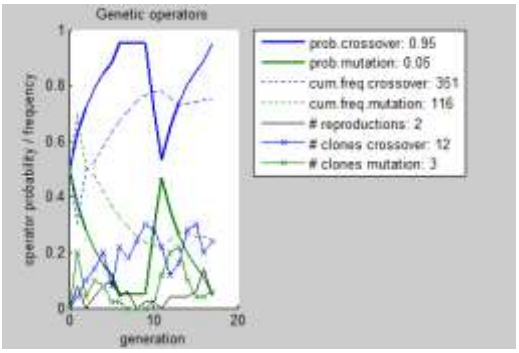

FIGURE 6 Genetic operators of the second dataset

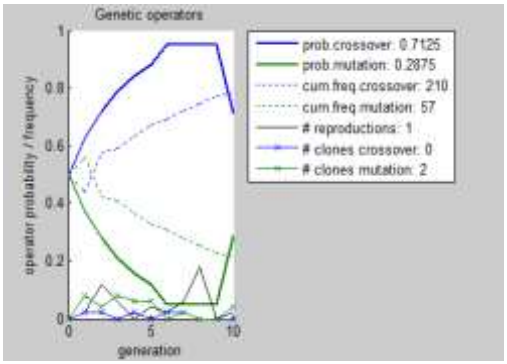

FIGURE 7 Genetic operators of the third dataset

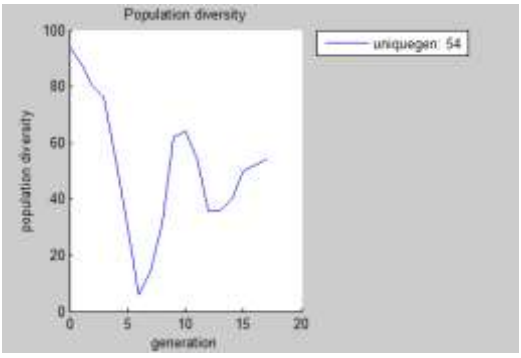

FIGURE 8 Population diversity of the first dataset

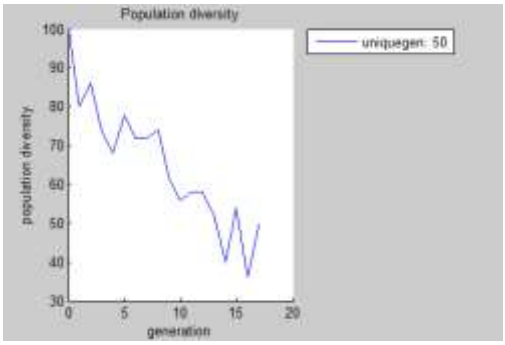

FIGURE 9 Population diversity of the second dataset

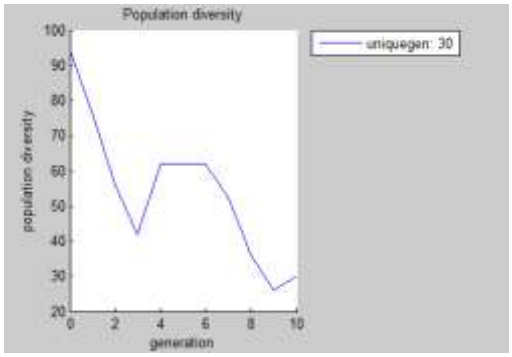

FIGURE 10 Population diversity of the third dataset

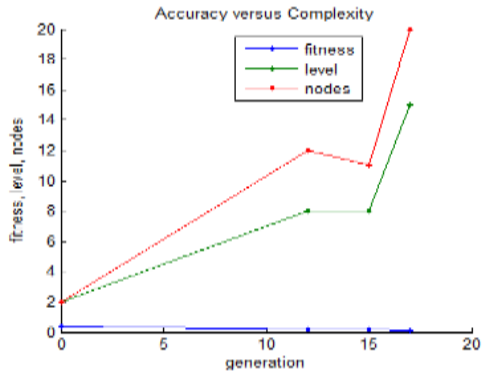

FIGURE 11 Accuracy versus complexity of the first dataset

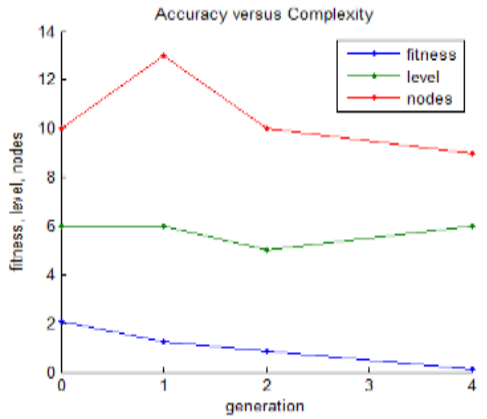

FIGURE 12 Accuracy versus complexity of the second dataset

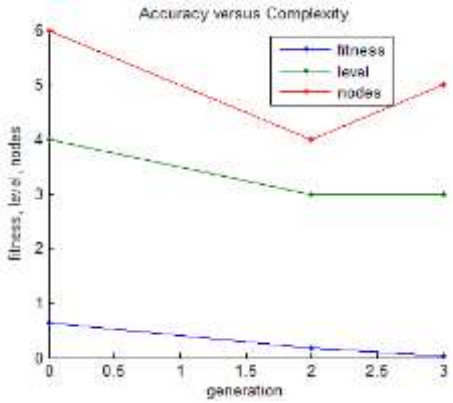

FIGURE 13 Accuracy versus complexity of the third dataset

As shown in Figure 14, the algorithm proposed in this study could be used to fit different datasets (Table 1) and finds the best fitting function. Table 2 shows that all the three methods, including Least Square method, GP and GPLS, have good results if the dataset is small and has a certain pattern. Under some circumstance, Least Square method can even lead to better results than GPLS. Compared to GP, GPLS performs better than GP since it is an improved method based on GP.

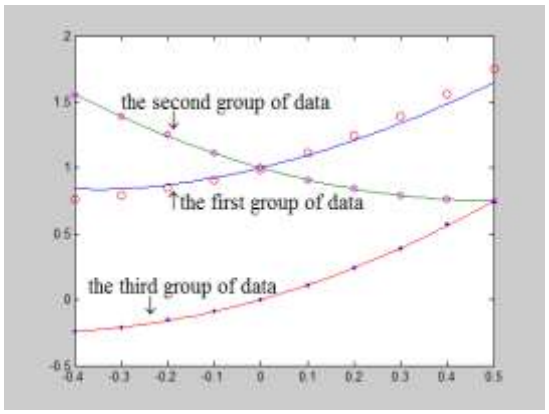

FIGURE 14 The fitting curve of three datasets

4.2 EXPERIMENT II

Two algorithms, including GPLS and Least Square method , are tested for dataset as shown in Table 3.

TABLE 3 Measuring data 2

x_i	0	0.31	0.63	0.94	1.26	1.57	1.89
y_i	0	1.71	2.49	2.07	0.95	0	-0.22
x_i	2.20	2.51	2.83	3.14	3.46	3.77	4.08
y_i	0.17	0.59	0.53	0	-0.53	-0.59	-0.17
x_i	4.40	4.71	5.03	5.34	5.65	5.97	6.28
y_i	0.22	0	-0.95	-2.07	-2.49	-1.71	0

Least Square method is applied to fitting a second order polynomial and 3rd degree polynomial for the dataset

listed in Table 3. Comparison among the three algorithms is listed in Table 4.

TABLE 4 Results 2

	Algorithm	Error
1	Second Order Polynomial	4.2064
2	Third Order Polynomial	4.2057
3	GPLS	0.2898

The changes in accuracy and complexity during GP evolution are shown in Figure 15.Genetic operators in GP evolution are shown in Figure 16. The fitting curve of three methods is shown in Figure 17.

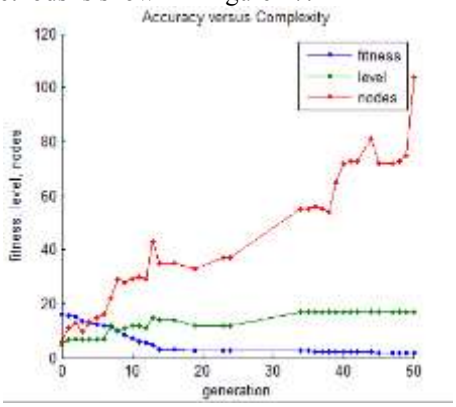

FIGURE 15 Accuracy versus complexity of the dataset

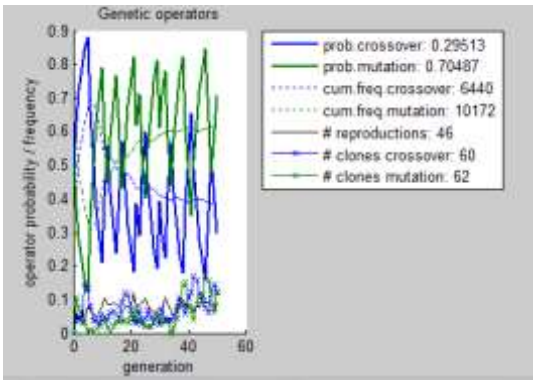

FIGURE 16 Genetic operators of the dataset

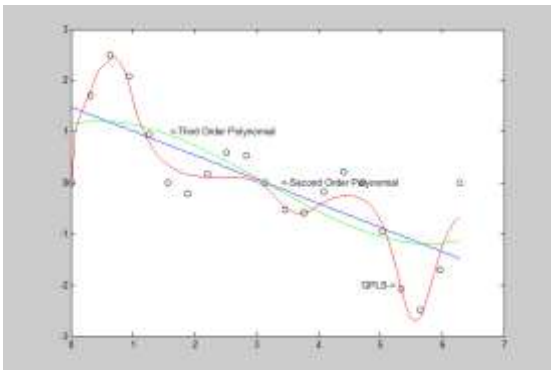

FIGURE 17 The fitting curve of three methods

Average error, calcuated in Eqn. (3), is used to evaluate genetic programming.

4.3 EXPERIMENT III

Least square method, GP and GPLS are used to fit the dataset shown in Table 5, the relationship in which cannot be easily determined.

TABLE 5 Measuring data 3

x_i	2	3	4	5	7	8	10
y_i	106.42	108.2	109.58	109.50	110	109.93	110.49
x_i	11	14	15	16	18	19	
y_i	110.59	110.6	110.90	110.76	111	111.20	

When applying Least Square method, it was used to fit both a second order polynomial and third degree polynomial. Results are shown in Table 6 and error term is calculated using Eqn. (3).

TABLE 6 Results 3

	Algorithm	Error
1	Second Order Polynomial	0.6285
2	GP	0.8177
3	GPLS	0.2736

As shown in Table 6 and Figure 18, compared to Least Square method and GP, GPLS has better performance in fitting dataset which has no easily determined pattern.

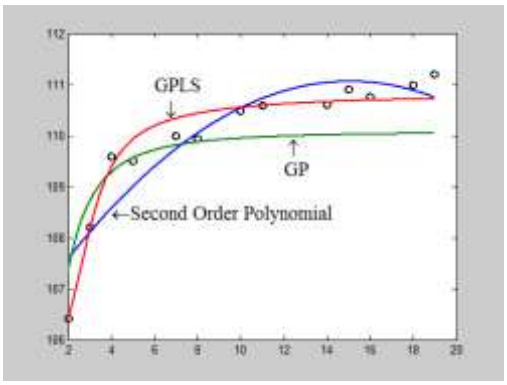

FIGURE 18 The fitting curve of three methods

5 Conclusions

Traditional data fitting methods are often required to pre-determine the structure of the fitting function based on experience, the programs designed by which can only solve specific problems in specific areas, i.e. low portability. However, GP possesses dynamic characteristics of the variability without experience prior knowledge[10], but its convergence is slow and fitted function is complicated. Hence GPLS is proposed in this study by integrating GP and Least Square method. GP is first adopted to optimize structure of the fitting function and then Least Square method is applied to optimizing parameters, thus the optimal solution can be achieved. In terms of different types of data, GPLS is compared with Least Square method in this paper. Results show that

GPLS has high accuracy and a general approach, which can be applied in different areas.

References

[1] Koza J R 1998 Genetic Programming MA:MIT Press, Cambridge

[2] Koza J R 2000 Automatic Creation of Human-Competitive Programs and Controllers by Means of Genetic Programming *Genetic Programming and Evolvable Machines* **1** 121-64

[3] Shang Xiu-qin, Lu Jian-gang, Sun You-xian, Liu Jun, Ying Yu-qian 2010 Data-Driven Prediction of Sintering Burn-Through Point Based on Novel Genetic Programming *Journal of Iron and Steel Research* **17**(12) 01-05

[4] Wu Y L, Lu J G, Sun Y X. A Segregated Genetic Programming for Bioprocess Modelling With Outliers Asia-Pacific *Journal of Chemical Engineering* **3**(6) 606

[5] Bolton C C, Gatica G, Parada V 2013 Automatically Generated Algorithms for the Vertex Coloring Problem *PLOS ONE* **8**(3)

[6] Blickle T 1997 Tournament selection Eds Back T, Fogel D B, Michalewicz Z *Handbook of Evolutionary Computation* Bristol, UK and New York NY Institute of Physics Publishing and Oxford University Press.

[7] Uy N Q, Hien N Th, Hoai N X, O'Neill M 2010 *Improving the Generalisation Ability of Genetic Programming with Semantic Similarity based Crossover* 2010 Eds A.I.Esparcia-Alcazar et al Euro GP LNCS 6021

[8] Silva S, Almeida J 2003 Dynamic maximum tree depth - a simple technique for avoiding bloat in tree-based GP2003 Proceedings of GECCO-2003 Berlin: Springer Verlag 1776–87

[9] Luke S, Panait L 2002 Lexicographic parsimony pressure Proceedings of GECCO-2002 San Francisco CA. Morgan Kaufmann 829–36

[10]He Dengxu. Genetic programming and applications 1999 *Guangxi Ethnic University Journal* **5**(3) 31-3

Authors

Pinchao Meng , born January 1978, China

Current position, grades: associate professor ,Department of Applied Mathematics, Changchun University of Science and Technology
University studies: mathematics
Scientific interest: computational mathematics ,Intelligent computing
Publications : 15

Weishi Yin , born January 1980, China

Current position, grades: lecturer ,Department of Applied Mathematics, Changchun University of Science and Technology
University studies: mathematics
Scientific interest: computational mathematics
Publications : 10

Yanzhong Li , born March 1965, China

Current position, grades: professor ,Beihua University
University studies: mathematics
Scientific interest: applied mathematics
Publications number or main: 9