WUJNS
WUHAN UNIVERSITY JOURNAL OF NATURAL SCIENCES

Vol.8 No.1B 2003 323-326

Article ID:1007-1202(2003)01B-0323-04

Point-Tree Structure Genetic Programming Method for Discontinuous Function's Regression

☐ **Xiong Sheng-wu, Wang Wei-wu**

School of Computer Science and Technology, Wuhan University of Technology, Wuhan 430070, Hubei, China

Abstract: A new point-tree data structure genetic programming (PTGP) method is proposed. For the discontinuous function regression problem, the proposed method is able to identify both the function structure and discontinuities points simultaneously. It is also easy to be used to solve the continuous function's regression problems. The numerical experiment results demonstrate that the point-tree GP is an efficient alternative way to the complex function identification problems.

Key words: genetic programming; symbolic regression; point-tree structure

CLC number: TP 301

Received date: 2002-10-12

Foundation item: Supported by the National Natural Science Foundation (60173046) and the Natural Science Foundation of Hubei Province (2002AB040)

Biography: Xiong Sheng-wu (1966-), male, Associate professor, research direction: evolutionary computing, parallel computing. E-mail: xiongsw@mail.whut.edu.cn

0 Introduction

We test a new GP structure called Point-Tree (PT) structure on symbolic regression of discontinuous function. In a regression problem we are searching for a function that closely matches an unknown function based on a finite set of sample points. Genetic programming (GP) introduced by Koza[1] uses a tree structure to represent an executable object or model. To understand how GP can find the unkown function automatically, please refer to Koza[1]. We would like to point out that when GP is applied to regress discontinuous functions, it gets a bad performance and bad candidate solutions. In attempt to improve both algorithm performance and solution quality, we use a new GP structure called PT structure. The new structure GP gets a good performance and solutions.

The next section defines the symbolic regression problem and how this is solved using genetic programming algorithm. In Section 2 we provide detail information on the PT structure and its genetic operations and show how this structure can be presented as a discontinuous function. In section 3 we explain our experiments and we provide numerical results. Finally we draw some conclusions and we provide some ideas for further research.

1 Symbolic Regression

The purpose of solving a symbolic regression problem is to find a function that closely matches some unknown function

on a certain interval[2]. More formally, given an unknown function $f(x)$ we want to find a function $g(x)$ such that $f(x_i)=g(x_i)\ \forall x_i \in X$, where X is a set of values drawn from the interval we are interested in. Note that we normally do not know $f(x)$ precisely. We only know the set of sample points $\{(x, f(x)) \mid x \in X\}$. In this study we use predefined functions and uniformly draw a set of 100 sample points from it to test our regression algorithm.

We use a genetic programming algorithm to generate candidate solutions, i. e. $g(x)$. These functions are presented as binary trees built up using binary functions and a terminal set. The precise definition of these sets and other parameter settings varies between the two experiments presented later. Therefore, we defer the presentation of these parameters until Section 3 where we conduct our experiments.

The selection scheme in an evolutionary algorithm is one of its basic components. It needs a way to compare the quality of two candidate solutions. This measurement, the fitness function, is calculated using knowledge of the problem. In symbolic regression we want to minimize the total error over all samples. This is defined as the absolute error in (1), which will be the fitness function for the GP algorithm.

$$\varepsilon = \sum_{x \in X} | f(x) - g(x) | \qquad (1)$$

Other fitness functions can also be used. For instance, based on the mean square error. We use this simple approach in this paper.

2 Point-Tree Structure

The point-tree structure is used to solve the regression problem of discontinuous function. As we know, there are several models in a discontionous function. Each of them is called a sub-function. Sometimes, these sub-functions are total different and they can't be represented as one model. The GP method tries to represent them as one model. This is the reason that GP gets a bad performance and bad solutions.

The point-tree structure uses an array of trees to present sub-functions and an array of floats to keep discontinuities' information. This is the right structure. Fig. 1 shows the structure. The target function is Eq. (2). We suppose that we had known how many discontinuities the function had. It can be guessed from prior information.

$$f(x) = \begin{cases} 2x+1 & -1 \leqslant x < 0 \\ x-3 & 0 \leqslant x < 1 \end{cases} \qquad (2)$$

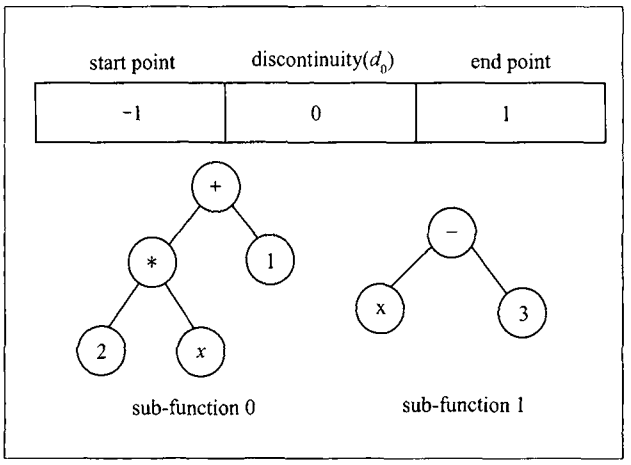

Fig. 1 Individual structure of a point-tree representation

2.1 Crossover

A crossover operation combines the genetic material of two parental individuals by swapping certain individual parts. The crossover for point-tree structure can be realized in two parts. The first part takes place on the sub-functions like tree-based crossover (T crossover). First we randomly select the number of the sub-function, so the sub-function in each parental individual which takes part in the tree-based crossover operation has decided. Then, in these sub-functions the crossover operator chooses a node randomly and exchanges the two corresponding subtrees. Fig. 2 illustrates this tree-bassed recombination method.

The second part takes place on the discontinuities(D crossover). First we randomly select the number of the discontinuity, so the discontinuity in each parental individual which takes part in the D crossover has been decided. Then the mean value of the two discontinuity takes the place of them to form new individuals. Fig. 3 illustrates this disscontinuity based recombination method.

For point-tree structure we use both methods but only one at a time. The following algorithm for the crossover of point-tree structure is applied for crossover.

procedure crossover (ind1, ind2)
1) randProb = random value between 0 and 1;
2) if (randProb < $\text{prob}_{\text{xover}}$)
3) perform a T cossover;
4) else
5) perform D crossover;
6) endif
end

In our tests the parameter $prob_{xover}$, which defines the probability whether the T or D crossover method is used, is set to the 50%.

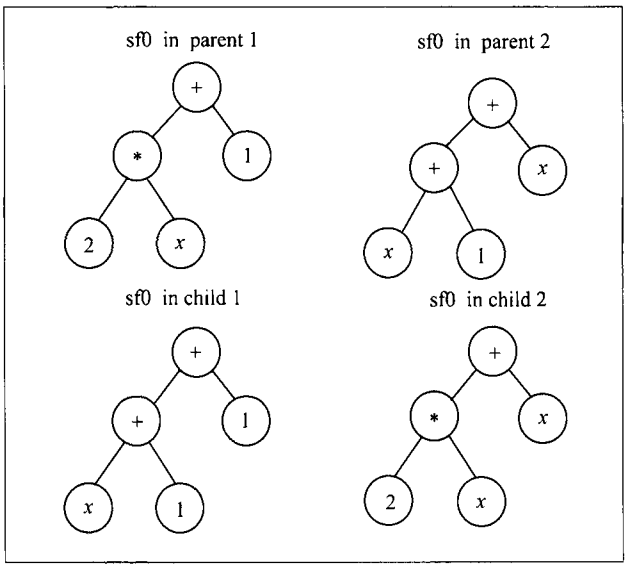

Fig. 2 Tree-based crossover

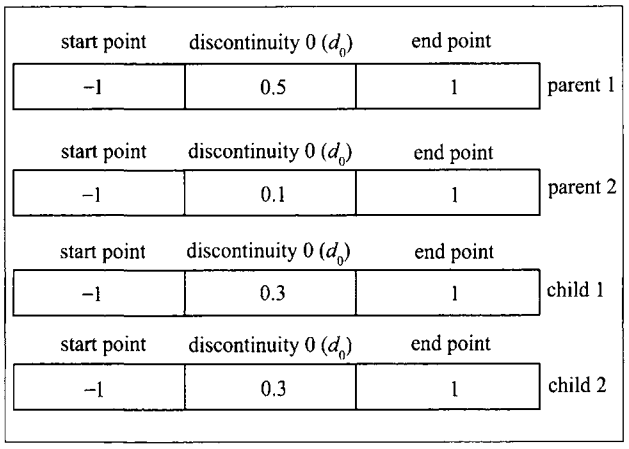

Fig. 3 Discontinuity-based crossover

2.2 Mutation

The difference between crossover and mutation is that mutation operates on a single individual only[3]. After the crossover of the population an individual is chosen with a given probability for mutation. If the mutation operator selects a sub-function for mutation, the sub-function swaps in a new randomly initialized subtree. If the mutation operator selects a discontinuity, the discontinuity is substituted by a new random value. The altered individual is then placed back into the population.

3 Experiment and Results

To test the performance of the point-tree structure GP we do a number of experiments. Koza's GP. and, PTGP ars tested with given setting as shown in Table1 and Table2. We use 100 independent runs for each setting in which we measure mean, median, stand deviation, minimum and maximum absolute error and the number of generations. We also count the number of successful runs. Where we define a run successful if the algorithm finds a function that has a zero absolute error. Furthermore, in PTGP algorithm, to know whether the PTGP algorithm can find the discontinuity automatically. we measure mean, median, standard deviation, minimum and maximum value of discontinuity.

Here we present two experiments with the given functions (3) and (4). Table3 and Table4 show the results. In function (3), GP has no any successful run, but PTGP has 67 successful runs out of 100. In function (4), GP has no successful run either, and PTGP has 80 successful runs out of 100. When we focus on fitness it appears that PTGP is significant better than GP. Its median value approaches zero. When we restrict our comparison to the number of generations, PTGP is also a winner. The GP always need 200 generations, the maximum generations. When we look on the median value of the discontinuities in PTGP, they all fall in areas of $[-2.04,-1.96]$, $[-0.04, 0.04]$, $[1.96, 2.04]$. We can infer PTGP can find discontinuites automatically.

$$f(x)=\begin{cases}1.0, & -4\leqslant x<-2\\ -1.0, & -2\leqslant x<0\\ 2.0, & 0\leqslant x<2\\ 1.0, & 2\leqslant x<4\end{cases} \tag{3}$$

$$f(x)=\begin{cases}1.0, & -4\leqslant x<-2\\ -(x+1), & -2\leqslant x<0\\ x-1, & 0\leqslant x<2\\ 1.0, & 2\leqslant x<4\end{cases} \tag{4}$$

4 Conclusions

We have shown how the point-tree structure can be used to boost performance in symbolic regression of discontinuous functions. We like to point out that the core factor of enhancement is the structure, i. e. point-tree structure accurately presents the essential structure of a discontinuous function.

We also like to point out that a continuous function's regression can also use this structure. When the parameter *number of discontinuities* is sets to zero, the whole function has only one sub-function and this sub-function

is continuous. So the whole function's evolution is a continuous function's evolution.

The point - tree structure has two parts. One is sub-functions, another is discontinuities. Each error of them can affect the individual's fitness. So how to make them cooperate perfectly is our future work.

Table 1 Setting in GP

Parameter	Value
stop criterion	maximum generation or perfect fit
function set	{+, −, *, pdiv, sin, cos}
terminal set	{1, x}
population size	500
crossover	0.7
mutation	0.2
reproduce	0.1
initial depth	2
maximum depth	10
maximum generation	200

Table 2 Setting in PTGP

Parameter	Value
stop criterion	maximum generation or perfect fit
function set	{+, −, *, pdiv, sin, cos}
terminal set	{1, x}
population size	500
crossover	0.7
mutation	0.2
reproduce	0.1
initial depth	2
maximum depth	6
maximum generation	200
number of discontinuites	3

Table 3 Result of function (3)

		median	mean	stddev	min	max
GP	fitness	24.916 4	25.318 1	6.813 3	11.807 0	43.161 3
	generation	200	200	0	200	200
PTGP	fitness	0	0.671 46	1.808 81	0	8.549 74
	generation	100	115.52	69.331 07	19	200
	d_0	−1.990 34	−1.982 14	0.068 59	−2.128 83	−1.756 49
	d_1	−0.007 85	−0.009 84	0.069 08	−0.188 25	0.266 32
	d_2	2.00 95	12.025 97	0.079 53	1.815 63	2.331 16

Table 4 Result of function (4)

		median	mean	stddev	min	max
GP	fitness	3.425 89	3.760 54	1.742 3	1.061 8	8.931 29
	generation	200	200	0	200	200
PTGP	fitness	0	0.358 29	1.257 7	0	7.482 59
	generation	77	99.92	67.037 1	20	200
	d_0	−1.987 35	−1.977 87	0.068 2	−2.258 16	−1.801 29
	d_1	−0.005 78	−0.001 7	0.065 48	−0.235 24	0.211 54
	d_2	2.026 4	2.037 25	0.075 11	1.816 1	2.298 22

References

[1] Koza J. *Genetic Programming*. Cambridge, MA: MIT Press, 1992.

[2] Eggermont J, Hemert J. Adaptive Genetic Programming Applied to New and Existing Simple Regression Problem. *Proceeding of Genetic Programming 4th European Conference*. Berlin: Springer, 2001.

[3] Lantschik W, Banzhaf W. Linear-Tree GP and Its Comparison with Other GP Structures. *Proceeding of Genetic Programming 4th European Conference*. Berlin: Springer, 2001.

□