Improved Gene Expression Programming to Solve the Inverse Problem for Ordinary Differential Equations

Kangshun Li[a], Yan Chen[a,*], Wei Li[a,b], Jun He[c], Yu Xue[d]

[a]*South China Agricultural University, College of Mathematics and Informatics Guangzhou, Guangdong, 510642, CN*
[b]*Jiangxi University of Science and Technology, School of Information Engineering, Ganzhou, 341000, CN*
[c]*Aberystwyth University, Department of Computer Science Aberystwyth, Ceredigion, UK*
[d]*Nanjing University of Information Science and Technology, School of Computer and Software, Nanjing, Jiangsu, CN*

Abstract

Many complex systems in the real world evolve with time. These dynamic systems are often modeled by ordinary differential equations in mathematics. The inverse problem of ordinary differential equations is to convert the observed data of a physical system into a mathematical model in terms of ordinary differential equations. Then the model may be used to predict the future behavior of the physical system being modeled. Genetic programming has been taken as a solver of this inverse problem. Similar to genetic programming, gene expression programming could do the same job since it has a similar ability of establishing the model of ordinary differential systems. Nevertheless, such research is seldom studied before. This paper is one of the first attempts to apply gene expression programming for solving the inverse problem of ordinary differential equations. Based on a statistic observation of traditional gene expression programming, an improvement is made in our algorithm, that is, genetic operators should act more often on the dominant part of genes than on the recessive part. This may help maintain population diversity and also speed up the convergence of the algorithm. Experiments show that this improved algorithm performs much better than genetic programming and traditional gene expression programming in terms of running time and prediction precision.

Keywords: gene expression programming, system of ordinary differential equations, inverse problem, Runge-Kutta algorithm

1. Introduction

There are many complex systems or non-linear phenomena varying with the time in the real world. Such systems are called dynamic systems, including weather change, population increase, disease diffusion and so on. In order to predict the development trend of such dynamic systems, it is often required to establish their mathematical models, that is, to establish the functional relationship or changing trend among variables of the systems. It is difficult to find the functional relations among variables in complicated changing processes, but it is still possible to find out the change rate or differential coefficients of some variables, and then to model them by ordinary differential equations (ODEs). If there are more than one unknown functions, we need to establish a group of ordinary differential equations (ODEs). Through the ODEs model of a physical system, it is possible to learn the development trend of the system and apply the prediction in the real world.

The problem of converting observed data of a physical system into a mathematical model in terms of differential equations is known as the inverse problem [1, 2] of differential equations [3, 4, 5, 6]. For instance, if we have previous data of a stock market, we may create an ODEs model for the stock market using previous data and then predict the development trend of the stock market. The inverse problem of

*Corresponding author
Email address: `cheny@scau.edu.cn` (Yan Chen)

differential equations plays an important role in many areas from scientific experiments to stock markets. However, given observed data, it is not an easy task to create models of ODEs for complex dynamical systems, because these problems are very complicated and usually belong to non-linear systems, so it is difficult to determine the structure of ODEs and parameters in ODEs in order to create a correct model.

In this paper, an improved Gene Expression Programming (GEP) is put forward to solve the inverse problems of ordinary differential equations. GEP is a kind of evolutionary algorithms based on genome and phenomena and referred to the gene expression rule in the genetics [7, 8]. It intends to combine the advantages of both GP and GA [9]. Unlike GP where an individual is expressed in the form of a tree, an individual in GEP is represented by the Isometric linear symbols. GEP [10] has been successfully applied in problem solving [7], combinatorial optimization [11], real parameter optimization [12], evolving and modeling the functional parameters [13], classification [14, 15], event selection in high energy physics [16].

Choosing GEP is based on several reasons. First, an GEP algorithm adopts a multi-gene structure, where each gene stands for an ODE and each chromosome for a group of differential equations. This is different from traditional algorithms in which an individual cannot be used to represent a group of ODEs directly. Secondly, previous experiments show that GEP algorithms have a better prediction effect in the shorter time and its time cost is so stable that it is seldom influenced by the complexity of dynamical systems. In addition, an improvement is made in our GEP algorithm. It is more suitable for studying the inverse problems of ordinary ODEs than the traditional ones because more genetic operations are centered at the dominant segment of the gene and fewer genetic operations are centered at the recessive segment[17].

The remainder of this paper is organized as follows: Section 2 introduces inverse problems of ODEs. Section 3 presents an improved GEP algorithm for solving the inverse problems of ODEs. Section 4 gives computer experiment results. Section 5 concludes the whole paper.

2. Inverse Problems for Ordinary Differential Equations

A dynamic system is represented by n correlated functions: $x_1(t), x_2(t), \cdots, x_n(t)$ where t denotes time. The system has a series of observed data collected at times $t_j = t_0 + j \times \Delta t$, $(j = 0, 1, \cdots, m-1)$, where t_0 represents the starting time, Δt the time increment, and $x_i(t_j)$ the observed value of x_i at the time t_j. Write the observation data in a matrix form:

$$\mathbf{X}_m := \begin{pmatrix} x_1(t_0), & x_2(t_0), & \cdots, & x_n(t_0) \\ x_1(t_1), & x_2(t_1), & \cdots, & x_n(t_1) \\ \cdots, & \cdots, & \cdots, & \cdots \\ x_1(t_{m-1}), & x_2(t_0), & \cdots, & x_n(t_0) \end{pmatrix}. \tag{1}$$

Denote

$$\mathbf{x}(t) := [x_1(t), x_2(t), \ldots, x_n(t)]^T, \tag{2}$$
$$\mathbf{f}(\mathbf{x}, t) := [f_1(\mathbf{x}, t), f_2(\mathbf{x}, t), \ldots, f_n(\mathbf{x}, t)]^T \tag{3}$$

where $f_j(\mathbf{x}, t) = f_j(x_1(t), x_2(t), \ldots, x_n(t), t)$ $(j = 1, 2, \ldots, n)$ is a composite function of several elementary functions involving of $x_i (i = 1, \cdots, n)$ and t. Let $\mathcal{F}$ denote the set of all possible composite functions.

A system of ordinary differential equations (ODEs) in the form of

$$\frac{dx_i(t)}{dt} = f_i(x_1, \cdots, x_n, t), \quad i = 1, \cdots, n \tag{4}$$

can be written in the vector form

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{f}(\mathbf{x}, t). \tag{5}$$

The goal of the inverse problem of ODEs is to find a mathematical model which is represented by a system of ODEs

$$\frac{d\mathbf{x}^*(t)}{dt} = \mathbf{f}(\mathbf{x}^*, t) \tag{6}$$

such that

$$\min\{\|\mathbf{X}_m^* - \mathbf{X}_m\|; \mathbf{f} \in \mathcal{F}\} \tag{7}$$

where the matrix norm

$$\|\mathbf{X}_m^* - \mathbf{X}_m\| := \sqrt{\sum_{j=0}^{m-1} \sum_{i=1}^{n} (x_i^*(t_j) - x_i(t_j))^2} \tag{8}$$

The above the matrix norm represents the difference between the observed data and the corresponding values derived from the ODEs model.

Then we may use the obtained ODEs (6) to predicate the future trend of the system. The above problem is called the inverse problem of ODEs.

Different approaches have applied to solving the inverse problem of ODEs. Linear modeling, such as Autoregressive model, Moving Average model, Autoregressive Moving Average model, are simple and popular [18, 19, 20]. However, there exist several restrictions for linear models. Firstly, they are linear models so that they can not represent non-linear dynamical systems. Secondly, identification and estimation of linear models requires strong mathematical knowledge and expertise, which often lacks in practice. Finally, once a model is established, it is not easy to constantly adjust the structure and parameters of the model based on updated observation data.

Another simple modeling approach is to take a form of differential equations which are pre-selected by experience, and then a numerical method is used to determine the variables [21]. However, how to pre-select the right differential equation model is a difficult task, especially for the differential equations whose number of variables increases.

Evolutionary modeling [22, 23, 24, 25] has been successfully used in studying the inverse problems of ordinary differential equations. Current evolutionary modeling are mainly based on Genetic Programming [22, 25, 26], [27] where an equation is represented in the form of tree. A hybrid evolutionary methods are proposed for evolving ODEs, for example, to predict small-time scale traffic measurements data in [28], which uses tree model to evolve but its speed is also slow. ECSID [29] found good models for linear pendulum, non-linear pendulum with friction, coupled mass-spring, and linear circuit. But the difference between model found by ECSID and original model becomes large when the model becomes complex. GEP-SWPM is proposed in [30], and the prediction is based on several generations of data before, so it is seriously affected by the noise. A new methods of GEP was proposed in [13, 31], which is very effective for identifying parameter functions. But it is based on the assumed model and doesn't provide a common solution and it can't be extended to most situations.

In this paper, we proposed an improved GEP algorithm for solving the inverse problem of ODEs.

3. Improved GEP for the Inverse Problem of ODEs

3.1. Gene Representation in GEP Algorithm

The genetic codes of GEP is the isometric linear symbols (GEP chromosome). Each chromosome can be composed of several genes. GEP gene consists of a head and a tail, where the former may contain both the functional symbols and termination symbols, while the latter only has the terminal symbols. For example, $\texttt{*+-aQ*+a}\underline{\texttt{ababb}}\ \underline{\texttt{baab}}$ is a legal gene, of which, `*` stands for the multiplication operation, `Q` the square root operation, the segment without underline belongs to the head, while the underlined segment is the tail. Figure 1 shows the expression of the gene in the form of a tree.

For each problem, the length of the tail t is a function of the length of the head h and the number of arguments of the function with the most arguments n, determined using the following formula [32].

$$t = h(n-1) + 1 \tag{9}$$

3.2. The Flowchart of GEP Algorithm for Soling the Inverse Problem of ODEs

GEP used to solve the inverse problem, is shown in Algorithm1. The details of procedures are described.

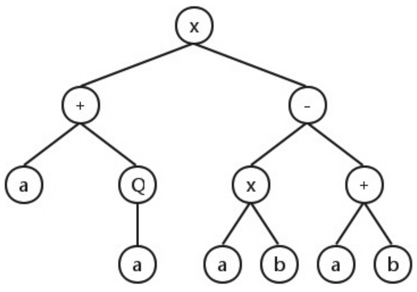

Figure 1: Expression tree 1

Algorithm 1 Improved GEP algorithm for inverse problem of ODEs

```
1: Determine the termination symbol set, operator set and control parameters;
2: Initialize a population;
3: while the termination requirement is not met do
4:     Convert the chromosome into the expression tree (this step might be replaced by the GRCM algorithm
       introduced later);
5:     Calculate the fitness;
6:     Apply genetic operation;
7:     Obtain a new population;
8: end while
9: Output the optimal solution;
```

3.3. Initialization

The first task in the initialization is to set control parameters, including the length of a gene's head and tail, the number of genes, a termination symbol set and a functional set.

The termination set used for the inverse problem of ODEs is $\{t, 1, 2, 3, \cdots\}$ where i stands for x_i. The functional set is

$$\{+, -, *, \div, s, c, Q, e, \ln\},$$

where $s = \sin, c = \cos, Q = \sqrt{}, e = \exp$. These sets are determined specifically for the inverse problem.

The settings of the length of a gene's head and tail, and the number of the gene rely on the problem. In our experiments, the head length is set to 8. Since the maximal number of the operators is 2, the tail length is set 9 according to (9).

The second task in the initialization is to create an initial population. A gene is generated at random using the termination set and functional set, subject to the constraints on the head and tail lengths of genes. A chromosome has k genes where k is fixed. For example, `*+-1Q*+3201321023` is a gene, and the chromosome consists of 3 genes: `*+-1Q*+3201321023` `*-*1+*+*2023 12032` `*+*1Q*+3210301323`. Then a number of chromosomes or individuals are generated, where the number of chromosomes is called the population size.

3.4. Fitness Evaluation and Chromosomes Ranking

The fitness evaluation of an individual is rather complex in the inverse problem of ODEs [33]. Firstly, we produce a system of ODEs or an ODEs model from each individual. Secondly, the model is used to produce prediction data using Runge-Kutta's method. Finally, the fitness is calculated by comparing prediction data and actual data. The detail is described as follows.

- Calculate genes

The traditional GEP method is to convert a chromosome into an expression tree, and then solve it via stacks. This is a complicated process. In this paper, we adopt an alternative method, called Gene Read

& Compute Machine (GRCM) algorithm [34] for calculating genes. The solutions of a chromosome can be achieved directly without converting the chromosome into an expression tree. GRCM algorithm is described in Algorithm2.

Algorithm 2 GRCM algorithm

```
1: Calculate the valid length of genes and determine the valid genetic sequence;
2: Read the valid genetic length forward one by one from the last operator until the first operator is
   achieved; record its position P)
3: while the valid length of the gene is 1 do
4:    if P targets to the unary operator then
5:       Read the genetic sequence in valid length from the last character and find out a terminal symbol,
         which is used as a parameter for operation, substitute the above operator with the operation results
         and convert the former operator into a terminal symbol;
6:       When the valid length of gene is subtracted by 1;
7:       The position of P moves forward by 1;
8:    else if P targets to the binary operator then
9:       Read the genetic sequence in valid length from the last character and find out two terminal symbols,
         which is used as a parameter for operation, substitute the above operator with the operation results
         and convert the former operator into a terminal symbol;
10:      When the valid length of gene is subtracted by 2;
11:      The position of P moves forward by 1;
12:   else
13:      No operation is conducted;
14:      Position P moves forward by 1;
15:   end if
16: end while
17: return the value of the terminal symbol
```

Comparing with the traditional methods, GRCM is easy for understanding and convenient for operation, what's more, the operation can be conducted at high speed. The more complicated the chromosome is, the greater advantages the algorithm has.

- Generate training and prediction data

The Runge-Kutta [35] is adopted in this paper to accomplish the prediction data. The Runge-Kutta is a iteration method for simulating the ODE solutions. At present, the commonly used Runge-Kutta is RK4, which is used in the condition that the differential coefficient of equation and the original value are known and omits the complicated process of solving the differential equation via the computer simulation. The Runge-Kutta can be used to achieve the precise solution especially for the complicated non-linear differential equation group and the non-linear ordinary differential equation group that is too complicate to obtain the precise solution.

For the ordinary differential equation with one variable $x(t)$

$$\frac{dx(t)}{dt} = f(x(t), t), \tag{10}$$

the RK4 formula is shown in (11).

$$\begin{cases} x(t_0) \text{ is the original value} \\ K_1 = f(x(t_j), t_j) \\ K_2 = f(x(t_j) + \frac{1}{2}K_1\Delta_t, t_j + \frac{1}{2}\Delta_t) \\ K_3 = f(x(t_j) + \frac{1}{2}K_2\Delta_t, t_j + \frac{1}{2}\Delta_t) \\ K_4 = f(x(t_j) + K_3\Delta_t, t_j + \Delta_t) \\ x(t_{j+1}) = x(t_j) + \frac{1}{6}(K_1 + 2K_2 + 2K_3 + K_4)\Delta_t \end{cases} \tag{11}$$

where $\Delta_t = t_{j+1} - t_j$.

For the ordinary differential equation with n variables $x_1(t), \cdots, x_n(t)$,

$$\frac{dx_i(t)}{dt} = f_i(x_1(t), \cdots, x_n(t), t), \quad i = 1, \cdots, n \tag{12}$$

the RK4 formula is shown in (13).

$$\begin{cases} x_1(t_0), x_2(t_0), \cdots, x_n(t_0) \text{ are orignal values,} \\ K_{i,1} = f_i(x_1(t_j), x_2(t_j), \cdots, x_n(t_j), t_j), \\ K_{i,2} = f_i(x_1(t_j) + \frac{1}{2}K_{1,1}\Delta_t, x_2(t_j) + \frac{1}{2}K_{2,1}\Delta_t, \\ \qquad \cdots, x_n(t_j) + \frac{1}{2}K_{n,1}\Delta_t, t_j + \frac{1}{2}\Delta_t) \\ K_{i,3} = f_i(x_1(t_j) + \frac{1}{2}K_{1,2}\Delta_t, x_2(t_j) + \frac{1}{2}K_{2,2}\Delta_t, \\ \qquad \cdots, x_n(t_j) + \frac{1}{2}K_{n,2}\Delta_t, t_j + \frac{1}{2}\Delta_t) \\ K_{i,4} = f_i(x_1(t_j) + K_{1,3}\Delta_t, x_2(t_j) + K_{2,3}\Delta_t, \\ \qquad \cdots, x_n(t_j) + K_{n,3}\Delta_t, t_j + \Delta_t) \\ x_i(t_{j+1}) = x_i(t_j) + \frac{1}{6}(K_{i,1} + 2K_{i,2} + 2K_{i,3} + K_{i,4})\Delta_t \\ \qquad i = 1, 2, \cdots, n \end{cases} \tag{13}$$

The calculation is as follows: first, achieve the matrix $\mathbf{X}$ from the observed data:

$$\mathbf{X} = \begin{pmatrix} x_1(t_0) & x_2(t_0) & \cdots & x_n(t_0) \\ x_1(t_1) & x_2(t_1) & \cdots & x_n(t_1) \\ \cdots & \cdots & \cdots & \cdots \\ x_1(t_{m-1}) & x_2(t_{m-1}) & \cdots & x_n(t_{m-1}) \end{pmatrix} \tag{14}$$

Then, select the data at first line of the matrix $\mathbf{X}$ as the original values of RK4, and calculate the prediction data $\mathbf{X}^*$ from the achieved model via the RK4 method as per the (14) by adopting the increment same with that of matrix $\mathbf{X}$, see (15).

$$\mathbf{X}^* = \begin{pmatrix} x_1(t_0) & x_2(t_0) & \cdots & x_n(t_0) \\ x_1^*(t_1) & x_2^*(t_1) & \cdots & x_n^*(t_1) \\ \cdots & \cdots & \cdots & \cdots \\ x_1^*(t_{m-1}) & x_2^*(t_{m-1}) & \cdots & x_n^*(t_{m-1}) \end{pmatrix} \tag{15}$$

The first row of matrix $\mathbf{X}^*$ are the original values of RK4 and they are same as the data at first line of matrix X. The remaining data of matrix X' are generated using (13).

- Construction of fitness function

Construction of fitness function is to ensure the algorithm to evolve in the required direction; it is the major drive for the evolvement of GEP groups. Different fitness function has different influences on the evolvement quality. The better the fitness function is, the individual is more adaptable to the environment and the greater probability that the individual evolves to the next generation [36]. The fitness function in this paper is constructed by the difference between matrix X and X', i.e. $\Delta = \|X - X'\|$.

Δ is the difference of the two matrixes, and the smaller it is, the better it is. Therefore, the greater the fitness is, the better it is after the conversion as per (16).

$$fitness = \frac{1}{\Delta + 1} \tag{16}$$

3.5. Genetic Operators

- Selection

The Roulette selection is adopted in the paper. The better the fitness, the greater probability an individual is reproduced to the next generation. The reproduction times will be determined as per the Roulette principle in the process and meanwhile the population size is unchanged.

- Mutation operator design

The mutation operation can be performed at any position of the chromosome and, according to relevant stipulations, the operator at the head can be mutated into any function or terminal symbol, but that at the tail can only be mutated into the terminal symbol, such mutation can ensure the newly generated chromosome is in the valid structure. In accordance with relevant requirements, the mutation probabilities set in this paper are 0.044, which is used for performing the mutation operation for the chromosome.

- Transposition operator design

The transposition operation are performed as per the set transposition probability and length, there are three kinds of transposition operations in this paper. In addition, the transposition probability is set as 0.1 and length is 5 as per the relevant experience.

Insertion Sequence Transposition: The conversion segment is selected randomly from the chromosome and the segment can be inserted at any position of the head except for the original position.

Root Insertion Sequence Transposition: It is similar to the IS transposition, but its conversion segment can only be inserted at the original position of the gene.

Gene Transposition: The complete gene is selected as the conversion segment and inserted at the original position of the chromosome; however, the selected gene will be deleted in the new chromosome [32, 37].

- Restructuring operator design

There are also three kinds of restructuring operators in this paper. They are single-point restructuring, double-point restructuring and gene restructuring. Single-point restructuring operator: This is similar to the single point mutation in GA, in this paper, we randomly selected an exchange point for two chromosomes, and then exchanged the chromosome segment behind the point. Double-point restructuring operator: This is similar to the double-point mutation in GA, we randomly selected two exchange points for two chromosomes, and then exchange the two strings between the exchange points. Gene restructuring operator: This operator only acts on chromosomes of multiple genes. We randomly selected a gene for two polygene chromosomes, and then exchanged the two chromosome of the corresponding genes. This is similar to the double-point restructuring operator, except that the swap substring must be a complete gene. Select randomly the restructuring segment or gene from the parent chromosome as per the specific restructuring operator, and then exchange the selected segments or genes. Care shall be taken that the gene restructuring operator cannot generate the new gene: the newly-generated individual is the array or combination of the existing gene. The probabilities for single-point restructuring, double-point restructuring and gene restructuring set in this paper are 0.3, 0.3 and 0.1 respectively.

3.6. Termination conditions

The algorithm will stop once any of the following requirements is met.

- The maximum number of generations is reached;
- The fitness of the best individual reaches the a predefined value, or it is unchanged for a predefined number of generations.

3.7. Improvement of GEP algorithm

A GEP gene has coding regions and noncoding regions. For example, consider the gene

$$\texttt{Q*+*a*Qaab}\underline{\texttt{abbaababa}}\ \underline{\texttt{ab}},$$

the part without underline is its Head while the underlined segment is the tail. Figure 2 illustrates the expression tree associated with the gene.

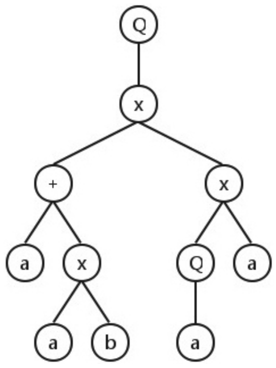

Figure 2: Expression tree 2

From the expression tree in Figure 2, we see that the last 10 characters of the gene are noncoding regions. This is the reason why GEP is a non-linear segment and its phenotype form and genotype form of the genes may be different.

Obviously, only when the mutation, transposition and restructuring operators are applied into the dominant segment of the gene which can affect the fitness directly, they increase the diversity of the population remarkably. If they are applied into the recessive segment which can not affect the fitness directly, they can only increase the probability for population diversity before performing the mutation operator. For the four standard genetic operators: mutation, IS transposition, single-point restructuring and double-point restructuring, all characters have the equal probability to become the position of the operator, therefore, the right selection of the position of these characters has a great impact on the validity of the genetic operators.

To verify the above statement, we demonstrate the average valid length of GEP genes used in recent references by Table 1.

Table 1: The valid length and the average utilization rate of gene

Reference	Head length	zero-param function	one-param function	two-param function	valid length of gene	length of gene	Average use ratio of gene
[38]	8	2	4	4	7.761	17	45.66%
[38]	8	5	0	4	3.827	17	22.51%
[38]	8	4	4	4	5.24	17	30.82%
[39]	10	1	0	4	13.657	21	65.03%
[40]	10	5	3	4	4.475	21	22.60%
[41]	8	5	4	4	4.478	17	26.34%
[42]	8	6	2	8	6.007	17	35.34%
[43]	8	8	14	4	4.195	17	24.68%
[44]	10	14	2	4	1.967	21	9.37%
[44]	10	15	2	4	1.886	21	8.98%
[44]	10	16	2	4	1.817	21	8.65%
[45]	19	1	0	4	24.466	39	62.73%
[45]	19	5	3	4	6.087	39	15.61%

From the utmost right column of Table 1, we see that the average utilization rate of gene is very low, especially for a function without parameter. The greater proportion of the function without parameter is, the lower the average utilization rate of gene is. In this case, genetic operators with varying positions may take place at the recessive segment of the gene with great probability even though the occurrence probability is met. It will greatly reduce the validity of genetic operators. Therefore, we propose that most of genetic operators can be centered at the dominant segment. This may help maintain population diversity and also increase the convergence rate of GEP. The setting of the occurrence probabilities of the dominant and recessive parts of the gene is determined by the problem. In the paper, this probability is assigned to 0.8, that is, 80% of genetic operation occurs at the dominant part of the gene and the remaining 20% at the recessive segment.

4. Computer Experiments

4.1. Settings of Experimental Parameters

The parameters used in the experiments are listed as follows:

- Set of termination symbols: I-II:{1,2,3,0}, III:{1,0}, IV:{1,2,0}
- Set of functional symbols: $\{+, -, *, \div, \sin, \cos, Q, \exp, \ln\}$
- Number of iterations: 10000
- Size of Populations: 50
- Number of gene: 3
- Head length of gene: 8
- Selection algorithm: Roulette
- Probability of gene restructuring: 0.1
- Probability of mutation: 0.044
- Probability of IS transposition: 0.1

- Length of IS transposition : 5
- Probability of RIS transposition: 0.1
- Length of RIS transposition: 5
- Length of Gene Transposition: 0.1
- Probability of single-point restructuring: 0.3
- Probability of double-point restructuring: 0.3

4.2. Experiment 1

We searched carefully with other state of the art solvers the inverse problem of ordinary differential equations including based the proposed method. There is only one literature [46] to discuss this inverse problem of ordinary differential equations. The training data set in experiment 1 is chosen from [46] which is constructed from the solution (18) to the ODEs (17).

$$\begin{cases} \dfrac{dx_1}{dt} = x_1 \\ \dfrac{dx_2}{dt} = x_1 + x_2 + x_3 \\ \dfrac{dx_3}{dt} = 2x_1 - x_2 + 3x_3 \end{cases} \tag{17}$$

$$\begin{cases} x_1 = e^t \\ x_2 = e^{2t}(t+1) \\ x_3 = e^{2t}(t+2) - e^t \end{cases} \tag{18}$$

Let $t_0 = 0$, $\Delta t = 0.01$, then we get data I described in Table 2. The part $\mathbf{X}_m$ is used as the observation data, and the part $\mathbf{X}_p$ as the prediction data for evaluate the quality of the model. For the dataset listed in Table 2, we performed 100 experiments and choose the best ODEs model produced in the experiments. Then, Runge-Kutta iteration is performed on the ODEs. The results of the best model obtained by using GEP is listed in Table 3. $\mathbf{E}_m$ represents the error of the ODE model on the training data and $\mathbf{E}_p$ the error of the ODE model under modelling on the prediction data.

Table 2: Dataset I where $t_0 = 0, \Delta t = 0.01$

		Observed data			Modeling data		
	t	$x_1(t)$	$x_2(t)$	$x_3(t)$	$x_1(t)$	$x_2(t)$	$x_3(t)$
X_m	t_0	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
	t_1	1.010050	1.030403	1.040555	1.010050	1.030455	1.040763
	t_2	1.020201	1.061627	1.082236	1.020201	1.061731	1.082565
	t_3	1.030455	1.093692	1.125074	1.030455	1.093851	1.125445
	t_4	1.040811	1.126619	1.169095	1.040811	1.126832	1.169441
	t_5	1.051271	1.160429	1.214329	1.051271	1.160698	1.214592
	t_6	1.061837	1.195147	1.260807	1.061837	1.195469	1.260939
X_p	t_7	1.072508	1.230793	1.308559	1.072508	1.231167	1.308526
	t_8	1.083287	1.267392	1.357616	1.083287	1.267817	1.357398
	t_9	1.094174	1.304967	1.408010	1.094174	1.305440	1.407599
	t_{10}	1.105171	1.343543	1.459775	1.105171	1.344063	1.459179

Table 3: Result I: $\mathbf{E}_m$ and $\mathbf{E}_p$ represent the error analysis results of comparison between improved GEP and Kang. et al. GP

		Improved GEP			Kang. et al. GP		
	t	$x_1(t)$	$x_2(t)$	$x_3(t)$	$x_1(t)$	$x_2(t)$	$x_3(t)$
E_m	t_0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
	t_1	0.000000	0.000052	0.000208	0.000000	0.000242	0.000011
	t_2	0.000000	0.000104	0.000329	0.000000	0.000142	0.000008
	t_3	0.000000	0.000159	0.000371	0.000001	0.000045	0.000006
	t_4	0.000000	0.000213	0.000346	0.000000	0.000052	0.000001
	t_5	0.000000	0.000269	0.000263	0.000000	0.000146	0.000010
	t_6	0.000000	0.000322	0.000132	0.000001	0.000242	0.000020
E_p	t_7	0.000000	0.000374	0.000033	0.000001	0.000334	0.000033
	t_8	0.000000	0.000425	0.000218	0.000000	0.000760	0.000081
	t_9	0.000000	0.000473	0.000411	0.000001	0.001274	0.000148
	t_{10}	0.000000	0.000520	0.000596	0.000000	0.001876	0.000236
Regression standard error			0.000000			0.000405	
Prediction standard error			0.001178			0.002432	

The experiment results in Table 3 demonstrates that the error of improved GEP has smaller than the Kang. et al. GP. This indicates the effectiveness of the improved GEP algorithm for solving the inverse problem of differential equations.

4.3. Experiment 2

In experiment 2, the training data set is still constructed from the solution (18) to the ODEs (17).

Set $t_0 = 0$ and $\Delta t = 0.05$, then we get data II described in Table 4. The part $\mathbf{X}_m$ is taken as observation data, and the part $\mathbf{X}_p$ as the prediction data for evaluating the solution quality of the model.

Table 4: experimental data II where $t_0 = 0, \Delta t = 0.05$

		Observed data			Modeling data		
	t	$x_1(t)$	$x_2(t)$	$x_3(t)$	$x_1(t)$	$x_2(t)$	$x_3(t)$
X_m	t_0	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
	t_1	1.051271	1.160429	1.214329	1.051271	1.159791	1.215778
	t_2	1.105171	1.343543	1.459775	1.105171	1.342855	1.462808
	t_3	1.161834	1.552338	1.740362	1.161834	1.552017	1.744937
	t_4	1.221403	1.790190	2.060611	1.221403	1.790444	2.066580
	t_5	1.284025	2.060902	2.425597	1.284025	2.061699	2.432779
	t_6	1.349859	2.368755	2.841015	1.349859	2.369794	2.849278
X_p	t_7	1.419068	2.718566	3.313251	1.419068	2.719254	3.322596
	t_8	1.491825	3.115758	3.849474	1.491825	3.115182	3.860125
	t_9	1.568312	3.566425	4.457716	1.568312	3.563335	4.470230
	t_{10}	1.648721	4.077424	5.146984	1.648721	4.070218	5.162366

For the dataset listed in Table 4, we performed 100 experiments and choose the best ODEs model generated in the experiments. Table 5 gives the error between the prediction by the best model and the solution to the original ODEs. $\mathbf{E}_m$ and $\mathbf{E}_p$ represent the error on training data and prediction data respectively.

Table 5: result II: $\mathbf{E}_m$ and $\mathbf{E}_p$ represent the error analysis results of comparison between improved GEP and Kang. et al. GP

		Improved GEP			Kang. et al. GP		
	t	$x_1(t)$	$x_2(t)$	$x_3(t)$	$x_1(t)$	$x_2(t)$	$x_3(t)$
E_m	t_0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
	t_1	0.000000	0.000052	0.000208	0.461439	0.509091	0.007282
	t_2	0.000000	0.000104	0.000329	0.000000	0.004718	0.005734
	t_3	0.000000	0.000159	0.000371	0.000000	0.002619	0.003613
	t_4	0.000000	0.000213	0.000346	0.000000	0.000192	0.000812
	t_5	0.000000	0.000269	0.000263	0.000001	0.002601	0.002799
	t_6	0.000000	0.000322	0.000132	0.000006	0.004801	0.007367
E_p	t_7	0.000000	0.000374	0.000033	0.000000	0.009453	0.013068
	t_8	0.000000	0.000425	0.000218	0.000000	0.024130	0.031253
	t_9	0.000000	0.000473	0.000411	0.000000	0.045175	0.055053
	t_{10}	0.000000	0.000520	0.000596	0.000000	0.073926	0.085000
Regression standard error			0.000000			0.000405	
Prediction standard error			0.001178			0.002432	

Experiment results in Table 5 demonstrates that prediction provided by the model show good agreement with the original ODEs. It indicates that the improved GEP algorithm is applicable for solving the inverse problem of ordinary differential equations.

As can be seen from Table 2 to Table 5, most of the results are better than the Kang. et al. GP algorithm, except for individual modeling results and predictive results. In addition, due to explore the improved GEP algorithm can transfer complex nonlinear curve into a linear problem, therefore, the use of improved GEP algorithm for ordinary differential equation have the more advantage of making the convergence faster than Kang. et al. GP algorithm in terms of computational complexity.

4.4. Experiment 3

The training data III, given in Table 6, come from [47]. The part $\mathbf{X}_m$ is taken as observation data, and the part $\mathbf{X}_p$ as the prediction data for evaluating the quality of the model found by the improved GEP algorithm.

Table 6: experimental data III where $t_0 = 0, \Delta t = 0.01$

III	t	$x(t)$	t	$x(t)$
$\mathbf{X}_m$	t_0	3.9	t_8	38.6
	t_1	5.3	t_9	50.2
	t_2	7.2	t_{10}	62.9
	t_3	9.6	t_{11}	76.0
	t_4	12.9	t_{12}	92.0
	t_5	17.1	t_{13}	106.5
	t_6	23.2	t_{14}	123.2
	t_7	31.4		
$\mathbf{X}_p$	t_{15}	131.7	t_{16}	150.7

For the dataset listed in Table 6, we performed 100 experiments and choose the best model for prediction. The results of the best models obtained by using GEP are listed in Table 7.

Table 7: result III: $\mathbf{E}_m$ and $\mathbf{E}_p$ represent the error of the ODE model and system under modeling on training data and prediction data respectively

Model	$\left\{\frac{dx}{dt} = \frac{(xx)+((x+t)-t)}{t}\right.$			
III	t	$x(t)$	t	$x(t)$
$\mathbf{E}_m$	t_0	0.000000	t_8	6.617181
	t_1	0.570360	t_9	4.533561
	t_2	1.492591	t_{10}	2.238262
	t_3	2.798630	t_{11}	0.764010
	t_4	4.128972	t_{12}	2.717210
	t_5	5.511010	t_{13}	3.693929
	t_6	5.962552	t_{14}	5.865052
	t_7	5.295318		
$\mathbf{E}_p$	t_{15}	1.170253	t_{16}	1.287381
Training standard error				9.551644
Prediction standard error				1.739782

The experiment results in Table 7 demonstrates that the dataset can predict the original ordinary differential equations by using the improved GEP algorithm, which indicates that it is feasible to use the improved GEP algorithm for the inverse problem of differential equations.

4.5. Experiment 4

The training data IV, given in Table 8, come from [47]. The part $\mathbf{X}_m$ is taken as observation data, and the part $\mathbf{X}_p$ as the prediction data for evaluate the quality of the model.

Table 8: experimental data IV where $t_0 = 0, \Delta t = 0.01$

IV	t	$x_1(t)$	$x_2(t)$
$\mathbf{X}_m$	t_0	6.846	1.567
	t_1	7.642	1.896
	t_2	8.531	2.180
	t_3	9.071	2.460
	t_4	9.963	2.785
$\mathbf{X}_p$	t_5	11.052	3.121

For the dataset listed in Table 8, we performed 100 experiments obtain the better model. The results of the best models obtained by using GEP are listed in Table 9.

The experiment results in Table 9 demonstrates that the dataset can predict the original ordinary differential equations by using the improved GEP algorithm, which indicates that it is feasible to use the improved GEP algorithm for the inverse problem of differential equations.

4.6. A Comparison of GP, Standard GEP and Improved GEP for the Inverse Problem of ODEs

To verify the advantages, disadvantages and existing problems of applying the GEP algorithm to the inverse problem of ordinary differential equations, we also conducted a comparative experiment. At present, state-of-the-art of the inverse problems using ordinary differential equations are solved by mathematical analysis. Furthermore, the limitations of inverse problems and complex differential equations are difficult to construct. There are different method of analytic solution in mathematics. The method of artificial intelligence is applied to the inverse problem of ordinary differential equations in this paper. Therefore, the

Table 9: result IV: $\mathbf{E}_m$ and $\mathbf{E}_p$ represent the error of the ODE model and system under modeling on training data and prediction data respectively.

Model	$\begin{cases} \frac{dx_1}{dt} = ((x_2x_1)+x_1)(x_1/x_2) - x_2(t+x_1) \\ \frac{dx_2}{dt} = ((x_2x_1)-(x_2+x_2))/\sin(x_2/x_1)+x_1 \end{cases}$		
IV	t	$x_1(t)$	$x_2(t)$
	t_0	0.000000	0.000000
	t_1	0.144472	0.041049
$\mathbf{E}_m$	t_2	0.310613	0.030504
	t_3	0.035360	0.003078
	t_4	0.006544	0.001825
$\mathbf{E}_p$	t_5	0.004764	0.013692
Training standard error			0.154365
Prediction standard error			0.014497

comparison experiment in this paper only chooses two kinds of algorithms of the same method, and analyzes the experimental results with this improved algorithm. All algorithms in this study in 100 experiment runs using the four datasets. The experiment results were compared with the GP, the standard GP and the improved GEP. The statistical errors listed in Table 10 were obtained.

Table 10: The result of 100 experiment runs using the four datasets for GP, Standard GEP and Improved GEP

	Data	GP	Standard GEP	Improved GEP
Training Standard Deviation	I	0.000405	0.000000	0.000000
	II	0.016548	0.008307	0.005477
	III	3.069454	17.115628	9.551644
	IV	0.043305	0.158723	0.154365
Prediction Standard Deviation	I	0.002432	0.001224	0.001178
	II	0.139932	0.027663	0.025618
	III	5.646193	2.424810	1.739782
	IV	0.048327	0.024367	0.014497
Running Time	I	121	23	20
	II	536	30	28
	III	1336	23	24
	IV	2062	16	16

From Table 10 we can observe that among the average running times (in seconds) of the four datasets, the standard deviation using standard GEP algorithm are significantly less than those for the GP algorithm. As the standard deviation when using the standard GEP algorithm are smaller than when using the GP algorithm, which indicates that it is excellent to apply the standard GEP algorithm to ordinary differential equations and that its prediction effects and computation efficiency are better than the GP algorithm. In addition, it can be seen from Table 10 that, in different data sets, GP algorithm's running time fluctuate a lot, on the other hand, GEP algorithm's running time fluctuate a few.

Table 10 shows that for these four datasets, the minimum prediction errors generated by the improved GEP algorithm were smaller than those generated by the standard GEP algorithm and that the maximum prediction errors generated by the improved GEP algorithm were also smaller than those generated by the standard GEP algorithm. Thus, the improvement of the standard GEP algorithm was not only effective but also overcame the problems of instability of the standard GEP algorithm. In terms of average running time (in seconds), there is no significant difference between the improved GEP algorithm and the standard GEP algorithm. However, the running time is decreased. Therefore, in terms of running efficiencies, the improved GEP algorithm is better than the GP algorithm.

4.7. Summary of Experiment Results

The described experiment results can be summarized as follows:

- For each dataset, using the GEP algorithm to solve the inverse problem of ordinary differential equations are efficient, and is able to produce a better model. GEP is effective improvement strategies.
- In terms of running time, there was no significant difference between the improved GEP and the standard GEP. However, the running time of GEP is significantly less than that of the GP algorithm.
- In terms of the stability, adopting the improved GEP algorithm to solve the inverse problem of ordinary differential equations is better than using the GP algorithm, particularly for complex problems.
- In terms of prediction accuracy, the improved GEP algorithm are better than standard GEP on the error of the model, while the GEP algorithm is superior to the GP algorithm on the standard deviation.
- As the constant items increase, the use of GEP algorithm is effective, and the overall effect is still better than the GP algorithm.

5. Conclusions

In this study, we proposed a GEP based algorithm to solve the inverse problem of ordinary differential equations, and GEP algorithm's shortcoming, evolution operation in the recessive segment contributes little is overcome. In view of the effectiveness of the improved GEP algorithm, this paper mainly discusses its application to practical problems, although the effect is good, but there is no theoretical issues in-depth discussion, so many problems need further study. Through experiments, we verified that the algorithm, in terms of solving the inverse problem of ordinary differential equations, was very effective with respect to training standard deviation, prediction standard deviation and running time. And improvement strategies can further enhance the efficiency of the algorithm convergence. The work of this paper is only a start, the problem of solving despite the engineering background, but the practical application of the project there is still a great distance. Our study on algorithms provides a powerful tool to automate the process of knowledge discovery for dynamic data, and we expect the approach to be promoted and applied in actual problems related to time series and time fields, such as weather forecasting, market prediction and ecological prediction.

Acknowledgements

This work is supported by the National Natural Science Foundation of China with the Grant No. 61573157, the Science and Technology Planning Project of Guangdong Province with the Grant No. 2017A010101037, the Fund of Natural Science Foundation of Guangdong Province of China with the Grant No. 2014A030313454, the Science and Technology Planning Project of Guangdong Province of China with the Grant No. 2015A020209108, the Science and Technology Research Project of Jiangxi Province with the Grant No. GJJ160631, the Science Foundation of Jiangxi University of Science and Technology under the grant No. NSFJ2015-K13.

6. References

[1] R. O. Alan L. Jennings, Optimal inverse functions created via population-based optimization, Cybernetics, IEEE Transactions on 44 (6) (2014) 950–965.

[2] F. O.-T. Riemann Ruiz-Cruz, Edgar N. Sanchez, Particle swarm optimization for discrete-time inverse optimal control of a doubly fed induction generator, Cybernetics, IEEE Transactions on 43 (6) (2013) 1698–1709.

[3] V.G.Romanov, Inverse Problems of Mathematical Physis, Netherlands, VNU Science Press BV, 1987.

[4] C. W. Groetsch, Inverse problems in the mathematical sciences, Vol. 52, Springer, 1993.

[5] V. Isakov, Inverse problems for partial differential equations, Vol. 127, Springer, 2006.

[6] J. L. Guofeng Feng, Bo Han, Widely convergent generalized pulse-spectrum methods for 2-d wave equation inversion, Chinese Journal of Geophysics 46 (2) (2003) 265–270.

[7] C. Ferreira, Gene expression programming: A new adaptive algorithm for solving problems, Complex Systems 13 (2) (2001) 87–129.

[8] C. Ferreira, Gene Expression Programming: Mathematical Modeling by an Artificial Intelligence, Springer-Verlag, 2006.

[9] Q. S. X. D. Jun Li, Mengchu Zhou, Colored traveling salesman problem, Cybernetics, IEEE Transactions on PP (99) (2014) 1.

[10] M. J. K. C. T. Su Nguyen, Mengjie Zhang, Automatic programming via iterated local search for dynamic job shop scheduling, Cybernetics, IEEE Transactions on 45 (1) (2014) 1–14.

[11] C. Ferreira, Combinatorial optimization by gene expression programming: inversion revisited, in: Proceedings of the Argentine symposium on Artificial Intelligence, 2002, pp. 160–174.

[12] R. T. Kaikuo Xu, Yintian Liu, A novel method for real parameter optimization based on gene expression programming, Applied Soft Computing 9 (2) (2009) 725–737.

[13] L. K. Dazhi Jiang, Zhijian Wu, Parameter identifications in differential equations by gene expression programming, in: Natural Computation, 2007. ICNC 2007. Third International Conference on, Vol. 5, IEEE, 2007, pp. 644–648.

[14] X. Wen, L. Shao, Y. Xue, W. Fang, A rapid learning algorithm for vehicle classification, Elsevier, Information Sciences 295 (1) (2015) 395–406.

[15] J. S. Z. F. X. L. N. L. Qi Liu, Weidong Cai, A speculative approach to spatial-temporal efficiency with multi-objective optimization in a heterogeneous cloud environment, Wiley Online Library, Security and Communication Network 9 (17) (2016) 4002–4012.

[16] L.Teodorescu, Gene expression programming approach to event selection in high energy physics, Nuclear Science, IEEE Transactions on 53 (4) (2006) 2221–2227.

[17] D. X. Q. W. H. Z. Yuhui Zheng, Byeungwoo Jeon, Image segmentation by generalized hierarchical fuzzy c-means algorithm, IOS Press, Journal of Intelligent & Fuzzy Systems 28 (2) (2015) 961–973.

[18] T. C. Mills, Time Series Techniques for Economists, Cambridge University Press, 1990.

[19] D. B. Percival, A. T. Walden., Spectral Analysis for Physical Applications, Cambridge University Press, 1993.

[20] S. W. Sudhakar M.Pandit, Time Series and System Analysis with Applications, John Wiley and Sons, 1983.

[21] W.F.Lucas, Differential Equation Models, New York: Springer-Verlag, 1983.

[22] Y. C. Hongqing Cao, Lishan Kang, The evolutionary modeling of higher-order ordinary differential equations for time-series analysis, Mini-Micro System (4) (2000) 344–349.

[23] H. L. Theodore W.Cornforth, Inference of hidden variables in systems of differential equations with genetic programming, Genetic Programming and Evolvable Machines 14 (2) (2013) 155–190.

[24] Y. Ikeda, Estimation of chaotic ordinary differential equations by coevolutional genetic programming, Electronics and Communications in Japan (Part III Fundamental Electronic Science) 86 (2) (2003) 1–12.

[25] T. G. Y. C. Hongqing Cao, Lishan Kang, A two-level hybrid evolutionary algorithm for modeling one-dimensional dynamic systems by higher-order ode models, IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 30 (2) (2000) 351–357.

[26] Y. Zhang, X. Sun, B. Wang, Efficient algorithm for k-barrier coverage based on integer linear programming, IEEE, China Communications (7) (2016) 16–23.

[27] L. K. Aimin Zhou, Hongqing Cao, The automatic modeling of complex functions based on genetic programming, Journal of System Simulation (6) (2003) 797–799.

[28] Q. M. Y. Z. A. A. Yuehui Chen, Bin Yang, Time-series forecasting using a system of ordinary differential equations, Information Sciences 181 (2011) 106114.

[29] M. G. Juan J. Flores, System identification using genetic programming and gene expression programming, Computer and Information Sciences - ISCIS 2005 3733 (2005) 503–511.

[30] C. L. C. Y. A. C. Jie Zuo, Changjie Tang, Time series prediction based on gene expression programming, Advances in Web-Age Information Management 3129 (2004) 55–64.

[31] K. Z. Z. D. Fangyu Li, Shouqian Sun, Improved gene expression programming algorithm for solving inverse problems in partial differential equations, Journal of Zhejiang University (Engineering Science) 43 (11) (2009) 2023–2027.

[32] C. Ferreira, Gene expression programming: a new adaptive algorithm for solving problems, Complex Systems 13 (2) (2001) 87–129.

[33] Q. L. L. Z. J. S. Zhangjie Fu, Xingming Sun, Achieving efficient cloud search services: multi-keyword ranked search over encrypted cloud data supporting parallel computing, The Institute of Electronics, Information and Communication Engineers, IEICE Transactions on Communications 98 (1) (2015) 190–200.

[34] L. K. Dazhi Jiang, Zhijian Wu, New method used in gene expression programming: Grcm, Journal of System Simulation (6) (2006) 1466–1468.

[35] H.Md.Azamathulla, Gene-expression programming to predict friction factor for southern italian rivers, Neural Computing and Applications 23 (5) (2013) 1421–1426.
[36] B. Z. T. M. Yu Xue, Jiongming Jiang, A self-adaptive artificial bee colony algorithm based on global best for global optimization, Springer, Soft Computing (2017) 1–18.
[37] C. Ferreira, Gene expression programming in problem solving, Soft Computing and Industry: Recent Applications (2002) 635–653.
[38] Y. L. Wenyin Gong, Zhihua Cai, Automatic modeling of complex functions based on gene expression programming, Journal of System Simulation 18 (6) (2006) 1450–1454.
[39] Z. C. Chong Li, Qiong Gu, Hyperspectral remoye sensing image classification based on de and gep, Microelectronics and Computer 29 (11) (2012) 103–106.
[40] X. L. Xianfeng He, Mining and application of reservoir landslide risk model based on gep, Yellow River 34 (3) (2012) 91–94.
[41] Z. H. Z. D. Dong, Prediction of gases content dissolved in power transformer oil based on gep sliding window model, Journal of North China Electric Power University 39 (4) (2012) 42–46.
[42] Z. D. Zhongmin Niu, Jitao Niu, Transformer fault diagnosis based on multi-gep classifier and dga, Journal of North China Electric Power University 39 (3) (2012) 35–40.
[43] T. L. Mengwei Liu, Xia Li, A gene expression programming algorithm for population prediction problem, Acta Scientiarum Naturalium Universitatis Sunyatseni 49 (6) (2010) 115–120.
[44] X. Qian, Prediction in stock-price index based on improved gene expression programming, Computer Engineering 35 (5) (2009) 200–202.
[45] Y. Wang, An improved gene expression programmingmethod and application, Journal of Qingdao Agricultural University (Natural Science) 26 (3) (2009) 242–245.
[46] Y. C. Lishan Kang, Hongqing Cao, The evolutionary modeling algorithm for system of ordinary differential equations, Chinese Journal of Computers-Chinese Edition 22 (1999) 871–876.
[47] Y. C. Hongqing Cao, Lishan Kang, A hybrid evolutionary modeling algorithm for dynamic system, Journal of Computer Research and Development (8) (1999) 923–931.