Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2014, Article ID 915058, 16 pages
http://dx.doi.org/10.1155/2014/915058

Research Article

Component Thermodynamical Selection Based Gene Expression Programming for Function Finding

Zhaolu Guo,[1] Zhijian Wu,[2,3] Xiaojian Dong,[4] Kejun Zhang,[5] Shenwen Wang,[6] and Yuanxiang Li[2,3]

[1] *School of Science, JiangXi University of Science and Technology, Ganzhou 341000, China*
[2] *State Key Laboratory of Software Engineering, Wuhan University, Wuhan 430072, China*
[3] *Computer School, Wuhan University, Wuhan 430072, China*
[4] *State-Owned Assets Supervision and Administration of Jiangxi Province, Nanchang 330006, China*
[5] *College of Computer Science and Technology, Zhejiang University, Hangzhou 310027, China*
[6] *School of Information Engineering, Shijiazhuang University of Economics, Shijiazhuang 050031, China*

Correspondence should be addressed to Zhaolu Guo; gzl@whu.edu.cn

Received 8 June 2013; Accepted 8 December 2013; Published 16 January 2014

Academic Editor: Wei-Chiang Hong

Gene expression programming (GEP), improved genetic programming (GP), has become a popular tool for data mining. However, like other evolutionary algorithms, it tends to suffer from premature convergence and slow convergence rate when solving complex problems. In this paper, we propose an enhanced GEP algorithm, called CTSGEP, which is inspired by the principle of minimal free energy in thermodynamics. In CTSGEP, it employs a component thermodynamical selection (CTS) operator to quantitatively keep a balance between the selective pressure and the population diversity during the evolution process. Experiments are conducted on several benchmark datasets from the UCI machine learning repository. The results show that the performance of CTSGEP is better than the conventional GEP and some GEP variations.

1. Introduction

Gene expression programming (GEP) [1, 2], improved genetic programming (GP) with linear representation [3, 4], is an artificial problem solver inspired in natural genotype/phenotype system. GEP combines both the simple, linear string of chromosomes with fixed length to represent the solutions similar to the ones utilized in genetic algorithm (GA) and the ramified structures with different sizes and shapes similar to the parse trees of GP [3, 5, 6]. Thus, GEP has the advantages of both GA and GP, while overcoming some of their individual limitations [3, 4]. Because of its high performance, GEP has attracted increasing attention recently as an efficient and effective data mining approach. Moreover, it has been successfully applied to many fields, such as function finding [7–9], symbolic regression [10–13], parameter optimization [14], rule mining [15], classification [3, 16], time series forecasting [2], prediction of flow number of asphalt mixes [17], prediction of material load [18, 19], prediction of the strength of concrete [20], engineering design [21], and machine scheduling [22, 23].

Although GEP has been successfully employed in a variety of areas, in practical applications, it is found that the conventional GEP usually suffers from premature convergence and slow convergence rate resulting in poor solution quality and/or large computational cost [2–4]. The main reason is that the conventional GEP cannot quantitatively keep a balance between the selective pressure and the population diversity during the evolution process. Therefore, this may lead to trapping in the local optimum and/or slowing down the search speed.

In general, increasing selective pressure and promoting population diversity in GEP are often in conflict with each other [3, 4]. This means that increasing selective pressure may lead to more individuals being close to the best individual, and then the average fitness of the population is better.

Hence, this can accelerate the convergence speed of the population. However, increasing selective pressure may result in an evolutionary state of which most of the individuals are approaching the best individual. As a result, the population diversity is significantly reduced after some generations, increasing the possibility of trapping into local optimum solutions. On the contrary, promoting population diversity can make the individuals distribute widely in the search space and increase the probability of finding the global optimum, but this may slow down the convergence speed.

To the best of our knowledge, there has been little research focusing on how to quantitatively balance the selective pressure and population diversity of GEP during the evolution process. Therefore, this motivates us to investigate a selection mechanism that can quantitatively keep a balance between the selective pressure and population diversity of GEP to enhance the global search ability and simultaneously to accelerate the convergence speed. Our work along this idea has produced a novel GEP based on component thermodynamical selection operator (CTS), called CTSGEP. This proposed approach, inspired by the principle of minimal free energy in thermodynamics, seeks to map the selective pressure and the population diversity into the mean energy and the entropy, respectively. In order to quantitatively balance the selective pressure and the population diversity of GEP, in the CTS, when selecting individuals for the next generation from the parent and offspring individuals, the selected individuals for the next generation should satisfy the principle of minimal free energy.

The rest of the paper is organized as follows. Section 2 introduces the notations and terminologies of GEP that are useful for the review of the previous works of GEP in Section 3. The proposed algorithm, CTSGEP, is elaborated in Section 4, with detailed explanations on the component thermodynamical selection operator. The computational results and comparisons are provided in Section 5. Finally, we end the paper with some conclusions in Section 6.

2. GEP Basic Concepts

2.1. Chromosomes Representation. The most innovative feature of GEP is the improved representation of chromosomes. GEP separates the genotype from the phenotype of the chromosomes [3], which is one of the greatest limitations of both GA [24, 25] and GP. In GEP, individuals are represented by linear strings and called chromosomes. In addition, the chromosomes consist of genes and link operators, in which the link operators connect the genes. The link operators usually can be arithmetic operators, such as +, −, ∗, and /. Moreover, the genes of GEP can be categorized into two types [2]: genotype and phenotype. The genotype is the code of genes similar to that used in GA and the genetic operators directly manipulate the genotype, while the phenotype is the decoding of the genes consisting of the same kind of ramified structures with different sizes and shapes similar to the parse trees of GP. For instance, the detailed transformation process of gene "$* - / + a\ a\ b\ c\ b$" can be shown in Figure 1. Hence, the merits are obvious to separate the genotype from phenotype

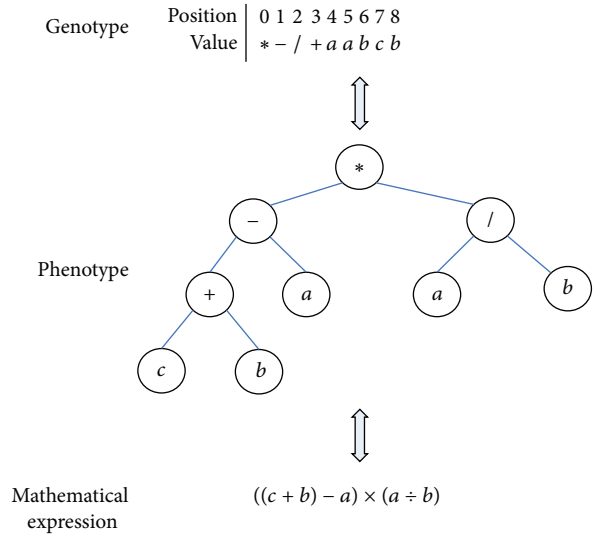

Figure 1: Transformation process of gene.

of the chromosomes. On the one hand, the representation of the chromosomes is simple and compact. Therefore, the genetic operators are easy to implement and very efficient. On the other hand, this mechanism makes GEP able to solve complex problems.

In GEP, each gene is composed of two parts: a head and a tail. The head contains functional symbols (e.g., +, −, ∗, /, etc.) and terminal symbols, but the tail contains only terminal symbols. Moreover, the length of the head h is, selected by the user, determined by specific problems, while the length of the tail t is a function of h and n. In addition, t should satisfy (1), which makes sure that any gene can be decoded to a correct mathematical expression, where n is the number of arguments for the function that takes the most arguments:

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

For example, we consider a gene composed of {+, −, ∗, /, Q, a, b}, where Q represents the square root function. In the set of functional symbols $F = \{+, -, *, /, Q\}$, n is 2. We assume h is 4; it can be concluded that $t = 4 \times (2 - 1) + 1 = 5$. Thus, the length of the gene is $4 + 5 = 9$.

2.2. Genetic Operators. There are many genetic operators in GEP, including selection operator, mutation operator, transposition-insertion operator, and recombination operator. These genetic operators should be subject to the following conditions. (1) The length of the head and that of the tail are subject to formula (1). (2) The tail contains only terminal symbols [2]. Moreover, these conditions ensure that the genetic operators can generate new genes that are decoded to correct mathematical expression. Therefore, these operators are simple and easy to implement. The detailed description of these operators can be referred in [1, 2].

2.3. Fitness Functions. Generally, different fitness functions are suitable for different problems. The choice of fitness functions is quite crucial for GEP. This is mainly because the fitness functions may directly affect the convergence speed and the solution quality. In GEP, there are many kinds of fitness functions: absolute error fitness function, relative error fitness function, and logic synthesis fitness function [1, 2]. They are described as follows:

$$f_i = \sum_{j=1}^{C_t} \left(\text{Max} - \left|C_{(i,j)} - T_{(j)}\right|\right), \tag{2}$$

$$f_i = \sum_{j=1}^{C_t} \left(\text{Max} - \left|\frac{C_{(i,j)} - T_{(j)}}{T_{(j)}} \cdot 100\right|\right), \tag{3}$$

$$\text{if } n \geq \frac{1}{2}C_t \quad \text{then } f_i = n; \quad \text{else } f_i = 1, \tag{4}$$

where Max is a constant, which determines the range of f_i, $C_{(i,j)}$ is the value calculated by the individual i for the sample instance j, $T_{(j)}$ is the target value for sample instance j, C_t is the total number of sample instances, and n is the number of sample instances correctly predicted. In general, fitness functions (2) and (3) are employed to solve function regression problems and fitness function (4) is applied to Boolean concept learning problems.

2.4. The Framework of GEP. The framework of GEP is similar to that of GA [26]. The major difference between GEP and GA is the representation of chromosomes. However, the essential idea of GEP is the same as the one of GA [2], which is based on the concepts of natural selection and survival of the fittest. The procedure of GEP is described in Algorithm 1.

3. Previous Work

In order to enhance the performance of the traditional GEP algorithm, many scholars recently have proposed several GEP variants. Moreover, these GEP variants can be classified into two categories: accelerating convergence speed and promoting population diversity.

3.1. Accelerating Convergence Speed. In order to accelerate the convergence speed of the traditional GEP, Karakasis and Stafylopatis [3] proposed a novel GEP for data mining tasks, which combined the principle inspired by the immune system, namely, the clonal selection principle. In the proposed algorithm, a receptor-editing step was added in order to achieve faster exploration of the antibody-antigen binding space. Experimental results showed that the proposed GEP variant outperformed the conventional GEP in terms of both prediction accuracy and computational efficiency. Zhang et al. [27] introduced an improved gene expression programming (IGEP), which employed a dynamic mutation operator to enhance the efficiency. The proposed algorithm can obtain better prediction results for the prediction of retention times for a larger set of pesticides than heuristic method. Further, IGEP as a nonlinear method had good generalized performance. By applying parallel taboo search, Rao et al. [28] presented an enhanced GEP to improve the local search ability of the conventional GEP. Wu et al. [29] proposed a parallel niche GEP based on general multicore processor to improve the evolution efficiency and the parallel model of niche GEP was designed by OpenMP. Based on analyzing the intelligibility and efficiency of expression-tree-based expression on GEP, Chen et al. [7] introduced a reduced GEP, of which the chromosomes were evaluated directly on the reduced gene without being expressed them into expression trees. Moreover, the result of the evolution by reduced GEP was simplified and easier to be understood and explained.

3.2. Promoting Population Diversity. For maintaining good population diversity of the conventional GEP, Jiang et al. [30] proposed an adaptive GEP algorithm based on cloud model. The proposed GEP algorithm employed an adaptive cloud strategy to determine the mutation and crossover rate dynamically to improve the population diversity. Li et al. [31] introduced an improved GEP (AMACGEP) by statistical analysis and critical velocity, which utilized statistical analysis of repeated bodies to enhance the diversity of the initial population. Moreover, it proposed a dynamic mutation operator to improve the diversity of individuals. Liu et al. [32] proposed a population diversity-oriented GEP (Mod-GEP) for function finding, in which two strategies including population updating and population pruning were used to increase the diversity of population. The experimental results showed that Mod-GEP can obtain more satisfactory solution than GP, GEP, and some other GEP variants. Zhang and Xiao [33] presented a population diversity strategy GEP (GEP-PDS). The presented GEP-PDS inherited the advantage of superior population producing strategy and various population strategies to maintain the diversity of population. Further, Zhang et al. [34] proposed an improved GEP based on block strategy (BS-GEP), in which the population was divided into several blocks according to the individual fitness of each generation and the genetic operators were reset differently in each block to preserve the population diversity. In addition, BS-GEP was also utilized in prediction of software failure sequence.

4. The Proposed CTSGEP Algorithm

4.1. Motivations. As pointed out in Section 3, some researchers have developed various GEP variants to improve the selective pressure in order to accelerate the convergence speed, whereas this may increase the possibility of trapping in local minima solutions [3, 27, 28]. Meanwhile, for the sake of decreasing the possibility of trapping in local minima solutions, many scholars have also attempted to encourage the population diversity during the evolution process. However, this may decelerate the searching speed [13, 31–33]. Therefore, a feasible solution to overcome these deficiencies of GEP cannot only improve one of the selective pressures or population diversities. Thus, a better approach is to keep a balance between the selective pressure and the population diversity during the evolution process. Actually, the essence

```
GEP Algorithm
Step 1  Initialize the parameters, t = 0, generate an initial population P(t);
Step 2  while (FES < Max_FES)
        {
            Evaluate population P(t);
            Save the best individual;
            Execute selection operator;
            Execute mutation operator;
            Execute transposition-insertion operator;
            Execute recombination operator;
            t = t + 1
        }
Step 3  Output the best individual.
```

Algorithm 1: Framework of GEP algorithm.

of reconciling the conflicts between the selective pressure and the population diversity is to solve a biobjectives optimization problem that can be formulated as follows.

In the parent population P_t of size N, M offspring individuals are created by GEP genetic operators. Hence, there are $M + N$ individuals in total. Further, the biobjectives optimization problem is to select N individuals from the parent and offspring individuals for the next generation population P_{t+1}, which make sure that the selective pressure measured by the average fitness AF and population diversity D of the next generation population P_{t+1} satisfy $\operatorname{Min} Y = (-\mathrm{AF}, D)$.

Notice that, in the above formulation, without loss of generality, we assume that the larger fitness value implies the better individual in GEP. In addition, the selective pressure can be measured by the average fitness AF.

Many existing approaches, such as evolutionary multiobjective optimization algorithms, can tackle the above biobjectives optimization problem. However, the solving process of this biobjectives optimization problem is executed for every generation of GEP. Therefore, the computational complexity of the solving process should be low. Otherwise, it may lead to very slow convergence speed of the overall GEP algorithm. Thus, approaches with high computational complexity (e.g., evolutionary multiobjective optimization algorithms) may not be suitable. Furthermore, it is unrealistic to obtain the accurate solution of the biobjectives optimization problem, because the computational complexity is $O(C_{N+M}^{N})$.

Based on the above considerations, we present a novel method, called CTS, to obtain the approximation solution of the above biobjectives optimization problem with very low computational complexity. Its primary idea is inspired by the principle of minimal free energy in thermodynamics. The principle of minimal free energy refers to [35, 36]; in the annealing process, a metal, starting with high temperature and disordered state, is gradually cooled in order that the system at any temperature approximately reaches thermodynamic equilibrium. This cooling process can be regarded as an adaptation procedure to achieve the stability of the final crystalline solid. In addition, any change from nonequilibrium to equilibrium of the system at each temperature follows the principle of minimum free energy. This means the system will change spontaneously to reach a lower total free energy and the system achieves equilibrium when its free energy seeks a minimum [36]. The free energy F is defined by

$$\operatorname{Min} F = E - T \cdot H, \tag{5}$$

where E is the mean energy of the system and H is the entropy. According to the principle of minimal free energy, we can know that any change of the system can be viewed as a result of the competition between the mean energy and the entropy, and the temperature T determines their relative weights in the competition [36]. In other words, the two objectives, namely, the mean energy and the entropy, are in conflict with each other, and the temperature T is the weight between the mean energy and the entropy. Moreover, the final objective can be converted into the minimal free energy. Thus, this is similar to the relationship between the selective pressure and the population diversity addressed before. Therefore, we can solve the above biobjectives optimization problem according to the principle of minimal free energy.

4.2. Basic Concepts of Component Thermodynamical Selection Operator. In order to utilize the principle of minimal free energy to reconcile the conflicts between the selective pressure and the population diversity, we should first map the selective pressure and the population diversity into the mean energy and the entropy, respectively. According to the characteristics of GEP and our previous works in [36, 37], we give the following definitions.

Definition 1. Let S be the search space; for any GEP individual $X_r \in S$, its fitness value is $F(X_r)$ and the characteristic of the fitness value is that the larger fitness value indicates that the individual is better. The absolute energy $e(X_r)$ of individual X_r is defined by

$$e\left(X_r\right) = -F\left(X_r\right). \tag{6}$$

Definition 2. Let $P_t = \{X_1, X_2, \ldots, X_N\} \in S^N$ be the GEP population of generation t. The absolute energy window W_t is defined as follows.

(1) When $t = 0$, $W_t = [l_0, u_0]$, where

$$l_0 = \min\{e(X_r) \mid X_r \in P_0\},$$
$$u_0 = \max\{e(X_r) \mid X_r \in P_0\}. \quad (7)$$

(2) When $t > 0$, $W_t = [l_t, u_t]$, where

$$l_t = \min(l_{t-1}, \min\{e(X_r) X_r \in O_t\}),$$
$$u_t = \min(u_{t-1}, \max\{e(X_r) \mid X_r \in O_t\}), \quad (8)$$

where $O_t = \{X_{N+1}, X_{N+2}, \ldots, X_{N+M}\} \in S^M$ is the offspring population.

Definition 3. Let $P_t = \{X_1, X_2, \ldots, X_N\} \in S^N$ and $W_t = [l_t, u_t]$ be the GEP population of generation t and the absolute energy window, respectively. For any GEP individual $X_r \in S$, its normalization energy $e'(W_t, X_r)$ within the absolute energy window W_t is defined by

$$e'(W_t, X_r) = \frac{l_t - e(X_r)}{l_t - u_t}. \quad (9)$$

Definition 4. The ith rank β_t^i in the absolute energy window $W_t = [l_t, u_t]$ is defined by

$$\beta_t^i = \left(\frac{a^{i-1} - 1}{a^{K-1} - 1} \cdot (u_t - l_t) + l_t, \frac{a^i - 1}{a^{K-1} - 1} \cdot (u_t - l_t) + l_t\right] \cap [l_t, u_t], \quad (10)$$

where $a > 1$, $i = 0, 1, \ldots, K-1$, and $K \geq 2$. a is a scaling factor, K is the number of ranks, and if $e(X_r) \in \beta_t^i$, it denotes that X_r is located in the rank β_t^i.

Definition 5. Let $P_t = \{X_1, X_2, \ldots, X_N\} \in S^N$ and $W_t = [l_t, u_t]$ be the GEP population of generation t and the absolute energy window, respectively. Moreover, the number of individuals located in the ith rank β_t^i is n_i. The rank entropy $H(W_t, P_t)$ is defined as follows

$$H(W_t, P_t) = -\sum_{i=0}^{K-1} \frac{n_i}{|P_t|} \log_K \frac{n_i}{|P_t|}. \quad (11)$$

Definition 6. Let $P_t = \{X_1, X_2, \ldots, X_N\} \in S^N$ and $W_t = [l_t, u_t]$ be the GEP population of generation t and the absolute energy window, respectively. The free energy $F(W_t, T, P_t)$ is defined as follows:

$$F(W_t, T, P_t) = E(W_t, P_t) - T \cdot H(W_t, P_t), \quad (12)$$

where T is the temperature and $E(W_t, P_t)$ is the mean energy, which is defined by:

$$E(W_t, P_t) = \frac{1}{N} \sum_{X_r \in P_t} e'(W_t, X_r). \quad (13)$$

Definition 7. Let $P_t = \{X_1, X_2, \ldots, X_N\} \in S^N$ and $W_t = [l_t, u_t]$ be the GEP population of generation t and the absolute energy window, respectively. For any GEP individual $X_r \in S$ located in the rank β_t^i where the number of the individuals is n_i, the component free energy $F_c(W_t, T, P_t, X_r)$ of individual X_r is defined by

$$F_c(W_t, T, P_t, X_r) = e'(W_t, X_r) + T \log_K \frac{n_i}{|P_t|}. \quad (14)$$

From the above definitions, we can obtain the following conclusion and the proof can be referenced in our previous work [36, 37]:

$$F(W_t, T, P_t) = \frac{1}{N} \sum_{X_r \in P_t} F_c(W_t, T, P_t, X_r). \quad (15)$$

As we know, our objective is the minimal free energy. Therefore, according to this conclusion, we can calculate the free energy by computing the mean of the component free energy of every individual in the population. Hence, the minimal free energy can be approximatively obtained by the minimal component free energy of every individual in the population. Next, we will present the component thermodynamical selection operator of GEP based on this conclusion.

4.3. Component Thermodynamical Selection Operator of GEP. Based on the definitions in Section 4.2, we will introduce the component thermodynamical selection operator (CTS) of GEP. The main idea of CTS is to pick M individuals, the component free energy of the picked individuals are the M largest ones from the parent and offspring population, and then eliminate the M individuals. Further, it can be proved that the remaining individuals approximately satisfy the principle of minimal free energy. The proof is similar to our previous work [36, 37]. The pseudocode of CTS operator is presented in Algorithm 2.

In the CTS of GEP, we first calculate the component free energy of the $N + M$ individuals of parent and offspring population, and then eliminate the M largest component free energy individuals to compose the next generation population. Using this method, we can select individuals for the next generation with very low computational cost and the computational complexity is $O((N + M) \cdot M)$. Furthermore, the process of computing the component free energy of each individual in the temporary population P'_{t+1} is shown in Algorithm 3, where K is the number of ranks, NR is an array which recorded the number of individuals in each rank, and P'_{t+1} is the temporary population.

4.4. Algorithm Description of the Proposed CTSGEP. Similar to the traditional GEP, CTSGEP starts with initializing a population of N individuals. Then at each temperature T, it evolves LK generations. At each generation, M new individuals are created by the uniform selection, mutation, transposition-insertion, and recombination operators, and then select N individuals from the $M + N$ individuals for the next generation using CTS. This process is repeated until

Component thermodynamical selection operator of GEP
Step 1 Combine offspring population O_t with parent population P_t to generate a temporary population P'_{t+1};
Step 2 Compute the component free energy of each individual in population P'_{t+1};
Step 3 Pick the M largest component free energy individuals from population P'_{t+1};
Step 4 Eliminate the M picked individuals from population P'_{t+1} to generate the population P_{t+1} for the next generation.

Algorithm 2: Pseudocode of CTS operator.

```
The process of computing the component free energy of each individual
  Step 1 /* initialize the number of individuals in each rank and compute rank β_t^i */
      for (i = 0; i < K; i + +)
      {
            NR[i] = 0; /* initialize the number of individuals in each rank */
            Compute rank β_t^i according to (10)
      }
  Step 2 /* Compute the number of individuals in each rank and obtain the rank of
each individual */
      for (i = 0; i < |P'_{t+1}|; i + +)
      {
          for (j = 0; j < K; j + +)
          {
              if (e(P'_{t+1}[i] ∈ β_t^i)
              {
                  NR[j] + +;
                  P'_{t+1} [i] .Rank = j;
                  break;
              }
          }
      }
  Step 3 /* Compute the component free energy of each individual */
      for (i = 0; i < |P'_{t+1}|; i + +)
      {
          n_i = NR[P'_{t+1} [i] .Rank];
          Compute the component free energy of individual P'_{t+1} [i]
          according to (9) and (14);
      }
```

Algorithm 3: Process of computing the component free energy of each individual.

the termination criterion is reached. The CTSGEP algorithm description is summarized in Algorithm 4.

5. Numerical Experiments

5.1. Experimental Setup. In order to evaluate the performance of our proposed CTSGEP algorithm for function finding, in this section we compare CTSGEP algorithm with the traditional GEP and some GEP variations on the function finding data sets, including IGEP [27], AMACGEP [31], and Mod-GEP [32]. In addition, all of the compared algorithms are implemented with C++ program language.

The function finding datasets are taken from the UCI machine learning repository [38]. There are about 200 test instances for the function finding problems in UCI [38], and we randomly select 15 test instances, which are instances 10, 21 35, 44, 49, 52, 76b, 84b, 103, 126a, 148c, 155, 163, 182c, and 203.

In our experimental studies, for each algorithm and each test instance, 30 independent runs are conducted with 400000 function evaluations (FES) as the termination criterion. To fairly compare the mentioned algorithms, the common parameter settings of all the algorithms, as used or recommended in [1, 2, 31], are shown as follows:

(i) head length: 20,
(ii) gene length: 41,
(iii) number of genes: 5,

```
GEP Based on Component Thermodynamical Selection
  Step 1  Create a random initial population P_0;
  Step 2  Evaluate the population P_0, and calculate the absolute energy of each individual
according to (6);
  Step 3  t = 0, k = 0, T = T0;
  Step 4  Compute the absolute energy window W_t according to (7);
  Step 5  while (FES < MAX_FES)
          {
              for (i = 0; i < LK; i + +)
              {
                  Create M new individuals by the uniform selection, mutation,
              transposition-insertion and recombination operator;
                  Establish the offspring population O_t by the M new individuals;
                  Evaluate the population O_t, and calculate the absolute energy of each
              individual according to (6);
                  Save the best individual;
                  Compute the absolute energy window W_{t+1} according to (8);
                  Utilize CTS operator to select N individuals from P_t ∪ O_t for the next generation;
                  t = t + 1;
              }
              k = k + 1;
              T = T0/(k + 1);
          }
  Step 6  Output the best individual.
```

Algorithm 4: Pseudocode of CTSGEP algorithm.

(iv) linking function: +,

(v) function set: +, −, ∗, /, pow, sqrt, sin, cos, log, and exp,

(vi) population size: 100,

(vii) mutation probability: 0.08,

(viii) one-point recombination rate: 0.3,

(ix) two-point recombination rate: 0.3,

(x) gene recombination rate: 0.3,

(xi) IS transposition rate: 0.1,

(xii) RIS transposition rate: 0.1,

(xiii) gene transposition rate: 0.1.

In addition, the other parameter values of IGEP [27], AMACGEP [31], and Mod-GEP [32] are the same as their original papers. K, M, $T0$, a, and LK in CTSGEP are set to 20, 20, 10, 2, 100, respectively. In our experiments, as recommended in [2], the average and standard deviation of the mean square error (MSE) are recorded for measuring the performance of each algorithm. The mean square error Err is calculated by [2]

$$\mathrm{Err} = \frac{1}{\mathrm{SN}} \sum_{i=0}^{\mathrm{SN}} \left(y^i - c^i\right)^2, \tag{16}$$

where y^i is the target value for sample i, c^i is the predicted value by the algorithms for sample i, and SN is the total number of samples in each dataset.

5.2. Comparison between CTSGEP and Other GEP Algorithms. The mean and the standard deviation of the MSE obtained by each algorithm for 15 test instances are summarized in Table 1. All the results are obtained from 30 independent runs. In addition, the best results among the five algorithms are marked in boldface. In order to have statistically sound conclusions, two-tailed t-test at a 0.05 significance level is conducted on the experimental results. The last three rows of Table 1 summarize the experimental results.

Clearly, CTSGEP is the best among the five algorithms on the 15 test instances. It performs significantly better than GEP, IGEP, AMACGEP, and Mod-GEP on fifteen, fourteen, thirteen, and ten test instances according to the two-tailed t-test, respectively. In addition, GEP cannot outperform CTSGEP on any test instance, while IGEP, AMACGEP, and Mod-GEP only surpass CTSGEP on one, one, and three test instances, respectively.

To compare the performance of these algorithms on the 15 test instances, the average ranking of the Friedman test is conducted by the suggestions considered in [39, 40]. Table 2 reports the average ranking of the five GEP algorithms on the 15 test instances. These GEP algorithms can be sorted by the average ranking into the following order: CTSGEP, Mod-GEP, AMACGEP, IGEP, and GEP. Thus, the best average ranking is obtained by the CTSGEP algorithm, which outperforms the other four GEP algorithms.

To compare the performance differences between CTSGEP and the other four GEP algorithms, we conduct a Wilcoxon signed-ranks test [41, 42] with a significance level equal to 0.05. Table 3 shows the resultant P values when comparing between CTSGEP and the other four GEP

Table 1: Experimental results of GEP, IGEP, AMACGEP, Mod-GEP, and CTSGEP over 30 independent runs for the 15 test instances.

Instance	GEP	IGEP	AMACGEP	Mod-GEP	CTSGEP
	Mean MSE ± Std Dev	Mean MSE ± Std Dev	Mean MSE ± Std Dev	Mean MSE ± Std Dev	Mean MSE ± Std Dev
10	$2.36E-06 \pm 3.17E-07-$	$1.40E-06 \pm 2.71E-06-$	$1.24E-06 \pm 9.38E-07-$	$4.07E-07 \pm 7.21E-08-$	**$9.61E-08 \pm 6.92E-09$**
21	$4.47E-04 \pm 1.06E-04-$	$7.70E-04 \pm 8.03E-05-$	$2.82E-04 \pm 4.33E-05-$	$4.56E-05 \pm 6.51E-06-$	**$9.59E-06 \pm 5.78E-07$**
35	$8.24E-03 \pm 2.51E-03-$	$1.83E-03 \pm 4.28E-04-$	$2.41E-03 \pm 5.08E-04-$	$1.04E-03 \pm 8.02E-04 \approx$	**$8.71E-04 \pm 2.07E-05$**
44	$1.13E-04 \pm 5.26E-05-$	$6.56E-05 \pm 1.82E-05-$	$3.52E-05 \pm 2.67E-06-$	$5.35E-05 \pm 5.11E-05-$	**$1.94E-05 \pm 9.69E-07$**
49	$8.85E-02 \pm 6.13E-03-$	$7.55E-02 \pm 2.61E-02-$	**$4.17E-02 \pm 1.93E-04+$**	$6.69E-02 \pm 1.48E-03-$	$5.22E-02 \pm 3.19E-03$
52	$5.57E-02 \pm 4.31E-02-$	$5.61E-03 \pm 2.57E-03-$	$7.54E-03 \pm 4.12E-04-$	$1.07E-03 \pm 1.21E-04-$	**$6.90E-04 \pm 2.63E-05$**
76b	$6.13E-06 \pm 2.87E-06-$	$4.22E-06 \pm 8.61E-07-$	$3.56E-06 \pm 9.47E-07-$	$2.71E-06 \pm 6.23E-07-$	**$1.01E-06 \pm 5.36E-07$**
84b	$9.59E-03 \pm 2.86E-03-$	$1.14E-03 \pm 6.35E-04-$	$2.71E-03 \pm 6.87E-04-$	**$6.82E-04 \pm 9.15E-06+$**	$7.89E-04 \pm 1.95E-05$
103	$1.85E-06 \pm 4.27E-07-$	$8.59E-08 \pm 6.29E-08-$	$6.58E-08 \pm 1.83E-09-$	$3.75E-08 \pm 5.18E-08-$	**$2.72E-09 \pm 6.16E-09$**
126a	$5.10E-04 \pm 3.68E-05-$	**$2.39E-05 \pm 7.53E-06+$**	$6.38E-05 \pm 4.79E-04 \approx$	$6.78E-05 \pm 1.63E-06-$	$5.10E-05 \pm 1.96E-06$
148c	$7.92E-03 \pm 5.15E-04-$	$9.15E-03 \pm 6.72E-04-$	$5.78E-03 \pm 7.61E-05-$	**$1.72E-03 \pm 3.14E-04+$**	$2.77E-03 \pm 6.52E-05$
155	$4.66E-02 \pm 2.90E-04-$	$3.77E-02 \pm 1.35E-04-$	$2.78E-02 \pm 9.22E-03-$	$1.66E-02 \pm 3.48E-02 \approx$	**$9.11E-03 \pm 4.62E-06$**
163	$1.49E-04 \pm 1.73E-04-$	$1.16E-04 \pm 2.58E-06-$	$8.10E-05 \pm 1.66E-06-$	$5.82E-05 \pm 6.15E-06-$	**$3.40E-05 \pm 2.83E-05$**
182c	$7.97E-05 \pm 3.82E-06-$	$1.08E-06 \pm 6.74E-07-$	$6.36E-06 \pm 3.52E-07-$	$4.95E-07 \pm 5.18E-07-$	**$9.25E-08 \pm 2.54E-10$**
203	$3.66E-03 \pm 6.47E-04-$	$1.26E-03 \pm 3.75E-04-$	$1.01E-03 \pm 3.87E-05-$	**$3.67E-04 \pm 1.19E-06+$**	$7.73E-04 \pm 1.54E-04$
−	15	14	13	10	
+	0	1	1	3	
≈	0	0	1	2	

"Mean MSE" and "Std Dev" indicate the average and standard deviation of the mean square error values obtained in 30 independent runs, respectively.
Two-tailed t-test at a 0.05 significance level is conducted between CTSGEP and each of GEP, IGEP, AMACGEP, and Mod-GEP.
"+", "−", "≈" denote that the performance of the corresponding algorithm is better than, worse than and similar to that of CTSGEP according to the two-tailed t-test, respectively.
The best results among the five algorithms are typed in bold.

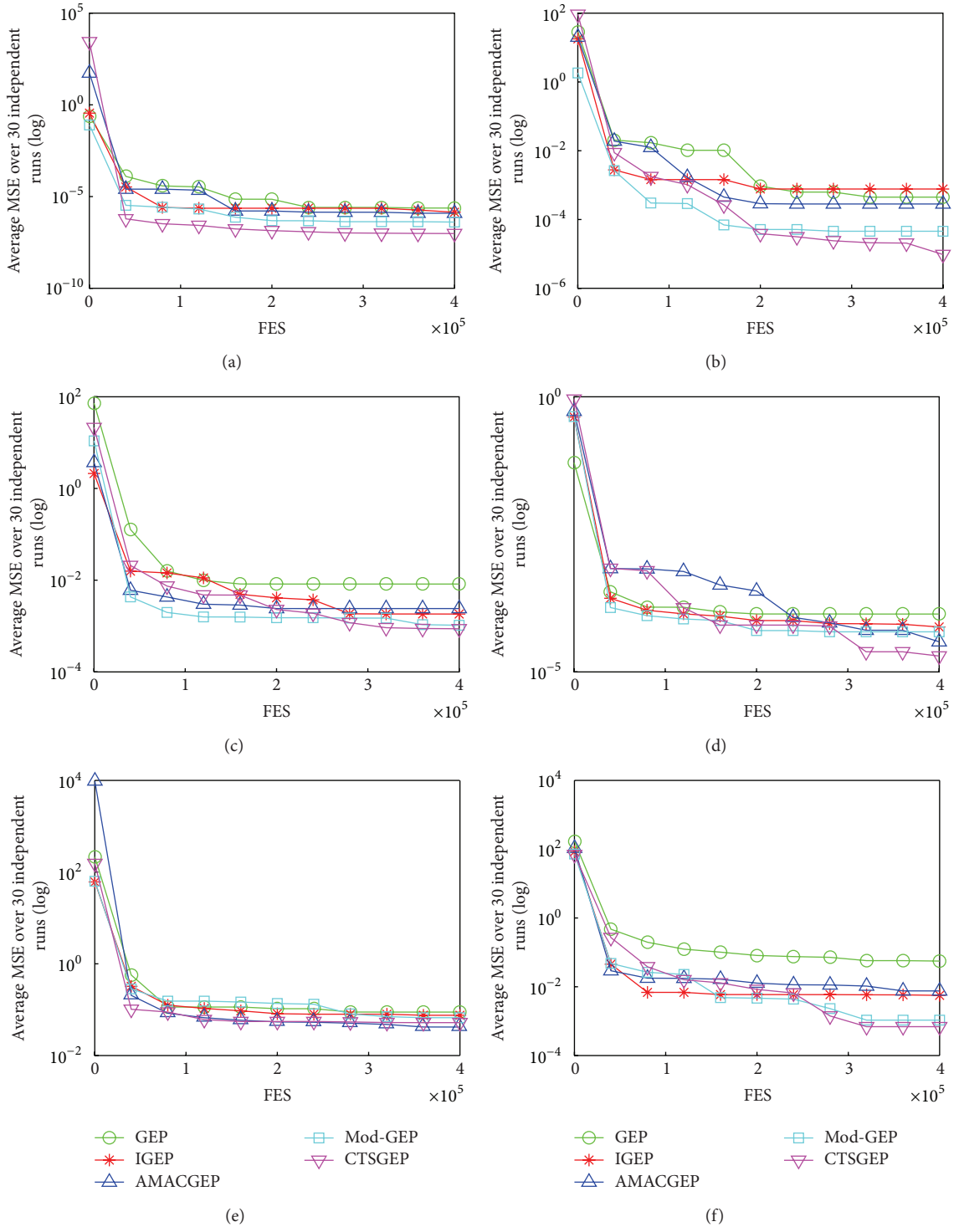

Figure 2: Evolution of the MSE derived from GEP, IGEP, AMACGEP, Mod-GEP, and CTSGEP versus the number of FES on six test instances. (a) Instance 10. (b) Instance 21. (c) Instance 35. (d) Instance 44. (e) Instance 49. (f) Instance 52.

algorithms. The P values below 0.05 are typed in bold. From the results, it can be observed that CTSGEP is significantly better than GEP, IGEP, and AMACGEP algorithms. Besides, CTSGEP is not significantly better than Mod-GEP. However, CTSGEP performs better than Mod-GEP according to the average rankings shown in Table 2.

In summary, CTSGEP is the winner on these 15 test instances. This can be because CTSGEP could quantitatively

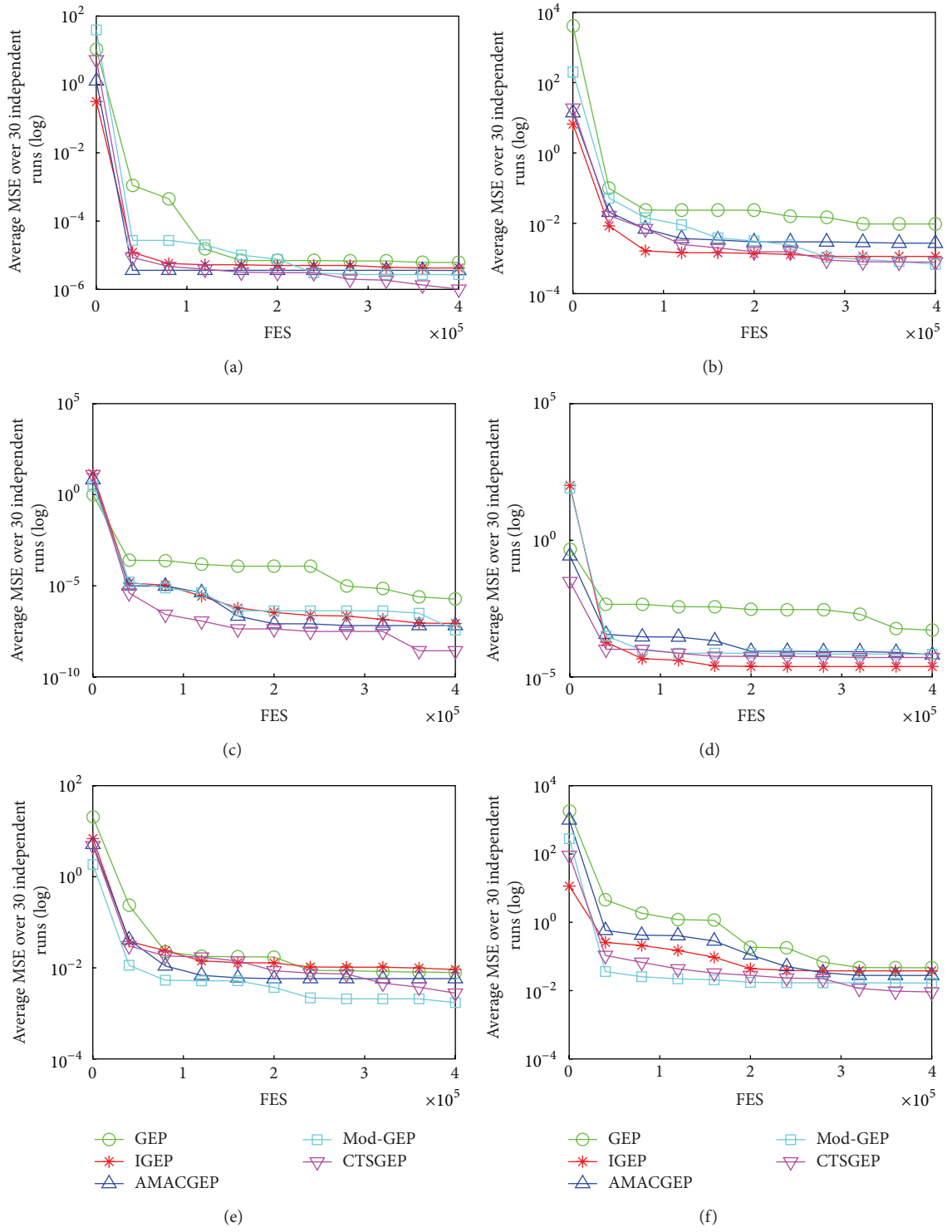

FIGURE 3: Evolution of the MSE derived from GEP, IGEP, AMACGEP, Mod-GEP, and CTSGEP versus the number of FES on six test instances. (a) Instance 76b. (b) Instance 84b. (c) Instance 103. (d) Instance 126a. (e) Instance 148c. (f) Instance 155.

keep a balance between the selective pressure and the population diversity during the evolution process, whereas IGEP only employs a dynamic mutation operator to enhance the convergence speed, while AMACGEP and Mod-GEP merely maintain the diversity of population.

For the convenience of illustration, the evolution of the mean MSE derived from GEP, IGEP, AMACGEP, Mod-GEP, and CTSGEP versus the number of FES is plotted in Figures 2, 3, and 4 for some typical test instances. From Figures 2, 3, and 4, it is clear that CTSGEP exhibits faster and more

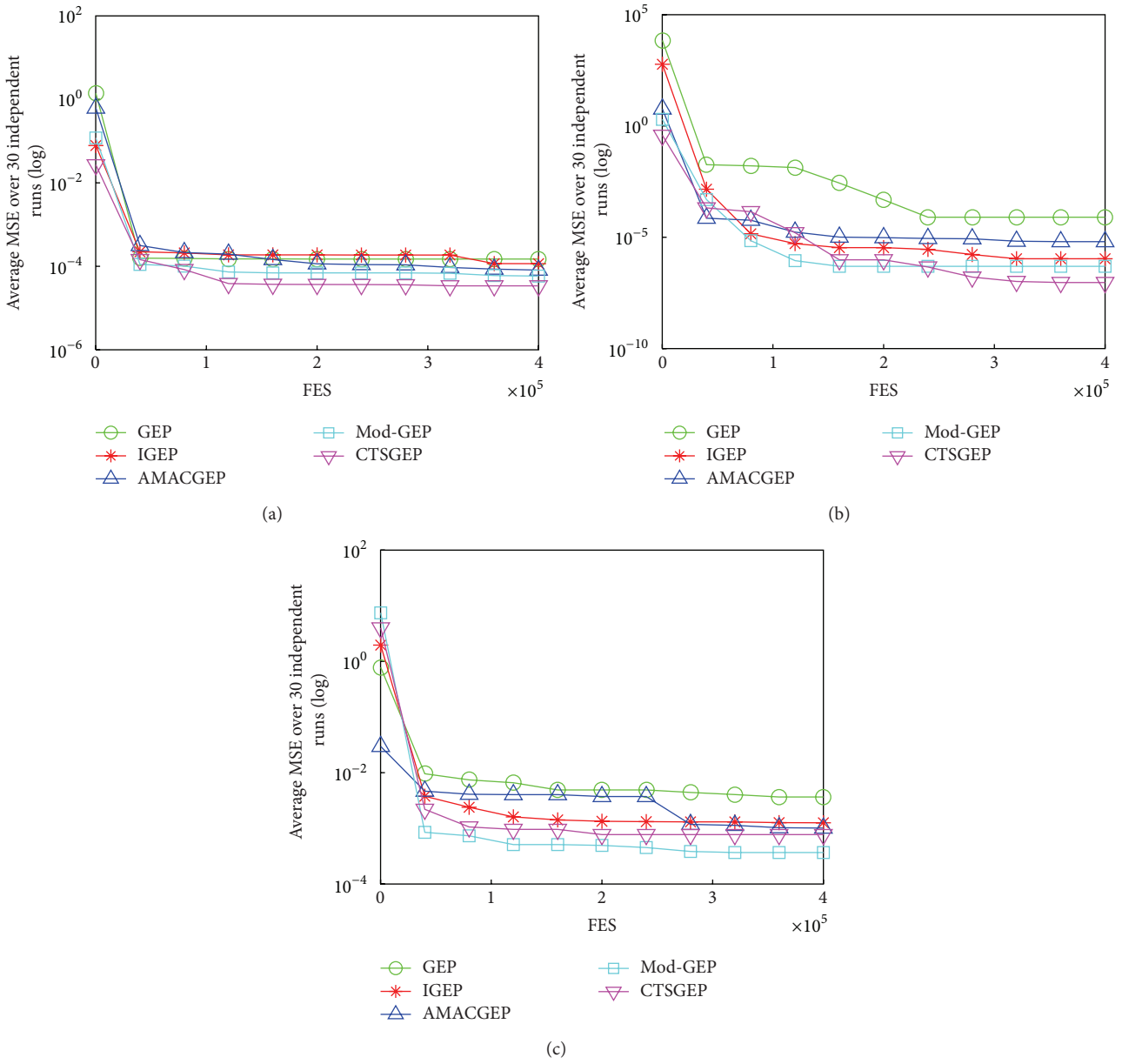

Figure 4: Evolution of the MSE derived from GEP, IGEP, AMACGEP, Mod-GEP, and CTSGEP versus the number of FES on three test instances. (a) Instance 163. (b) Instance 182c. (c) Instance 203.

Table 2: Average Rankings of the five GEP algorithms for the 15 test instances achieved by Friedman test.

Algorithm	Ranking
CTSGEP	4.67
Mod-GEP	3.93
AMACGEP	2.93
IGEP	2.33
GEP	1.13

Table 3: Wilcoxon test between CTSGEP and the other four GEP variations for the 15 test instances.

CTSGEP versus	P values
Mod-GEP	0.1398
AMACGEP	**0.0090**
IGEP	**0.0018**
GEP	**0.0007**

The P values below 0.05 are typed in bold.

stable convergence, for it can obtain a compromise between the selective pressure and the population diversity.

5.3. Parameter Sensitivity Study. In this section, we conduct a series of experiments to study the two important parameters of CTSGEP, which are the offspring population size M and

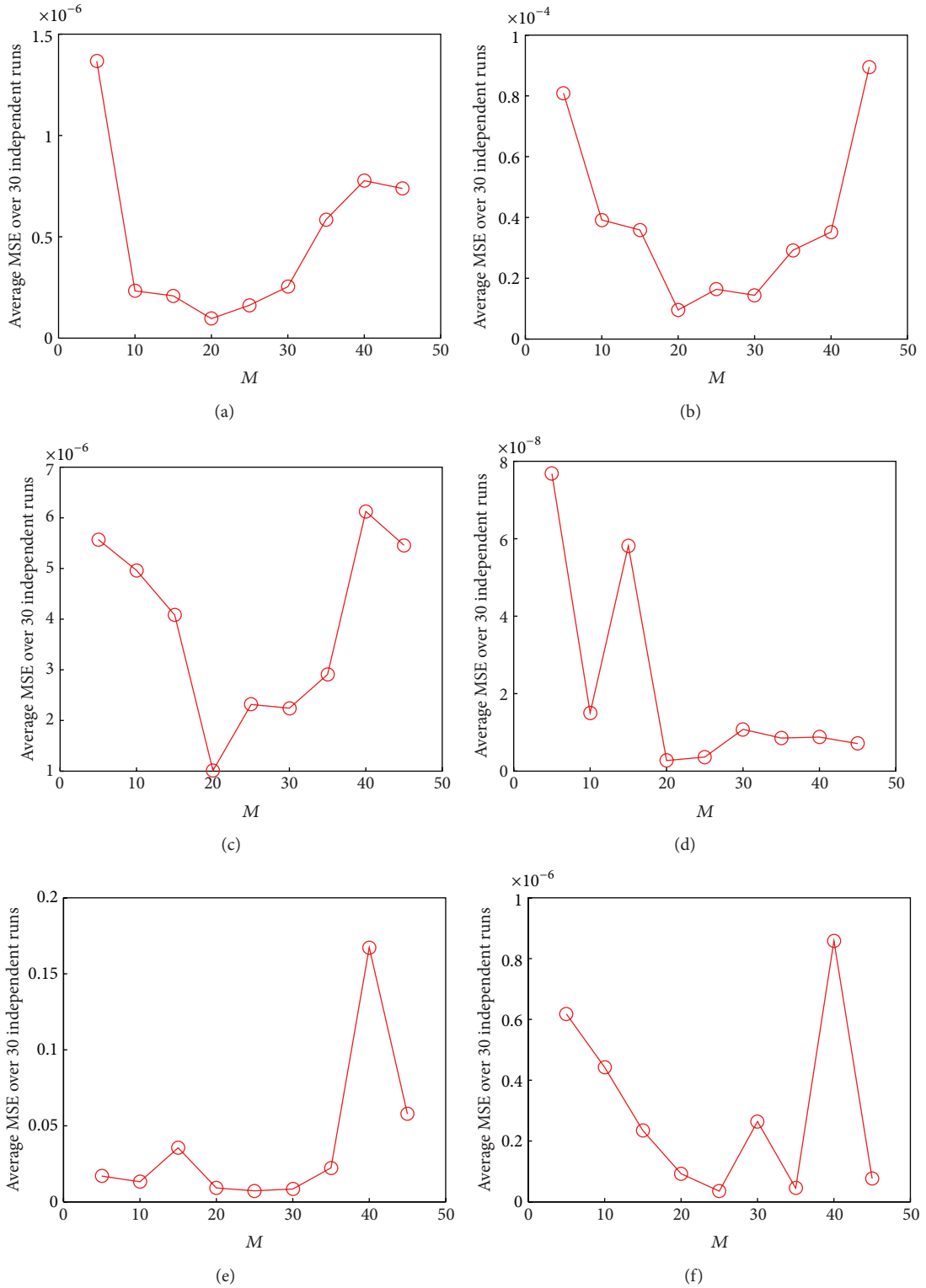

Figure 5: Average MSE over 30 independent runs with different offspring population size M values on six test instances. (a) Instance 10. (b) Instance 21. (c) Instance 76b. (d) Instance 103. (e) Instance 155. (f) Instance 182c.

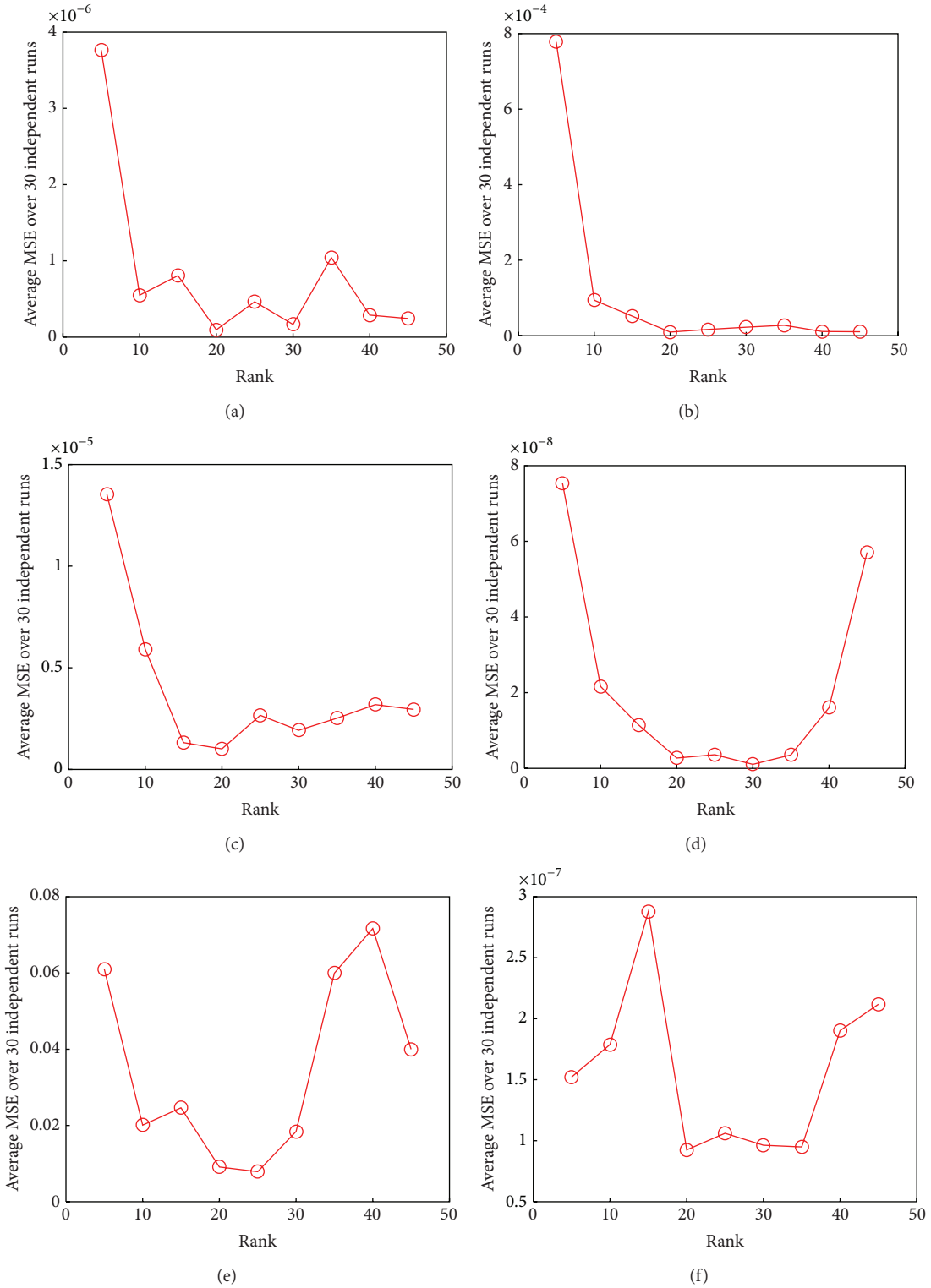

Figure 6: Average MSE over 30 independent runs with different number of ranks K values on six test instances. (a) Instance 10. (b) Instance 21. (c) Instance 76b. (d) Instance 103. (e) Instance 155. (f) Instance 182c.

the number of ranks K. The former is related to the selective pressure, while the latter is correlated with the population diversity.

5.3.1. Sensitiveness to Offspring Population Size M. An experiment is conducted to investigate the sensitivity of CTSGEP algorithm to variations in offspring population size M based on the 15 test instances described in Section 5.1 over 30 independent runs. Obviously, the offspring population size M is related to population size N. Therefore, we set M which varies from $N*5\%$ to $N*50\%$ with a step equal to 5 in the experiment. In addition, all the other parameters of CTSGEP are the same as those in Section 5.1. Results for some typical test instances, reported in Figure 5, show that the performance of CTSGEP changes with offspring population size M. Here, we omit plots for all other test instances as they exhibit a similar behavior. The X-coordinate of each plot in Figure 5 represents the offspring population size M, while the Y-coordinate stands for the average MSE over 30 independent runs. It can be easily seen from Figure 5 that CTSGEP performs best when the offspring population size M is selected in the range $[N * 15\%, N * 30\%]$.

5.3.2. Sensitiveness to Number of Ranks K. The impact of the number of ranks K is investigated using the 15 test instances described in Section 5.1 over 30 independent runs. We fix the parameters of CTSGEP the same as those in Section 5.1 except that K ranges from $N*5\%$ to $N*50\%$ with a step of 5. The results for some typical test instances are shown in Figure 6. Here, we also omit results for all other test instances since they show the similar tendency as well. In the figure, it is clear that CTSGEP works best with the number of ranks $K \in [N * 15\%, N * 35\%]$.

6. Conclusion

GEP is an increasingly popular tool for data mining. However, it tends to suffer from premature convergence and slow convergence rate when solving complex problems. Aiming at this drawback of GEP, we present a novel GEP based on the component thermodynamical selection operator. CTSGEP, proposed in this paper, is inspired by the principle of minimal free energy in thermodynamics, which maps the selective pressure and the population diversity into the mean energy and the entropy, respectively. Further, due to the chosen individuals for the next generation satisfying the principle of minimal free energy, the proposed approach can quantitatively keep a balance between the selective pressure and population diversity of GEP.

The experimental studies in this paper were conducted on 15 test instances of function finding problems taken from the UCI machine learning repository. CTSGEP was compared with the conventional GEP and three GEP variations, that is, IGEP, AMACGEP, and Mod-GEP. The experimental results demonstrated that its overall performance was better than the four competitors. Moreover, the parameters sensitivity study of CTSGEP was also experimentally investigated.

In the future, we will perform more detailed evaluation of CTSGEP for the large scale data-mining problems, which is considered as a challenge by the data mining community. In addition, it is also interesting to study how to incorporate parameter adaptation schemes to CTSGEP.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of the paper.

Acknowledgments

The authors would like to thank anonymous reviewers for their detailed and constructive comments which help them to improve the quality of this work significantly. This work was supported in part by the National Natural Science Foundation of China (nos. 61070008, 61303137, and 61364025), by Startup Foundation for Ph.D. of JiangXi University of Science and Technology (no. jxxjbs13028), and by the Natural Science Youth Foundation of Hebei Educational Committee (no. QN20131053).

References

[1] C. Ferreira, "Gene expression programming: a new adaptive algorithm for solving problems," *Complex Systems*, vol. 13, no. 2, pp. 87–129, 2001.

[2] C. Ferreira, *Gene Expression Programming*, Mathematical Modeling by an Artificial Intelligence, Springer, Berlin, Germany, 2nd edition, 2006.

[3] V. K. Karakasis and A. Stafylopatis, "Efficient evolution of accurate classification rules using a combination of gene expression programming and clonal selection," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 6, pp. 662–678, 2008.

[4] C. Zhou, W. Xiao, T. M. Tirpak, and P. C. Nelson, "Evolving accurate and compact classification rules with gene expression programming," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 6, pp. 519–531, 2003.

[5] A. H. Gandomi and A. H. Alavi, "A new multi-gene genetic programming approach to nonlinear system modeling. Part I: materials and structural engineering problems," *Neural Computing and Applications*, vol. 21, no. 1, pp. 171–187, 2012.

[6] A. H. Gandomi and A. H. Alavi, "Multi-stage genetic programming: a new strategy to nonlinear system modeling," *Information Sciences*, vol. 181, no. 23, pp. 5227–5239, 2011.

[7] Y. Chen, C. J. Tang, R. Li, M. F. Zhu, C. Li, and J. Zuo, "Reduced-GEP: improving gene expression programming by gene reduction," in *Proceedings of the 2nd International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC '10)*, pp. 176–179, Jiangsu, China, August 2010.

[8] J.-L. Zheng, C.-J. Tang, K.-K. Xu, N. Yang, L. Duan, and H.-J. Li, "Gene expression programming evolution difficulty prediction based on posture model," *Journal of Software*, vol. 22, no. 5, pp. 899–913, 2011.

[9] M. Zhu, C. Tang, S. Dai, Y. Chen, S. Qiao, and Y. Xiang, "Naïve gene expression programming based on genetic neutrality," *Computer Research and Development*, vol. 47, no. 2, pp. 292–299, 2010.

[10] E. A. Colbourn, S. J. Roskilly, R. C. Rowe, and P. York, "Modelling formulations using gene expression programming—a comparative analysis with artificial neural networks," *European Journal of Pharmaceutical Sciences*, vol. 44, no. 3, pp. 366–374, 2011.

[11] A. Mollahasani, A. H. Alavi, and A. H. Gandomi, "Empirical modeling of plate load test moduli of soil via gene expression programming," *Computers and Geotechnics*, vol. 38, no. 2, pp. 281–286, 2011.

[12] S. S. S. Hosseini and A. H. Gandomi, "Short-term load forecasting of power systems by gene expression programming," *Neural Computing and Applications*, vol. 21, no. 2, pp. 377–389, 2012.

[13] J.-J. Hu, C.-J. Tang, L. Duan, J. Zuo, J. Peng, and C.-A. Yuan, "Strategy for diversifying initial population of gene expression programming," *Chinese Journal of Computers*, vol. 30, no. 2, pp. 305–310, 2007.

[14] K. Xu, Y. Liu, R. Tang, J. Zuo, J. Zhu, and C. Tang, "A novel method for real parameter optimization based on gene expression programming," *Applied Soft Computing Journal*, vol. 9, no. 2, pp. 725–737, 2009.

[15] T. Zeng, C. Tang, Y. Xiang, P. Chen, and Y. Liu, "A model of immune gene expression programming for rule mining," *Journal of Universal Computer Science*, vol. 13, no. 10, pp. 1484–1497, 2007.

[16] J. L. Ávila, E. L. Gibaja, A. Zafra, and S. Ventura, "A gene expression programming algorithm for multi-label classification," *Journal of Multiple-Valued Logic and Soft Computing*, vol. 17, no. 2-3, pp. 183–206, 2011.

[17] A. H. Gandomi, A. H. Alavi, M. R. Mirzahosseini, and F. M. Nejad, "Nonlinear genetic-based Models for prediction of flow number of asphalt mixtures," *Journal of Materials in Civil Engineering*, vol. 23, no. 3, pp. 248–263, 2010.

[18] F. Özcan, "Gene expression programming based formulations for splitting tensile strength of concrete," *Construction and Building Materials*, vol. 26, no. 1, pp. 404–410, 2012.

[19] S. M. Mousavi, P. Aminian, A. H. Gandomi, A. H. Alavi, and H. Bolandi, "A new predictive model for compressive strength of HPC using gene expression programming," *Advances in Engineering Software*, vol. 45, no. 1, pp. 105–114, 2012.

[20] A. H. Gandomi, S. K. Babanajad, A. H. Alavi, and Y. Farnam, "Novel approach to strength modeling of concrete under triaxial compression," *Journal of Materials in Civil Engineering*, vol. 24, no. 9, pp. 1132–1143, 2012.

[21] L. Gao, M. Xiao, X. Shao, P. Jiang, L. Nie, and H. Qiu, "Analysis of gene expression programming for approximation in engineering design," *Structural and Multidisciplinary Optimization*, vol. 46, no. 3, pp. 399–413, 2012.

[22] A. Baykasoğlu and M. Göçken, "Gene expression programming based due date assignment in a simulated job shop," *Expert Systems with Applications*, vol. 36, no. 10, pp. 12143–12150, 2009.

[23] L. Nie, X. Shao, L. Gao, and W. Li, "Evolving scheduling rules with gene expression programming for dynamic single-machine scheduling problems," *International Journal of Advanced Manufacturing Technology*, vol. 50, no. 5–8, pp. 729–747, 2010.

[24] Y. Xu and Z. Wang, "Genetic algorithm optimized CCEM for complex topology," *Mathematical Problems in Engineering*, vol. 2012, Article ID 383248, 14 pages, 2012.

[25] L. W. Qu, K. Chen, B. H. Li, J. S. Ma, and W. Tao, "The optimized transport scheme of empty and heavy containers with novel genetic algorithm," *Mathematical Problems in Engineering*, vol. 2013, Article ID 434125, 5 pages, 2013.

[26] Z. Miao, K. Fu, and F. Yang, "A hybrid genetic algorithm for the multiple crossdocks problem," *Mathematical Problems in Engineering*, vol. 2012, Article ID 316908, 18 pages, 2012.

[27] K. Zhang, S. Sun, and H. Si, "Prediction of retention times for a large set of pesticides based on improved gene expression programming," in *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation (GECCO '08)*, pp. 1725–1726, ACM, Atlanta, Ga, USA, July 2008.

[28] Y. Rao, R.-C. Wang, and C.-A. Yuan, "Improving gene expression programming using parallel taboo search," in *Proceedings of the IEEE 5th International Conference on Natural Computation (ICNC '09)*, pp. 144–148, Tianjin, China, August 2009.

[29] J. Wu, T. Li, B. Fang, Y. Jiang, Z. Li, and Y. Liu, "Parallel niche gene expression programming based on general multicore processor," in *Proceedings of the International Conference on Artificial Intelligence and Computational Intelligence (AICI '10)*, vol. 3, pp. 75–79, Sanya, China, October 2010.

[30] Y. Jiang, C.-J. Tang, H.-C. Zheng et al., "Adaptive gene expression programming algorithm based on cloud model," in *Proceedings of the IEEE 1st International Conference on BioMedical Engineering and Informatics (BMEI '08)*, vol. 1, pp. 226–230, Sanya, China, May 2008.

[31] K. Li, W. Pan, W. Zhang, and Z. Chen, "Automatic modeling of a novel gene expression programming based on statistical analysis and critical velocity," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '08)*, pp. 641–647, Hong Kong, June 2008.

[32] R. C. Liu, Q. F. Lei, J. Liu, and L. C. Jiao, "A population diversity-oriented gene expression programming for function finding," in *Simulated Evolution and Learning*, vol. 6457 of *Lecture Notes in Computer Science*, pp. 215–219, Springer, Berlin, Germany, 2010.

[33] Y. Q. Zhang and J. Xiao, "Population diversity strategy in gene expression programming," *Advanced Materials Research*, vol. 204–210, pp. 288–292, 2011.

[34] Y. Zhang, J. Xiao, and S. Sun, "BS-GEP algorithm for prediction of software failure series," *Journal of Software*, vol. 7, no. 1, pp. 243–248, 2012.

[35] N. Mori, J. Yoshida, H. Tamaki, H. Kita, and Y. Nishikawa, "Thermodynamical selection rule for the genetic algorithm," in *Proceedings of the IEEE International Conference on Evolutionary Computation (CEC '95)*, vol. 1, pp. 188–192, Perth, Australia, December 1995.

[36] W. Q. Ying, Y. X. Li, S. J. Peng, and W. W. Wang, "A steep thermodynamical selection rule for evolutionary algorithms," in *Computational Science*, vol. 4490 of *Lecture Notes in Computer Science*, pp. 997–1004, Springer, Berlin, Germany, 2007.

[37] W.-Q. Ying, Y.-X. Li, and P. C.-Y. Sheu, "Improving the computational efficiency of thermodynamical genetic algorithms," *Journal of Software*, vol. 19, no. 7, pp. 1613–1622, 2008.

[38] C. Blake, E. Keogh, and C. J. Merz, "Repository of machine learning databases," 2012, http://mlearn.ics.uci.edu/databases/function-finding/function-finding.data.

[39] S. García, A. Fernández, J. Luengo, and F. Herrera, "Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: experimental analysis of power," *Information Sciences*, vol. 180, no. 10, pp. 2044–2064, 2010.

[40] S. García and F. Herrera, "An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons," *Journal of Machine Learning Research*, vol. 9, pp. 2677–2694, 2008.

[41] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.

[42] S. García, D. Molina, M. Lozano, and F. Herrera, "A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the CEC'2005 special session on real parameter optimization," *Journal of Heuristics*, vol. 15, no. 6, pp. 617–644, 2009.