*Journal of Mathematical Research & Exposition*
Mar., 2009, Vol. 29, No. 2, pp. 237–246
DOI:10.3770/j.issn:1000-341X.2009.02.006
Http://jmre.dlut.edu.cn

# Triangle Evolution–A Hybrid Heuristic for Global Optimization

*LUO Chang Tong*[1,2], *YU Bo*[3]
(1. Institute of Mathematics, Jilin University, Jilin 130012, China;
2. Jilin Institute of Architecture and Civil Engineering, Jilin 130021, China;
3. Department of Applied Mathematics, Dalian University of Technology, Liaoning 116024, China)
(E-mail: c.t.luo@163.com)

**Abstract** This paper presents a hybrid heuristic–triangle evolution (TE) for global optimization. It is a real coded evolutionary algorithm. As in differential evolution (DE), TE targets each individual in current population and attempts to replace it by a new better individual. However, the way of generating new individuals is different. TE generates new individuals in a Nelder-Mead way, while the simplices used in TE is 1 or 2 dimensional. The proposed algorithm is very easy to use and efficient for global optimization problems with continuous variables. Moreover, it requires only one (explicit) control parameter. Numerical results show that the new algorithm is comparable with DE for low dimensional problems but it outperforms DE for high dimensional problems.

**Keywords** global optimization; evolutionary computation; differential evolution; simplex method.

**Document code** A
**MR(2000) Subject Classification** 80M50; 68W20
**Chinese Library Classification** O242.23

## 1. Introduction

Consider global optimization problem with continuous variables of the form

$$\min_{X\in D} f(X) \tag{1}$$

where $D \subset R^n$ and $f : D \longrightarrow R$ is of the black-box type, which may be discontinuous.

Direct search optimization algorithm is the first choice for such problems. A typical determinate direct search technique is simplex method which was first introduced by Spendley, Hext and Himsworth[8]. Nelder and Mead developed a modification to it that allows the procedure to adjust its search step according to the evaluation result of the new point generated[3]. Press presented an annealed Nelder and Mead strategy[5]. Evolutionary algorithm (EA) is a family of stochastic direct search algorithms. Varieties of evolutionary computational models that have been proposed and studied[11] include: genetic algorithm (GA), evolutionary programming (EP), evolution strategy (ES), genetic programming (GP), etc. However often the convergence speed of them is not fast.

**Received date**: 2007-03-19; **Accepted date**: 2007-03-24
**Foundation item**: the National Natural Science Foundation of China (No. 10671029).

Hybridizing EA with a traditional optimization algorithm is a good way to accelerate the convergence of original EA. There are two possible ways for designing hybrid algorithm: interweaving EA with traditional algorithm and introducing new operators inspired from traditional optimization algorithms. Several hybrid methods, such as GPL and simplex-GA[7], have been developed and show better behavior. GPL, a coupling of real-coded GA and Powell's method, is a representative of the first kind of hybridization. In GPL, a Powell's process for each individual in the current population is executed using the current individuals as the initial point and, the individual is replaced by the termination point found by the Powell's process. In Simplex-GA, the simplex operators are used as the crossover operator with a given probability. Consequently, Simplex-GA falls into the second kind of hybridization. Nevertheless, the reliability and the computing efficiency are not satisfactory yet.

In 1995, Rainer Storn and Kenneth Price broke through the framework of crossover, mutation and selection, and designed a new effective algorithm–differential evolution (DE)[9,10]. However, the convergence speed of DE becomes slow as the region of global minimum is approached. Let the optimization process be divided into two phases, global approach and local improvement. Then DE is fast in the phase of global approach but slow in the phase of local improvement. As is known, Nelder-Mead method is good at local searching. So we try to use the idea from Nelder-Mead method to accelerate the local improvement process of DE. To this end, we use the evolutionary operators for the explorative (global approach) process and simplex operators for the exploitative (local improvement) process. However, both interweaving EA with simplex search and introducing the simplex operators as the crossover operator turned out to be disappointing to attain an efficient algorithm, and we found that full dimensional simplex operators could slow down the evolutionary process remarkably.

In this paper, we will present a new heuristic–triangle evolution. Different from [7], it does not use full dimensional simplex, and uses 1 or 2 dimensional simplex operators: triangle reflection, triangle contraction and last struggle as the main search operators in the evolutionary process. It takes both advantages of strong local search ability of the simplex method and global convergence of evolutionary algorithms and, since only 1 or 2 dimensional simplex operators are used, the speed will not be slowed down as in [7].

The rest of this paper is organized as follows: two related algorithms, simplex-GA and DE, are briefly described in Section 2. In Section 3, our TE algorithm is described in detail and its relationships with simplex-GA and DE are also analyzed. Numerical experiments are presented in Section 4 and concluding remarks are made in Section 5.

## 2. Brief overviews of relevant algorithms

### 2.1. Simplex-GA method

Simplex-GA is a kind of real-coded evolutionary algorithm proposed by Renders and Bersini[7]. The population size is set to $\lambda \cdot (n+1)$. At each iteration (generation), every $n+1$ points form a mating group, and $\lambda$ mating groups are formed. Each group generates one individual, and the

$\lambda$ new indiviudals will replace the $\lambda$ worst individuals of the current population. Simplex-GA uses standard ranking selection scheme (proportional) for parents selection. Besides standard crossover operators (discrete crossover and average crossover), simplex-GA uses simplex crossover with a given probability $P_s$. After crossover, standard mutation operator is applied to produce a new generation.

Simplex crossover used in simplex-GA belongs to standard simplex operators (reflection, expansion or contraction). Suppose points $X_1, X_2$, and $X_3$ form an original simplex, and the point $X_1$ is the worst point. Point $\bar{X}$ represents the centroid of $X_2$ and $X_3$. Figure 1(a) illustrates how a new point is generated with the simplex crossover in $R^2$.

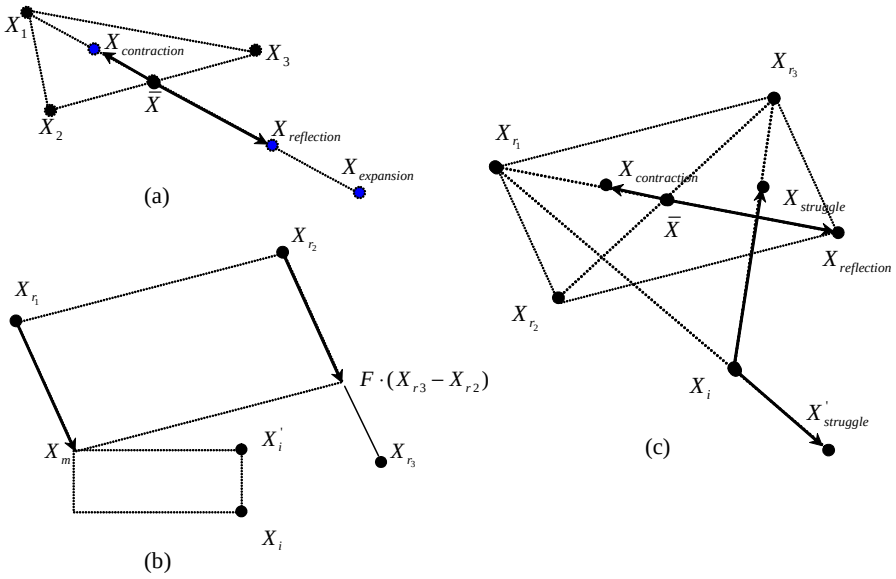


Figure 1 Generation of a new point in three Algorithms:
(a) Simplex-GA (if simplex crossover is used); (b) DE algorithm; (c) TE algorithm.

### 2.2. Differential evolution

Differential evolution (DE) was proposed by Storn and Price in 1995. It was proved to be the fastest evolutionary algorithm among the 1st ICEO conference entries[10]. Some modified versions of DE have been provided by Kaelo and Ali[1], in which localization techniques are combined.

DE is a population based algorithm. For each individual $X_i(t)$ (a point in $R^n$) in current population $\vec{X}(t)$, DE reproduces a trial point by mutation and crossover. In the mutation phase DE randomly selects three distinct points $X_{r_1}(t)$, $X_{r_2}(t)$, $X_{r_3}(t)$ from the current population $\vec{X}(t)$. These points should be different from $X_i(t)$. The mutated point $X_m(t)$ is a permutation of any of three point along the differential variation of the other two: $X_m(t) = X_{r_1}(t) + F \cdot (X_{r_2}(t) - X_{r_3}(t))$, where $F$ is a scaling factor. The trial point $X^{'}(t)$ is then generated by crossover

$x'_{i,j}(t) = \begin{cases} x_{m,j}(t), & \text{if rand}\,[0,1) < C_R \text{ or } j = j_{\text{rand}}; \\ x_{i,j}(t), & \text{otherwise,} \end{cases}$ $j = 1, 2, \ldots, n$. The way of generating a new individual in TE from the $i$-th individual and three randomly selected individuals is shown in Figure 1(b). The trial point replaces $X_i(t)$ from the current population if and only if it is not worse than it.

The outline of the differential evolution algorithm is as follows.

Procedure DE:

Step 1. Input population size $N$, initial bounds $lb$, $ub$, crossover probability $C_R$ and crossover factor $F \in (0, 1)$. Set $t = 0$; Initialize population $\vec{X}(0) = \{X_1(0), X_2(0), \ldots, X_N(0)\}$, where $X_i(0) = (x_{i,1}(0), x_{i,2}(0), \ldots, x_{i,n}(0))$;

Step 2. Repeat

1) Evaluate $\vec{X}(t)$: Compute $f(X_i(t))$ for each individual;

2) Reproduce $\vec{X}(t)$: For every $i \in \{1, 2, \ldots, N\}$, randomly generate three mutually different integers $r_1, r_2, r_3 \in \{1, 2, \ldots, N\}$ and $j_{\text{rand}} \in \{1, 2, \cdots, n\}$. Let

$$x'_{i,j}(t) = \begin{cases} x_{r_1,j}(t) + F \cdot (x_{r_2,j}(t) - x_{r_3,j}(t)), & \text{if rand}[0,1) < C_R \\ & \text{or } j = j_{\text{rand}}; \\ x_{i,j}(t), \text{ otherwise.} \end{cases}$$

3) Update $\vec{X}(t)$: $X_i(t+1) = \begin{cases} X'_i(t), & \text{if } f(X'_i(t)) \le f(X_i(t)); \\ X_i(t), & \text{otherwise.} \end{cases}$ until some stopping criterion is satisfied.

## 3. Triangle evolution (TE)

Motivated by DE and Simplex-GA, we designed a new real coded evolutionary algorithm–triangle evolution (TE). As in DE, it targets each individual in current population and attempts to replace it by a new better individual. But the method of generating a new individual is different, it generates a new individual with 1 or 2 dimensional simplex operators.

### 3.1. Evolutionary operators

For each individual in the current population $\vec{X}(t)$, TE selects three mutually different points (individuals) $X_{r_1}(t), X_{r_2}(t), X_{r_3}(t)$ from $\vec{X}(t)$ to form a 2-simplex (i.e., a triangle). Then the Nelder-Mead style evolutionary operators are based on this triangle. Let the three individuals satisfy $f(X_{r_1}(t)) > f(X_{r_2}(t)) > f(X_{r_3}(t))$. Then the Evolutionary operators introduced in TE can be described as follows.

1) Triangle reflection. Reflect the worst point across the centroid of the other two, $X_{\text{reflection}} = X_{r_2}(t) + X_{r_3}(t) - X_{r_1}(t)$.

2) Triangle contraction. Contract to the centroid of the three randmom selected parents, $X_{\text{contraction}} = \frac{1}{3}(X_{r_1}(t) + X_{r_2}(t) + X_{r_3}(t))$.

3) Last struggle. Step towards the best individual or away from the worst individual,

$$X_{\text{struggle}} = \begin{cases} X_i(t) + 0.618 \cdot (X_{r_3}(t) - X_i(t)), & \text{if } f(X_{r_3}(t)) < f(X_i(t)); \\ X_i(t) + 0.382 \cdot (X_i(t) - X_{r_1}(t)), & \text{else.} \end{cases}$$

For the $i$-th individual in current population $X_i(t)$, it has two chances to improve. The first chance is provided by triangle reflection. If the reflection point $X_{\text{reflection}}$ is better than $X_i(t)$, then $X_i(t)$ is replaced by $X_{\text{reflection}}$, thus $X_i(t)$ gets improved. If the reflection is not successful, the $i$-th individual $X_i(t)$ failed to get improved by the reflection operator, then triangle contraction is used trying to improve $X_i(t)$. If the contraction point $X_{\text{contraction}}$ is better than $X_i(t)$, then $X_i(t)$ is replaced by $X_{\text{contraction}}$. If the $i$-th individual $X_i(t)$ lost the previous two chances and it cannot receive average profit, that is, its function value is no less than the average value of the current population, it will take its last struggle, and $X_i(t)$ will be replaced by the struggle point $X_{\text{struggle}}$, regardless of their function values. Usually, not all of the three operators take effect to generate an acceptable new trial. In fact, the triangle contraction takes effect only if the triangle reflection failed to improve the quality of the $i$-th individual $X_i(t)$, and the last struggle takes effect only if the triangle reflection failed again. To keep track of the percent of each operator, we take the 17 problems in Section 4 to do our numerical experiments. Numerical results show that the success rates of the three operators are 43.7%, 46.6% and 38.6%, respectively. Here an operator is said to be successful if it generates a new trial better than the $i$-th individual $X_i(t)$.

The last struggle operator could also be regarded as a simplex operator, 1-dimensional reflection or 1-dimensional contraction. Note that the last struggle operator might degenerate the $i$-th individual, but it could increase the diversity of the population. The way of generating a new individual in TE from the $i$-th individual and three randomly selected individuals is shown in Figure 1(c).

### 3.2. Main procedure

The outline of the triangle evolution algorithm is as follows.

Procedure TE:

Step 1. Input population size $N$, initial bounds $lb$, $ub$. Set t = 0; Initialize population $\vec{X}(0) = \{X_1(0), X_2(0), \ldots, X_N(0)\}$, where $X_i(0) = (x_{i,1}(0), x_{i,2}(0), \ldots, x_{i,n}(0))$;

Step 2. Repeat

1) Evaluate $\vec{X}(t)$: Compute $f(X_i(t))$ for each individual;

2) Reproduce and update $\vec{X}(t)$: For every $i \in \{1, 2, \ldots, N\}$, randomly generate three mutually different integers $r_1, r_2, r_3 \in \{1, 2, \ldots, N\}$, $r_k \neq i, k = 1, 2, 3$. Rearrange $r_1, r_2, r_3$ to make that $f(X_{r_1}(t)) > f(X_{r_2}(t)) > f(X_{r_3}(t))$. Let $X_{\text{reflection}} = X_{r_2}(t) + X_{r_3}(t) - X_{r_1}(t)$.
If $f(X_{\text{reflection}}) < f(X_i(t))$, then $X_i(t+1) = X_{\text{reflection}}$
else

Let $X_{\text{contraction}} = \frac{1}{3}(X_{r_1}(t) + X_{r_2}(t) + X_{r_3}(t))$;

If $f(X_{\text{contraction}}) < f(X_i(t))$, then $X_i(t+1) = X_{\text{contraction}}$

else if $f(X_i(t)) \geq \frac{1}{N}\sum_i f(X_i(t))$ then

$$X_i(t+1) = \begin{cases} X_i(t) + 0.618 \cdot (X_{r_3}(t) - X_i(t)), & \text{if } f(X_{r_3}(t)) < f(X_i(t)) \\ X_i(t) + 0.382 \cdot (X_i(t) - X_{r_1}(t)), & \text{else.} \end{cases}$$

Until some stopping criterion is satisfied.

### 3.3. Similarities and differences with Simplex-GA and DE

TE shares similar idea in using heuristic information to Simplex-GA. Both of the algorithms have so-called reflection operator and contraction operator. But there are many differences between the two algorithms.

1) Most of the evolutionary operators in Simplex-GA are standard evolutionary operators, except that it uses the simplex crossover with a given probability $P_s$. While TE does not use other standard evolutionary operators to generate new individuals except the triangle reflection, triangle contraction and last struggle.

2) Simplex-GA uses full dimensional simplex. While TE uses 1 or 2 dimensional simplex. Thus the reflection and contraction of TE are relatively simpler than those of simplex-GA.

3) The simplex crossover in simplex-GA has three operators: reflection, expansion and contraction. But TE has only reflection and contraction. In TE, a trial point would not expand even though it succeeded in reflection.

4) The criteria to judge whether a reflection/contraction is successful is different. Simplex-GA uses the worst point in the simplex as the reference point, but TE uses the current individual $X_i(t)$.

5) Simplex-GA uses new individuals to replace the $\lambda$ worst individuals of the current population after all of the new $\lambda$ individuals are generated. But TE uses a new individual to replace the current individual $X_i(t)$.

6) TE introduces a new evolutionary operator, the last struggle operator.

TE shares similar method for parent selection with DE. Both of the algorithms produce a new individual from the $i$-th individual and three randomly selected mutually different parents. However, the reproduction process is quite different.

1) DE produces a new individual by perturbing one of the selected parents with the differential of the other two parents for some components of the vector. While TE produces a new individual by reflecting or contracting the worst parent to the centroid of other two parents or by stepping towards the best parent.

2) DE replaces the $i$-th individual with the new child only if it is worse than the child. Therefore, the average cost value of the population decreases steadily in DE. While the last struggle operator of TE may degenerate the $i$-th individual. Thus, the cost value of the population may fluctuate in TE.

3) DE does not take into consideration the parents' function value when creating a new trial. While TE utilizes these heuristic information in a Nelder-Mead way. Thus, TE is more reasonable.

The differences of generating of a new point in the three algorithms are shown in Figure 1. As can be seen from the above discussion, TE has some similarities to simplex-GA and DE, but it is far from either of them.

## 4. Numerical experiments

TE algorithm is implemented in C language and executed on a notebook PC with 800MHz AMD CPU and 224M memory. The main search engine of TE is in only about twenty lines of C codes. Because DE has outperformed other well known stochastic algorithms as already described, we compare numerical results of TE with DE. The source code of DE we use was downloaded from http://www.icsi.berkeley.edu/~ storn/. For convenience of comparison, we use the same notations as those in [10]. For all functions, an initial parameter range (IPR) and a value to reach (VTR) are defined. At the beginning of the optimization, the initial parameter values are drawn randomly from IPR. If a minimizer gets below the VTR, the problem is assumed to be solved. VTR is generally set to $f^* + 10^{-6}$, where $f^*$ is the global minimum of the objective function. The evolutionary process terminates if the current minimum becomes below VTR or the maximum number of function evaluations (which is set to $10 \cdot n^2$ in our test) is reached. The average number of function evaluations (nfe) over successful runs and the success rate (rate) are used as two criterions of the performance of the algorithms. The results were computed by averaging 100 trial runs for each test case. If the corresponding field for the average number of function evaluations contains a hyphen (−), it means that the global minimum could not be found in the 100 runs.

Control parameters of DE used in our experiments follow [10]. Several combinations of different parameters are carried out for each problem, and only the best results are listed in the following tables.

### 4.1. Low dimensional test problems

This set of test functions is from [10]. They are frequently used for testing the performance of stochastic algorithms. The test results are listed in Table 1.

1) Second De Jong function (Rosenbrock's saddle, $f_2(X)$ in [10])

$$f_1(X) = 100 \cdot (x_1^2 - x_2)^2 + (1 - x_1)^2; \quad \text{IPR} : x_j \in [-2.048, 2.048], \quad \text{VTR} : 10^{-6}.$$

2) Fifth De Jong function (Shekel's Foxholes, $f_5(X)$ in [10])

$$f_2(X) = \frac{1}{0.002 + \sum_{i=0}^{24} \frac{1}{(i+1) + \sum_{j=1}^{2} (x_j - a_{ij})^6}},$$

where $a_{i1} = \{-32, -16, 0, 16, 32\}$ for $i = 0, 1, 2, 3, 4$ and $a_{i1} = a_{i \bmod 5,1}$ as well as $a_{i2} = \{-32, -16, 0, 16, 32\}$ for $i = 0, 5, 10, 15, 20$ and $a_{i2} = a_{i+k,2}, k = 1, 2, 3, 4$. Note that $(i + 1)$ in the expression of $f_4(X)$ is mistakenly printed as $i$ in [10]. IPR: $x_j \in [-65.536, 65.536]$, VTR: $0.998004 + 10^{-6}$.

3) Griewangk's function (Griewangk ,1981, $f_7(X)$ in [10])

$$f_3(X)=\sum_{j=1}^{10}\frac{x_j^2}{4000}-\prod_{j=1}^{10}\cos(\frac{x_j}{\sqrt{j}})+1;\quad \text{IPR}: x_j\in[-400,400],\quad \text{VTR}: 10^{-6}.$$

4) Six-hump camel function ( $f_{20}(X)$ in [10])

$f_4(X)=(4-2.1\cdot x_1^2+\frac{1}{3}x_1^4)\cdot x_1^2+x_1\cdot x_2+(-4+4\cdot x_2^2)\cdot x_2^2$ ; Note that, $f_7(X)$ is mistakenly printed as $(4-2.1\cdot x_1^2+\frac{1}{3}x_1^4)+x_1\cdot x_2+(-4+4\cdot x_2^2)\cdot x_2^2$ in [10]. IPR: $x_j\in[-10,10]$, VTR: $-1.0316285+10^{-6}$.

5) Polynomial ($f_{26}(X)$ in [10])

$f_5(X)=0.25\cdot x_1^4-0.5\cdot x_1^2+0.1\cdot x_1+0.5\cdot x_2^2$; IPR: $x_j\in[-10,10]$, VTR: $-0.3523861+10^{-6}$.

Table 1 shows that TE has almost the same performance as DE for low dimensional problems.

### 4.2. High dimensional test problems

All problems in [10] are low dimensional. To test the performance of our algorithm for high dimensional problems, we choose some problems from other literatures. The chosen problems are all $n$-dimensional. The function family $f_6(X)$ is from [2]. $f_7(X)$ is the First De Jong function. And $f_8(X)$, $f_9(X)$ are from [4]. The test results are listed in Table 2.

1) (Goffin in §4.1, [2])

$f_6(X)=n\cdot\max_{j\in Q}x_j-\sum_{j\in Q}x_j$; $Q=\{1,2,\ldots,n\}$, IPR: $x_j\in[-1,1]$, VTR: $10^{-6}$

2) $f_7(X)=\sum_{j=1}^{n}x_j^2$; IPR: $x_j\in[-5,5]$, VTR: $10^{-6}$

3) (Ex. 6.15 in [4])

$f_8(X)=\max_{j\in Q}x_j^2$ ; $Q=\{1,2,\ldots,q\}, n=q$, IPR: $x_j\in[-1,1]$, VTR: $10^{-6}$

4) (Ex. 6.17 in [4])

$f_9(X)=\max_{j\in Q}(x_{4j-3}^2+x_{4j-2}^2+x_{4j-1}^2+x_{4j}^2)$ ; $Q=\{1,2,\ldots,q\}, n=4\cdot q$, IPR: $x_j\in[-1,1]$, VTR: $10^{-6}$

Table 1 Comparison of DE and TE in low dimensional cases

| func | problem | DE | | | | | TE | | |
|---|---|---|---|---|---|---|---|---|---|
| | | $F$ | $C_R$ | $N$ | $nfe$ | $rate$ | $N$ | $nfe$ | $rate$ |
| $f_1$ | Rosenbrock | 0.9 | 0.9 | 10 | 542 | 100% | 10 | 428 | 100% |
| $f_2$ | Foxholes | 0.5 | 0.1 | 40 | 1224 | 100% | 35 | 1546 | 100% |
| $f_3$ | Griewangk | 0.5 | 0.1 | 100 | 46385 | 100% | 1000 | 43457 | 100% |
| $f_4$ | Six-hump Camel | 0.5 | 0.1 | 20 | 1137 | 100% | 20 | 598 | 100% |
| $f_5$ | Polynomial | 0.5 | 0.1 | 20 | 669 | 100% | 30 | 773 | 100% |

Table 2 Comparison of DE and TE in high dimensional cases

| func | dimension | **DE** | | | **TE** | | |
|---|---|---|---|---|---|---|---|
| | | $N$ | $nfe$ | $rate$ | $N$ | $nfe$ | $rate$ |
| $f_6$ | $n=3$ | 50 | 134191 | 90% | 50 | 1536 | 100% |
| $f_6$ | $n=5$ | up to 100 | – | 0% | 50 | 2551 | 100% |
| $f_6$ | $n=20$ | up to 300 | – | 0% | 800 | 187235 | 100% |
| $f_7$ | $n=100$ | 120 | 180696 | 100% | 800 | 105347 | 100% |
| $f_7$ | $n=150$ | 200 | 293161 | 100% | 1000 | 131540 | 100% |
| $f_7$ | $n=200$ | up to 400 | – | 0% | 1200 | 149194 | 100% |
| $f_8$ | $n=20$ | 50 | 42785 | 100% | 500 | 57495 | 100% |
| $f_8$ | $n=40$ | 100 | 229409 | 100% | 1500 | 98576 | 100% |
| $f_8$ | $n=50$ | up to 200 | – | 0% | 1800 | 115427 | 100% |
| $f_9$ | $n=20$ | 50 | 27220 | 100% | 500 | 43723 | 100% |
| $f_9$ | $n=40$ | 50 | 69896 | 100% | 800 | 84270 | 100% |
| $f_9$ | $n=80$ | up to 300 | – | 0% | 1200 | 91591 | 100% |

From Table 2, we can see that TE outperforms DE for high dimensional problems. Therefore, TE is a good candidate for higher dimensional optimization.

### 4.3. Choice of control parameter

TE requires only one control variable, the population size $N$. Thus it is not difficult to choose in order to obtain a good result. According to our experience, a reasonable choice for $N$ is between $3n$ and $30n$, and a good first choice is $5n$.

## 5. Conclusions

Motivated by simplex-GA and differential evolution (DE), a new evolutionary algorithm–Triangle Evolution (TE) has been presented. TE selects parents and updates individuals in DE style, and the new individuals are generated in a Nelder-Mead way. The algorithm is very easy to implement. Its main search engine is in only about twenty lines of C codes. Furthermore, TE is very easy to use. It requires only one control variable, the population size $N$. Numerical experiments indicate that it is very efficient for global optimization problems with continuous variables. Comparison results show that it has a similar performance with DE for low dimensional problems and outperforms DE for high dimensional problems.

Hybridizing EA with some traditional optimization algorithm is a good way to improve the performance of the original EA. However, a simple combination of traditional optimization algorithm with EA might not work well. Usually some essential modifications need to be done for the original algorithms to get a successful hybridized algorithm. As a matter of fact, although TE could be regarded as a hybrid of simplex-GA and DE, it is far from either of the original algorithm.

Although TE has proved to be a kind of evolutionary computation model which is easy to

use, reliable and efficient, its working mechanism is still not clear yet, that is, we cannot tell why on earth TE works so well. TE is purely heuristic with no theoretical proof. The parameter $N$ of TE could be very large for some problems to converge to global minimum. This slows down TE's convergence rate. Further research is underway in developing the theoretical results and in developing more efficient TE requiring less population size.

## References

[1] KAELO P, ALI M M. *A numerical study of some modified differential evolution algorithms* [J]. European J. Oper. Res., 2006, **169**(3): 1176–1184.
[2] MÄKELÄ M M. *Nonsmooth Optimization: Analysis and Algorithms with Applications to Optimal Control* [M]. World Scientific, 1992.
[3] NELDER J A, MEAD R. *A simplex method for function minimization* [J]. Comp. J., 1965, **7**: 308–313.
[4] POLAK E, ROYSET J O, WOMERSLEY R S. *Algorithms with adaptive smoothing for finite minimax problems* [J]. J. Optim. Theory Appl., 2003, **119**(3): 459–484.
[5] PRESS W, TEUKOLSKY S, VETTERLING W. et al. *Numerical Recipes in FORTRAN* [M]. Second Edition, Cambridge University Press, Cambridge, 1992.
[6] PRICE K, STORN R. *Minimizing the real functions of the ICEC'96 contest by differential evolution* [C]. IEEE International Conference on Evolutionary Computation, 1996, 842–844.
[7] RENDERS J M, BERSINI H. *Hybridizing genetic algorithms with hill-climbing methods for global optimization: Two possible ways* [C]. Proceedings of the First IEEE Conference on Evolutionary Computation, 1994, 312–317.
[8] SPENDLEY W, HEXT G R, HIMSWORTH F R. *Sequential application of simplex designs in optimisation and evolutionary operation* [J]. Technometrics, 1962, **4**: 441–461.
[9] STORN R, PRICE K. *DE-a simple and efficient heuristic for global optimization over continuous space* [J]. J. Global Optim., 1997, **11**: 341–359.
[10] YAO X. *Evolutionary Computation: Theory and Applications* [M]. World Scientific, 1999.