International Journal of Innovative Computing, Information and Control
Volume **8**, Number **5(B)**, May **2012**

ICIC International ©2012 ISSN 1349-4198
pp. 3689–3703

A MINIMUM-ELIMINATION-ESCAPE MEMETIC ALGORITHM FOR GLOBAL OPTIMIZATION: MEEM

Lei Fan and Yuping Wang

School of Computer Science and Technology
Xidian University
No. 2, South Taibai Road, Xi'an 710071, P. R. China
lfan@mail.xidian.edu.cn; ywang@xidian.edu.cn

Received November 2010; revised March 2011

Abstract. *Smoothing function method and filled function method are two of the most efficient methods for global optimization problems. The former can eliminate many local minima during the optimization process, but it often loses some useful information on looking for descent directions. The later can escape from local minima and find a better minimum, but it is usually parameter sensitive. To overcome these shortcomings, an auxiliary function is designed which integrates the advantages of both smoothing function and filled function; that is, it not only can eliminate many local minima and escape from local minima, but also cannot lose the useful information and is not parameter sensitive. By using such a function, many local minima will be eliminated and the algorithm successively goes from one local minimum to another better local minimum during optimization process, and finds the global minimum finally. To enhance the efficiency of the algorithm, a local search called square search is designed and integrated into the algorithm. Based on these techniques, a minimum-elimination-escape memetic algorithm called MEEM is proposed in this paper. The simulations are made on 30 standard benchmark problems and the performance of the proposed algorithm is compared with that of some well performed existing algorithms. The results indicate the performance of the proposed algorithm is more effective.*

Keywords: Memetic algorithm, Minimum elimination, Smoothing function, Square search, Global optimization

1. Introduction.

Global minimization problems considered in this paper can be formulated as follows:

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

where $D \subset R^n$ is a bounded closed set. For convenience, the global minimization problems are denoted as GPs for short. They can be divided into two groups: differentiable GPs in which the objective function $f(x)$ is differentiable and non-differentiable GPs in which the objective function $f(x)$ is nondifferentiable.

For differentiable GPs, many traditional optimization methods have been developed, such as Newton method, quasi Newton methods, steepest descent method, trust region method, conjugate gradient method. These methods can converge rapidly to a solution with high precision, but the solution found is usually not the global optimal solution. For the review of recent development of this kind of methods one can refer to the work [1]. Another type of traditional methods is auxiliary function methods, in which an auxiliary function is designed to help algorithms to escape from the local optimum. Some well known methods have been proposed, such as filled function method [2, 3], stretching function method [4], tunneling function method [5] and cut-peak function method [6]. It has been proved that, with proper parameters, these methods have good performance in jumping out the local optimal solutions; however, most of them (e.g., stretching function method, filled function method and tunneling function method) will appear the so called "Mexican hat" effect [7] if the parameters are improperly chosen [8]. Therefore, these existing algorithms are parameter sensitive. Smoothing function [9] is a simple but effective auxiliary function, which can eliminate the solutions worse than the best solution found so far. Thus, it can reduce the number of local optimal solutions and make the search for global optimal solutions much easier. However, by using the smoothing function, some useful information such as descent direction will lost, and this makes the local search more difficult.

For nondifferentiable GPs with many local optima, it is more difficult and challengeable for solution algorithms. When designing an algorithm for nondifferentiable GPs, one has to face the following three issues: (1) how to escape from the local optima; (2) how to improve the convergent speed with high precision efficiently; (3) the gradients cannot be used.

Over the past decades, nondifferentiable global optimization has found many applications in real life and attracted more and more attention. Many efficient and effective algorithms have been developed for nondifferentiable optimization problems in order to deal with aforementioned three issues. For example, evolutionary algorithms (EAs) are one of the most efficient and popular such algorithms, but the convergent speed of EAs is often slow, and the solution found is often not the global optimal solution for high-dimensional problems with a large number of local optimal solutions. In order to overcome these shortcomings, many techniques have been developed and incorporated into EAs to enhance their performance [10, 11]. One of the most popular strategies is to embed local search (LS) techniques into EAs to form a new kind of algorithms called memetic algorithms (MAs) [12-15]. The local search techniques are often employed to speed up the search procedure [12]. Based on the usage of local search techniques, MAs can be classified into the following two groups [16]:

- LS is applied to each or some individuals of each population directly, aiming to find better solutions. Quasi-Newton, conjugate gradient, SQP, and steepest descent direction, etc. are examples of this kind of LS [15, 17-19].
- LS is integrated into the evolutionary operators to improve the performance of the evolutionary operators, such as crossover-based LS. Using crossover-based LS can produce several offspring around the parents [20].

In an MA, the attention has to be paid to both global search ability and local search ability. As mentioned in [16], global search can provide reliability and local search can provide accuracy for an MA. By keeping the diversity of the population, the global search ability can be improved, while a proper local search scheme can greatly improve the performance of an MA. When designing an efficient and powerful MA, one has to balance the global search and the local search [13].

The following sections are arranged as follows. In Section 2, smoothing function [9] and filled function [2] are introduced. In Section 3, a new local search method is proposed based on the minimum-elimination-escape function. Sections 4 and 5 introduce the uniform design based crossover operator and square search respectively. The proposed algorithm MEEM is proposed in Section 6. Section 7 presents the experimental results. The conclusions are made at last.

2. **Smoothing Function and Filled Function.**

2.1. **Smoothing function.**

Smoothing function [9] is utilized to eliminate the worse solutions which might influence the search ability of the solution algorithm. By using the smoothing function, the number of the local minima of the original global optimization problem will be reduced substantially. According to [9], we transform the formula of the smoothing function to an equivalent form as follows:

$$F(x, x^*) = \min_{x \in D}\{f(x), f(x^*)\} \tag{2}$$

where $f(x)$ is the original objective function, and x^* is the best solution found so far (changed with generation). Thus, at any point x no better than x^*, its function value is changed to $f(x^*)$, i.e., $F(x, x^*) = f(x^*)$, meanwhile, at any point x better than x^*, its function value keeps unchanged, i.e., $F(x, x^*) = f(x)$. This can be intuitively illustrated by smoothing a function $f(x)$ of one variable as an example in Figure 1, in which the dashed line represents the original function, the black solid line represents the smoothing function, and x^* is the best solution found so far. The idea of the smoothing function method is that, when a local minimum x^* is found, a smoothing function is constructed by Formula (2), and it is minimized to get another better local minimum. Then, the smoothing function is updated by using the new local minimum and it is minimized again. The process is repeated until the global minimum is found.

The advantage of the smoothing function method is as follows, it may eliminate many local minima at each iteration and speed up the search for global minimum. However, after flattened by the smoothing function, there will be a large plat area, in which some information such as descent direction will lost. This might bring difficulties for local search.

2.2. **Filled function.** The filled function method, proposed by Ge [2] in 1990, is a popular and an efficient approach to look for the global minima of multi-modal functions. Suppose that x^* is the best solution found so far. The filled function at x^* was constructed as follows [2]:

$$P(x, x^*, r, \rho) = \frac{1}{r + f(x)} \exp\left(-\frac{\|x - x^*\|^2}{\rho^2}\right) \tag{3}$$

where r and ρ are parameters which need to be chosen properly. Motivated by this idea, a number of efficient filled functions have been designed (e.g., [2, 3]). However, these filled functions are often sensitive to their parameters. Improper parameters will lead to the phenomenon called "Mexican hat" effect [7] which will cause difficulties for minimization process.

To overcome the shortcomings of both smoothing function and filled function, a simple auxiliary function called minimum-elimination-escape function is proposed and is denoted as MEE function. The function has only one parameter and is not parameter sensitive.

3. Minimum-Elimination-Escape Function Method.

3.1. **Minimum-elimination-escape function.** The minimum-elimination-escape function can be defined as follows:

$$P(x, x^*, \gamma) = F(x, x^*) - \gamma\|x - x^*\| \tag{4}$$

where $\gamma > 0$ is a parameter, and x^* is the best solution found so far.

Figure 1 shows the original function, smoothing function and MEE function with one variable as an example. Figures 2 to 4 show the original function, its smoothing function and its MEE function with two variables as an example. It can be seen from Figure 4 that, after minimum elimination at the best solution x^* found so far, descent directions can be found easily for MEE function. Thus, a better local minimum can be found easily by using MEE function from x^*.

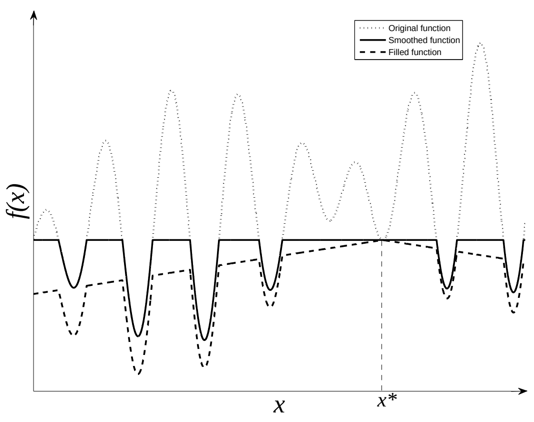

FIGURE 1. 2-D example: MEE function at x^*

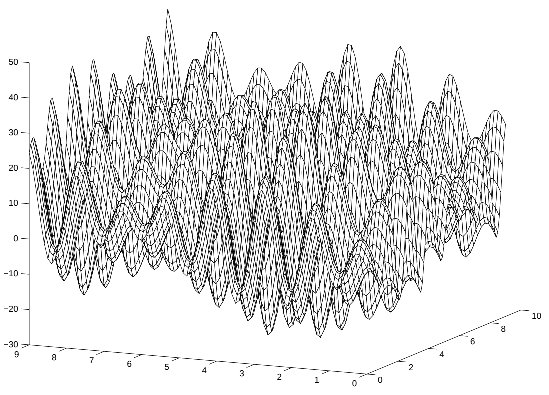

FIGURE 2. 3-D example: Original function

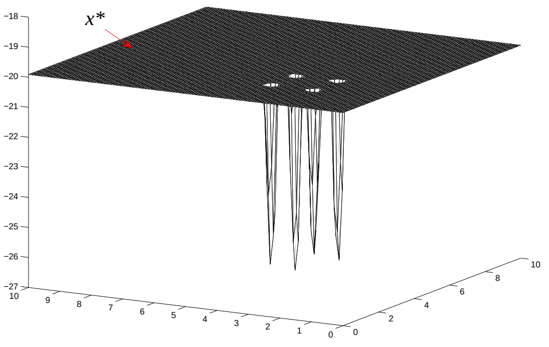

FIGURE 3. 3-D example: Smoothing function

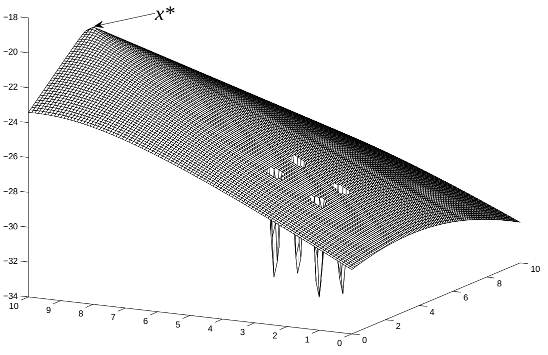

FIGURE 4. 3-D example: MEE function

Also note that for MEE function in (4)

$$J(x, x^*, \gamma) \triangleq \frac{P(x^*, x^*, \gamma) - P(x, x^*, \gamma)}{||x - x^*||} = \frac{F(x^*, x^*) - F(x, x^*)}{||x - x^*||} + \gamma \tag{5}$$

with $x \in D$ and $x \neq x^*$. Obviously, it has the following properties:

1) For $\forall x \in D$, one has $P(x, x^*, \gamma) \leq P(x^*, x^*, \gamma) = F(x^*, x^*) = f(x^*)$.
2) If x is a local minimum of (4), then x is located in a basin[1] of (2), and also in a basin of the objective function $f(x)$.
3) If $x \neq x^*$ and $f(x) \geq f(x^*)$, then $F(x, x^*) = f(x^*)$ and $J(x, x^*, \gamma) = \gamma$;
4) If $x \neq x^*$ and $f(x) < f(x^*)$, then $F(x, x^*) = f(x)$ and $J(x, x^*, \gamma) > \gamma$.

The first property means that x^* is the unique global maximum of $P(x, x^*, \gamma)$. It can be seen from the second property that for $\forall x \in D$, if $x \neq x^*$, the direction from x^* to x is a descent direction of $P(x, x^*, \gamma)$. Also, from properties 3) and 4), one can see that if x is no better than x^*, $J(x, x^*, \gamma) = \gamma$ holds. If x is better than x^*, $J(x, x^*, \gamma) > \gamma$ holds.

[1]A basin [21] of $f(X)$ at an isolated minimizer x_1^* is a connected domain B_1 which contains x_1^* and in which starting from any point the steepest descent trajectory of $f(X)$ converges to x_1^*, but outside which the steepest descent trajectory of $f(X)$ does not converge to x_1^*.

Another issue is that how to estimate the parameter γ. In the existing auxiliary function methods, estimating parameters is usually a hard task. However, in the proposed MEE function, parameter γ can be easily estimated and is not sensitive. Generally, when x^* is the best solution found so far, the search will go away from x^*. However, when the search goes to a point x far away from x^* (usually x is away from x^* in order to jump out the current basin), the main problem arisen will be that $\gamma||x - x^*||$ is too large to result in arithmetic overflow. So γ should be small in order to avoid $\gamma||x - x^*||$ too large. Thus, in the proposed algorithm, it is no problem if γ is taken not too large. It is suggested that parameter γ is taken in $(0, 1]$. In experiments, parameter γ was taken as 1.0 and 10^{-9} respectively to test its influence.

3.2. **Minimum-elimination-escape function method.** Based on the above strategies, we design a new local search method called minimum-elimination-escape function method. In the designed method, the best solution found so far is employed as a reference point. In the search process, a solution with poor objective value might locate in a lower basin. Therefore, in the proposed local search method, not only the best solution found so far but also any other solution will be updated. It is expected that using this strategy could increase the chance of finding a better solution. This idea can be described by the framework of the following Algorithm 1.

Algorithm 1 Minimum-Elimination-Escape Function Method

```
Step 1) Given the parameter $\gamma > 0$. Suppose that $x^* \in D$ is the best solution found so
        far, and $x_0$ is one picked solution needs to be updated.
Step 2) Generate the smoothing function $F(x, x^*)$ according to (2); then, generate the
        auxiliary function $P(x, x^*, \gamma)$ according to (4).
Step 3) Randomly generate $K$ uniformly distributed vectors $e_i$ $(i = 1 \sim K)$ on the unit
        ball.
Step 4) Let $\bar{X} = \Phi$. Along each $e_i$ $(i = 1 \sim K)$, using line search scheme to obtain a
        solution $\bar{x}_i \in D$. If $P(\bar{x}_i, x^*, \gamma) > \gamma$, let $\bar{X} = \bar{X} \cup \{\bar{x}_i\}$. If $\bar{X} \neq \Phi$, go to Step 5);
        else, go to Step 3);
Step 5) For each $\bar{x}_j \in \bar{X}$, minimize $F(\bar{x}_j, x^*)$ till a local minimum $\tilde{x}_j$ is obtained. Select
        $\tilde{x}$, which satisfies $F(\tilde{x}, x^*) = \min_j F(\tilde{x}_j, x^*)$. Let $x^* = \tilde{x}$, stop.
```

4. **Crossover Operation Based on Uniform Design.** First we denote

$$C_n = \big\{(x_1, x_2, \cdots, x_n) | 0 \le x_1, \cdots, x_n \le 1\big\}$$

The uniform design methods [22] aim to generate several uniformly distributed points in C_n. By the following two steps, q approximately uniformly distributed points in C_n can be generated.

Step 1. Construct an n-dimensional reference vector $\gamma = (\gamma_1, \gamma_2, \cdots, \gamma_n)$ using any of the following 3 methods $(i = 1, 2, \cdots, n)$:

a) $\gamma_i = \sqrt{p_i}$, where $p_1, \cdots, p_n$ are the first n positive primes.

b) $\gamma_i = p^{i\lambda}$, where $\lambda = \frac{1}{n+1}$ and p is a positive prime.

c) $\gamma_i = 2\cos(i\omega)$, where $p \ge 2n + 3$ is a positive prime and $\omega = 2\pi/p$.

Step 2. Let $\{\alpha\}$ denote the decimal part of the real number α. Generate a set of q approximately uniformly distributed points in C_n as follows:

$$\big\{\big(\{k\gamma_1\}, \{k\gamma_2\}, \cdots, \{k\gamma_n\}\big) | k = 1 \sim q\big\}$$

Suppose that $x = (x_1, \cdots, x_n)$ and $y = (y_1, \cdots, y_n)$ are any two parents chosen for crossover. Let $l_i = \min\{x_i, y_i\}$ and $u_i = \max\{x_i, y_i\}$ for $i = 1 \sim n$. Now, a new crossover operator is designed to generate q approximately uniformly distributed points in set $[l, u] = \{x | l_i \le x_i \le u_i,\ i = 1 \sim n\}$. The crossover operator is described in the following Algorithm 2.

Algorithm 2 Crossover Operator Based on Uniform Design [9]

```
Step 1) Generate $q$ approximately uniformly distributed points in $C_n$ by the above steps, and denote the set of these points by $\big\{(c_{k1}, \cdots, c_{kn}) | k = 1 \sim q\big\} = \big\{\big(\{k\gamma_1\}, \{k\gamma_2\}, \cdots, \{k\gamma_n\}\big) | k = 1 \sim q\big\}$.
Step 2) Generate $q$ uniformly distributed points in set $[l, u]$ by $B = \big\{(b_{k1}, \cdots, b_{kn}) | b_{kj} = l_j + c_{kj}(u_j - l_j),\ j = 1 \sim n,\ k = 1 \sim q\big\}$. Then, the points in $B$ are offspring of $x$ and $y$.
```

5. Square Search.

Flattened by the smoothing function, the flat area of $f(x)$ will be very large, especially at the end of the evolution process. In order to find a better point, it is needed to explore the domain as completely as possible. Therefore, we propose a new search method called square search in this paper to search the domain effectively. In the square search, a group of squares will be constructed according to some selected solutions, and several uniformly distributed points in (or on) each square will be sampled.

At the end of the process, the focus of a global optimization algorithm will turn to improve the precision of the solutions. Enlarging the search squares only can not improve the precision effectively, which also might bring much computational cost. In order to handle this issue, we do the square search outward and inward from the initial square alternately. Under this way, a better solution can be found with less computational cost and also the solution precision can be improved. The detail is described in Algorithm 3.

Algorithm 3 Square Search

```
Step 1) Let $\Gamma > 0$ be the maximum number of search squares and $L = (l_1, l_2, \cdots, l_n)$, $U = (u_1, u_2, \cdots, u_n)$ be the lower bound and upper bound of the search domain respectively. Suppose that $\bar{x} = (\bar{x}_1, \bar{x}_2, \cdots, \bar{x}_n)$ is the best solution in the current population. Let $k = 1$;
Step 2) If $k$ is odd, let $\alpha = \frac{k}{\Gamma}$; else, let $\alpha = \frac{1}{\Gamma + 2k}$. Go to Step 3);
Step 3) Compute $\bar{L} = (\bar{l}_1, \cdots, \bar{l}_n)$ and $\bar{U} = (\bar{u}_1, \cdots, \bar{u}_n)$, in which $\bar{l}_i = x_i - \alpha(\bar{x}_i - l_i)$, $\bar{u}_i = x_i + \alpha(u_i - \bar{x}_i)$, for $i = 1 \sim n$. Generate $J$ points uniformly in the hyperrectangle between $\bar{L}$ and $\bar{U}$.
Step 4) Compute the smoothing function values of these $J$ points. If there is a point $x^j$ such that $F(\bar{x}, x^j) < f(\bar{x})$, go to Step 5); Otherwise, let $k = k + 1$, go to Step 2);
Step 5) Use a local search method to update $x^j$. Denote the updated solution as $\tilde{x}$. Let $\bar{x} = \tilde{x}$, stop.
```

6. Minimum-Elimination-Escape Memetic Algorithm: MEEM.

Based on the above techniques, we propose a novel memetic algorithm called minimum-elimination-escape memetic algorithm (MEEM) as follows.

```
Algorithm 4 Minimum-Elimination-Escape Memetic Algorithm: MEEM
Step 1) (Initialization) Given population size $N$, crossover probability $p_c > 0$ and update probability $p_u$. Randomly generate initial population $POP(0)$. Find out the best individual $\tilde{x}_0$ in $POP(0)$. Let $k = 0$.
Step 2) (Smoothing Function) Define smoothing function $F(x, \tilde{x})$ by Formula (2).
Step 3) (Crossover) Randomly choose $[p_c \times N/2]$ pairs of parents from $POP(k)$, where $[\alpha]$ represents the integer part of number $\alpha$. For each pair, use Algorithm 2 to produce offspring. The set of all these offspring is denoted as $O_1$.
Step 4) (Square Search) Randomly choose $[p_u \times N]$ individuals from $POP(k) \cup O_1$. For each individual, use Algorithm 3 to generate offspring. Denote the set of all these spring as $O_2$.
Step 5) (Selection) Select the first $[N/2]$ best individuals from $POP(k) \cup O_1 \cup O_2$ to put into $POP(k+1)$ using original function $f(x)$, then randomly select $N - [N/2]$ individuals from $POP(k) \cup O_1 \cup O_2$ to put into $POP(k+1)$.
Step 6) (Local Search) Find out the best individual $\tilde{x}_{k+1}$ in $POP(k+1)$. Use Algorithm 1 to update $\tilde{x}_{k+1}$. Randomly choose $[p_u \times (N-1)]$ individuals from $POP(k+1)$. Use Algorithm 1 to update each individual.
Step 7) (Termination) If stop condition is satisfied, stop; otherwise, let $k = k + 1$, go to Step 2.
```

Table 1. Basic characteristics of test functions $F_{21\sim30}$

Func	n	Search Space	$f_{\min}$	Func	n	Search Space	$f_{\min}$
F_{21} (see f_{14} in [23])	2	$[-65.536, 65.536]^n$	0.9980	F_{26} (see f_{23} in [23])	4	$[0, 10]^n$	−10.54
F_{22} (see f_6 in [23])	30	$[-100, 100]^n$	0	F_{27} (see f_9 in [24])	30	$[-32, 32]^n$	0
F_{23} (see f_7 in [23])	30	$[-1.28, 1.28]^n$	0	F_{28} (see f_{10} in [24])	30	$[-600, 600]^n$	0
F_{24} (see f_{21} in [23])	4	$[0, 10]^n$	−10.15	F_{29} (see f_7 in [24])	30	$[-5, 5]^n$	0
F_{25} (see f_{22} in [23])	4	$[0, 10]^n$	−10.54	F_{30} (see f_{12} in [24])	30	$[-5.12, 5.12]^n$	0

7. Experimental Results.

7.1. Test problems and existing algorithms for comparison.

30 widely used and challenging benchmarks are selected to test the performance of the proposed algorithm MEEM, where F_1 to F_{20} are the same as that used in [29], and F_{21} to F_{30} are chosen from [23] to [28]. The basic properties of the last 10 problems are listed in Table 1, where $f_{\min}$ represents the function value at minimum and n the dimension of the problem. In each table from Table 2 to Table 7, Func, M-fun, Best, Worst, M-best, Std represent test functions, the mean number of function evaluations, the best function value, the worst function value, the mean best function value and the standard deviation of best function values respectively in 50 runs. "NA" represents the results are not provided in the related reference, and "–" means the experiment was not carried out on the related function.

We choose 8 algorithms for comparison: Fast Evolutionary Programming with Cauchy Mutation (FEP) [23], Comprehensive Learning Particle Swarm Optimizer (CLPSO) [24], Hybrid Estimation of Distribution Algorithm (EDA/L) [25], Hybrid Taguchi-Genetic Algorithm (HTGA) [26], Orthogonal Genetic Algorithm with Quantization (OGA/Q) [27], Evolutionary Programming with Adaptive Levy Mutation (ALEP) [28], Evolutionary Algorithm based on the Level-set Evolution and Latin Squares (LEA) [29], and DE with Global and Local Neighborhoods and Self-adaptive Weight Factor (DEGL/SAW) [30]. Each of the above algorithms was executed to solve some of the test functions, and the results are reported in related articles. We use these existing results for a direct comparison.

7.2. **Parameters setting for MEEM.** The proposed algorithm MEEM is executed 50 independent runs for each test problem. In experiments, we adopted the parameters as follows:

- Population size: $N = 20$.
- Crossover and mutation parameters: $p_u = 0.1$, $p_c = 0.2$, $p = 5$, $q = 7$.
- Parameters in Algorithms 1 and 3: $\gamma = 1.0e - 9$ and 1.0, $\Gamma = 6$, $K = \lceil n/10 \rceil$, where n is the dimension of the problem, and $\lceil \alpha \rceil$ denotes the smallest integer no less than α.
- Stopping criterion: when the best solution can not be improved further in successive 50 generations or after 400 generations, the execution of the algorithm is stopped.

7.3. **Results and comparisons.** In the experiments, in order to test the influence of parameter γ, we took parameter γ as 1.0 and 1.0e-9 respectively, and the results are presented in Tables 2 and 3. From Tables 2 and 3, we can find that for test functions, except for $F_{7,8,12}$, the mean best function values found by MEEM are very close to the global optimal values, and the standard deviations of the results are very small. Comparing the results listed in Tables 2 and 3, one can find obviously that the difference between the results in these two tables is so slight that it can be ignored. This confirms that parameter γ has no big influence for the proposed algorithm.

From Table 7, it can be seen that, for 100-dimensional problems $F_{1\sim 6}$, MEEM can find close-to-optimal solutions with high precision. Also, in the 50 independent runs for each test function, the difference of the objective function values between the best solutions and the worst solutions is very small except for F_7, which indicates that the proposed algorithm

Table 2. The results of the test problems obtained by MEEM with $\gamma = 1.0$

Func	M-fun	Best	Worst	M-best	Std
F_1	49924.12	−1.256948661e+04	−1.256948660e+04	−1.256948661e+04	3.876432735871869e-06
F_2	25302.98	0	9.563732975e -15	2.385643531e-15	4.253762041054699e-15
F_3	18690.20	2.534801583e-11	4.859312462e-09	5.863520138e-10	1.487512863528164e-10
F_4	77524.42	0	2.683972689e-13	3.013643561e-14	2.698293760435360e-14
F_5	55711.10	1.023842523e-21	4.364281265e-18	2.936533168e-19	5.853531283624362e-19
F_6	65104.32	1.862383517e-20	6.401352262e-17	4.912532839e-18	2.439256246437241e-17
F_7	179872.46	−79.451305826	−58.880842725	−62.828383918	7.183906351
F_8	165704.12	2.001825360e-10	9.351017386e-09	2.134326693e-09	1.028960234328506e-09
F_9	95536.74	−78.332331407	−78.332331407	−78.332331407	2.563836025327626e-12
F_{10}	99079.32	2.103102871e-10	3.831378210e-05	3.933672977e-06	5.130753647238653e-04
F_{11}	42693.54	2.563120346e-32	1.242309943e-19	4.001034528e-20	2.351462532870132e-20
F_{12}	70086.52	1.114137321e-04	5.925609e-03	1.962119e-03	2.019351623024e-03
F_{13}	18633.58	0	0	0	0
F_{14}	59706.76	0	8.968561882e-07	5.7211068e-07	1.092125583397502e-07
F_{15}	46398.14	7.783253659e-37	1.359300187e-19	3.495821643e-20	6.365208636133355e-20
F_{16}	3088.98	−1.031363937	−1.031363937	−1.031363937	4.763836925132368e-13
F_{17}	3216.64	0.397894048	0.397894068	0.397894051	5.231853576703662e-09
F_{18}	3673.18	3.00000000	3.00000000	3.000000000	6.768234664933617e-10
F_{19}	29690.40	3.075076257e-04	3.094726485e-04	3.084352577e-04	6.024336152192296e-07
F_{20}	19267.82	−3.321995171	−3.321995171	−3.321995171	6.001478135963665e-14
F_{21}	2057.86	0.998003837	0.998003851	0.998003844	7.000540134362388e-10
F_{22}	1913.06	0	0	0	0
F_{23}	64683.36	6.203724630e-06	8.996438931e-03	1.078365295e-03	2.536096236499e-03
F_{24}	2480.20	−10.15319968	−10.15319968	−10.15319968	4.061362765391308e-12
F_{25}	2301.34	−10.40294057	−10.40294057	−10.40294057	2.143625673177430e-12
F_{26}	1960.50	−10.53640982	−10.53640982	−10.53640982	5.332604361917626e-13
F_{27}	20713.78	2.683717056e-11	3.846620132e-09	2.836725823e-10	5.967314836240213e-10
F_{28}	63503.30	0	2.106947426e-11	1.019589723e-12	5.896137584260245e-12
F_{29}	27217.60	0	5.671335579e-14	2.7683576211e-15	5.667133105338553e-15
F_{30}	25788.04	8.964351672e-16	3.907643761e-14	1.837695134e-14	5.169810359235277e-15

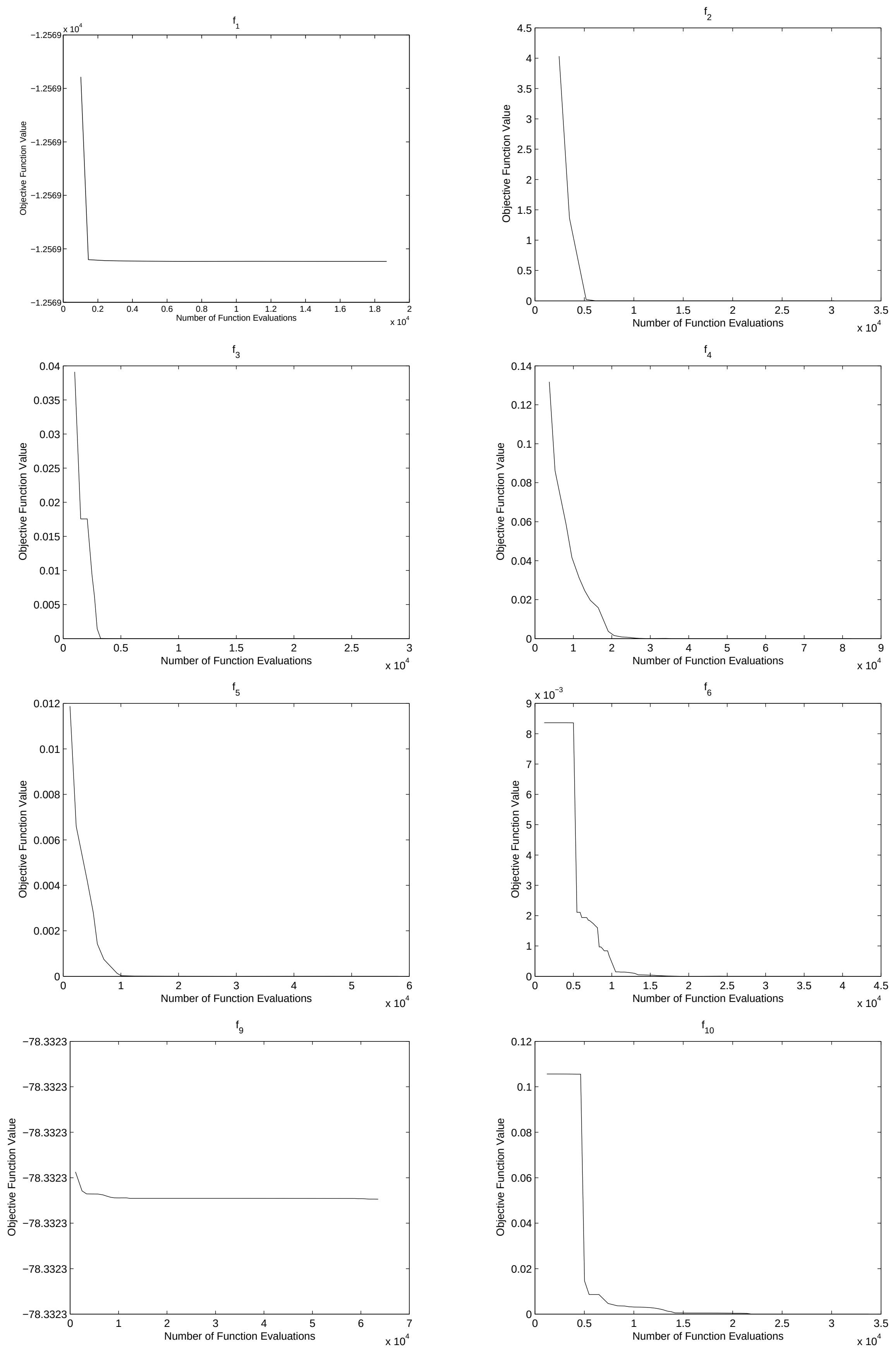

FIGURE 5. Convergence results for several test functions by MEEM

TABLE 3. The results of the test problems obtained by MEEM with γ =1.0e-9

Func	M-fun	Best	Worst	M-best	Std
F_1	49783.26	−1.256948661e+04	−1.256948660e+04	−1.256948661e+04	4.352138463579719e-06
F_2	25467.30	0	1.635368369e-14	1.635368369e-15	3.001841704840774e-15
F_3	18873.44	4.270635796e-11	7.258964376e-09	6.536389265e-10	1.976553225148764e-10
F_4	76825.62	0	4.258563867e-13	1.954283295e-14	1.856258415716020e-14
F_5	56482.38	8.8567391e-21	1.225327620e-18	6.859725426e-19	8.041890784161613e-19
F_6	64946.76	3.261482385e-20	2.312488697e-17	7.341356412e-18	1.250142385318736e-17
F_7	179906.78	−79.736346712	−56.074589249	−62.813745231	7.372856190
F_8	165468.26	1.457934631e-10	1.715238622e-08	1.001152733e-09	1.237674872836286e-09
F_9	95977.46	−78.332331407	−78.332331407	−78.332331407	5.649273206701539e-12
F_{10}	99532.62	1.956350635e-10	4.762456031e-05	5.210146351e-06	1.752761639218210e-04
F_{11}	42563.28	1.274130266e-32	2.057164352e-19	2.790214303e-20	1.021283535413243e-20
F_{12}	69745.18	1.008759429e-04	6.746284e-03	2.311738e-03	1.769245736822e-03
F_{13}	18752.02	0	0	0	0
F_{14}	59587.90	0	1.135081479e-06	7.5120743e-07	1.216234973490162e-07
F_{15}	46532.20	1.394520156e-36	2.571603077e-19	2.539301883e-20	4.450167997062091e-20
F_{16}	3241.52	−1.031363937	−1.031363937	−1.031363937	5.895904638713140e-13
F_{17}	3241.52	0.397894048	0.397894069	0.397894050	8.763941758498781e-09
F_{18}	3712.22	3.00000000	3.00000000	3.000000000	5.357141679044328e-10
F_{19}	29487.54	3.075076256e-04	3.096398825e-04	3.085263685e-04	5.546848450346350e-07
F_{20}	19358.62	−3.321995171	−3.321995171	−3.321995171	5.023768938325215e-14
F_{21}	2087.42	0.998003837	0.998003851	0.998003845	5.283475729653587e-10
F_{22}	1865.22	0	0	0	0
F_{23}	65011.40	9.430563839e-06	1.441165279e-02	1.890234872e-03	1.999768845546e-03
F_{24}	2498.22	−10.15319968	−10.15319968	−10.15319968	2.413624840505037e-12
F_{25}	2287.68	−10.40294057	−10.40294057	−10.40294057	4.634421329174964e-12
F_{26}	1936.28	−10.53640982	−10.53640982	−10.53640982	7.215419216734349e-13
F_{27}	20634.80	3.545370923e-11	5.925740054e-09	4.404834726e-10	2.365725342586147e-10
F_{28}	63271.44	0	5.425720268e-11	1.560637523e-12	7.706591561478561e-12
F_{29}	27088.72	0	7.103625845e-14	4.1165372231e-15	6.352710467247366e-15
F_{30}	25906.38	7.253845211e-16	2.309263891e-14	2.004678520e-14	4.873012534776256e-15

is robust and effective. Figure 5 shows the convergence results of several problems. It can be seen obviously that MEEM can rapidly converge to the solutions which are very close to the global optimal solutions using few function evaluations. These results indicate that MEEM can find close-to-optimal solutions and has a good performance.

From Tables 4, 5 and 6, one can find that MEEM can find much better solutions with much fewer function evaluations than FEP and ALEP for high dimensional problems $F_{1,6,11,14}$. It seems that both FEP and ALEP have difficulty in solving high dimensional problems. For 2-dimensional problems $F_{16,18}$, MEEM and ALEP perform equally. Thus, MEEM outperforms FEP and ALEP, especially in solving high dimensional problems.

Compared the results obtained by OGA/Q and MEEM, it can be seen that, for $F_{3,12,14,15}$, although OGA/Q obtains better results than MEEM, MEEM can use much fewer function evaluations to get the results with relatively high precision. For F_7, neither OGA/Q nor MEEM can find close-to-optimal solution. For the 100-dimensional problems, the results obtained by MEEM are closer-to-optimal and much better than those obtained by OGA/Q. For the remaining problems, the results obtained by MEEM are as good as OGA/Q. Thus, MEEM outperforms OGA/Q in dealing with high dimensional problems.

Compared with HTGA, it can be seen that MEEM can find better solutions using fewer function evaluations than HTGA on problems $F_{1,5,6,8,9}$. However, for F_{10}, MEEM obtains much better solution using more function evaluations than HTGA, so that it is difficult to judge which one has better performance for this problem. It can be found easily that, MEEM can find better solutions using fewer function evaluations for almost

Table 4. Comparisons between MEEM and other algorithms on $F_{1\sim 10}$

Func		MEEM	FEP	OGA/Q	HTGA	EDA/L	ALEP	CLPSO	LEA
F_1	M-fun	49,783	900,000	302,116	163,468	52,216	150,000	–	287,365
	M-best	−12569.49	−12554.5	−12569.4537	−12569.46	−12569.48	−11469.2	–	−12569.4542
	Std	4.532e-06	52.6	6.447e-04	0	NA	58.2	–	4.831e-04
F_2	M-fun	25,467	500,000	224,710	16,267	75,014	150,000	200,000	223,803
	M-best	1.635e-15	0.046	0	0	0	5.85	4.85e-10	2.103e-18
	Std	3.002e-15	0.012	0	0	NA	2.07	3.63e-10	3.359e-18
F_3	M-fun	18,873	150,000	112,421	16,632	106,061	150,000	200,000	121,435
	M-best	4.270e-11	0.018	4.440e-16	0	4.141e-15	0.019	0	2.5993
	Std	1.976e-10	0.021	3.989e-17	0	NA	0.001	0	0.09245
F_4	M-fun	76,825	200,000	134,000	20,999	79,096	150,000	200,000	130,498
	M-best	1.954e-14	0.016	0	0	0	0.024	3.14e-10	6.104e-16
	Std	1.856e-14	0.022	0	0	NA	0.028	4.64e-10	3.001e-17
F_5	M-fun	56,482	150,000	134,556	66,457	89,925	150,000	–	132,642
	M-best	6.860e-19	9.2e-06	6.019e-06	1.000e-06	3.654e-21	6.0e-06	–	2.482e-06
	Std	8.042e-19	3.6e-06	1.159e-06	0	NA	1.0e-06	–	2.276e-06
F_6	M-fun	64,946	150,000	134,143	59,003	114,570	150,000	–	130,213
	M-best	7.341e-18	1.6e-04	1.869e-04	1.000e-04	3.485e-21	9.8e-05	–	1.734e-04
	Std	1.250e-17	7.3e-05	2.615e-05	0	NA	1.2e-05	–	1.205e-04
F_7	M-fun	189,906	–	302,773	265,693	169,887	–	–	289,863
	M-best	54.279	–	92.83	−92.83	−94.3757	–	–	−93.01
	Std	9.037	–	0.02626	0	NA	–	–	0.02314
F_8	M-fun	165,468	–	190,031	186,816	124,417	–	–	189,427
	M-best	1.001e-09	–	4.672e-07	5.869e-05	3.294e-08	–	–	1.627e-06
	Std	1.237e-09	–	1.293e-07	8.325e-05	NA	–	–	6.527e-03
F_9	M-fun	95,977	–	245,930	216,535	153,116	–	–	243,895
	M-best	−78.3323	–	−78.3000296	−78.303	−78.31077	–	–	−78.31
	Std	5.469e-12	–	6.288e-03	0	NA	–	–	6.127e-04
F_{10}	M-fun	99,532	–	167,863	60,737	128,140	–	200,000	168,910
	M-best	5.210e-06	–	0.752	0.7	4.324e-03	–	2.1e+01	0.5609
	Std	1.483e-05	–	0.114	0	NA	–	2.98e+00	0.1078

100-dimensional problems. For $F_{2\sim 4,11,14,15}$, HTGA can found exactly optimal solutions using fewer function evaluations than MEEM. For F_{13}, both MEEM and HTGA can find the exactly optimal solution, but HTGA uses fewer function evaluations. By comparing Figure 5 and the convergence graphs reported in [26], we can find that the convergence speed of MEEM is much faster than HTGA. Therefore, MEEM outperforms HTGA in solving high-dimensional problems.

The results of EDA/L are shown in Table 4, where one can find that both MEEM and EDA/L can find optimal or close-to-optimal solutions except for F_7. For $F_{1,8\sim 10}$, MEEM can find better solutions with fewer function evaluations than EDA/L. For $F_{2\sim 6}$, MEEM obtains a little worse solutions, while uses much fewer function evaluations than EDA/L. From the above, it can be concluded that MEEM outperforms EDA/L.

Comparing MEEM and CLPSO, it can be found that for $F_{2,4,10,11,27\sim 30}$, MEEM can obtain solutions with higher precision using much fewer function evaluations. For F_3, the result obtained by CLPSO looks a little better than that obtained by MEEM, but the function evaluations used by MEEM are much fewer than CLPSO. Thus MEEM outperforms CLPSO.

The results of LEA and MEEM show that, MEEM can find optimal or close-to-optimal solutions except for $F_{7,12}$ (with precision 1.0e-4 or better). However, for $F_{1,3,6\sim 10,12,20}$, the solutions obtained by LEA can not reach the precision 1.0e-4. For 15 problems, MEEM can use fewer function evaluations to find better solutions than LEA. On the remaining problems, MEEM can use much fewer function evaluations than LEA to find solutions

TABLE 5. Comparisons between MEEM and other algorithms on $F_{11\sim 20}$

Func		MEEM	FEP	OGA/Q	HTGA	EDA/L	ALEP	CLPSO	LEA
F_{11}	M-fun	42,563	150,000	112,559	20,844	–	150,000	200,000	110,647
	M-best	2.790e-20	5.7e-04	0	0	–	6.32e-04	4.46e-14	4.727e-16
	Std	1.021e-20	1.3e-04	0	0	–	7.6e-05	1.73e-14	6.218e-17
F_{12}	M-fun	69,745	–	112652	20,065	–	–	–	111,093
	M-best	2.312e-03	–	6.301e-03	0.001	–	–	–	5.136e-03
	Std	1.769e-03	–	4.069e-04	0	–	–	–	4.432e-4
F_{13}	M-fun	18,752	200,000	112,612	14,285	–	–	–	110,031
	M-best	0	8.1e-03	0	0	–	–	–	4.247e-19
	Std	0	7.7e-04	0	0	–	–	–	4.236e-19
F_{14}	M-fun	59,587	500,000	112,576	26,469	–	150,000	–	11,064
	M-best	7.512e-07	0.016	0	0	–	0.04185	–	6.783e-18
	Std	1.216e-07	0.014	0	0	–	0.05969	–	5.429e-18
F_{15}	M-fun	46,532	500,000	112,893	21,261	–	–	–	111,105
	M-best	2.539e-20	0.3	0	0	–	–	–	2.683e-16
	Std	4.450e-20	0.5	0	0	–	–	–	6.257e-17
F_{16}	M-fun	3,241	10,000	–	–	–	3,000	–	10,823
	M-best	−1.0314	−1.03	–	–	–	−1.031	–	−1.03108
	Std	5.896e-13	4.9e-07	–	–	–	0	–	3.364e-07
F_{17}	M-fun	3,792	10,000	–	–	–	–	–	10,538
	M-best	0.39789	0.398	–	–	–	–	–	0.398
	Std	8.764e-09	1.5e-07	–	–	–	–	–	2.652e-08
F_{18}	M-fun	3,712	10,000	–	–	–	3,000	–	11,721
	M-best	3.0000	3.02	–	–	–	3.000	–	3.0003
	Std	5.357e-10	0.11	–	–	–	0	–	6.245e-05
F_{19}	M-fun	29,487	400,000	–	–	–	–	–	55,714
	M-best	3.085e-04	5.0e-04	–	–	–	–	–	3.512e-04
	Std	5.547e-07	3.2e-04	–	–	–	–	–	7.361e-05
F_{20}	M-fun	19,358	20,000	–	–	–	–	–	28,428
	M-best	−3.322	−3.27	–	–	–	–	–	−3.301
	Std	5.024e-14	0.059	–	–	–	–	–	7.832e-03

TABLE 6. Comparisons between MEEM and other algorithms on $F_{21\sim 30}$[†]

Func		MEEM	FEP	CLPSO	Func		MEEM	FEP	CLPSO
F_{21}	M-fun	2,087	10,000	–	F_{26}	M-fun	1,936	10,000	–
	M-best	0.9980	1.22	–		M-best	−10.5364	−6.57	–
	Std	5.283e-10	0.56	–		Std	7.215e-13	3.14	–
F_{22}	M-fun	1,865	150,000	–	F_{27}	M-fun	20,634	–	200,000
	M-best	0	0	–		M-best	4.4048e-10	–	3.43e-04
	Std	0	0	–		Std	2.366e-10	–	1.91e-04
F_{23}	M-fun	65,011	300,000	–	F_{28}	M-fun	63,271	–	200,000
	M-best	1.890e-03	7.6e-03	–		M-best	1.5606e-12	–	7.04e-10
	Std	2.000e-03	2.6e-03	–		Std	7.706e-12	–	1.25e-11
F_{24}	M-fun	2,498	10,000	–	F_{29}	M-fun	27,088	–	200,000
	M-best	−10.1532	−5.52	–		M-best	4.1165e-15	–	4.36e-10
	Std	2.414e-12	1.59	–		Std	6.353e-15	–	2.44e-10
F_{25}	M-fun	2,287	10,000	–	F_{30}	M-fun	25,906	–	200,000
	M-best	−10.4029	−5.52	–		M-best	2.0047e-14	–	3.46e+01
	Std	4.634e-12	2.12	–		Std	4.873e-15	–	4.59e+00

[†] There is no result obtained by OGA/Q, HTGA, EDA/L, ALEP and LEA on $F_{21\sim 30}$

Table 7. Comparisons between MEEM and DEGL/SAW

Func		MEEM	DEGL/SAW	Func		MEEM	DEGL/SAW
F_1	M-fun	279,308.72	5,000,000	F_{17}	M-fun	3792.30	5,000,000
	M-best	$-4.189828872e+04$	$-4.18983e+04$		M-best	0.397894050	0.39788170
	Std	2.224628885e-06	6.98e-06		Std	8.763941758e-09	8.544e-04
F_2	M-fun	137,970.14	5,000,000	F_{18}	M-fun	3712.22	5,000,000
	M-best	3.438618648e-15	1.7728e-22		M-best	3.000000000	3.000000
	Std	1.013791431e-14	3.838e-23		Std	5.357141679e-10	NA
F_3	M-fun	192,341.26	5,000,000	F_{19}	M-fun	29487.54	5,000,000
	M-best	3.485860489e-10	3.52742e-17		M-best	3.085263685e-04	3.7041849e-04
	Std	5.217535743e-10	1.365e-16		Std	5.546848450e-07	2.11e-09
F_4	M-fun	375,756.00	5,000,000	F_{21}	M-fun	2087.42	5,000,000
	M-best	8.817814987e-16	4.11464e-15		M-best	0.998003845	0.99800390
	Std	9.734963968e-16	6.02e-16		Std	5.283475729e-10	1.15e-18
F_5	M-fun	156,937.82	5,000,000	F_{24}	M-fun	2498.22	5,000,000
	M-best	3.625324561e-17	8.00496e-19		M-best	-10.15319968	-10.15323
	Std	1.269571356e-17	4.82e-17		Std	2.413624840e-12	7.341e-08
F_6	M-fun	112,358.60	5,000,000	F_{25}	M-fun	2287.68	5,000,000
	M-best	1.685358743e-20	-1.142823		M-best	-10.40294057	-10.40295
	Std	2.073497151e-17	9.032e-05		Std	4.634421329e-12	5.923e-04
F_{10}	M-fun	99532.62	5,000,000	F_{26}	M-fun	1936.28	5,000,000
	M-best	5.210146351e-06	1.5463e-25		M-best	-10.53640982	-10.53641
	Std	5.64927320e-12	7.301e-22		Std	7.215419216e-13	3.90e-08
F_{16}	M-fun	3241.52	5,000,000				
	M-best	-1.031363937	-1.031630				
	Std	5.895904638e-13	1.749e-10				

with very high precision. These indicate that the performance of MEEM is better than that of LEA.

In order to compare with the algorithm DEGL/SAW, we expand the dimension of $F_{1\sim6}$ to 100. The global optimal value of 100-dimensional F_1 is -41898.3, and the global optimal values of $F_{2\sim6}$ are still 0. The properties of the other functions in Table 7 are the same as those listed in Table 1. Table 7 presents the results obtained by MEEM and DEGL/SAW. From Table 7, it can be seen that, MEEM uses much fewer function evaluations than DEGL/SAW. Also, MEEM can find better solutions than DEGL/SAW on $F_{1,4,19,21}$. For the remaining problems, the solutions obtained by DEGL/SAW are a little better than those obtained by MEEM, but the difference is very small. On average, DEGL/SAW needs much more computational cost than MEEM to find a solution with the same precision.

It can be indicated from the above comparison and analysis that MEEM outperforms the compared algorithms (except for EDA/L and HTGA for a few problems), especially on solving high dimensional problems. In addition, the standard deviations of function values given by MEEM is very small, which indicates that the performance of MEEM is stable.

8. Conclusions.

In this paper, a minimum-elimination-escape function is proposed to eliminate all the local minima no better than the best solution found so far, and it can provide a proper descent direction for the local search method. Then, we designed a local search method which can jump out from the current local minimum quickly and find the better local minimum with high precision. Also, in order to improve the global search ability of the proposed algorithm, a new search scheme called square search is proposed.

This search scheme can explore the overall domain effectively. Based on these strategies, a minimum-elimination-escape memetic algorithm (MEEM) is designed. The simulation results indicate the effectiveness and soundness of MEEM.

Acknowledgement. This work is partially supported by National Natural Science Foundation of China (60873099), Ph.D. Programs Foundation of Ministry of Education of China (20090203110005), the Fundamental Research Funds for Central Universities (K505 10030014), and 973 Program with Grant No. 2010CB328102. The authors also gratefully acknowledge the helpful comments and suggestions of the anonymous reviewers.

REFERENCES

[1] C. A. Floudas and C. E. Gounaris, A review of recent advances in global optimization, *Journal of Global Optimization*, vol.45, no.1, pp.3-38, 2009.

[2] R. Ge, A filled function method for finding a global minimizer of a function of several variables, *Mathematical Programming*, vol.46, pp.191-204, 1990.

[3] Y. Shang, D. Pu and A. Jiang, Finding global minimizer with one-parameter filled function on unconstrained global optimization, *Applied Mathematics and Computation*, vol.191, pp.176-182, 2007.

[4] K. E. Parsopoulos, V. P. Plagianakos, G. D. Magoulas and M. N. Vrahatis, Improving particle swarm optimizer by function stretching, *Advances in Convex Analysis and Global Optimization, Non-convex Optimization and Applications*, vol.54, pp.445-457, 2001.

[5] A. Levy and A. Montalvo, The tuneling algorithm for the global minimization of functions, *SIAM J. Scientific and Statistical Computing*, vol.6, pp.15-29, 1985.

[6] Y. Wang, W. Fang and T. Wu, A cut-peak function method for global optimization, *Journal of Computational and Applied Mathematics*, vol.230, pp.135-142, 2009.

[7] K. E. Parsopoulos and M. N. Vratatis, On the computation of all global minimizers through particle swarm optimization, *IEEE Trans. Evol. Comput.*, vol.8, no.3, pp.211-224, 2004.

[8] Y.-J. Wang and J.-S. Zhang, A new constructing auxiliary function method for global optimization, *Mathematical and Computer Modelling*, vol.47, pp.1396-1410, 2008.

[9] Y. Wang and L. Fan, A smoothing evolutionary algorithm with circle search for global optimization, *Proc. of the 4th International Conference on Network and System Security*, pp.412-418, 2010.

[10] P.-W. Tsai, J.-S. Pan, B.-Y. Liao and S.-C. Chu, Enhanced artificial bee colony optimization, *International Journal of Innovative Computing, Information and Control*, vol.5, no.12(B), pp.5081-5092, 2009.

[11] M. Rashid and A. R. Baig, PSOGP: A genetic programming based adaptable evolutionary hybrid particle swarm optimization, *International Journal of Innovative Computing, Information and Control*, vol.6, no.1, pp.287-296, 2010.

[12] P. Merz and B. Freisleben, Fitness landscapes, memetic algorithms, and greedy operators for graph bipartitioning, *Evol. Comput.*, vol.8, no.1, pp.61-91, 2000.

[13] D. Molina, M. Lozano, C. García-Martínez and F. Herrera, Memetic algorithms for continuous optimisation based on local search chains, *Evolutionary Computation*, vol.18, no.1, pp.27-63, 2010.

[14] M. Lozano, F. Herrera, N. Krasnogor and D. Molina, Real-coded memetic algorithms with croosover hill-climbing, *Evolutionary Computation*, vol.12, no.2, pp.181-201, 2004.

[15] S. Ono, Y. Hirotani and S. Nakayama, A memetic algorithm for robust optimal solution search-hybridization of multi-objective genetic algorithm and quasi-newton method, *International Journal of Innovative Computing, Information and Control*, vol.5, no.12(B), pp.5011-5019, 2009.

[16] M. Lozano, F. Herrera, N. Krasnogor and D. Molina, Real-coded memetic algorithms with crossover hill-climbing, *Evolutionary Computation*, vol.12, no.3, pp.273-302, 2004.

[17] C.-K. Zhang and H.-H. Shao, A hybrid strategy: Real-coded genetic algorithm and chaotic search, *Proc. of IEEE Int. Conf. on Systems, Man, and Cybernetics*, pp.2361-2364, 2001.

[18] W. E. Hart, *Adaptive Global Optimization with Local Search*, Ph.D. Thesis, University of California, 1994.

[19] J. A. Joines and G. M. Kay, Hybrid genetic algorithms and random linkage, *Proc. of the Congress on Evolutionary Computation*, Piscataway, New Jersey, pp.1733-1738, 2002.

[20] S.-H. Chen, P.-C. Chang, Q. Zhang and C.-B. Wang, A guided memetic algorithm with probabilistic models, *International Journal of Innovative Computing, Information and Control*, vol.5, no.12(B), pp.4753-4764, 2009.

[21] L. C. W. Dixon, J. Gomulka and S. E. Herson, Reflections on global optimization problem, in *Optimization in Action*, L. C. W. Dixon (ed.), Academic Press, NY, 1976.
[22] F. T. Fang and Y. Wang, *Number-Theoretic Methods in Statistics*, Chapman & Hall, London, UK, 1994.
[23] X. Yao, Y. Liu and G. Lin, Evolutionary programming made faster, *IEEE Trans. Evol. Comput.*, vol.3, no.2, pp.82-102, 1999.
[24] J. J. Liang, A. K. Qin, P. N. Suganthan and S. Baskar, Comprehensive learning particle swarm optimizer for global optimization of multimodal functions, *IEEE Trans. Evol. Comput.*, vol.10, no.3, pp.281-295, 2006.
[25] Q. Zhang, J. Sun, E. Tsang and J. Ford, Hybrid estimation of distribution algorithm for global optimization, *Engineering Computations*, vol.21, no.1, pp.91-107, 2004.
[26] J.-T. Tsai, T. Liu and T.-H. Chou, Hybrid Taguchi-genetic algorithm for global numerical optimization, *IEEE Trans. Evol. Comput.*, vol.8, no.4, pp.365-377, 2004.
[27] Y. W. Leung and Y. P. Wang, An orthogonal genetic algorithm with quantization for global numerical optimization, *IEEE Trans. Evol. Comput.*, vol.5, no.1, pp.41-53, 2001.
[28] C. Y. Lee and X. Yao, Evolutionary programming using mutations based on the levy probability distribution, *IEEE Trans. Evol. Comput.*, vol.8, no.1, pp.1-13, 2004.
[29] Y. Wang and C. Dang, An evolutionary algorithm for global optimization based on level-set evolution and Latin squares, *IEEE Trans. Evol. Comput.*, vol.13, no.2, pp.454-472, 2007.
[30] S. Das, A. Abraham, U. K. Chakraborty and A. Konar, Differential evolution using a neighborhood-based mutation operator, *IEEE Trans. Evol. Comput.*, vol.13, no.3, pp.526-533, 2009.