Journal of Computers Vol. 29 No. 5, 2018, pp. 160-176
doi:10.3966/199115992018102905013

A Basin Escaping Auxiliary Function Method for Global Optimization

Haiyan Liu[1], Yuping Wang[1*], Liwen Liu[1]

[1] School of Computer Science and Technology, Xidian University,
South Taibai road, Xi'an, China
hyliu83@126.com; ywang@xidian.edu.cn; 378020179@qq.com

Received 7 June 2017; Revised 7 November 2017; Accepted 7 December 2017

Abstract. In this paper, a basin escaping auxiliary function method is proposed. This method adopts the idea of the filled function to move from a local minimum to better ones successively. The main advantages of the new proposed basin escaping auxiliary function are: (1) it is easy to construct, (2) it has no parameters to adjust, and (3) the formula expression of the new auxiliary function is quite simple which can result in less computation and hence can improve the efficiency. The properties of the auxiliary function is studied and discussed theoretically. And an evolutionary algorithm with a new crossover operator was proposed based on the basin escaping auxiliary function. The experiments are conducted on 8 often used benchmarks in the field and the results show that the proposed algorithm is more efficient.

Keywords: auxiliary function, basin escaping, global optimization, local minima escaping

1 Introduction

Many problems in science, economics and engineering involve global optimization [1, 2, 7]. Scientists have already proposed many useful ways to solve the global optimization problems. The classical gradient descent methods, Newton and quasi-Newton method are very efficient for problems with known or easy to get gradients. The population based evolutionary algorithm (EA) is another kind of optimization method inspired by the idea of Darwin's theory of evolution. In EA algorithms, points in the search space are called individuals or solutions and multiple individuals form the population which provide a parallel searching mechanism.

EA has developed very fast and a large number of different variants is proposed such as Particle Swarm Optimization (PSO) [10], Artificial Bee Colony Algorithm (ABC) [8] and Differential Evolution (DE) [21]. EAs are efficient and helpful especially when little knowledge of the optimization problem is known since they use the function value or transformed function value as the fitness instead of gradients. Also, the randomness mechanism in EA can help the algorithm to escape from local optima. However, EA may suffer from the premature convergence due to the lack of diversity of the population and may have slow convergence speed. Many efforts have been devoted to improve the efficiency of EA in variety ways such as using better initialization technique [9, 11, 29], auto enhancing the population diversity [26], making the evolutionary operator self-adaptive [20, 27], using elitism strategy [32-34].

Other efficient way to escape local optima is the filled function method first proposed by Ge [3-5]. Because of the prevalence of local minima, algorithms may often be trapped in local minima, or may enter other local minima worse than the former ones. So, a major problem is how to escape from local minima to find better ones successively till the global minima is found. The filled function method is an effective way to tackle this kind of problem.

Consider the following global optimization problem:

* Corresponding Author

$$\begin{aligned} & Min\, F(x) \\ & s.t. \quad x \in R^n \end{aligned} \tag{1}$$

where $F(x): \mathrm{R}^n \to R$ is a continuously differentiable function.

In practice, it is often assumed that $F(x)$ is coercive, that is:

$$F(x) \to +\infty \text{ when } \| x \| \to +\infty$$

So that all minima are in a closed bounded box $\Omega \subset R^n$ and there is only a finite number of minima in Ω. Usually, Ω is known or can be estimated before the optimization. So the above optimization problem is equivalent to the following one:

$$\begin{aligned} & Min\, F(x) \\ & s.t. \quad x \in \Omega = [l,u] = \{x \mid l \le x \le u, l, u \in R^n\} \end{aligned} \tag{2}$$

We will consider problem (2) in this paper and we use the following symbols for convenience:

n: The dimension of the problem.

x^* : The global optimal solution of the problem.

k : The iteration counter.

x_k^* : The local minimum of the problem in the kth iteration.

x_{ki}^* : The ith element of x_k^* where i=1,2, $\cdots$ n.

B_k^* : The basins of the objective function at an isolated local minimum x_k^*.

$F(x)$: The objective function.

The remainder of this article is organized as follows: Section 2 gives a brief overview of related work. Section 3 presents the basin escaping auxiliary function method and its properties. Based on it, a new basin escaping auxiliary function algorithm is proposed in section 4. Numerical experiments are carried out in section 5. Section 6 is the conclusions.

2 Related Work

In this section, we give an overview of the related work manly about the auxiliary function method and the elitism based evolutionary algorithm.

2.1 Auxiliary Function Method

Auxiliary function method is widely used in many field and in various ways. Some authors define functions for the fitness evaluation instead of the original function in evolutionary algorithms in order to make the algorithm more efficient [35]. More information about the transformed fitness functions for EA can be found in the survey paper [36]. Authors in [37-38] used transformed function in to help the algorithm to locate the global minimum potential energy of a Lennerd Joins cluster easily. Another two widely used auxiliary function method is the smooth function method and the filled function method.

The smoothing function method. The smoothing function method is first proposed by Wei and Wang [39] as follows:

$$\begin{aligned} & F(x,\tilde{x}) = f(\tilde{x}) + \frac{1}{2}\left\{1 - sign\left[f(x) - f(\tilde{x})\right]\right\} \bullet \left[f(x) - f(\tilde{x})\right] \\ & sign(y) = \begin{cases} -1 & if\ y < 0 \\ 1 & if\ y \ge 0 \end{cases} \end{aligned} \tag{3}$$

where $f(\mathrm{x})$ is the objective function to be optimized and $\tilde{x}$ is the current best point obtained by the algorithm. The smoothing function has two good properties:

(1) $F(x,\tilde{x})$ will keep $f(\mathrm{x})$ unchanged at any point x better than $\tilde{x}$, i.e., for $\forall x \in D$, if $f(\mathrm{x}) < f(\tilde{x})$,

then $F(x,\tilde{x}) = f(x)$.

(2) $F(x,\tilde{x})$ will flatten the landscape at any point no better than $\tilde{x}$, i.e., for $\forall x \in D$, if $f(\mathrm{x}) \geq f(\tilde{x})$, then $F(x,\tilde{x}) = f(\tilde{x})$.

The main idea of the smoothing function is to smooth the landscape of the objective function, that is, to reduce the number of local minima that are definitely won't be the global minima. Fig. 1 and Fig. 2 from [41] show the effect to function landscape before and after the use of smoothing filled function.

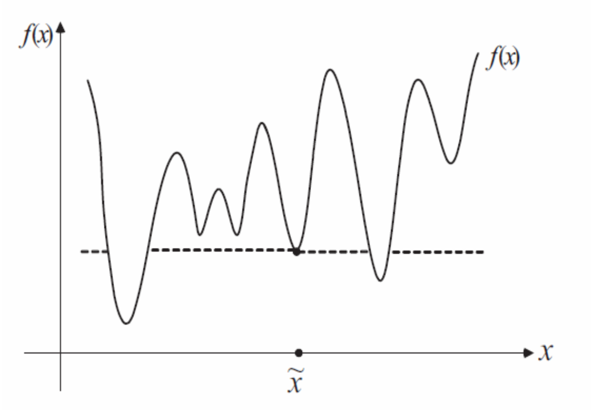

Fig. 1. The objective function to be optimized

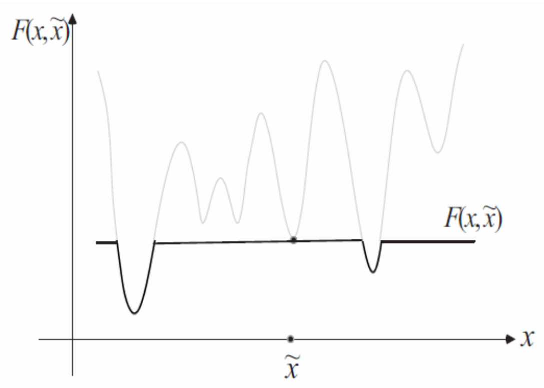

Fig. 2. The effect of the smoothing function

The smoothing function method is efficient in finding the global optimal solution by means of reducing the number of local minima. However, from Fig. 2 we can see that there are large flat areas that the gradient keeps unchanged in these areas. Gradient based algorithms may fail in such area. Authors in [40] proposed a spherical search method in which the searching radius is adjust to make attempt to walk out of the flat area. In [41], the authors use uniform design method to distribute points in these flat areas in order to get out of the flat areas.

The filled function method. Filled function method is another kind of auxiliary function aimed at escaping local minima successively to get the global optimal solutions. The definition of a filled function and the related concepts are first proposed by Ge [3] as follows:

Definition 1: The basin B_k^* of F(x) at an isolated minimum x_k^* defined in [3] is a connected domain which contains x_k^*, and in this domain the steepest descent trajectory of F(x) will converge to x_k^* by starting from any initial point.

A basin B_1^* at x_1^* is lower (or higher) than basin B_2^* at x_2^* iff:

$$F(x_1^*) \leq (\text{or} >) \mathrm{F}(\mathrm{x}_2^*) \tag{4}$$

The definition of a filled function is first proposed by Ge [3] as follows:

Definition 2: A function $P(x,x_1^*)$ is called a filled function of F(x) at a local minimum x_1^* if it satisfies the following properties:

(1) x_1^* is a strictly local maximum of $P(x,x_1^*)$, and the whole basin B_1^* of F(x) becomes a part of a hill of $P(x,x_1^*)$.

(2) $P(x,x_1^*)$ has no minima or stable points in any basin of F(x) higher than B_1^*.

(3) If F(x) has a lower basin than B_1^*, then there is a point x' in such a basin that minimizes $P(x,x_1^*)$ on the line through x and x_1^*.

From property (1) we can see that it is easy to escape from the local minimum x_1^* by minimizing the filled function $P(x,x_1^*)$. property (2) ensures the filled function will not find a worse local minimum by minimizing the filled function $P(x,x_1^*)$, and property (3) means that if a better local minimum than x_1^* exists, the filled function is able to enter a lower basin of F(x) in theory, thus it is possible to get a better local minimum by minimizing the filled function $P(x,x_1^*)$ along a line from x to x_1^*.

The first filled function method proposed by Ge [3] is:

$$P(x,r,\rho)=\frac{1}{r+F(x)}exp(-\frac{\|x-x_1^*\|^2}{\rho^2}) \tag{5}$$

Where r and ρ are two parameters need to be carefully chosen and adjusted to make $P(x,r,\rho)$ satisfy the above mentioned three properties of the filled function. Another disadvantage is that $P(x,r,\rho)$ will approach to zero very fast and its values change slowly when $\|x-x_1^*\|^2$ becomes larger and larger because of the term $[exp(-\|x-x_1^*\|^2/\rho^2)]$. So, when the value of $P(x,r,\rho)$ changes too slowly to distinguish, worse local minima(such as pseudo-minima or saddle points) could be located. To reduce the possibility of locating worse local minimum, the author use $\|x-x1^*\|$ instead of $\|x-x1^*\|^2$ as shown in equation (6)

$$\tilde{P}(x,r,\rho)=\frac{1}{r+F(x)}exp(-\frac{\|x-x_1^*\|}{\rho^2}) \tag{6}$$

Afterwards, many other filled function had been proposed to overcome these drawbacks [6, 13-18, 22, 24-25, 28, 30-31]. In [16] a filled function with only one parameter is proposed.

$$H(x)=\frac{1}{\ln(1+F(x)-F(x_1^*))}-a\|x-x_1^*\|^2 \tag{7}$$

However, the logistic operation will still make the filled function prone to ill conditioning. Liu [17] a filled function without exponential or logistic forms to eliminate this effect.

$$M(x,a)=\frac{1}{F(x)-F(x_1^*)^{1/m}}-a\|x-x1^*\|^2 \tag{8}$$

Then in [15] a class of continuously differentiable filled function was first proposed:

$$\begin{aligned}&C(X,a)=-u\left[f(X)-f(X_1)\right]w^a\left(\|X-X_1\|^p\right)\\&u(0)=0 \quad \omega(0)=1/a>0\\&u'(t)>0 \quad \omega'(t)>0,\quad \forall t\in[0,\infty)\\&\lim_{t\to 0}\frac{u(t)\omega'(t)}{u'(t)\omega(t)}=0\end{aligned} \tag{9}$$

And some other continuous differentiable filled functions have also been proposed [13-14, 24-25]. In [19], the authors proposed a parameter free filled function to overcome the disadvantage of tuning parameters.

$$P\left(x, x1^*\right) = -sign(\mathrm{F(x)} - \mathrm{F(x_1^*)})\arctan(\| x - x_1^* \|^2)$$
$$sign\left(t\right) = \begin{cases} 1, t \geq 0 \\ -1, t < 0 \end{cases} \tag{10}$$

However, it is still liable to getting trapped in local minima because of the $arctan(\| x - x^* \|)$ term. Also, the computation of $\| x - x_1^* \|^2$ is expensive as it needs n times of addition operation and n times of multiplication operation each time.

From the above introduction, we can get the conclusion that the existing filled functions are either prone to ill conditioning or have parameters not easy to tune. Also, these filled functions have complex in formula expression and need considerable computation resources. So, design a filled function with relatively easier formula expression and without any adjusting parameter to make is necessary. Also, it is not easy to construct filled functions satisfying property (3) of the definition of the filled function because it is not clear what is point x in property (3) of the filled function definition (definition 2)? Is it any point or a fixed point? It is confusing and not clear, so how to construct filled function satisfying property (3) is difficult. So, in this paper, we modify the definition of the filled function to make it easier to design functions that helps the algorithm to find optimal solutions. And we designed a parameter free auxiliary function method based on the new definition to solve global optimization problems.

Note that we call the new proposed function an basin escaping auxiliary function instead of filled function to make the distinguish between them.

2.2 Elitism Based Evolutionary Algorithm

EAs are seen as population based stochastic search algorithms inspired by the process of Biological evolution.

Because of the stochastic characteristic, there is no guarantee that the new generation after the population selection, crossover and mutation will be better than their parents generation. As an operational characteristic of GAs, elitism provides a way to reduce genetic drift [46]. The main idea is that the best individual(s) will be selected in any case and will pass to the next generation. Elitism has been shown to improve the performance and convergence of the genetic algorithms [47]. Elitism based evolutionary algorithms is widely used [32-34, 42-44]. In literature [32], a genetic algorithm with elitism is proposed to solve the transit network design problem. Authors in [33] proposed a immune genetic algorithm with the elitist selection and elitist crossover operator. Authors in [34] proposed two Elitism-Based Compact Genetic Algorithms to solve difficult optimization problems. The elitism mechanism is also adopted in multi objective optimization problems [42-44]. In this paper, we also use elitism mechanism and an elitism based crossover operator is designed to help the algorithm for optimization.

3 The Basin Escaping Auxiliary Function and Its Properties

The filled function method offers the insight of escaping from local minima which is the key problem of some optimization algorithms. It can help the algorithm gradually move from a local minimum to better ones using the following mechanism:

(1) First, from a local minimum x_1^* of the original function F(x) a filled function $P(x, x_1^*)$ is constructed at x_1^*.

(2) Then a minimization process is executed on the filled function $P(x, x_1^*)$ to get its local minimum x' from an initial point y_1 near x_1^*.

From property (1), we can see that $P(x, x_1^*)$ at y_1 has a descent direction and thus the minimization of $P(x, x_1^*)$ can be easily executed by any descent direction method such as the steepest descent method.

Furthermore, x' will not be in a basin higher (worse) than B_1^* which is guaranteed by property (2).

(3) Finally, by minimizing F(x) from the initial point x' we can get a better local minimum x_2^* of F(x) by property (3).

Repeat this process until no better minimum of F(x) can be found.

Property (3) ensures that the filled function will enter the basin contains a better local minimum in theory, however, property (3) is not clear and confusing. Also, in practice with the limit of computation resources and the ability of the search method, this will not fulfill every time. That may be one of the reasons that the filled function method can't get the global minima in a 100% success rate.

To overcome the drawbacks of the definition of the filled function and make it easier to construct functions to assist the optimization algorithm, we use the evolutionary algorithm with uniformly distributed population and multiple elite mechanism to replace property (3) which has the similar functionality. We will call this kind of function basin escaping auxiliary function because it adopts the main concept of the filled function to escape local minima.

Accordingly, we propose a new parameter free basin escaping auxiliary function (PF_AF briefly) as follows:

$$PF\_AF(x,x_k^*) = A^T x + F(x_k^*) - A^T x_k^* \tag{11}$$

where $A = (A_1, A_2, \cdots, A_n)^T$ is a column vector with the same dimension as x, and A^T is the transpose of A.

A is defined as follows:

$$A_i = \begin{cases} -1 & (x_i - x_i^{k*})[F(x) - F(x_k^*)] \geq 0 \\ 1 & (x_i - x_i^{k*})[F(x) - F(x_k^*)] < 0 \end{cases} \tag{12}$$

In the following, we will prove that the PF_AF proposed above has the following good properties.

Theorem 1: Suppose x_k^* is a local minimum of F(x) and $PF\_AF(x,x_k^*)$ is a auxiliary function at x_k^*, then x_k^* is a strictly local maximum of $PF\_AF(x,x_k^*)$.

Proof: Let B_k^* be the basin of F(x) at x_k^*. For $\forall x \in B_k^*, x \neq x_k^*$, we need to prove that

$PF\_AF(x_k^*,x_k^*) > PFF(x,x_k^*)$, that is $PF\_AF(x,x_k^*) - PFF(x_k^*,x_k^*) < 0$.

Since $PF\_AF(x_k^*,x_k^*) = F(x_k^*)$,

and $PF\_AF(x,x_k^*) = A^T x + F(x_k^*) - A^T x_k^* = F(x_k^*) + A^T (x - x_k^*)$,

We only need to prove that $A^T (x - x_k^*) < 0$.

Note that $A^T (x - x_k^*) = \sum_{i=1}^{n} A_i^T (x_i - x_{ki}^*)$, and Since x_k^* is a strictly local minimum of F(x), we have $F(x) > F(x_k^*)$ for any x in B_k^* with $x \neq x_k^*$.

From equation (12), we know that:

When $x_i - x_{ki}^* < 0$, we get $A_i = 1$. In this case $A_i^T (x_i - x_{ki}^*) < 0$.

When $x_i - x_{ki}^* > 0$, we get $A_i = -1$. In this case $A_i^T (x_i - x_{ki}^*) < 0$.

When $x_i - x_{ki}^* = 0$, we get $A_i = -1$. In this case, $A_i^T (x_i - x_{ki}^*) = 0$.

Thus each term $A_i^T (x_i - x_{ki}^*) \leq 0$ in $A^T (x - x_k^*) = \sum_{i=1}^{n} A_i^T (x_i - x_{ki}^*)$.

Note that $x \neq x_k^*$, we know that there exists at least one i such that $x_i - x_{ki}^* > 0$ or $x_i - x_{ki}^* < 0$.

Thus there exists at least one i such that $A_i^T (x_i - x_{ki}^*) < 0$.

Therefore $A^T (x - x_k^*) = \sum_{i=1}^{n} A_i^T (x_i - x_{ki}^*) < 0$.

That is x_k^* is a strictly local maximum of $PF\_AF(x,x_k^*)$.

Theorem 2: Suppose x_k^* is a local minimum of $F(x)$, $\Omega_1=\{x\in\Omega \mid F(x)\geq F(x_k^*), x\neq x_k^*\}$, then $PF\_AF(x,x_k^*)$ has no stationary points or saddle points in Ω_1.

Proof: there are two classes of points in Ω_1:

Class 1: $\{x \mid (x_i-x_{ki}^*)[F(x)-F(x_k^*)]=0, x\in\Omega_1\}$

Class 2: $\{x \mid (x_i-x_{ki}^*)[F(x)-F(x_k^*)]\neq 0, x\in\Omega_1\}$

Obviously, $PF\_AF(x,x_k^*)$ is not differentiable on the points in class 1. Except points in class 1, $PF\_AF(x,x_k^*)$ is continuous differentiable, and $\nabla PF\_AF(x,x_k^*)=A\neq 0$.

From the above property, we can see that the basin escaping auxiliary function has the main advantage of the filled function which can help the algorithm to escape local minima easily. Also, the new auxiliary function has no parameter to tune which overcome the disadvantage of the adjustment of parameters on different kind of problems which is not an easy task. In the following section, we propose an evolutionary algorithm based on the basin escaping auxiliary function to solve global optimization problem.

4 The New Basin Escaping Auxiliary Function Algorithm

In this section, we proposed an elitism-based evolutionary algorithm using the basin escaping auxiliary function.

4.1 A New Elitism-based Crossover Operator

Elitism selection is the mechanism to keep the best individual(s) in the current generation to the next generation in order to ensure the solution quality will not degrade. In this paper, we adopt the idea of elitism, and propose a elitism-based crossover operator. The main idea is: from the best individual of one generation may not generate the best solution in the next generation because it may be a local minimum. So we keep multiple best individuals in the hope that they will converge to different basins containing better minima so as to increase the chance of getting better solution in the next generation.

To further ensure the possibility of getting better solution when the algorithm stagnate with the improvement less than a pre-set threshold, we generate the next generation using the following way:

First, we set a neighborhood in the right hand (with higher value) of the elite solution:

$$u=elite(i)+[ubound(i)-lbound(i)]/(\mathrm{n}*10) \qquad \textbf{(13)}$$

where $i=1,2,...,n$ is the dimension of the problem, *ubound* and *lbound* is the upper and lower boundary of the problem respectively.

Then we generate a vector for crossover:

$$vector=(u-best\_x)*\mathrm{rand(n,1)} \qquad \textbf{(14)}$$

The new individual will be generated using the following equation:

$$\begin{aligned}&for\ j=1:popsize\\&newIndividule(j)=elite+vector;\\&end\end{aligned} \qquad \textbf{(15)}$$

If the solution is still not improved, then we will set a neighborhood in the left hand (with smaller value) of the elite solution in the similar way:

$$l=elite(i)-[ubound(i)-\mathrm{lbound(i)}]/(\mathrm{n}*10) \qquad \textbf{(16)}$$

where $i=1,2,...,n$ is the dimension of the problem, *ubound* and *lbound* is the upper and lower boundary of the problem respectively.

Then we generate a vector for crossover:

$$vector = (best\_x - l) * \mathrm{rand(n,1)} \tag{17}$$

Similarly, the new individual will be generate using the equation (9).

Then we choose the best individual from the crossover operator as the initial point for the basin escaping auxiliary function method in the next subsection.

4.2 The Basin Escaping Auxiliary Function Algorithm

Based on the above discussion, a new basin escaping auxiliary function algorithm is proposed as follows:
Step 1. (initialization). Randomly and uniformly distribute NP points in Ω, set $\epsilon = 1.0e-10$ as the stopping criteria. We use best_x and best_val to record the best point and its function value we found so far. Initially we set best_val to infinity and best_x to null.
Step 2. Evaluate the function values of the initial points and put the best m points that have the smallest function values into a set S.

For example when m=3, suppose $S = \{x_0, x_1, x_2\}$ in which x_0 is the best point and followed by the second-best x_1 and third-best x_2.

For small scale problems $(n < 10)$, we recommend $m = 1$, and for problems with dimension $n > 10$, we recommend $m = 3$.
Step 3. Minimize $F(x)$ starting from the points in S respectively, we get the corresponding local minima of $F(x)$ and their function values, we choose the point with the best(lowest) function value as x_k^*, Then go to step 6.
Step 4. Construct the basin escaping auxiliary function at x_k^*

$$PF\_AF(x, x_K^*) = A^T x + F(x_k^*) - A^T x_k^*$$

Where A is a coefficient column vector as shown above in equation (12).
Step 5. Make a tiny disturbance on x_k * and then minimize $PF\_AF(x, x_k^*)$ using the disturbed point.

We put m best points of $PF\_AF(x, x_k^*)$ into set S. Then go to step 3.
Step 6. (termination). If $F(x_k^*) - best\_val < -\epsilon$ then update best_x to x_k^* and best_val to $F(x_k^*)$. Go to step 4. Otherwise set $x^* = best\_x$ as the global minimum and the algorithm will stop.

Note that we always record the best point and its function value we have found so far in each cycle and will update them if a better point appears, for example we use best_x and best_val to record them.

The reason we use uniformly distributed points other than randomly given points in some literature is that, for an unknown problem, it has more opportunity to get closer to the local or even global minimum and have a better diversity of points which can help the algorithm to obtain good results. For detailed information about uniform design, the readers can refer to [12, 23].

5 Numerical Experiment

In this section, the proposed algorithm is implemented in Matlab R2013b and is tested on some widely used benchmark problems. The result is compared with that of literature [13, 28]. The reason why the new basin escaping auxiliary function can save computational resources is analyzed.

5.1 The Benchmark Functions

The benchmark functions taken from literature [13, 28] are listed below:
Two-dimensional function.

$$\begin{aligned} &\min \quad F(x) = [1 - 2x_2 + c\sin(4\pi x_2) - x_1]^2 + [x_2 - 0.5\sin(2\pi x_1)]^2 \\ &s.t. \quad 0 \le x_1 \le 10, \quad -10 \le x_2 \le 0 \end{aligned} \tag{18}$$

where c=0.2, 0.5 and 0.05. The global minimum value is F(x*)=0 for all c.

Three-hump back camel function.

$$\begin{aligned} &\min \quad F(x) = 2x_1^2 - 1.05x_1^4 + \frac{1}{6}x_1^6 - x_1x_2 + x_2^2 \\ &s.t. \quad -3 \le x_1 \le 3, \quad -3 \le x_2 \le 3 \end{aligned} \tag{19}$$

The global minimum solution is x*= (0, 0)^T and F(x*)=0.

Six-hump back camel function.

$$\begin{aligned} &\min \quad F(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4 \\ &s.t. \quad -3 \le x_1 \le 3, \quad -3 \le x_2 \le 3 \end{aligned} \tag{20}$$

The global minimum solution is x*=(-0.0898, -0.7127)^T or x*=(0.0898, 0.7127)^T and F(x*)=-1.0316.

Treccani function.

$$\begin{aligned} &\min \quad F(x) = x_1^4 + 4x_1^3 + 4x_1^2 + x_2^2 \\ &s.t. \quad -3 \le x_1 \le 3, \quad -3 \le x_2 \le 3 \end{aligned} \tag{21}$$

The global minimum solution is x*=(-2, 0)^T and F(x*)=0.

Goldstein-Price function.

$$\begin{aligned} \min \quad F(x) = &\left[1 + (x_1 + x_2 + 1)^2 \left(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2\right)\right] \\ &\times \left[30 + (2x_1 - 3x_2)^2 \left(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2\right)\right] \\ s.t. \quad -3 \le x_1 \le 3,& \quad -3 \le x_2 \le 3 \end{aligned} \tag{22}$$

The global minimum solutions are x*=(0, -1)^T and F(x*)=3.

Shubert function.

$$\begin{aligned} &\min \quad F(x) = \left\{\sum_{i=1}^{5} i\cos\left[(i+1)x_1\right] + i\right\}\left\{\sum_{i=1}^{5} i\cos\left[(i+1)x_2\right] + 1\right\} \\ &s.t. \quad 0 \le x_1 \le 10, \quad 0 \le x_2 \le 10 \end{aligned} \tag{23}$$

This function has 760 minima in total. The global minimum value is F(x*)=-186.7309. The illustration of this function for n=2 is shown in Fig. 3.

Shekel function.

$$\begin{aligned} &\min \quad F(x) = -\sum_{i=1}^{5}\left[\sum_{j=1}^{n}\left(x_j - a_{ij}\right)^2 + c_i\right]^{-1} \\ &s.t. \quad 0 \le x_i \le 10, \quad i = 1, 2, \cdots 4. \end{aligned} \tag{24}$$

Where c= (0.1, 0.2, 0.3, 0.4, 0.5) and

$$a = \begin{bmatrix} 4.0 & 1.0 & 8.0 & 6.0 & 3.0 \\ 4.0 & 1.0 & 8.0 & 6.0 & 7.0 \\ 4.0 & 1.0 & 8.0 & 6.0 & 3.0 \\ 4.0 & 1.0 & 8.0 & 6.0 & 7.0 \end{bmatrix}$$

The global minimum solutions are x* = (4, 4, 4, 4)^T, and F(x*) is -10.1532.

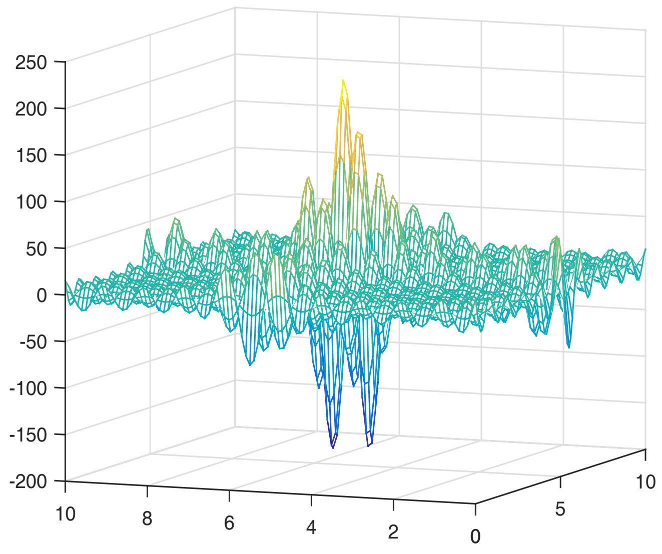

Fig. 3. Shubert function

n-dimensional function.

$$\begin{aligned}&\min \quad F(x)=\frac{\pi}{n}[10\sin^2 \pi \mathrm{x}_1+\mathrm{g(x)}+(\mathrm{x}_n-\mathrm{x})^2]\\&\mathrm{s.t.} \quad -10\le x_i\le 10, \quad i=1,2,...n.\end{aligned} \tag{25}$$

Where $g(x)=\sum_{i=1}^{n-1}[(\mathrm{x}_i-1)^2(1+10\sin^2\pi\,\mathrm{x}_{i+1})]$

The global minimum solution is x*=(1, 1, …1)$^{\mathrm{T}}$ for all n. The illustration of the function for n=2 is shown in Fig. 4.

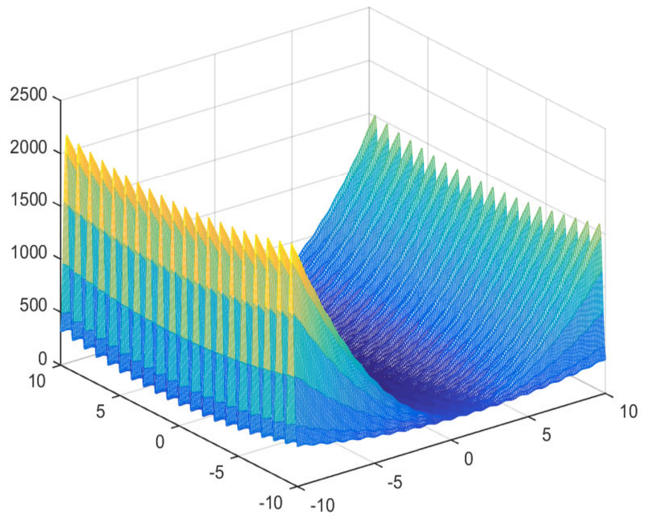

Fig. 4. Sine square function

5.2 Experimental Results

The proposed basin escaping auxiliary function method is tested on the above 8 problems, and the performance is compared with the two algorithms in literature [13, 28]. The results are listed in Table 1 to Table 11. In these tables, the following symbols are used:

K: The iteration number in finding the kth local minimum of the objective function.

x_k^* : The kth local minimum of the objective function.

f_k^* : The function value at x_k^*

PF_AF: The proposed algorithm in this paper.

CDFA: The algorithm proposed in [13]

NFA: The algorithm proposed in [28]

From Table 1 to Table 11, we can see that all the three algorithms can find the global optimum solutions for all the 8 test problems but with different iteration numbers. The proposed PF_AF algorithm outperforms both CDFA and NFA in problem 1 with c=0.2 and 0.05, problem 2, problem 4, problem 5 and problem 6. Note that for problem 4, PF_AF uses 2 iterations to find one optimum solution which is the same as the iteration of CDFA and NFA, But PF_AF continues to find another different optimum solution $x^* = (-2,0)^T$. So we can say that PF_AF performs better than CDFA and NFA on problem 4. For problem 3 and problem 7, PF_AF performed equally well as CDFA and NFA. For problem 1 with c=0.5, PF_AF is as good as CDFA but better than NFA. For problem 8 with n=7 and n=10, PF_AF is as good as NFA but takes one more iteration to find the global optimum solution than CDFA. So, we can get a conclusion that the proposed PF_AF algorithm is more efficient than CDFA in [13] and NFA in [28]. The reason partially lies in that, the auxiliary function designed in the proposed PF_AF algorithm is simpler and saved some computational resources. The complexity of the auxiliary function is further animalized in section 5.3.

Table 1. Results of benchmark function 1 with c=0.2.

	PF_AF		CDFA		NFA	
k	x_k^*	f_k^*	x_k^*	f_k^*	x_k^*	f_k^*
1	$(2.1086,-0.1611)^T$	1.1603	$(5.7221,-1.8806)^T$	2.5070	$(5.7221,-1.8806)^T$	2.5070
2	$(1.8784,-0.3459)^T$	2.3602e-14	$(3.7387,-1.2649)^T$	0.6165	$(4.7387,-1.7417)^T$	1.6212
3			$(1.5909,-0.2703)^T$	2.8126e-9	$(4.7096,-1.3985)^T$	1.3566
4					$(3.7387,-1.2649)^T$	0.61647
5					$(2.7380,-0.78836)^T$	0.088673
6					$(1.8784,-0.34585)^T$	0

Table 2. Results of benchmark function 1 with c=0.5

	PF_AF		CDFA		NFA	
k	x_k^*	f_k^*	x_k^*	f_k^*	x_k^*	f_k^*
1	$(1.4527,-0.2833)^T$	0.2851	$(0.0420,-0.0948)^T$	0.5175	$(0.042023,-0.094772)^T$	0.51745
2	$(1.8974,-0.3005)^T$	4.8219e-15	$(1.0000,0)^T$	5.7949e-016	$(0.99991e-4,-1.2524e-4)^T$	2.2389e-7
3					$(1.0000e-14,-2.2205e-14)^T$	0

Table 3. Results of benchmark function 1 with c=0.05

	PF_AF		CDFA		NFA	
k	x_k^*	f_k^*	x_k^*	f_k^*	x_k^*	f_k^*
1	$(4.7978,-1.2792)^T$	2.1343	$(8.7299,-3.2965)^T$	9.0733	$(8.7299,-3.2965)^T$	9.0733
2	$(4.7129,-1.4891)^T$	1.5351	$(7.7280,-0.4022)^T$	6.5031	$(7.7280,-2.8347)^T$	6.5031
3	$(1.8513,-0.4021)^T$	1.3220e-14	$(1.8513,-0.4021)^T$	4.3885e-011	$(6.7248,-2.3724)^T$	4.3943
4					$(5.7198,-1.9162)^T$	2.7434
5					$(4.7129,-1.4891)^T$	1.5351
6					$(3.7305,-1.2306)^T$	0.61844
7					$(2.7300,-0.79341)^T$	0.10216
8					$(1.8513,-0.40209)^T$	0

Table 4. Results of benchmark function 2

	PF_AF		CDFA		NFA	
k	x_k^*	f_k^*	x_k^*	f_k^*	x_k^*	f_k^*
1	(-0.3922,-0.1359)^T	0.2486	(−1.7476,−0.8738)^T	0.2986	(−1.7476,−0.87378)^T	0.29864
2	(0.1299e-6,0.1336e-6)^T	3.4245e-14	(−0.0000,−0.0000)^T	4.0157e-010	(0,0)^T	0

Table 5. Results of benchmark function 3

	PF_AF		CDFA		NFA	
k	x_k^*	f_k^*	x_k^*	f_k^*	x_k^*	f_k^*
1	(-0.6700,-0.7126)^T	-0.0747	(1.6071,-0.5687)^T	2.1043	(1.6071,-0.56865)^T	2.1043
2	(-0.0898,-0.7127)^T	-1.0316	(-0.0898,-0.7127)^T	−1.0316	(-0.089842,-0.71266)^T	−1.0316

Table 6 Results of benchmark function 4

	PF_AF		CDFA		NFA	
k	x_k^*	f_k^*	x_k^*	f_k^*	x_k^*	f_k^*
1	(0.0051,-0.0791)T	0.0064	(-1.0000,0)T	1.0000	(-1.0000,0)T	1.0000
2	(0.7307e-8,-0.7343e-8)T	2.6748e-16	(-0.0000,-0.0000)T	2.4048e-017	(0,0)T	0
3	(0.0109e-7,-0.1147e-7)T	1.3639e-16				
4	(-2.0000,0.0000)T	1.1740e-18				

Table 7. Results of benchmark function 5

	PF_AF		CDFA		NFA	
k	x_k^*	f_k^*	x_k^*	f_k^*	x_k^*	f_k^*
1	(0.1083,-0.9229)T	6.3570	(-0.6000,-0.4000)T	30.000	(-0.60000,-0.40000)T	30.000
2	(0.0000,-1.0000)T	3.0000	(0.0000,-1.0000)T	3.0000	(0,-1.0000)T	3.0000

Table 8. Results of benchmark function 6

	PF_AF		CDFA		NFA	
k	x_k^*	f_k^*	x_k^*	f_k^*	x_k^*	f_k^*
1	(-0.8940,-1.4518)^T	-166.0075	(2.0467,2.0467)^T	0	(1.0865,1.0865)^T	2.8841e-17
2	(-0.8003,-1.4251)^T	-186.7309	(3.2800,4.8581)^T	−46.511	(1.3200e-12,1.8703e-12)^T	−13.052
3			(4.2760,4.8581)^T	−79.411	(1.3200,4.8581)^T	−37.681
4			(5.4892,4.8581)^T	−186.739	(3.2800,4.8581)^T	−46.511
5					(4.2760,4.8581)^T	−79.411
6					(5.4892,4.8581)^T	−186.73

Table 9. Results of benchmark function 7

	PF_AF		CDFA		NFA	
k	x_k^*	f_k^*	x_k^*	f_k^*	x_k^*	f_k^*
1	(3.5672,4.1449 3.2483,3.9304)^T	-1.2859	(1.0001,.0002 1.0001,1.0002)^T	−5.0552	(1.0001,.0002 1.0001,1.0002)^T	−5.0552
2	(4.0000,4.0001 4.0000,4.0001)^T	-10.1532	(4.0000,4.0000 4.0000,4.0000)^T	−10.1529	(4.0000,4.0001 4.0000,.0001)^T	−10.153

Table 10. Results of benchmark function 8 with n=7

	PF_AF		CDFA		NFA	
k	x_k^*	f_k^*	x_k^*	f_k^*	x_k^*	f_k^*
1	(6.0000,1.0060, 0.7933,-1.9449, -1.5355,2.0160, 8.0848)^T	79.8858	(1.0000,1.0000, 1.0000,1.0000, 1.0000,1.0000, 1.0000)^T	2.3538e-013	(1.9899,1.9897, 1.9896,1.9896, 1.9896,1.9896, 1.9898)^T	−3.1095
2	(1.0000,1.0000, 1.0000,1.0000, 1.0000,1.0000, 1.0000)^T	3.9121e-13			(1.0000,1.0000, 1.0000,1.0000, 1.0000,1.0000, 1.0000)^T	0

Table 11. Results of benchmark function 8 with n=10

	PF_AF		CDFA		NFA	
k	x_k^*	f_k^*	x_k^*	f_k^*	x_k^*	f_k^*
1	(-8.5205,2.9287 3.5550,-3.0893 2.0085,1.5289 -2.9035,3.2940 -3.4533,-4.4657)^T	200.9943	(0.0101,0.0103, 0.0103,0.0104 0.0103,0.0102, 1.0000,6.0000 6.0000,6.0000)^T	2.6653	(5.9490,5.9979, 5.9980,5.9980 5.9980,5.9980, 5.9980,5.9980 5.9980,5.9980)^T	78.432
2	(-6.9159,9.9985 7.9991,0.0002 0.0110,3.9690 0.0019,-6.9156 0.0002,-0.9799)^T	85.9293	(1.1615,0.1651, 0.4418,0.9258 0.9638,-0.4809, 0.9926,6.0000 6.0000,6.0000)^T	2.4443	(-1.9696,0.9943, 5.9980,5.9980 5.9980,5.9980, 5.9980,5.9980 5.9980,5.9980)^T	73.450
3	(1.0145,-0.7174 0.9612,1.5399 1.1809,0.9754 1.0759,1.1073 1.2214,8.8954)^T	21.0815	(1.9900,0.0000, 1.0000,1.0000 1.0000,1.0000, 1.0000,6.0000 6.0000,6.0000)^T	0.4443	(-0.97956,5.9871, 5.9980,5.9980, 5.9980,5.9980, 5.9980,5.9980 5.9980,5.9980)^T	71.884
4	(0.9985,0.6487 1.9483,1.9937 -1.9745,0.9985 1.1498,-0.8454 1.0115,2.8191)^T	5.5888	1.0000,1.0000, 1.0000,1.0000 1.0000,1.0000, 1.0000,1.0000 1.0000,1.0000	0	(0.012709,5.9476, 5.9979,5.9980 5.9980,5.9980, 5.9980,5.9980 5.9980,5.9980)^T	70.890
5	(1.0000,1.0000, 1.0000,1.0000, 1.0000,1.0000, 1.0000,1.0000, 1.0000,1.0000)^T	3.7571e-13			(1.0000,1.0000, 1.0000,1.0000 1.0000,1.0000, 1.0000,1.0000 1.0000,1.0000)^T	0

Note that there is a slight difference between PF_AF and the other two algorithms: CDFA and NFA. The result of CDFA and NFA is based on a given initial point as shown in Table 12, while PF_AF uses the best point out of uniformly random distributed points which is more flexible. To compare with CDFA and NFA, PF_AF use m=1 in the algorithm which means only one point is calculated in the set S each time. But setting a larger m value can get better performance for some problems.

Table 12. The initial points used in CDFA and NFA

Problem number	x_0	Problem number	x_0
problem 1 with c=0.2	$x_0=(6,-2)^T$	problem 1 with c=0.5	$x_0=(0, 0)^T$
problem 1 with c=0.05	$x_0=(10, -10)^T$	problem 2	$x_0=(-2, -1)^T$
problem 3	$x_0=(-2, 1)^T$	problem 4	$x_0=(-1, 0)^T$
problem 5	$x_0=(-1, -1)^T$	problem 6	$x_0=(1, 1)^T$
problem 7	$x_0=(1, 1, 1, 1)^T$	problem 8 with n=7	$x_0=(2, 2, 2, 2, 2, 2, 2)^T$
problem 8 with n=10	$x_0=(6, 6, 6, 6, 6, 6, 6, 6, 6, 6)^T$		

5.3 The Complexity Analysis of the Basin Escaping Auxiliary Function

In this subsection, we analyze and explain why PF_AF is easier and less computationally expensive than NFA and CDFA. As is well known, in computers multiplication operation is implemented by repeatedly adding and shifting bits. So the cost of multiplication operation is much higher than addition. Take Pentium for example, an addition operation takes one clock cycle while a multiplication operation takes 10 clock cycles.

Table 13 shows the three filled functions' number of operations in one function evaluation. We can see that for one function evaluation, PF_AF needs 3+2n operations in total which is less than NFA's 9+2n and CDFA's 8+2n. Moreover, PF_AF does not need any multiplication operation while both NFA and CDFA needs 3+n times for each function evaluation. For one function evaluation, the differences among the three function is not significant at all. However, Note that whenever a local minimum x*k of the objective function is found, the filled function will be constructed and then many function evaluations (usually more than 103) of the filled function will be calculated to find the local minimum in this iteration. The accumulation of many iterations will greatly enlarge the differences.

Note that the calculation of $\| x - x_1^* \|^2$ needs n times of addition and n times of multiplication. The opposite operation is not counted. And for piecewise function, we count the worst situation.

Table 13. number of operations in one function evaluation

function	+ or -	×	÷	total
NFA	6+n	3+n	0	9+2n
CDFA	3+n	3+n	2	8+2n
PF_AF	3+2n	0	0	3+2n

The proposed algorithm is run on a desktop computer with Intel (R) Core (TM) i7-3770 CPU @3.4GHZ. Figure 5 shows the convergence curve on function 8 with dimension 10. The x axis is the time used in seconds and the y axis is the function value. From this figure, we can see that the proposed algorithm takes very little time to get the global optimal solution.

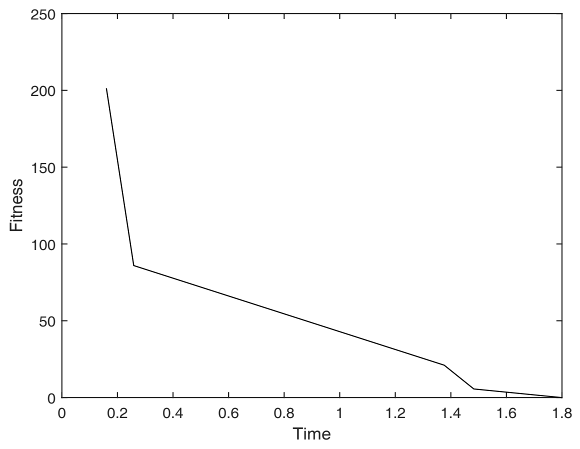

Fig. 5. Convergence curve on function 8

6 Conclusions

In this paper, a basin escaping auxiliary function method is proposed to solve multimodal global optimization problems. We adopt the idea of the filled function method of moving from local minima to better ones and proposed a basin escaping auxiliary function that overcome the disadvantage of the filled function: not easy to construct, with relatively complex forms and sensitive to parameters which are not so easy to tune. The new basin escaping auxiliary function is parameter free and with quite simple formula expression which can result in less computation and hence can improve the efficiency. The reason why it is less computational expensive is analyzed. Also, the new auxiliary function does not contain exponential or logistic forms which are prone to ill-conditioning. Based on it, a new evolutionary algorithm with a new crossover operator is proposed. Numerical experiments are carried out on some widely used benchmark functions. The results show that the new proposed algorithm is effective and can save time because of its simplicity.

The proposed algorithm is designed for differentiable functions and it is only effective for small scale (low dimension) problems as other filled function algorithm. So our future work is to extend it to larger scale problems.

Acknowledgements

This work was supported by National Natural Science Foundation of China (No. 61472297 and No. 61402350) and the Fundamental Research Funds for the Central Universities (BDZ021430).

References

[1] E.F. Campana, G. Liuzzi, S. Lucidi, D. Peri, V. Piccialli, A. Pinto, New global optimization methods for ship design problems, Optimization and Engineering 10(4)(2009) 533-555.

[2] C. Dang, W. Ma, J. Liang, A deterministic annealing algorithm for approximating a solution of the min-bisection problem, Neural Networks 22(1)(2009) 58-66.

[3] R.P. Ge, A filled function method for finding a global minimizer of a function of several variables, Mathematical Programming 46(1-3)(1990) 191-204.

[4] R.P. Ge, The theory of filled function method for finding global minimizers of nonlinearly constrained minimization problems, Journal of Computational Mathematics 5(1987) 1-9.

[5] R.P. Ge, Y.F. Qin, A class of filled functions for finding global minimizers of a function of several variables, Journal of Optimization Theory and Applications 54(2)(1987) 241-252.

[6] S. He, W. Chen, H. Wang, A new filled function algorithm for constrained global optimization problems, Applied Mathematics and Computation 217(12)(2011) 5853-5859.

[7] R. Qing-dao-er-ji, Y. Wang, A new hybrid genetic algorithm for job shop scheduling problem, Computers & Operations Research 39(10)(2012) 2291-2299.

[8] D. Karaboga, An idea based on honey bee swarm for numerical optimization, Vol. 200. Technical Report-tr06, 2005.

[9] B. Kazimipour, X. Li, A.K. Qin, Initialization methods for large scale global optimization, in: Proc. Evolutionary Computation (CEC), 2013.

[10] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Proc. IEEE International Conference on Neural Networks, 1995.

[11] Y.-W. Leung, Y. Wang, An orthogonal genetic algorithm with quantization for global numerical optimization, IEEE Transactions on Evolutionary Computation 5.1(2001) 41-53.

[12] Y.W. Leung, Y. Wang, Multiobjective programming using uniform design and genetic algorithm, IEEE Transactions on System, Man, and Cybernetics- Part C: Application and Reviews 30(3)(2000) 293-304.

[13] H. Lin, Y. Gao, Y. Wang, A continuously differentiable filled function method for global optimization, Numerical Algorithms 66(3)(2014) 511-523.

[14] H. Lin, Y. Wang, L. Fan, A filled function method with one parameter for unconstrained global optimization, Applied Mathematics and Computation 218(7)(2011) 3776-3785.

[15] X. Liu, A class of continuously differentiable filled functions for global optimization, IEEE Transactions on Systems, Man,

and Cybernetics-Part A: Systems and Humans 38(1)(2008) 38-47.

[16] X. Liu, Finding global minima with a computable filled function, Journal of Global Optimization 19(2)(2001) 151-161.

[17] X. Liu, W. Xu, A new filled function applied to global optimization, Computers & Operations Research 31(1)(2004) 61-80.

[18] S. Lucidi, V. Piccialli, New classes of globally convexized filled functions for global optimization, Journal of Global Optimization 24(2)(2002) 219-236.

[19] S. Ma, Y. Yang, H. Liu, A parameter free filled function for unconstrained global optimization, Applied Mathematics and Computation 215(10) 3610-3619.

[20] A.K. Qin, P.N. Suganthan, Self-adaptive differential evolution algorithm for numerical optimization, in: Proc. 2005 IEEE Congress on Evolutionary Computation, 2005.

[21] R. Storn, K. Price, Differential evolution- A simple and efficient heuristic for global optimization over continuous spaces, Journal of Global Optimization 11(4)(1997) 341-359.

[22] W. Wang, Y. Shang, L. Zhang, A filled function method with one parameter for box constrained global optimization, Applied Mathematics and Computation 194(1)(2007) 54-66.

[23] Y. Wang, K.T. Fang, A note on uniform distribution and experimental design, Chinese Science Bulletin 6(1981) 485-489.

[24] F. Wei, Y. Wang, A new filled function method with one parameter for global optimization. <https://www.hindawi.com/journals/mpe/2013/532325/>, 2013.

[25] F. Wei, Y. Wang, H. Lin. A new filled function method with two parameters for global optimization, Journal of Optimization Theory Applications 163(2)(2014) 510-527.

[26] Y. Yang, Y. Gao, A new filled function method for global optimization, in: Proc. 2015 IEEE International Conference on Digital Signal Processing (DSP), 2015.

[27] Z. Yang, K. Tang, X. Yao, Self-adaptive differential evolution with neighborhood search, in: Proc. 2008 IEEE World Congress on Computational Intelligence, 2008.

[28] L.-S. Zhang, C.-K. Ng, D. Li, W.-W. Tian., A New Filled Function Method for Global Optimization, Journal of Global Optimization 28(1)(2004) 17-43. doi:10.1023/B:JOGO.0000006653.60256.f6

[29] Q. Zhang, Y.W. Leung, An orthogonal genetic algorithm for multimedia multicast routing, IEEE Transactions on Evolutionary Computation 3(1) 53-62.

[30] Y. Zhang, Y.T. Xu, A one-parameter filled function method applied to nonsmooth constrained global optimization, Computers & Mathematics with Applications 58(6)(2009) 1230-1238.

[31] Y. Zhang, L.-S. Zhang, Y. Xu, New filled functions for non-smooth global optimization, Applied Mathematical Modelling 33(7)(2009) 3114-3129.

[32] M.A. Nayeem, M.K. Rahman, M.S. Rahman, Transit network design by genetic algorithm with elitism, Transportation Research Part C: Emerging Technologies 46(2014) 30-45.

[33] G.-Z. Tan, D.-M. Zhou, B. Jiang, M.I. Duiybate, Elitism-based immune genetic algorithm and its application to optimization of complex multi-modal functions, Journal of Central South University of Technology 15(6)(2008) 845-852.

[34] C.W. Ahn, R.S. Ramakrishna, Elitism-based compact genetic algorithms, IEEE Transactions on Evolutionary Computation 7(4)(2003) 367-385.

[35] Q. Chen, K. Worden, P. Peng, A.Y.T. Leung, Genetic algorithm with an improved fitness function for (N) ARX modelling, Mechanical Systems and Signal Processing 21(2)(2007) 994-1007.

[36] L. A. Nelson, G.J. Barlow, L. Doitsidis, Fitness functions in evolutionary robotics: a survey and analysis, Robotics and

Autonomous Systems 57(4)(2009) 345-370.

[37] X. Lai, R. Xu, W. Huang, Prediction of the lowest energy configuration for Lennard-Jones clusters, Science China Chemistry 54(6)(2011) 985-991.

[38] D.J. Wales, J.P. Doye, Global optimization by basin-hopping and the lowest energy structures of Lennard-Jones clusters containing up to 110 atoms, The Journal of Physical Chemistry A 101(28)(1997) 5111-5116. doi:10.1021/jp970984n

[39] Y.P. Wang, D.L. Liu, A global optimization evolutionary algorithm and its convergence based on a smooth scheme and line search, Chinese Journal of Computers 29(4)(2006) 670.

[40] Y. Wang, L. Fan, A smoothing evolutionary algorithm with circle search for global optimization, in: Proc. 2010 4th International Conference on Network and System Security (NSS), 2010.

[41] F. Wei, Y. Wang, Z. Meng, A smoothing function method with uniform design for global optimization, Pacific Journal of Optimization 10(2)(2014) 385-399.

[42] L. Costa, P. Oliveira, An adaptive sharing elitist evolution strategy for multiobjective optimization, Evolutionary Computation 11(4)(2003) 417-438.

[43] C. Mu, L. Jiao, Y. Liu, Y. Li, Multiobjective nondominated neighbor coevolutionary algorithm with elite population, Soft Computing 19(5)(2015) 1329-1349.

[44] Y. Xiang, Y. Zhou, H. Liu, An elitism based multi-objective artificial bee colony algorithm, European Journal of Operational Research 245(1)(2015) 168-193.

[45] Z. Yang, K. Tang, X. Yao, Self-adaptive differential evolution with neighborhood search, in: Proc. 2008 IEEE World Congress on Computational Intelligence, 2008.

[46] P.M. Reed, B.S. Minsker, D.E. Goldberg, The practitioner's role in competent search and optimization using genetic algorithms, in: Proc. Bridging the Gap: Meeting the World's Water and Environmental Resources Challenges, 2001.

[47] E. Zitzler, K. Deb, L. Thiele, Comparison of multiobjective evolutionary algorithms: empirical results, Evolutionary Computation 8(2)(2000) 173-195.