Yokohama Publishers
ISSN 1349-8169 ONLINE JOURNAL

Pacific Journal of Optimization
Vol. 15, No. 1, pp. 23 – 44, 2019

A NEW FILLED FUNCTION METHOD COMBINING AUXILIARY FUNCTION FOR GLOBAL OPTIMIZATION*

Xin Sui, Yuping Wang† and Junhua Liu

Abstract: In this paper, we introduce an auxiliary function into traditional filled function method, and construct a new one-parameter filled function based on this novel framework. The auxiliary function (which we call flatten function) can simplify the objective function so as to make construction of the filled function easier. One coefficient is used to the filled function to adjust the range of the objective function value, which can greatly reduce the difficulty of adjusting the parameter of filled function. Furthermore, we propose an adaptive strategy to determine the initial step size at the initial point by using the local search to minimize the filled function. The strategy overcomes the disadvantages caused by the fixed step size used in the existing algorithms. Based on these schemes, a new filled function algorithm is proposed. Numerical experiments are conducted based on 14 widely used test problems. Performance of the proposed algorithm is compared with that of several representative algorithms. The results demonstrate the proposed algorithm is more effective and efficient than the compared algorithms for global optimization problems.

Key words: *filled function method, auxiliary function, global optimization, adaptive initial point strategy, filled function parameter adjustment*

Mathematics Subject Classification: *90C26, 90C30*

1 Introduction

More and more practical problems existing in many fields, such as social science, engineering design and economics, can be formulated as global optimization problems. Global optimization has become a popular research field for researchers. Due to the complexity of the objective function and the existence of multiple local minima, it is usually not easy to find a global optimal solution of the objective function. Thus, in recent years, many works have been done in this area and some effective methods have been proposed.

Global optimization methods can be divided into two types: deterministic methods and stochastic methods. Deterministic methods often use the analytic properties of objective function, such as monotonicity, continuity and convexity, to solve the problem [1,6,7,10,14, 15,25]. While stochastic methods are mainly based on biology or statistical physics and use the heuristic or probabilistic mechanism to find the global optimal solution [2,3,9,24].

Filled function method, proposed by Ge [14], is an efficient deterministic global optimization method [5,12–16,21,23,26]. It modifies the objective function through a filled function. By optimizing the filled function that is constructed at the previous minimizer, the algorithm can jump out of the current minimum and obtain a better local minimum of objective

*This work was supported by National Natural Science Foundation of China (No.61872281).

function. In the study of [14], Ge gave a definition of filled function and constructed a two-parameter filled function. This filled function has the following form:

$P(x, x_1^*, \rho) = \frac{1}{\gamma + f(x)} \exp(-\frac{||x - x_1^*||^2}{\rho})$

There are two parameters γ and ρ in this filled function and the efficiency of the filled function method strongly depends on these two parameters. However, the values of γ and ρ are difficult to be adjusted. In the study of [15], Ge and Qin tried to solve this problem by constructing a one-parameter filled function as follows:

$Q(x, x_1^*, A) = -[f(x) - f(x_1^*)]exp(-A||x - x_1^*||^2)$

but the exponential term makes some false stationary points when A and $||x - x_1^*||$ are very large. Later, Liu proposed a new one-parameter filled function to overcome the shortcomings mentioned above in [13]:

$H(x, x^*, \alpha) = \frac{1}{\ln[1 + f(x) - f(x^*)]} - \alpha||x - x^*||^2$

where $\alpha > 0$ is a parameter.

Thereafter, many one-parameter filled functions were proposed [11, 16, 18, 26], and even some parameter-free filled functions were proposed in [6, 8]. However, there are two main drawbacks for the filled function method: The original definition of the filled function in Ge [14] makes it difficult to design a filled function; The initial step size δ at the initial point is hard to determine in minimizing the filled function. To make the design of the filled function much easier and improve the efficiency of the filled function algorithms, many researchers gave some new definitions for filled function and constructed different kinds of filled functions according to the new definitions [4, 11, 17, 18, 27]. However, the new filled functions using the new definitions still face great difficulty in jumping out the too many local optimal solutions and finally reaching the global optimal solution when there are a lot of local optimal solutions.

To overcome this drawback and further improve the efficiency of the filled function method, firstly, we introduce a flatten function into traditional filled function method for eliminating many local minimizers and thus making the optimal solution search much easier. Secondly, we propose an adaptive strategy to determine the distance δ. Third, we design a new filled function. Moreover, we use a constant coefficient to the new filled function, which helps filled function go fast from a nearly flat landscape.

The rest of this paper is organized as follows: Section 2 gives an introduction to the flatten function we used, and then a new filled function based on the flatten function is proposed and its properties are investigated. Section 3 describes a new filled function algorithm and demonstrate the details of an adaptive strategy to determine the distance. Section 4 applies the new filled function algorithm to 14 classical test problems and presents the numerical results. Finally, Section 5 concludes this paper.

2 A New Filled Function Combining Flatten Function and its Properties

Consider the following global optimization problem (P):

$$\begin{aligned} &\min f(x) \\ &s.t. \quad x \in \Omega \end{aligned}$$

where $f : R^n \to R$ is continuous in R^n; $\Omega \subset R^n$ is a closed bounded box that contains all global minimizers of $f(x)$. It is assumed that the number of minimizers of problem (P) can be infinite, but the number of different function values at the minimizers is finite.

In this paper, we use a flatten function to process (simplify) the objective function [19, 20, 22], and construct a new filled function on the simplified objective function.

The flatten function is first proposed in 2006 [19]. Simplified by the flatten function, all those local minimizers whose function values are larger than or equal to that of the current local minimizer of the original function can be eliminated. In this way, the number of local minimizers for original function will be greatly reduced, so that globally optimal solution finding process becomes easier and simpler. Its form is as follows [19]

$$s(x, x_1^*) = f(x_1^*) + \frac{1}{2}\left\{1 - sign\left[f(x) - f(x_1^*)\right]\right\}\left[f(x) - f(x_1^*)\right]$$

Namely,

$$s(x, x_1^*) = \begin{cases} f(x_1^*), & f(x) \geq f(x_1^*) \\ f(x), & f(x) < f(x_1^*) \end{cases}$$

where $f(x)$ is the objective function, x_1^* is the current local minimizer. $s(x, x_1^*)$ is called flatten function, which has two properties [19]:

Property 2.1. $s(x, x_1^*)$ keeps the local minimizers of $f(x)$ unchanged at any point x which is better than x_1^*.

That is, $\forall x \in \Omega$, if $f(x) < f(x_1^*)$, then $s(x, x_1^*) = f(x)$.

Property 2.2. $s(x, x_1^*)$ flattens the landscape at any point x which is not better than x_1^*.

That is, $\forall x \in \Omega$, if $f(x) \geq f(x_1^*)$, then $s(x, x_1^*) = f(x_1^*)$.

In this paper we use the definition of filled function in [23] described as follows:

Definition 2.1. $p(x, x_1^*)$ is called a filled function of $f(x)$ at a local minimizer x_1^* if $p(x, x_1^*)$ has the following properties:

1. x_1^* is a strictly local maximizer of $p(x, x_1^*)$.
2. $p(x, x_1^*)$ has no stationary point in the region

$$\Omega_1 = \{x | f(x) \geq f(x_1^*), x \in \Omega \backslash x_1^*\}$$

3. If x_1^* is not a global minimizer of $f(x)$, then $p(x, x_1^*)$ will have a minimizer in the region $\Omega_2 = \{x | f(x) < f(x_1^*), x \in \Omega\}$.

These properties of the new filled function ensure that when a descent method, for example, a steepest descent method, is employed to minimize the constructed filled function, the sequence of iteration points will not terminate at any point at which the objective function value is larger than $f(x_1^*)$; if x_1^* is not a global minimizer, then there must be a minimizer of the filled function at which the objective function value is less than $f(x_1^*)$, that is, any local minimizer of $p(x, x_1^*)$ must belong to the set $\Omega_2 = \{x | f(x) < f(x_1^*), x \in \Omega\}$. So that the descent method can jump out the current local minimizer of the objective function and find a better minimizer.

Let $L(P)$ stand for the set of local minimizers of the objective function $f(x)$, and $G(P)$ stand for the set of global minimizers of the objective function $f(x)$.

First, apply the above flatten function on $f(x)$ at the current local minimizer x_1^* to obtain a new objective function $s(x, x_1^*)$, and then we can design a new filled function for $s(x, x_1^*)$ as follows:

$$p(x, x_1^*, \mu) = \frac{1}{1 + ||x - x_1^*|| + s(x_1^*, x_1^*) - s(x, x_1^*)} + \mu \cdot \exp\left(\alpha\left[s(x, x_1^*) - s(x_1^*, x_1^*)\right]\right)$$

where α is a positive constant, and its value is the reciprocal of the order of magnitude of $s(x_1^*, x_1^*)$ (e.g. if $s(x_1^*, x_1^*) = 1.8e-3$, then $\alpha = 1e+3$). So α is not a parameter and it takes the fixed value at any moment. Thus, the new filled function can be considered as a one-parameter filled function with parameter μ.

Next we will prove that the function $p(x, x_1^*, \mu)$ is a filled function that satisfies Definition 2.1.

Theorem 2.2. *Suppose that x_1^* is a local minimizer of $f(x)$, then x_1^* is a strictly local maximizer of $p(x, x_1^*, \mu)$.*

Proof. Since $x_1^* \in L(P)$, then there exists a neighborhood $N(x_1^*, \delta)$ of x_1^* with $\delta > 0$, such that for $\forall x \in N(x_1^*, \delta)$, we have $f(x) \geq f(x_1^*)$, i.e., $s(x, x_1^*) = f(x_1^*)$.

For all $x \in N(x_1^*, \delta)$, and $x \neq x_1^*$, we have

$$\begin{aligned} p(x, x_1^*, \mu) &= \frac{1}{1 + ||x - x_1^*|| + f(x_1^*) - f(x_1^*)} + \mu \cdot exp(\alpha[f(x_1^*) - f(x_1^*)]) \\ &= \frac{1}{1 + ||x - x_1^*||} + \mu \end{aligned}$$

$$\begin{aligned} p(x, x_1^*, \mu) - p(x_1^*, x_1^*, \mu) &= \frac{1}{1 + ||x - x_1^*||} + \mu - (\frac{1}{1 + ||x_1^* - x_1^*||} + \mu) \\ &= \frac{1}{1 + ||x - x_1^*||} - 1 \\ &< 0 \end{aligned}$$

Thus, x_1^* is a strict local maximizer of $p(x, x_1^*, \mu)$. □

Theorem 2.3. *Suppose that $x_1^* \in L(P)$, $\forall x \in \Omega_1 = \{x | f(x) \geq f(x_1^*), x \in \Omega \backslash x_1^*\}$, then $p(x, x_1^*, \mu)$ has no stationary points.*

Proof. $\forall x \in \Omega_1$, $s(x, x_1^*) = f(x_1^*)$, $p(x, x_1^*, \mu) = \frac{1}{1+||x-x_1^*||} + \mu$, we have

$$\nabla p(x, x_1^*, \mu) = \frac{-1}{(1 + ||x - x_1^*||)^2} \cdot \frac{x - x_1^*}{||x - x_1^*||}$$

$$(\nabla p(x, x_1^*, \mu))^T \cdot \frac{x - x_1^*}{||x - x_1^*||} = \frac{-1}{(1 + ||x - x_1^*||)^2} < 0$$

Thus, $p(x, x_1^*, \mu)$ has no stationary points in the region Ω_1. □

Theorem 2.4. *Suppose that $x_1^* \in L(P)$ but $x_1^* \notin G(P)$. If $\mu > M$, then $p(x, x_1^*, \mu)$ must have a local minimizer in the set $\Omega_2 = \{x \in \Omega | f(x) < f(x_1^*)\}$.*
where

$$\begin{aligned} M = & \max_{\substack{||x - x_1^*|| > ||x_2^* - x_1^*|| \\ x \in \Omega_\varepsilon^=}} \\ & \{\frac{||x - x_1^*|| - ||x_2^* - x_1^*|| + f(x_2^*) - f(x)}{[1 + ||x_2^* - x_1^*|| + f(x_1^*) - f(x_2^*)][1 + ||x - x_1^*|| + f(x_1^*) - f(x)]} \\ & \cdot \frac{\exp(\alpha \cdot f(x_1^*))}{\exp(\alpha \cdot f(x)) - \exp(\alpha \cdot f(x_2^*))}\} \end{aligned}$$

$x_2^* \in L(P)$ with $f(x_2^*) < f(x_1^*)$, $\Omega_\varepsilon^= = \{x \in \Omega | f(x) = f(x_2^*) + \varepsilon\}$, and $\varepsilon > 0$ is small enough.

Proof. Since $x_1^* \in L(P)$ but $x_1^* \notin G(P)$, then $\exists\ x_2^* \in L(P)$ such that $f(x_2^*) < f(x_1^*)$.

There exists an $\varepsilon > 0$ small enough, such that
$\forall x \in \Omega_\varepsilon^= = \{x \in \Omega | f(x) = f(x_2^*) + \varepsilon\}$, $f(x) < f(x_1^*)$, i.e. $s(x, x_1^*) = f(x)$,

$$p(x, x_1^*, \mu) = \frac{1}{1 + ||x - x_1^*|| + f(x_1^*) - f(x)} + \mu \cdot \exp\left(\alpha[f(x) - f(x_1^*)]\right)$$

For any $x \in \Omega_\varepsilon^=$,

$$\begin{aligned}
& p(x_2^*, x_1^*, \mu) - p(x, x_1^*, \mu) \\
= & \frac{1}{1 + ||x_2^* - x_1^*|| + f(x_1^*) - f(x_2^*)} + \mu \cdot \exp\left(\alpha[f(x_2^*) - f(x_1^*)]\right) \\
& - \left\{ \frac{1}{1 + ||x - x_1^*|| + f(x_1^*) - f(x)} + \mu \cdot \exp\left(\alpha[f(x) - f(x_1^*)]\right) \right\} \\
= & \frac{||x - x_1^*|| - ||x_2^* - x_1^*|| + f(x_2^*) - f(x)}{[1 + ||x_2^* - x_1^*|| + f(x_1^*) - f(x_2^*)][1 + ||x - x_1^*|| + f(x_1^*) - f(x)]} \\
& + \mu \left\{ \exp\left(\alpha[f(x_2^*) - f(x_1^*)]\right) - \exp\left(\alpha[f(x) - f(x_1^*)]\right) \right\}
\end{aligned}$$

Consider the following two cases

1. When $||x_2^* - x_1^*|| \geq ||x - x_1^*||$, we have
   $\forall x \in \Omega_\varepsilon^=$, $f(x) > f(x_2^*)$,
   $||x - x_1^*|| - ||x_2^* - x_1^*|| + f(x_2^*) - f(x) < 0$,
   $\exp\left(\alpha[f(x_2^*) - f(x_1^*)]\right) - \exp\left(\alpha[f(x) - f(x_1^*)]\right) < 0$,
   that is, $p(x_2^*, x_1^*, \mu) - p(x, x_1^*, \mu) < 0$.

   For $\mu > 0$, we have $p(x_2^*, x_1^*, \mu) < p(x, x_1^*, \mu)$.

2. When $||x_2^* - x_1^*|| < ||x - x_1^*||$, we first want to prove $p(x_2^*, x_1^*, \mu) < p(x, x_1^*, \mu)$, that is, to prove
   $$\begin{aligned}
   & p(x_2^*, x_1^*, \mu) - p(x, x_1^*, \mu) \\
   = & \tfrac{1}{1+||x_2^* - x_1^*|| + f(x_1^*) - f(x_2^*)} + \mu \cdot \exp\left(\alpha[f(x_2^*) - f(x_1^*)]\right) \\
   & - \left\{ \tfrac{1}{1+||x - x_1^*|| + f(x_1^*) - f(x)} + \mu \cdot \exp\left(\alpha[f(x) - f(x_1^*)]\right) \right\} \\
   = & \tfrac{||x - x_1^*|| - ||x_2^* - x_1^*|| + f(x_2^*) - f(x)}{[1+||x_2^* - x_1^*|| + f(x_1^*) - f(x_2^*)][1+||x - x_1^*|| + f(x_1^*) - f(x)]} \\
   & + \mu \left\{ \exp\left(\alpha[f(x_2^*) - f(x_1^*)]\right) - \exp\left(\alpha[f(x) - f(x_1^*)]\right) \right\} \\
   < & 0,
   \end{aligned}$$

   which is equivalent to

   $$\mu > \tfrac{||x - x_1^*|| - ||x_2^* - x_1^*|| + f(x_2^*) - f(x)}{[1+||x_2^* - x_1^*|| + f(x_1^*) - f(x_2^*)][1+||x - x_1^*|| + f(x_1^*) - f(x)]} \cdot \tfrac{\exp(\alpha \cdot f(x_1^*))}{\exp(\alpha \cdot f(x)) - \exp(\alpha \cdot f(x_2^*))}.$$

   Denote

   $$M = \max_{\substack{||x - x_1^*|| > ||x_2^* - x_1^*|| \\ x \in \Omega_\varepsilon^=}} \left\{ \tfrac{||x - x_1^*|| - ||x_2^* - x_1^*|| + f(x_2^*) - f(x)}{[1+||x_2^* - x_1^*|| + f(x_1^*) - f(x_2^*)][1+||x - x_1^*|| + f(x_1^*) - f(x)]} \cdot \tfrac{\exp(\alpha \cdot f(x_1^*))}{\exp(\alpha \cdot f(x)) - \exp(\alpha \cdot f(x_2^*))} \right\}.$$

Thus, when $\mu > M$, for all $x \in \Omega_{\varepsilon}^{=}$, we will have $p(x_2^*, x_1^*, \mu) < p(x, x_1^*, \mu)$.

We denote

$$\Omega_{\varepsilon}^{\leq} = \{x \in \Omega | f(x) \leq f(x_2^*) + \varepsilon\}.$$

Since $\Omega_{\varepsilon}^{\leq}$ is a compact set and $p(x, x_1^*, \mu)$ is continuous, there exists $x_3^* \in \Omega_{\varepsilon}^{\leq}$ such that

$$p(x_3^*, x_1^*, \mu) = \min_{x \in \Omega_{\varepsilon}^{\leq}} p(x, x_1^*, \mu)$$

$\forall x \in \Omega_{\varepsilon}^{=}, p(x_3^*, x_1^*, \mu) \leq p(x_2^*, x_1^*, \mu) < p(x, x_1^*, \mu)$, and $\Omega_{\varepsilon}^{\leq} \subset \Omega_2$, thus $x_3^* \in \Omega_2$ is a local minimizer of $p(x, x_1^*, \mu)$.
Based on the above discussions, we get that when $\mu > M$, $p(x, x_1^*, \mu)$ must have a local minimizer in the region $\Omega_2 = \{x \in \Omega | f(x) < f(x_1^*)\}$. □

Theorems 2.2-2.4 prove that the function $p(x, x_1^*, \mu)$ at x_1^* is a filled function satisfying Definition 2.1 under some assumptions on parameter μ.

Although flatten function makes the proposed filled function nondifferentiable at some points, the filled function may be nondifferentiable at only the boundary points between two regions, where the original function values are equal to $f(x_k^*)$ in one region, and smaller than $f(x_k^*)$ in the other region, where x_k^* is the current local minimizer obtained. Note that there are relatively few such points because the boundary points are much fewer than the points in these two regions. Also, even if the algorithm goes to such point x, we only need to move x to its quite near point $x + \varepsilon e_i (\varepsilon > 0$ is a proper small number, $e_i = (0, 0, ...1, ...0) \in R^n)$, then the algorithm can be executed smoothly. Therefore, it can not cause the numerical instability.

The constant coefficient α adopted in the aforementioned way can adjust the range of objective function value effectively. In fact, without this parameter, when the objective function value is too small, the change of objective function is not enough to generate the obvious change of the filled function; or when the objective function value is too large, the change of objective function may be too big to result in the overflow of the filled function. In both cases, the filled function may not satisfy Definition 2.1. However, when we use coefficient α whose value is equal to the reciprocal of the order of magnitude of $s(x_1^*, x_1^*)$, in the neighborhood of x_1^*, the values of $\alpha \cdot s(x_1^*, x_1^*)$ and $\alpha \cdot s(x, x_1^*)$ are both in the range (-10,10) and $\exp(\alpha [s(x, x_1^*) - s(x_1^*, x_1^*)])$ can have enough change but does not cause the overflow. This strategy can greatly reduce the difficulty of adjusting the parameter of filled function and improve solution accuracy and algorithm performance. Since α is completely adaptive to objective function, this strategy can be used for general filled function construction.

In order to demonstrate the effect of α, we gave an example. In problem 12, both the value of $s(x_1^*, x_1^*)$ and its change are too small. If we don't use α, a common parameter value of μ couldn't make $p(x, x_1^*, \mu)$ be an eligible filled function. But when we use α with the same μ, the new $p(x, x_1^*, \mu)$ will be a satisfactory filled function. The flatten function and two filled functions with and without α are shown in Figures 1-3 respectively.

3 A New Filled Function Method Combining Flatten Function

In this section, we first introduce an adaptive strategy to determine the distance δ, then present a filled function method combining flatten function(F-FFM).

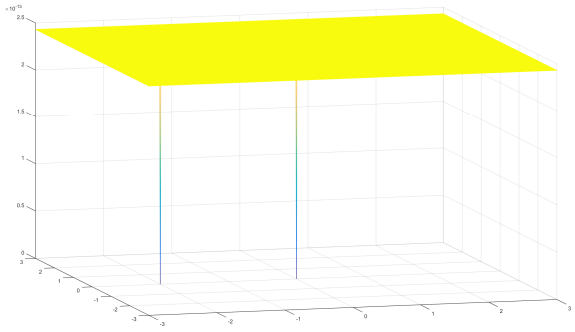

Figure 1: flatten function of $f12$

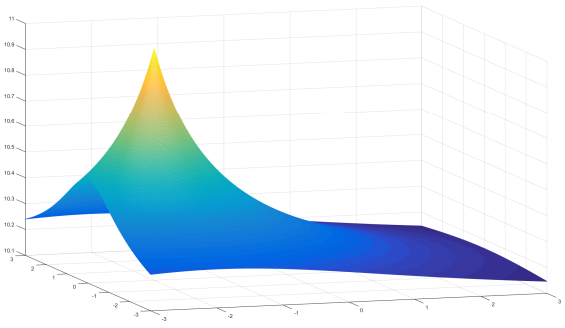

Figure 2: Filled function on the $s(x, x_k^*)$ of $f12$ without α

3.1 An adaptive strategy to determine the distance δ

When we construct a filled function at a local minimizer x_k^*, we should find an initial point x_2 that is near x_k^* to implement local search process on the filled function. We denote x_2 as $x_2 = x_k^* + \delta \cdot e_j$, where e_j is usually the unit direction vector along the j-th axis and δ is the distance between x_k^* and x_2. However, the existing methods to determine the distance δ are not efficient. Most of the previous works only suggest that the distance δ is taken by a fixed and small constant, such as $\delta = 0.1$ [4, 27]. However, determining δ in this way is not effective for many problems because it does not take into account the position relationship among x_k^*, the search domain and the previous local minimizer, which may cause that x_2 crosses the border or takes excessive search on the neighborhood of the previous local minimizer. To solve this problem, we present an adaptive strategy which is based on the relative relationship among x_k^*, the search domain and the previous local minimizer.

Suppose that the current local minimizer is x_k^*, and U and L to denote upper and lower bound of search domain. The new strategy is described as follows:

Strategy 3.1

1. Compute the distance from x_k^* to the upper and lower bound in each dimension:

$$du(i) = U(i) - x_k^*(i), i = 1 \sim n$$

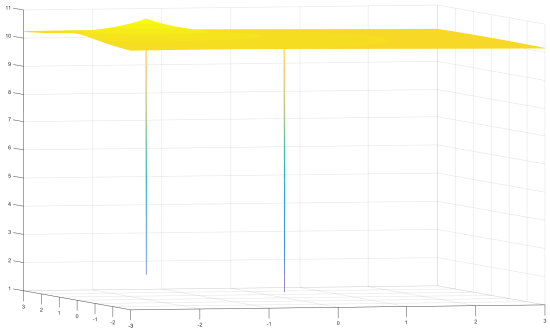

Figure 3: Filled function on the $s(x, x_k^*)$ of $f12$ with α

$$dl(i) = x_k^*(i) - L(i), i = 1 \sim n$$

Where $x_k^*(i)$ is the i-th element of x_k^*, $U(i)$ and $L(i)$ are the upper and lower bounds of the i-th dimension, $du(i)$ and $dl(i)$ represent the distances from $x_k^*(i)$ to $U(i)$ and to $L(i)$, respectively.

2. Compute the distance dp from x_k^* to previous local minimizer (represented by x_{k-1}^*) in each dimension and update the corresponding $du(i)$ or $dl(i)$:

```
i=1
while i ≤ n do
  if x_k^*(i) < x_{k-1}^*(i) then
    dp(i) = x_{k-1}^*(i) − x_k^*(i)
    du(i) = dp(i)
  else
    dp(i) = x_k^*(i) − x_{k-1}^*(i)
    dl(i) = dp(i)
  end if
end while
```

If $k = 1$, that is x_k^* is the first local minimizer we found, then previous local minimizer x_{k-1}^* is represented by the initial point x_0 of problem(P).

3. Divide $du(i)$, $dl(i)$ into N parts equally, then the size of one part stands for the distance δ in the corresponding direction. Detailed procedure is as follows:

```
if the searching is in the positive direction of i-th dimension then
  δ = du(i)/N
else
  δ = dl(i)/N
end if
```

Remark 3.1. N is a parameter. We select N=50 in this paper in all experiments.

3.2 A new filled function method (F-FFM)

Algorithm 1 F-FFM

1. Initialization Steps:
   1: Choose $\mu = 10$, and $\mu U > 0$ as the tolerance parameter for terminating the minimization process.
   2: Choose an initial point $x_1^0 \in \Omega$.
   3: Choose direction $e_i,\ i = 1, 2, ..., k_0$ with $k_0 = 2n$, where n represents the dimension of variable and $e_{n+i} = e_i$ for $i = 1, 2, \cdots, n$.
   4: Compute the distance δ by using $N = 50$.
   5: Choose $\alpha = 1$ to adjust the filled function.
   6: Let $k = 1$, where k stands for the number of iteration.
2. Main Steps:
   1: Implement a local search method at x_k^0 to get a local minimizer x_k^* of problem (P). If $x_k^*(j) \notin [L(j), U(j)]$, then let $x_k^*(j) = L(j) + r[U(j) - L(j)]$, where r is a random number in $[0, 1]$.
   2: Let $i = 1$ and take flatting process on $f(x)$:
   $s(x, x_k^*) = f(x_k^*) + \frac{1}{2}\{1 - sign[f(x) - f(x_k^*)]\}[f(x) - f(x_k^*)]$
   3: Construct a filled function on $s(x, x_k^*)$:
   $p(x, x_1^*, \mu) = \frac{1}{1+||x-x_1^*||+s(x_1^*,x_1^*)-s(x,x_1^*)} + \mu \cdot \exp(\alpha[s(x, x_1^*) - s(x_1^*, x_1^*)])$
   4: If $i \leq k_0$, go to step 5; otherwise, go to step 7.
   5: Calculate δ through strategy 3.1, then let $\bar{x}_k^* = x_k^* + \delta e_i$. Implement a local search method from $\bar{x}_k^*$ on the filled function $p(x, x_k^*, \mu)$ to obtain a local minimizer p_k^*. If $p_k^* \notin \Omega$, let $i = i + 1$ and go to step 4, otherwise, go to step 6.
   6: Let $k = k + 1$. Implement a local search method from p_k^* on $s(x, x_k^*)$ to obtain a new local minimizer x_k^* of $f(x)$, then update α through $f(x_k^*)$ and go to step 2.
   7: If $\mu \geq \mu U$, stop and x_k^* is considered as the global minimizer of problem (P), otherwise, let $\mu = 10 * \mu$, go to step 3.

This algorithm mainly contains two parts. The first part is to minimize the objective function $f(x)$ to get a local minimizer x_k^*. Since local search method can't escape from local minimizer, we adopt the second part to solve this problem. In the second part, we first design a flatten function $s(x, x_k^*)$ on $f(x)$ which can eliminate a lot of local minima and make the number of local minima reduced greatly for the ease and convenience of the followed globally optimal solution search, then construct a filled function $p(x, x_k^*, \mu)$ on $s(x, x_k^*)$. Through minimizing $p(x, x_k^*, \mu)$, we can find its local minimizer p_k^*. It's obvious that p_k^* is in a new better basin of $f(x)$, in which there is a better local minimizer. Therefore, we can return to the first part to get a new better local minimizer. Execute such iteration continuously until $p(x, x_k^*, \mu)$ has no longer minimizer, then we adjust parameter μ. If parameter μ is sufficiently large and $p(x, x_k^*, \mu)$ still has no minimizer, we consider x_k^* as a global minimizer of problem(P), and the algorithm is terminated.

4 Numerical Experiment

4.1 Experiment environment

In this section, we execute our new algorithm F-FFM on 14 standard and frequently-used test problems [4, 27]. All experiments are implemented using Matlab R2014b, under windows 7 and Intel Xeon with 2.53G CPU and 4G RAM. Fminunc function in the optimization toolbox of Matlab is taken as our local search method. Two classical filled function methods (Ge [14], Y. Zhang [27]) and two state-of-art methods (C. Wang [17], T. El-Gindy [4]) are selected to compare with our method, and the initial points in our experiments are all set to be same as those in compared algorithms. Numerical results show that our method is effective and efficient.

4.2 Test problems

14 test problems are as follows:

Problem 1

$$\begin{aligned}&\min f(x) = \left|\tfrac{x-1}{4}\right| + \left|\sin(\pi(1+\tfrac{x-1}{4}))\right| + 1\\ &s.t. -10 \le \mathrm{x} \le 10\end{aligned}$$

The new filled function method SF-FFM succeeds in finding the global minimizer $x^* = 1.00000000$ with $f(x^*) = 1.000000000325805$. The computational results are summarized in Table 2.

Problem 2

$$\begin{aligned}&\min f(x) = 2x^2 - 1.05x^4 + \tfrac{1}{6}x - |x|\\ &s.t. -0.8 \le \mathrm{x} \le 1\end{aligned}$$

The new filled function method F-FFM succeeds in finding the global minimizer $x^* = -0.329088554$ with $f(x^*) = -0.179653263511957$. The computational results are summarized in Table 3.

Problem 3

$$\begin{aligned}&\min f(x) = |x-1|\,(1+10\,|\sin(x+1)|) + 1\\ &s.t. -10 \le \mathrm{x} \le 10\end{aligned}$$

The new filled function method F-FFM succeeds in finding the global minimizer $x^* = 0.999999995$ with $f(x^*) = 1.000000048856826$. The computational results are summarized in Table 4.

Problem 4

$$\begin{aligned}&\min f(x) = \begin{cases} x^2 \sin\frac{1}{x}, & \text{if } \mathrm{x} \neq 0\\ 0, & \text{if } \mathrm{x} = 0\end{cases}\\ &s.t. -0.4 \le \mathrm{x} \le 0.4\end{aligned}$$

The new filled function method F-FFM succeeds in finding the global minimizer $x^* = 0.233930054$ with $f(x^*) = -0.049566607874611$. The computational results are summarized in Table 5.

Problem 5

$$\min f(x) = \sum_{i=1}^{5} i\,|\cos((i+1)x+i)| + 5$$
$$s.t. -10 \le \mathrm{x} \le 10$$

The new filled function method F-FFM succeeds in finding the global minimizer $x^* = 8.31061870$ with $f(x^*) = 6.699793778161520$. The computational results are summarized in Table 6.

Problem 6

$$\min f(x) = \sum_{i=1}^{n} |x_i - 0.5|$$
$$s.t. -5 \le \mathrm{x}_i \le 5, \mathrm{i} = 1, 2, ..., \mathrm{n}$$

The new filled function method F-FFM succeeds in finding the global minimizer $x^* = (0.500000000, 0.500000000)$ with $f(x^*) = 0$ for n=2. The computational results are summarized in Table 7.

Problem 7

$$\min f(x) = \max\left\{5x_1 + x_2, -5x_1 + x_2, x_1^2 + x_2^2 + 4x_2\right\}$$
$$s.t. -4 \le \mathrm{x}_1 \le 4, -4 \le \mathrm{x}_2 \le 4$$

The new filled function method F-FFM succeeds in finding the global minimizer $x^* = (-3.34376972E-04, -3.00055715)$ with $f(x^*) = -2.998885266142280$. The computational results are summarized in Table 8.

Problem 8

$$\min f(x) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}|x_i|}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20$$
$$s.t. -20 \le \mathrm{x}_i \le 30, \mathrm{i} = 1, 2, ..., \mathrm{n}$$

The new filled function method F-FFM succeeds in finding the global minimizer $x^* = (2.41384061E-15, 0))$ with $f(x^*) = -2.718281689496010$ for $n = 2$. The computational results are summarized in Table 9.

Problem 9

$$\min f(x) = x_1^2 + x_2^2 - \cos(18x_1) - \cos(18x_2)$$
$$s.t. -1 \le \mathrm{x}_1 \le 1, -1 \le \mathrm{x}_2 \le 1$$

The new filled function method F-FFM succeeds in finding the global minimizer $x^* = (0, 0)$ with $f(x^*) = -2$. The computational results are summarized in Table 10.

Problem 10

$$\min f(x) = [1 - 2x_2 + csin(4\pi x_2) - x_1]^2 + [x_2 - 0.5sin(2\pi x_1)]^2$$
$$s.t. 0 \le \mathrm{x}_1 \le 10, -10 \le \mathrm{x}_2 \le 0$$

The new filled function method F-FFM succeeds in finding the global minimizer with $f(x^*) = 0$ for c=0.2,0.5. The computational results are summarized in Tables 11.

Problem 11

$$\min f(x) = 4x_1^2 - 2.1x_1^4 + \tfrac{1}{3}x_1^6 - x_1x_2 - 4x_2^2 + 4x_2^4$$
$$s.t. -3 \le \mathrm{x}_1 \le 3, -3 \le \mathrm{x}_2 \le 3$$

The new filled function method F-FFM succeeds in finding the global minimizer $x^* = (8.98420215E{-}02, 0.712656441)$ with $f(x^*) = -1.031628453489865$ and $x^* = (8.98419853E{-}02, 0.712656395)$ with $f(x^*) = -1.031628453489875$, for two initial points $x_0 = (-2, 1)$ and $x_0 = (-3, 3)$. The computational results are summarized in Table 12.

Problem 12

$$\min f(x) = x_1^4 + 4x_1^3 + 4x_1^2 + x_2^2$$
$$s.t. -3 \le \mathrm{x}_1 \le 3, -3 \le \mathrm{x}_2 \le 3$$

The new filled function method F-FFM succeeds in finding the global minimizer x^*=(0, 0), and $f(x^*)$=0. The computational results are summarized in Table 13.

Problem 13

$$\min f(x) = \left\{\sum_{i=1}^{5} i\cos[(i+1)x_1 + i]\right\}\left\{\sum_{i=1}^{5} i\cos[(i+1)x_2 + i]\right\}$$
$$s.t. 0 \le \mathrm{x}_1 \le 10, 0 \le \mathrm{x}_2 \le 10$$

The new filled function method F-FFM succeeds in finding the global minimizer $x^* = (5.48286419, 4.85805684)$ with $f(x^*) = -1.867309088310200E + 02$. The computational results are summarized in Table 14.

Problem 14

$$\min f(x) = \tfrac{\pi}{n}\left\{10\sin^2\pi x_1 + \sum_{i=1}^{n-1}\left[(x_i - 1)^2(1 + 10\sin^2\pi x_{i+1})\right] + (x_n - 1)^2\right\}$$
$$s.t. -10 \le \mathrm{x}_i \le 10, i = 1, 2, ..., n$$

The new filled function method F-FFM succeeds in finding the global minimizer $x^* = (1, 1, ..., 1)$ with $f(x^*) = 0$ for n=2,5,7,10,20,30,50,100. The computational results are summarized in Table 15-17 respectively. For making tables concise and direct, we omit x_k^*, and only demonstrate the values of k and $f(x_k^*)$.

4.3 Experiment results

General computational results on all 14 test problems are summarized in Table 1. The detailed computational results for each problem are listed in the followed Tables respectively. The meanings of the symbols used in tables are as follows:

No. : The order of the problems.

n: The number of variables.

k_0: The total number of direction e_i.

Iter: The total number of iterations.

F_f: The total number of function evaluations of $f(x)$, $s(x, x_k^*)$ and $p(x, x_k^*, \mu)$.

μ: The parameter value of filled function $p(x, x_k^*, \mu)$.

μU: The upper bound of parameter μ.

N: The parameter in Strategy 3.1 for determining δ. We set N=50 in this paper for all experiments.

k: The iteration number in finding the kth local minimizer.

x_k^0: The kth initial point.

x_k^*: The kth local minimizer.

$f(x_k^*)$: The function value of the kth local minimizer.

Table 1 gives the general computational results on 14 test problems, including the initial values of algorithm parameters (k_0, μU, N), the final parameter value μ of filled function, and the consumed numbers of iterations as well as total function evaluations.

Tables 2-17 give the detailed computational results for each problem. All initial points in our experiments are set to be the same as those in compared methods. By comparing Tables 2-9 in this paper with the corresponding Tables 4.1-4.8 in [27] respectively, we can find that our method can obtain the global minimizers for all problems and even get more accurate results, such as Problems 1 and 2. By comparing Tables 10-17 in this paper with the results in [4,14,17], it can be seen that our method can obtain almost global minimizers for all problems. And the accuracy of our results is higher than that of all results in [14,17], but lower than that of some results in [4]. However, the number of function evaluations used in the proposed algorithm is fewer than that used in [4], especially for high dimension cases (more than 10 dimensions).

Table 18 gives the general comparison of the results obtained by our method and those in [4, 14, 17]. Since we use the function $fminunc$ as the local search method and $fminunc$ doesn't provide the number of expended gradient evaluations, we don't compare gradient evaluations in Table 18. From the table, we find the total number of iterations in our algorithm is less than that in [14, 17] for almost all problems and it's similar to that in [4]. Meanwhile, the number of the function evaluations F_f used in the proposed algorithm is greatly less than that used in [14,17] for all problems. And in nine experiments, the number of the function evaluations F_f used in our method is less than that used in [4]. Especially, in dealing with the problems of high dimension, the advantage of our method is more obvious.

5 Conclusions

In this paper, we propose a new one-parameter filled function combining the flatten function to solve unconstrained global optimization problems and investigate its properties. In order to improve the performance of the filled function and make the adjustment of parameter as easy as possible, we introduce a constant α to the new filled function so that the change of the exponential term become more easily controlled and the proposed filled function is numerically stable (can not overflow). Furthermore, we introduce an adaptive strategy to determine the initial step size δ of the initial point in order to minimize the filled function efficiently, which is based on the relative position relationship among the current local minimizer, the problem domain and the previous local minimizer. This strategy breaks the limitation of the fixed step size in the existing filled function methods. Based on all above strategies, a new filled function method is proposed. The numerical results of experiments imply that our method is more effective and robust. Especially in the numbers of iterations and function evaluations, our method shows greater advantage. And with the increasing of

Table 1: General computational results.

No.	n	k_0	Iter	F_f	μ	μU	N
1	1	2	2	304	10	10	50
2	1	2	2	150	10	10	50
3	1	2	2	238	10	10	50
4	1	2	2	170	10	10	50
5	1	2	2	288	10	10	50
6	2	4	1	412	10	10	50
7	2	4	3	942	10	10	50
8	2	4	7	1108	10	10	50
9	2	4	2	458	10	10	50
10	2	4	2	443	10	10	50
		4	2	395	10	10	50
11	2	4	1	316	10	10	50
		4	2	398	10	10	50
12	2	4	4	724	10	10	50
13	2	4	6	1020	10	10	50
14	2	4	2	509	10	10	50
	5	10	5	3995	10	10	50
	7	14	4	4700	10	10	50
	10	20	2	11453	10	10	50
	20	40	3	29424	10	10	50
	30	60	3	75364	10	10	50
	50	100	3	169578	10	10	50
	100	200	5	689936	10	10	50

Table 2: Numerical results for Problem 1 with initial point 6.

k	x_k^0	x_k^*	$f(x_k^*)$
1	6	4.99999996	2.000000024043301
2	1.00000000	1.00000000	1.000000000325805

Table 3: Numerical results for Problem 2 with initial point 0.5.

k	x_k^0	x_k^*	$f(x_k^*)$
1	0.5	0.408163934	-0.036083585083927
2	-0.611481763	-0.329088554	-0.179653263511957

Table 4: Numerical results for Problem 3 with initial point -1.5.

k	x_k^0	x_k^*	$f(x_k^*)$
1	-1.5	-0.999999997	3.000000048646088
2	0.999999995	0.999999995	1.000000048856826

Table 5: Numerical results for Problem 4 with initial point 0.1.

k	x_k^0	x_k^*	$f(x_k^*)$
1	0.1	0.213863609	-0.045707160044852
2	0.234006345	0.233930054	-0.049566607874611

Table 6: Numerical results for Problem 5 with initial point 3.

k	x_k^0	x_k^*	$f(x_k^*)$
1	3	3.28407044	14.070575989849708
2	8.31061870	8.31061870	6.699793778161520

Table 7: Numerical results for Problem 6 with initial point (3,3).

k	x_k^0	x_k^*	$f(x_k^*)$
1	(3,3)	(0.500000000,0.500000000)	0

Table 8: Numerical results for Problem 7 with initial point (1,1).

k	x_k^0	x_k^*	$f(x_k^*)$
1	(1,1)	(-0.00000000,-1.20000000)	-1.199999999999999
2	(-0.00000000,-1.25600000)	(0.00000000,-1.25600000)	-1.255999999999999
3	(1.15785915E-03,-3.00192806)	(-3.34376972E-04,-3.00055715)	-2.998885266142280

Table 9: Numerical results for Problem 8 with initial point (-16,-1).

k	x_k^0	x_k^*	$f(x_k^*)$
1	(-16,-1)	(-15.9964294,-0.996429560)	6.117673641171395
2	(-13.9958236,-0.996061400)	(-13.9960614,-0.996061538)	5.715581170386420
3	(-11.9953632,-0.995605276)	(-11.9956054,-0.995605445)	5.269594080166323
4	(-9.99477384,-0.995021602)	(-9.99502182,-0.995021882)	4.768359205573809
5	(-0.984037201,-0.984495066)	(-0.984500073,-0.984500071)	0.894453876550717
6	(0.000000000,-0.978345246)	(0.000000000,-0.978345246)	-0.094970424106275
7	(2.41384061E-15,0)	(2.41384061E-15,0)	-2.718281689496010

Table 10: Numerical results for Problem 9 with initial point(1,1).

k	x_k^0	x_k^*	$f(x_k^*)$
1	(1.00000000,1.00000000)	(0.167141238,0.0236398399)	0.108831490871375
2	(-3.03702204E-03,-4.28113879E-04)	(6.42457742E-10,-4.15377437E-11)	-2

Table 11: Numerical results for Problem 10 with c=0.2 and c=0.5.

c=0.2			
k	x_k^0	x_k^*	$f(x_k^*)$
1	(3.00000000,-3.00000000)	(1.01749345,-0.287411848)	0.536955348422852
2	(1.89734105,-0.347766769)	(1.878431037,-0.3458499667)	2.109423746787797E-15
c=0.5			
k	x_k^0	x_k^*	$f(x_k^*)$
1	(0.00000000,0.00000000)	(0.0420235949,-0.0947718087)	0.517453552567242
2	(0.954908405,-0.175854948)	(0.943226261,-0.174601213)	3.441691376337985E-15

Table 12: Numerical results for Problem 11 with initial point(-2,1) and (-3,3).

x_0=(-2,1)			
k	x_k^0	x_k^*	$f(x_k^*)$
1	(-2.00000000,1.00000000)	(8.98420215E-02,0.712656441)	-1.031628453489865
x_0=(-3,3)			
k	x_k^0	x_k^*	$f(x_k^*)$
1	(-3.00000000,3.00000000)	(-1.60710464,0.568655141)	2.104250310361998
2	(9.80241678E-02,0.713480350)	(8.98419853E-02,0.712656395)	-1.031628453489875

Table 13: Numerical results for Problem 12 with initial point(-1,-2).

k	x_k^0	x_k^*	$f(x_k^*)$
1	(-1.00000000,-2.00000000)	(-1.00000003,-1.4901161E-08)	0.999999999999998
2	(-2.10127904,2.98974504E-07)	(-1.99999976,1.01949740E-07)	2.431388423929093E-13
3	(-1.99999997,9.4125955E-08)	(-1.99999997,9.4125955E-08)	1.243449787580175E-14
4	(-2.00000003,9.4125955E-08)	(-2.00000003,9.4125955E-08)	1.243449787580175E-14

Table 14: Numerical results for Problem 13 with initial point(1,1).

k	x_k^0	x_k^*	$f(x_k^*)$
1	(-1.00000000,-2.00000000)	(2.04669717,2.04669717)	2.291785049429219E-15
2	(2.20576323,2.04669710)	(2.20576323,2.04669710)	-3.174772226760831E-06
3	(2.36164796,2.04669138)	(2.36164796, 2.04669138)	-2.817139588048570E-04
4	(5.82432135,2.04595877)	(5.82432135,2.04595877)	-0.001851942899270
5	(5.48199101,1.80531284)	(5.48286417,1.80565642)	-39.588744510923043
6	(5.48287226,4.85826986)	(5.48286419,4.85805684)	-1.867309088310200E+02

Table 15: Numerical results for Problem 14 with n=2,5.

n=2			
k	x_k^0	x_k^*	$f(x_k^*)$
1	(-4.00000000,-4.00000000)	(2.97983349,2.99484262)	12.487063715415195
2	(0.999970886,0.997071937)	(0.999999999,1.00000003)	1.776356839400251E-15
n=5			
k	x_k^0	x_k^*	$f(x_k^*)$
1	$\begin{pmatrix} 2.00000000 \\ 3.00000000 \\ 2.00000000 \\ 1.00000000 \\ -2.00000000 \end{pmatrix}$	$\begin{pmatrix} 1.98954884 \\ 2.97944758 \\ 1.99742064 \\ 0.999999993 \\ 0.999999999 \end{pmatrix}$	3.736222357935198
2	$\begin{pmatrix} 0.999762391 \\ 0.953260142 \\ 0.979914240 \\ 0.999999992 \\ 0.999999982 \end{pmatrix}$	$\begin{pmatrix} 0.999999993 \\ 1.00000007 \\ 0.999999728 \\ 0.999999988 \\ 0.999999982 \end{pmatrix}$	5.329070518200751E-14
3	$\begin{pmatrix} 0.999999993 \\ 1.00000007 \\ 0.999999937 \\ 0.999999988 \\ 0.999999982 \end{pmatrix}$	$\begin{pmatrix} 0.999999993 \\ 1.00000007 \\ 0.999999937 \\ 0.999999988 \\ 0.999999982 \end{pmatrix}$	9.325873406851315E-15
4	$\begin{pmatrix} 0.999999993 \\ 1.00000007 \\ 1.00000000 \\ 0.999999988 \\ 1.00000002 \end{pmatrix}$	$\begin{pmatrix} 0.999999993 \\ 1.00000007 \\ 1.00000000 \\ 0.999999988 \\ 1.00000002 \end{pmatrix}$	7.105427357601002E-15
5	$\begin{pmatrix} 1.00000002 \\ 1.00000007 \\ 1.00000000 \\ 0.999999988 \\ 1.00000002 \end{pmatrix}$	$\begin{pmatrix} 1.00000002 \\ 1.00000007 \\ 1.00000000 \\ 0.999999988 \\ 1.00000002 \end{pmatrix}$	7.105427357601002E-15

Table 16: Numerical results for Problem 14 with n=7,10.

n=7			
k	x_k^0	x_k^*	$f(x_k^*)$
1	$\begin{pmatrix} 1.00000000 \\ -2.00000000 \\ 2.00000000 \\ 3.00000000 \\ -3.00000000 \\ 4.00000000 \\ -4.00000000 \end{pmatrix}$	$\begin{pmatrix} 1.02936822 \\ 0.666061276 \\ 0.946659250 \\ 1.280192384 \\ 0.998375814 \\ 0.821589035 \\ -3.79463820 \end{pmatrix}$	10.532657378329942
2	$\begin{pmatrix} -0.979931824 \\ 1.00000194 \\ 1.00012250 \\ 0.999367058 \\ 0.999997839 \\ 1.00039940 \\ 1.01082399 \end{pmatrix}$	$\begin{pmatrix} -0.979886051 \\ 0.999999992 \\ 0.999999941 \\ 0.999999884 \\ 0.999999956 \\ 0.999995759 \\ 1.00000019 \end{pmatrix}$	1.777165398167645
3	$\begin{pmatrix} 1.00028452 \\ 0.999999929 \\ 1.00000001 \\ 0.999999908 \\ 1.00000003 \\ 1.00000003 \\ 0.999999891 \end{pmatrix}$	$\begin{pmatrix} 1.00000000 \\ 0.999999929 \\ 1.00000001 \\ 0.999999909 \\ 1.00000003 \\ 1.00000003 \\ 0.999999892 \end{pmatrix}$	1.243449787580175E-14
4	$\begin{pmatrix} 1.00000000 \\ 1.00000000 \\ 1.00000001 \\ 0.999999979 \\ 1.00000003 \\ 1.00000003 \\ 0.999999962 \end{pmatrix}$	$\begin{pmatrix} 1.00000000 \\ 1.00000000 \\ 1.00000001 \\ 0.999999979 \\ 1.00000003 \\ 1.00000003 \\ 0.999999962 \end{pmatrix}$	1.776356839400251E-15
n=10			
k	x_k^0	x_k^*	$f(x_k^*)$
1	$\begin{pmatrix} 1.00000000 \\ 2.00000000 \\ 0.00000000 \\ -2.00000000 \\ 3.00000000 \\ -3.00000000 \\ 4.00000000 \\ -4.00000000 \\ 5.00000000 \\ -5.00000000 \end{pmatrix}$	$\begin{pmatrix} 0.996200138 \\ 1.18287718 \\ -0.112765254 \\ 1.00921981 \\ 0.866041398 \\ 1.02914747 \\ 2.72807235 \\ -3.98793487 \\ 4.99841520 \\ -4.99632739 \end{pmatrix}$	25.518680669731722
2	$\begin{pmatrix} 0.999268497 \\ 0.976017807 \\ 1.37082226 \\ 1.36960648 \\ 0.008813370 \\ 1.01367276 \\ 1.38686430 \\ 2.64127411 \\ 1.00692460 \\ -3.17513417 \end{pmatrix}$	$\begin{pmatrix} 0.999999990 \\ 1.00001051 \\ 0.999999277 \\ 0.999993965 \\ 1.00001373 \\ 0.999999244 \\ 0.999999954 \\ 0.999998713 \\ 0.999997731 \\ 0.999995802 \end{pmatrix}$	1.133635407768452E-10

Table 17: Numerical results for Problem 14 with n=20,30,50,100.

n=20		n=30	
k	$f(x_k^*)$	k	$f(x_k^*)$
1	25.024269320349966	1	26.290377511895549
2	0.681833637838455	2	0.157966086563075
3	4.831690603168681E-13	3	1.058708676282549E-12
n=50		n=100	
k	$f(x_k^*)$	k	$f(x_k^*)$
1	24.185850534414463	1	25.756905387376211
2	0.438637301206239	2	10.386247553207347
3	1.057642862178909E-11	3	0.205102803215617
		4	3.659295089164516E-13
		5	2.060573933704291E-13

Table 18: Comparison of the results.

No.	n	filled function in[14]		filled function in[17]		filled function in[4]		new filled function	
		Iter	F_f	Iter	F_f	Iter	F_f	Iter	F_f
9	2	4	21276	4	11097	2	315	2	458
10	2	15	27500	5	903	2	778	2	443
		34	54505	3	47204	2	310	2	395
11	2	44	35171	3	6295	2	303	1	316
		6	5703	3	2931	2	279	2	398
12	2	14	15759	2	2544	2	265	4	724
13	2	20	103988	3	8061	3	635	6	1020
14	2	22	107899	6	83516	3	549	2	509
	5	16	1229860	3	329956	2	5291	5	3995
	7	21	1443686	19	222630	2	12793	4	4700
	10	16	1829898	35	276386	2	33810	2	11453
	20	—	—	—	—	2	96223	3	29424
	30	—	—	—	—	3	376885	3	75364
	50	—	—	—	—	4	$> 10^6$	3	169578
	100	—	—	—	—	9	$> 10^6$	5	689936

the dimension of the problems, the advantage is more obvious.

References

[1] Jacob Barhen, Vladimir Protopopescu and David Reister, Trust: A deterministic algorithm for global optimization, *Science* 276 (1997) 1094–1097.

[2] M. Clerc and J. Kennedy, The particle swarm - explosion, stability, and convergence in a multidimen- sional complex space, *IEEE Trans. Evol. Comput.* 6 (2002) 58–73.

[3] Djurdje Cvijovic and Jacek Klinowski, Taboo search: an approach to the multiple minima problem, *Science* 267 (1995) 664.

[4] TM El-Gindy, MS Salim and AI Ahmed, A new filled function method applied to unconstrained global optimization, *Appl. Math. Comput.* 273 (2016) 1246–1256.

[5] Yuelin Gao, Yongjian Yang and Mi You, A new filled function method for global optimization, *Appl.Math.Comput.* 268 (2015) 685–695.

[6] Liu Haiyan, Wang Yuping, Guan Shiwei and Liu Xuyan, A new filled function method for unconstrained global optimization, *Int. J. Comput. Math.* 94 (2017) 2283–2296.

[7] Reiner Horst and Hoang Tuy, *Global optimization - deterministic approaches (3. Aufl.)*, Springer, New York, 1996.

[8] A Lan, Z Liansheng and C Meilin, A parameter-free filled function for unconstrained global optimiza tion, *ShanghaiUniv. Engl. Ed.* 8 (2004) 117–123.

[9] Yiu-Wing Leung and Yuping Wang, An orthogonal genetic algorithm with quantization for global numerical optimization, *IEEE Trans. Evol. Comput.* 5 (2001) 41–53.

[10] A. V. Levy and A. Montalvo, The tunneling algorithm for the global minimization of functions, *SIAM. J. Sci. Comput.* 6 (1985) 15–29.

[11] Hongwei Lin, Yuping Wang and Lei Fan, A filled function method with one parameter for unconstrained global optimization, *Appl. Math. Comput.* 218 (2011) 3776–3785.

[12] Hongwei Lin, Yuping Wang, Lei Fan and Yuelin Gao, A new discrete filled function method for finding global minimizer of the integer programming, *Appl. Math. Comput.* 219 (2013) 4371–4378.

[13] Xian Liu, Finding global minima with a computable filled function, *J. Global. Optim.* 19 (2001) 151–161.

[14] Ge R.P., A filled function method for finding a global minimizer of a function of several variables, *Math. Program.* 46 (1990) 191–204.

[15] Ge. R.P. and Qin Y.F., A class of filled functions for finding global minimizers of a function of several variables, *J. Optim. Theory. Appl.* 54 (1987) 241–252.

[16] You-lin Shang, Ding-guo Pu and Ai-ping Jiang, Finding global minimizer with one-parameter filled function on unconstrained global optimization, *Appl. Math. Comput.* 191 (2007) 176–182.

[17] Chengjun Wang, Yongjian Yang and Jing Li, A new filled function method for unconstrained global optimization, *J. Comput. Appl. Math.* 225 (2009) 68–79.

[18] Weixiang Wang, Youlin Shang and Liansheng Zhang, A filled function method with one parameter for box constrained global optimization, *Appl. Math. Comput.* 194 (2007) 54–66.

[19] Y Wang and Da-Lian Liu, A global optimization evolutionary algorithm and its convergence based on a smooth scheme and line search, *Journal of Computers-Chinese Edition.* 29 (2006) 670-675.

[20] Yuping Wang and Lei Fan, A smoothing evolutionary algorithm with circle search for global optimization, *International Conference on Network and System Security.* (2010) 412–418.

[21] Fei Wei, Yuping Wang and Hongwei Lin, A new filled function method with two parameters for global optimization, *J. Optim. Theory. Appl.* 163 (2014) 510–527.

[22] Fei Wei, Yuping Wang and Zhiqing Meng, A smoothing function method with uniform design for global optimization, *Pac. J. Optim.* 10 (2014) 385-399.

[23] Yongjian Yang and Youlin Shang, A new filled function method for unconstrained global optimization, *Appl. Math. Comput.* 173 (2006) 501–512.

[24] Xin Yao, Yong Liu and Guangming Lin, Evolutionary programming made faster, *IEEE Trans. Evol. Comput.* 3 (1999) 82–102.

[25] Y. Yao, Dynamic tunneling algorithm for global optimization, *IEEE Trans. Syst. Man. Cybern.* 19 (1989) 1222–1230.

[26] Ying Zhang and Ying-Tao Xu, A one-parameter filled function method applied to nonsmooth con- strained global optimization, *Comput. Math. Appl.* 58 (2009) 1230–1238.

[27] Ying Zhang, Liansheng Zhang and Yingtao Xu, New filled functions for nonsmooth global optimization, *Appl. Math. Model.* 33 (2009) 3114–3129.

Manuscript received 6 July 2017
revised 18 January 2018
accepted for publication 19 March 2018

Xin Sui
School of Computer Science and Technology
Xidian University, Xi'an 710071, China
E-mail address: suixin_1115@163.com

Yuping Wang
School of Computer Science and Technology
Xidian University, Xi'an 710071, China
E-mail address: ywang@xidian.edu.cn

Junhua Liu
School of Computer Science and Technology
Xidian University, Xi'an 710071, China
E-mail address: msliujunhua@163.com