Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2014, Article ID 895629, 12 pages
http://dx.doi.org/10.1155/2014/895629

Research Article

An Improved Central Force Optimization Algorithm for Multimodal Optimization

Jie Liu[1,2] and Yu-ping Wang[3]

[1]*School of Mathematics and Statistics, Xi'dian University, Xi'an 710071, China*
[2]*College of Science, Xi'an University of Science and Technology, Xi'an 710054, China*
[3]*School of Computer, Xi'dian University, Xi'an 710071, China*

Correspondence should be addressed to Jie Liu; tears191@foxmail.com

Received 24 June 2014; Revised 22 September 2014; Accepted 12 October 2014; Published 7 December 2014

Academic Editor: Boris Andrievsky

This paper proposes the hybrid CSM-CFO algorithm based on the simplex method (SM), clustering technique, and central force optimization (CFO) for unconstrained optimization. CSM-CFO is still a deterministic swarm intelligent algorithm, such that the complex statistical analysis of the numerical results can be omitted, and the convergence intends to produce faster and more accurate by clustering technique and good points set. When tested against benchmark functions, in low and high dimensions, the CSM-CFO algorithm has competitive performance in terms of accuracy and convergence speed compared to other evolutionary algorithms: particle swarm optimization, evolutionary program, and simulated annealing. The comparison results demonstrate that the proposed algorithm is effective and efficient.

1. Introduction

Unconstrained optimization is a mathematical programming technique that attempts to solve nonlinear objective functions. It has become an important branch of operations research and has a wide variety of applications in network security, cloud computation, mobile search, pattern recognition, data classification, power systems, protein structures, medical image registration, and financial market. Unconstrained optimization problems with ND variables may be written in the following form:

$$\begin{aligned} &\min \quad f(x), \quad x = (x_1, x_2, \ldots, x_{ND}) \\ &\text{s.t.} \quad x^{\min} \le x \le x^{\max}, \end{aligned} \tag{1}$$

where objective function $f(x)$ may be continuous or discontinuous, highly multimodal, or "smooth"; ND is the dimension of object function; and $x^{\min}$ and $x^{\max}$ are the lower and upper boundaries of the vector x, respectively.

In the past few decades, various academics proposed many methods in response to this problem. These methods may be grossly classified into two categories: swarm intelligent techniques and traditional direct search techniques. Firstly, swarm intelligent methods, such as the genetic algorithm developed by Leung and Wang, where a genetic algorithm with quantization for global numerical optimization is proposed [1], ant colony optimization developed by Neto and Filho, where an improving ant colony optimization approach to a permutational flow shop scheduling problem with outsourcing allowed is proposed [2], an improving particle swarm optimization developed by Green II et al., where neural networks trained by particle swarm optimization [3], simulated annealing developed by Kirkpatrick et al., which comes from annealing in metallurgy and is often used when the search space is discrete [4], gravitational search algorithm developed by Rashedi et al., which is based on the law of gravity and the notion of mass interactions [5], are efficient to well explore the whole search space and to localize the "best" areas. However, these methods have one shortage: results were never able to exactly repeat their results, for true random variables are used in algorithms. Recently, a new swarm intelligent algorithm, namely, central force optimization (CFO), is developed by Formato [6–9]. This algorithm demonstrates and leverages two characteristics that make it unique when compared to other methodologies: a basis in Newton's law of gravity and a deterministic nature. As CFO is a very

young algorithm, it has yet to be compared and contrasted against other algorithms for many different applications [9]. Pseudorandomness of CFO is discussed in [10]. Every CFO run with the same setup returns precisely the same values step by step throughout the entire run. Nevertheless, effective implementations substantially benefit from a "pseudorandom" component that enters the algorithm indirectly, not through its basic equations. Although pseudorandomness is not required in CFO, numerical experiments show that it is an important feature in effective implementations.

Secondly, direct search methods include hill climbing which is an iterative algorithm that starts with an arbitrary solution to a problem and then attempts to find a better solution by incrementally changing a single element of the solution and simplex method (SM) which is developed by Nelder and Mead [11] that is a well-defined numerical method for problems for which derivatives may not be known. These methods are more efficient than the previous ones for the exploitation of the best areas already detected. However, one has to be very careful when using these methods since it is very sensitive to the choice of initial points and not guaranteed to find the best possible solution (the global optimum) out of all possible solutions.

Although standard CFO is a deterministic algorithm, the inherent drawback with most of the population based stochastic algorithms is premature convergence. Any swarm intelligence algorithm is regarded as an efficient algorithm if it is fast in convergence and able to explore the minimum area of the search space. In other words, if CFO is capable of balancing between exploration and exploitation of the search space, then CFO is regarded as an efficient algorithm. Also some numerical experiments proved that stagnation is another inherent drawback with CFO; that is, CFO sometimes stops proceeding towards the global optima even though the population has not converged to local optima or any other point [8, 12]. The problems of premature convergence and stagnation worth considering for designing an efficient improving CFO algorithm.

In this study, a hybrid algorithm is proposed. The motivation of such a hybrid is to explore a better trade-off between computational cost and global optimality of the solution attained. Generally hybrid methods achieve better solutions than "pure" methods and converge more quickly. Similar ideas have been discussed in hybrid methods using evolutionary algorithm and direct search technique [13–15]. However, these hybrid methods are stochastic.

The current study investigates the deterministic hybridization of the simplex method and central force optimization, and performance of the hybrid algorithm is compared with other pertinent alternatives via simulations. In order to overcome the shortage of the simplex method, we propose a clustering algorithm to select suitable vertices from population. To have a better clustering effect, a unique method, namely, good points set method, is embedded in clustering algorithm.

The structure of the rest of the paper is as follows. Section 2 gives us a review on the related works. In Section 3, we point out the drawback of CFO and present a new hybrid algorithm. In Section 4, we test the new algorithm against a variety of multidimension functions, all drawn from recognized "benchmark suites" (complex functions with analytically known minima). Finally, Section 5 outlines the conclusions, followed by Appendix which shows the benchmark functions.

Table 1: Unimodal test functions.

Test function	S
$f_1(x) = \sum_{i=1}^{n} x_i^2$	$[-100, 100]^n$
$f_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	$[-10, 10]^n$
$f_3(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^2$	$[-100, 100]^n$
$f_4(x) = \max_i \{ \lvert x_i \rvert, 1 \le i \le n \}$	$[-100, 100]^n$
$f_5(x) = \sum_{i=1}^{n-1} \left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2 \right]$	$[-30, 30]^n$
$f_6(x) = \sum_{i=1}^{n} \left([x_i - 1] \right)^2$	$[-100, 100]^n$
$f_7(x) = \sum_{i=1}^{n} i x_i^4 + \text{random}[0, 1)$	$[-1.28, 1.28]^n$

2. Central Force Optimization and Simplex Method

2.1. Central Force Optimization Algorithm (CFO). In CFO, every individual of population is named probe. Probes are attracted by gravitation based on the defined mass. Probes are considered as objects and their performances are measured by the fitness function. In other words, each mass represents a solution, which is navigated by adjusting the position properly according to the Newton universal law of gravitation.

CFO comprises three steps: (1) initialization; (2) computation of probe's acceleration; and (3) motion. A general working flow of CFO is shown in Table 1.

In the first step, a population of probes is created in search space. The initial position and acceleration vectors are set to zero. In the second step, the compound acceleration vector of one probe from components in each direction is calculated according to the Newton universal law. Mass is a user-defined function from the object function to be minimized. In the ND-dimensional search space with NP probes $x^p = (x_1^p, x_2^p, \ldots, x_{ND}^p)$, $(p = 1, 2, \ldots, NP)$, the pth probe operates according to the following formula:

$$\mathbf{A}_t^p = G \sum_{\substack{k=1,\\ k \neq p}}^{NP} U\left(M_{t-1}^k - M_{t-1}^p\right) \left(M_{t-1}^k - M_{t-1}^p\right)^{\alpha} \times \left(\mathbf{x}_{t-1}^k - \mathbf{x}_{t-1}^p\right) \left\| \mathbf{x}_{t-1}^k - \mathbf{x}_{t-1}^p \right\|^{-\beta}, \tag{2}$$

where $\mathbf{x}_t^p$, $\mathbf{A}_t^p$ are the position and acceleration vectors for the pth probe at the tth generation, respectively, $M_{t-1}^p = f(x_1^{p,t-1}, x_2^{p,t-1}, \ldots, x_{ND}^{p,t-1})$ is the fitness of the pth probe,

namely, the objective function value, and $U(z)$ represents a piecewise function and t is the number of iterations:

$$U(z) = \begin{cases} 1, & z \geq 0 \\ 0, & \text{otherwise.} \end{cases} \quad (3)$$

In (2), G, α, and β do not represent the concrete gravitational fundamentals. In addition, in order to prevent the probe from flying away from the search space, it is required to detect the position of the probe. If the probe is out of the range, it is pulled back to the search space.

In third step, according to the acceleration calculated previously, the position vectors of probes are updated based on the Newtonian formula. If acceleration $\mathbf{A}_t^p$ is exerted, the pth probe will move from $\mathbf{x}_t^p$ to $\mathbf{x}_{t+1}^p$ according to the motion equation

$$\mathbf{x}_{t+1}^p = \mathbf{x}_t^p + 0.5 \times \mathbf{A}_t^p \Delta t^2, \quad (4)$$

where Δt represents the step. The position of probe is updated based on last "mass" information as a deterministic gradient algorithm.

The convergence conditions of CFO have revealed that it will converge to the optima that have searched so far, which is not worse than the predefined one in initial distributions [9].

Standard Central Force Optimization Algorithm

Step 1 (parameters initialization). Set the objective function's dimension ND, boundaries $x^{\min}$, $x^{\max}$, total number of probes NP, gravitational constant G, and acceleration parameters α and β.

Step 2 (population initialization). Compute initial probe distribution X, fitness matrix M, position vectors $\mathbf{R}$, and acceleration vectors $\mathbf{A}$.

Step 3 (loop on time step). Consider the following.

Step 3.1. Compute probe position vectors $\mathbf{R}$.

Step 3.2. If probe flies out of the boundary, we retrieve it.

Step 3.3. Update fitness matrices of current probes.

Step 3.4. Compute accelerations vectors $\mathbf{A}$ for next time step.

Step 4. Increase time step and repeat Step 3 until stopping criterion has been met.

2.2. Nelder-Mead Simplex Method (SM). Nelder-Mead simplex method (SM) is a local search method designed for unconstrained optimization, known as direct search methods. In contrast to more local optimization methods, SM is a derivative-free method as it does not require any information about the gradient (or higher derivative) of the objective function when searching for an optimal solution. Therefore, SM is particularly appropriate for solving noncontinuous, nondifferentiable, and multimodal optimization problems.

It uses four parameters: reflection, expansion, contraction, and size of the simplex to move in the design space based on the values at the vertices and center of the triangle.

It is an ND-dimensional, closed geometric figure in space that has straight line edges intersecting at $ND+1$ vertices. The values of $ND + 1$ vertex points functions are calculated, and the vertex point W with the maximum value, the vertex point B with the minimum value, and the vertex point N with the second largest value of the objective function are obtained, respectively. The basic move is a reflection to generate a new vertex point R. The choice of reflection direction and the choice of the new vertex depend on the location of the worst point W in the simplex. The new point R is called the "complement" of the worst point W. If any "new point" is the worst point in the new simplex, the algorithm would oscillate; in other words, it would bounce back and forth between this point and the earlier worst point. When this happens, the second worst point is used as the point to use to find the next "new point." As the simplex moves through the design space, its centroid moves toward the extremum. If the edges of the sides of the simplex are allowed to contract and expand, we can see how the method could accelerate toward the optimum. This method has the versatility of adapting itself to the local landscape of the merit surface.

Nelder-Mead Simplex Method Algorithm

Step 1 (order). Evaluate objective function values $f(x)$ at the $ND + 1$ vertices of simplex and sort it according to the function values $f(x_1) \leq f(x_2) \leq \cdots \leq f(x_{ND+1})$.

Step 2 (reflection). Compute the reflection point R from $x_R = \overline{x} + \alpha(\overline{x} - x_{ND+1})$, where $\overline{x}$ is the center of gravity of all points except x_{ND+1}; that is, $\overline{x} = (1/ND) \sum_{i=1}^{ND} x_i$. If $f(x_1) \leq f(x_R) \leq f(x_{ND})$, replace x_{ND+1} with x_R.

Step 3 (expansion). If $f(x_R) < f(x_1)$, then compute the expansion point x_E from $x_E = \overline{x} + \alpha(x_R - \overline{x})$. If $f(x_E) < f(x_R)$, replace x_{ND+1} with x_E; otherwise replace x_{ND+1} with x_R.

Step 4 (outside contraction). If $f(x_{ND}) \leq f(x_R) \leq f(x_{ND+1})$, compute the outside contraction point $x_{C^+} = \overline{x} + \beta(x_R - \overline{x})$. If $f(x_{C^+}) \leq f(x_R)$, replace x_{ND+1} with x_{C^+}. Otherwise go to Step 6.

Step 5 (inside contraction). If $f(x_R) \geq f(x_{ND+1})$, compute the inside contraction point x_{C^-} from $x_{C^-} = \overline{x} + \gamma(\overline{x} - x_R)$. If $f(x_{C^-}) < f(x_{ND+1})$, replace x_{ND+1} with x_{C^-}. Otherwise, go to Step 6.

Step 6 (reduction). For all but the point x_1, replace the point with $x_i = x_1 + \delta(x_i - x_1)$ for all $i \in \{2, \ldots, ND + 1\}$. If the stopping criterion is satisfied, then STOP. Otherwise, go to Step 1.

3. Hybrid CSM-CFO Method

This section introduces the hybrid method and, in doing so, also demonstrates that the convergence of the SM and the

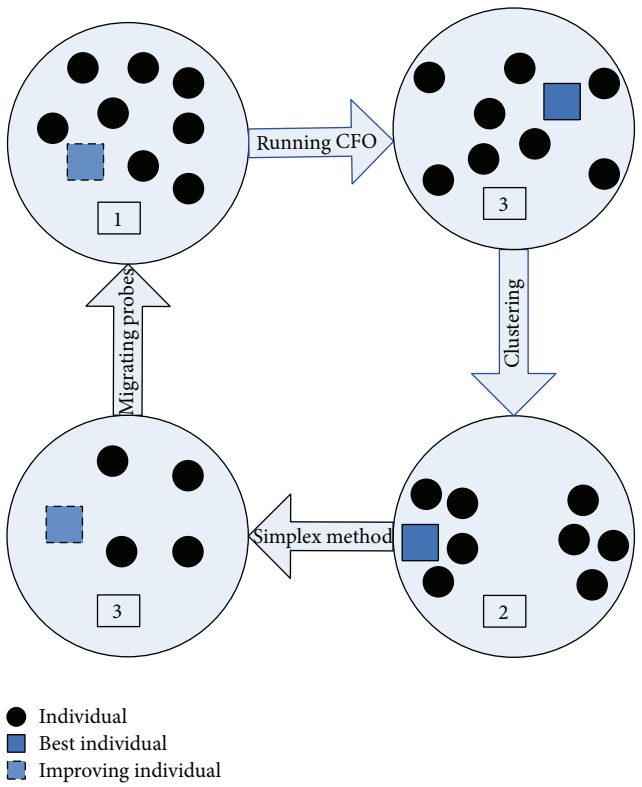


Figure 1: Model of CSM-CFO algorithm.

accuracy of the CFO can be further improved simultaneously.

A good mix strategy should be able to integrate the global exploration and the local search organically in the design of the algorithm: global exploration algorithm can constantly search in the decision domain of the issue, avoiding premature convergence phenomenon; local search can make a deep search in the region which may contain the optimal solution, promoting a faster convergence for the global optimal solution. Based on this idea, the simplex method (SM) is introduced into the CFO in this paper, improving the local search capability. With this approach, simplex structures are periodically formed starting from the current optimal solution of CFO, which helps the algorithm be able to deeply develop the global optimum.

The model of SM-CFO algorithm is shown in Figure 1, containing three main steps: global search with CFO, local search with clustering SM, and probes migration. The migration operation of the probe can effectively link the exploration and exploitation for the convergence of CSM-CFO algorithm. In order to improve the response speed of SM and ensure the convergence speed, we design the following migration strategy: after several evolutionary generations of the probe populations, the population will be subdivided into two classes. One class which includes the best probe will be chosen to construct a simplex and run the Nelder-Mead simplex method. The worst probe is altered by the exploration point generated by the SM operation. The flow chart of SM-CFO algorithm is shown in Figure 2.

Numerical experiments show that it is not very satisfactory to evaluate the probe by simply using the objective function value as the fitness function. The range of function values varies largely for different problems. This will make the algorithm not convergent or overflowing when parameters' setting is inappropriate. So, parameters' adjustment is an extremely difficult assignment. In order to solve this problem, we give the following definition on fitness:

$$\text{fitness}_p(t) = \frac{M(p,t) - \min_{p\in\{1,\ldots,NP\}} M(p,t)}{\max_{p\in\{1,\ldots,NP\}} M(p,t) - \min_{p\in\{1,\ldots,NP\}} M(p,t)}. \tag{5}$$

Then we can select easily a set of fixed parameters of CSM-CFO by numerical experiments for different problems. Moreover, (5) can also make CSM-CFO more robust and efficient.

In CSM-CFO the main structure and properties of CFO are preserved. However, to give the ability of handling the complex or ill functions to CFO, we apply some modification on CFO inspired by simplex method and cluster. It is noted that the proposed algorithm is able to form niches. According to the above definition, steps of the proposed CSM-CFO are as follows.

Algorithm 1 (CSM-CFO). Consider the following.

Step 1 (initialization). Generate the initial swarm of NP individuals with "3-D Fano load matching network" [6].

For $i = 1$ to ND, $n = 1$ to NP/ND:

$$p = n + \frac{(i-1)\,NP}{ND},$$
$$x(p,i,0) = x_i^{\min} + \frac{(n-1)\left(x_i^{\max} - x_i^{\min}\right)}{NP/ND - 1}, \tag{6}$$

and set $t = 0$. Each individual represents a candidate solution and is considered as a real valued vector x_i $(i = 1, 2, \ldots, NP)$, where NP is the population size, ND is the dimension of objective function, and x_i is decision variables to be optimized. $x^{\max}$ and $x^{\min}$ are upper and lower boundaries of variables. Set parameters of simplex method: reflection, contraction, and expansion coefficients. Operation period of cluster simplex method is set to Δt. Current populations are divided into N_c subpopulations.

Step 2 (fitness evaluation). Calculate fitness value of each individual $x_i(t)$, $(i = 1, 2, \ldots, NP)$ with (5).

Step 3 (computation of position, acceleration, and velocity). At time t, the position, acceleration, and velocity of each individual are computed according to (2) and (4), respectively.

Step 4 (cluster). If $\text{mod}(t, \Delta t) = 0$, populations are divided into N_c subpopulations with Algorithm 2; then go to Step 5.

Step 5 (simplex method search). One subpopulation is chosen which includes the best individual. Simplex method is run in this subpopulation. The improving best individual is moved

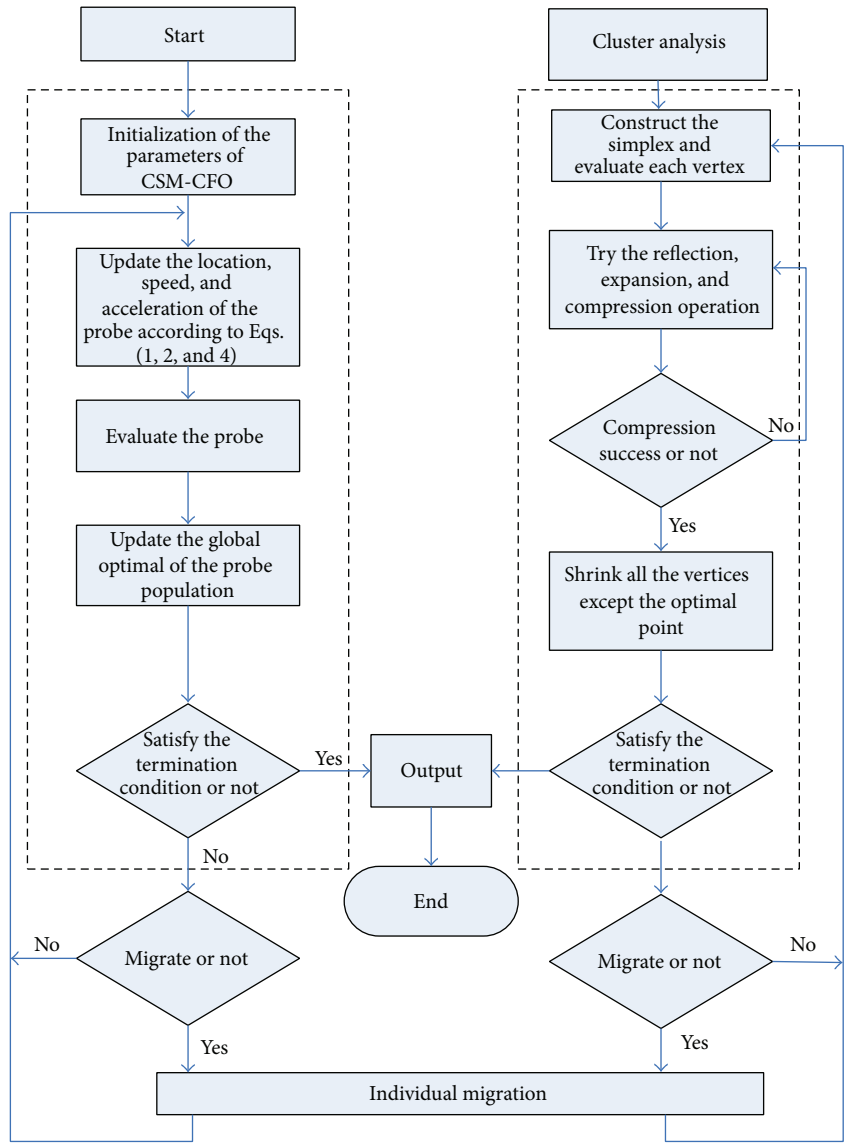


Figure 2: Schematic flow diagram of CSM-CFO algorithm.

into populations and replaces the worst individual; then go to Step 6.

Step 6. $t := t + 1$; Steps 2–5 are repeated until the stopping criterion is satisfied.

Algorithm 2 (clustering algorithm). Consider the following.

Step 1. Produce a reference point R from the search space by Algorithm 3.

Step 2. Search the nearest individual $\widetilde{X}$ to R, which means that point $\widetilde{X}$ has the minimum distance from R.

Step 3. Repeat Step 2 until M individuals of population are selected. These individuals form a subpopulation.

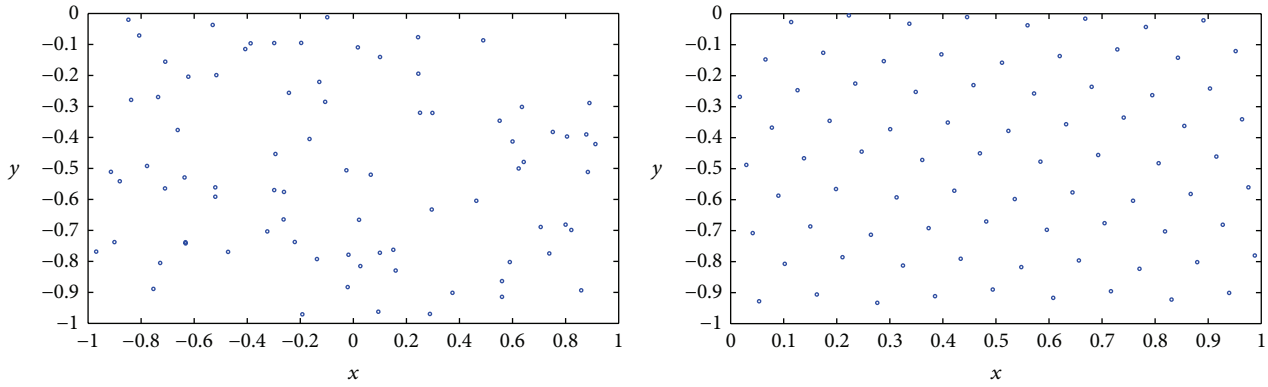
Step 4. These M individuals are removed from current population.

Step 5. Repeat Steps 2–4 until population is divided into N_p/M subpopulations.

Algorithm 3 (good points set algorithm). Consider the following.

Step 1. Generate a point $r = (r_1, r_2, \ldots, r_{ND})$, $r_i = \{2\cos(2\pi i/p)\}$, where p is minimum prime number content with $(p-3)/2 \geq ND$.

Step 2. Let $P_n(k) = \{\{r_1 * k\}, \{r_2 * k\}, \ldots, \{r_{NP} * k\}\}$, where $\{r_i * k\}$ is the decimal fraction of $r_i * k$, $k = 1, 2, \ldots, n$; then points set $\{P_1, P_2, \ldots, P_n\}$ is called good points set.



(a) 80 initial points generated by uniform random point

(b) 80 initial points generated by good points set method

Figure 3: Comparison of good points set method and uniform random point.

Step 3. Map is defined as follows: $f(\{r_i * k\}) = x_i^{\min} + \{r_i * k\}(x_i^{\max} - x_i^{\min})$, which means good point is mapped to search region S.

Note 1. Vertices of simplex are generated by Algorithm 2. The population is divided into two subgroups. One group which includes the best currently probes will be chosen. If the number of this group is larger than $ND + 1$, the best $ND + 1$ probes construct the simplex. If the number of this group is less than $ND + 1$, some probes from another group which is nearest to the best probe will be chosen and added to the group to construct simplex. Note that the clustering algorithm is not strictly the classical clustering algorithm, for example, K-medoids algorithm, Birch algorithm, Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm, and others. In this paper, we focus on dividing the whole population into a number of subpopulations. However, enhancing the result of Algorithm 2, a reference point for Algorithm 2 is generated by Algorithm 3 (good points set method) [16, 17].

The idea of good points set make the points set distributes more evenly than random points. When we generate the reference point of Algorithm 2 by means of good points set method, obviously, the points are even and representative (see Figure 3). This will help Algorithm 2 get better result than generating reference point randomly.

Note 2. The local stopping criterion is defined as

$$\max_{i,j=1,\ldots,ND+1} \left\| P_i - P_j \right\| \leq \varepsilon, \tag{8}$$

where P_i denotes the vertices of current simplex and $\| \cdot \|$ denotes the Euclidean norm. In order to achieve quicker convergence, the simplex method will stop when either (8) is satisfied or the given number of iterations is reached. In the following numerical experiment, ε is 10^{-2} and the number of iterations is 10. Stopping criterion of CSM-CFO is the maximum number of function evaluations reached.

4. Numerical Experiments and Analysis

To evaluate the performance of the proposed algorithms, a number of internationally recognized standard test functions (i.e., f_1–f_{23}) [18] and CEC benchmark functions [19] are selected. The standard test functions are divided into three classes: benchmark function set (1) unimodal functions, benchmark function set (2) multimodal functions, and benchmark functions set (3) fix dimension functions, as shown in Tables 1, 2, and 3. These functions are widely used in published papers and are very difficult to track the global minimum. For example, for f_2–f_7, the existing local optima can misguide the population to move away from the true global optimum; f_8–f_{13} have more global minimum, many local minima around them. CEC benchmark functions are more complex than standard test functions.

4.1. Standard Test Functions. Tables 1–3 represent the functions used in our numerical experiment. In these tables, n is the dimension of function and S is the search space. A detailed description of the functions 1–23 is given in the appendix.

4.2. Experimental Setting. A comparative study is done to assess the effectiveness of the CSM-CFO by experiments on test functions. The parameter settings of CSM-CFO are summarized as follows. The initial population of CSM-CFO is generated by "3-D Fano load matching network" in [6]. The population size is $N = 100$. The gravitation constant α is 1 and β is 2. The constant G is 15. Reflection coefficient, contraction coefficient, and expansion coefficient are 1, 1, and 0.5, respectively. Termination parameter is dimension of test functions $f_1 \sim f_{13}$ and dimension of CEC benchmark functions $f_{\text{CEC1}} \sim f_{\text{CEC5}}$ is 30.

We compared the performance of CSM-CFO with that of three different evolutionary algorithms:

(1) the NM-SM PSO method developed by Fan and Zahara (NM-PSO) [14];

(2) evolutionary programming developed by Yao et al. (EP) [18];

Table 2: Multimodal test functions.

Test function	S
$f_8(x) = \sum_{i=1}^{n} -x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)$	$[-500, 500]^n$
$f_9(x) = \sum_{i=1}^{n} \left[x_i^2 - 10\cos(2\pi x_i) + 10\right]$	$[-5.12, 5.12]^n$
$f_{10}(x) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$	$[-32, 32]^n$
$f_{11}(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$[-600, 600]^n$
$f_{12}(x) = \frac{\pi}{n}\left\{10\sin(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2\left[1 + 10\sin^2(\pi y_{i+1})\right] + (y_n - 1)^2\right\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$, $y_i = 1 + \frac{(x_i + 1)}{4}$, $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a \\ 0, & -a < x_i < a \\ k(-x_i - a)^m, & x_i < -a \end{cases}$	$[-50, 50]^n$
$f_{13}(x) = 0.1\left\{\sin^2(3\pi x_1) + \sum_{i=1}^{n}(x_i - 1)^2\left[1 + \sin^2(3\pi x_i + 1)\right] + (x_n - 1)^2\left[1 + \sin^2(2\pi x_n)\right]\right\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$	$[-50, 50]^n$

Table 3: Multimodal test functions with fix dimension.

Test function	S
$f_{14}(x) = \left(\frac{1}{500} + \sum_{j=1}^{25}\left(j + \sum_{i=1}^{2}(x_i - a_{ij})^6\right)^{-1}\right)^{-1}$	$[-65.53, 65.53]^2$
$f_{15}(x) = \sum_{i=1}^{11}\left\{a_i - [x_1(b_i^2 + b_i x_2)][b_i^2 + b_i x_3 + x_4]^{-1}\right\}^2$	$[-5, 5]^4$
$f_{16}(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	$[-5, 5]^2$
$f_{17}(x) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos x_1 + 10$	$[-5, 0]\ [0, 15]$
$f_{18}(x) = \left[1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)\right] \times \left[30 + (2x_1 - 3x_2)^2 \times (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)\right]$	$[-5, 5]^2$
$f_{19}(x) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2\right)$	$[0, 1]^3$
$f_{20}(x) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2\right)$	$[0, 1]^6$
$f_{21}(x) = -\sum_{i=1}^{5}\left[(x - a_i)(x - a_i)^T + c_i\right]^{-1}$	$[0, 10]^4$
$f_{22}(x) = -\sum_{i=1}^{7}\left[(x - a_i)(x - a_i)^T + c_i\right]^{-1}$	$[0, 10]^4$
$f_{22}(x) = -\sum_{i=1}^{10}\left[(x - a_i)(x - a_i)^T + c_i\right]^{-1}$	$[0, 10]^4$

(3) hybrid simulated annealing method developed by Hedar and Fukushima (DSSA) [20].

All the control parameters, for example, mutation rate of the EP, inertia weight of the NM-PSO, temperature of DSSA, and so forth, were set to be default as recommended in original articles. In addition, the maximum number of function evaluations for the standard test functions was set to be 150 000. The maximum number of evaluations for the CEC benchmark functions was set to be 300 000.

4.3. Comparison of Algorithm Performance. The results of the performance evaluation, as achieved by the four algorithms for the four types of functions, unimodal, multimodal high

Table 4: Minimization result of benchmark functions in Table 1 with $n = 30$.

Function	CSM-CFO	NM-PSO		EP		DSSA	
		Mean	Std.	Mean	Std.	Mean	Std.
f_1	1.53×10^{-8}	1.94×10^{-8}	1.16×10^{-8}	3.17	1.66	$\mathbf{3.69 \times 10^{-37}}$	2.45×10^{-36}
f_2	2.9×10^{-4}	3.70×10^{-5}	8.61×10^{-5}	0.57	0.13	$\mathbf{2.91 \times 10^{-24}}$	1.13×10^{-23}
f_3	$\mathbf{5.5 \times 10^{-10}}$	5.78	3.68	9749.91	2594.95	1.19×10^{-3}	2.11×10^{-3}
f_4	**0**	0.10	3.99	7.96	1.50	0.41	0.25
f_5	**35.28**	49.83	30.17	338.56	361.49	37.35	32.14
f_6	8.4×10^{-2}	$\mathbf{1.60 \times 10^{-2}}$	1.30×10^{-1}	3.69	1.95	1.4×10^{-1}	4.1×10^{-1}
f_7	$\mathbf{7.3 \times 10^{-4}}$	7.37×10^{-2}	9.25×10^{-2}	0.10	3.62	9.90×10^{-3}	3.53×10^{-2}

Table 5: Minimization result of benchmark functions in Table 2 with $n = 30$.

Function	CSM-CFO	NM-PSO		EP		DSSA	
		Mean	Std.	Mean	Std.	Mean	Std.
f_8	**−12569.48**	−12569.48	2.21×10^{-2}	−12566.09	2.10	−9659.69	463.78
f_9	$\mathbf{1.21 \times 10^{-1}}$	1.01	0.95	0.65	0.35	20.78	5.94
f_{10}	**−2.06e − 5**	2.65×10^{-5}	3.08×10^{-5}	0.86	0.28	1.34×10^{-3}	4.23×10^{-2}
f_{11}	$6.1e - 2$	$\mathbf{3.07 \times 10^{-2}}$	3.08×10^{-2}	1.00	6.75×10^{-2}	0.23	0.44
f_{12}	-2.36×10^{-3}	$\mathbf{2.74 \times 10^{-11}}$	9.16×10^{-11}	4.35×10^{-2}	5.05×10^{-2}	3.95×10^{-2}	9.14×10^{-2}
f_{13}	$\mathbf{4.18 \times 10^{-5}}$	4.69×10^{-5}	7.00×10^{-4}	0.16	7.06×10^{-2}	5.05×10^{-2}	0.56

dimensional, multimodal functions with fixed-dimensional, and CEC benchmark functions, are shown.

4.3.1. Unimodal Functions. Functions f_1 to f_7 are unimodal functions for which the focus is on the convergence rate because current optimization algorithms are already able to present global optima that are close to the actual optima. However, unimodal functions have been adopted to assess the convergence rate of evolutionary algorithms. Therefore, the aim is to obtain the best global minimum in the least number of required iterations. We tested the CSM-CFO on a set of unimodal functions in comparison with the other three algorithms. Table 4 lists the mean and standard deviations of the function values in the last iteration. As Table 4 illustrates, CSM-CFO generated significantly better results than EP on all the unimodal functions. From comparisons of CSM-CFO, NM-PSO, and DSSA, we can see that CSM-CFO had significantly better performance on functions 3, 4, 5, and 7. In summary, the search performance of the four algorithms tested can be ordered as CSM-CFO > NM-PSO > DSSA > EP.

4.3.2. Multimodal Functions. This set of benchmark functions have more than one local optimum but can either have a single or more than one global optimum. For multimodal functions, the final results are more important since they reflect the ability of the algorithm in escaping from poor local optima and locating a near-global optimum. We have carried out experiments on f_8 to f_{13} where the number of local minima increases exponentially as the dimension of the function increases. The dimension of these functions is set to 30. The results of NM-PSO, EP, and DSSA are averaged over 30 runs and mean, standard deviations of the function values are reported for these functions in Table 5.

From Table 5, it is clear to see that for four of the tested benchmark functions, CSM-CFO outperformed other algorithms. For functions 8, 9, 10, and 13 especially, CSM-CFO provides much better solution than other algorithms. However, for functions f_{11} and f_{12}, CSM-CFO cannot tune itself and has not a good performance. NM-PSO outperformed CSM-CFO statistically. This is also in accord with "no free lunch" theorem. It can be concluded from Table 5 that the order of the search performance of these four algorithms is CSM-CFO > NM-PSO > DSSA > EP.

4.3.3. Functions in Fixed Dimension. This set of benchmark functions f_{14}–f_{23} has fixed dimensions and has only a few local minima. Compared to the multimodal functions with many local minima (f_8–f_{13}), this set of functions is not challenging: some of them can even be solved efficiently by deterministic algorithms.

From Table 6, we can see that, in comparison to EP, NM-PSO and DSSA achieved better results on all benchmark functions. In comparison with NM-PSO, it can be seen that CSM-CFO has a better performance on most of the functions except the function 21 where NM-PSO generated better average results than CSM-CFO. Table 6 shows the comparison between CSM-CFO, CFO, and PSO on multimodal fixed-dimensional test functions of Table 3. The results show that CSM-CFO, CFO, and PSO have similar solutions and the performances are almost the same. It indicates that there is no big difference between CSM-CFO and other algorithms when dimension of test functions is lower. From Table 6 we can see that the order of the search performance of these four algorithms is CSM-CFO ≈ NM-PSO > DSSA > EP.

4.3.4. CEC Benchmark Functions. For the sake of testing the performance of the proposed algorithm and illustrating our

Table 6: Minimization result of benchmark functions in Table 3.

Function	CSM-CFO	NM-PSO		EP		DSSA	
		Mean	Std.	Mean	Std.	Mean	Std.
f_{14}	**0.99**	0.99	0	0.99	0	1.02	0.14
f_{15}	$\mathbf{3.0 \times 10^{-4}}$	3.77×10^{-4}	2.59×10^{-4}	7.08×10^{-3}	7.85×10^{-3}	3.80×10^{-4}	2.50×10^{-4}
f_{16}	**−1.03**	−1.03	0	−1.02	3.13×10^{-4}	−1.01	1.27×10^{-2}
f_{17}	**0.39**	0.39	0	0.40	1.03×10^{-2}	0.40	6.88×10^{-2}
f_{18}	**3.00**	3.0	0	7.50	10.39	3.00	1.21×10^{-3}
f_{19}	**−3.86**	−3.86	3.84×10^{-6}	−3.86	6.28×10^{-4}	−3.85	6.10×10^{-2}
f_{20}	**−3.32**	−3.26	5.96×10^{-2}	−3.26	6.03×10^{-2}	−3.18	6.10×10^{-2}
f_{21}	−5.11	**−6.09**	3.45	−5.16	2.92	−7.54	3.03
f_{22}	$\mathbf{-1.07 \times 10^{-1}}$	−6.55	3.24	−5.16	2.92	−8.35	2.01
f_{23}	$\mathbf{-1.04 \times 10^{-1}}$	−7.40	3.21	−4.91	3.48	−8.94	1.63

Table 7: Minimization result of CEC benchmark functions.

Function	CSM-CFO	NM-PSO		EP		DSSA	
		Mean	Std.	Mean	Std.	Mean	Std.
f_{CEC1}	0	**0**	0	7.90	5.71×10^{1}	3.33	1.83×10^{1}
f_{CEC2}	$\mathbf{1.17 \times 10^{2}}$	1.20×10^{2}	6.10×10^{1}	2.68×10^{1}	3.47×10^{1}	1.90×10^{2}	1.06×10^{2}
f_{CEC3}	5.56×10^{2}	$\mathbf{2.39 \times 10^{2}}$	5.76×10^{1}	8.69×10^{3}	2.12×10^{3}	7.67×10^{2}	4.72×10^{2}
f_{CEC4}	9.51×10^{2}	$\mathbf{9.00 \times 10^{2}}$	0	1.07×10^{3}	7.8×10^{1}	9.18×10^{2}	5.53
f_{CEC5}	$\mathbf{1.29 \times 10^{1}}$	1.32×10^{1}	2.76	5.05×10^{1}	1.99×10^{1}	3.18×10^{1}	3.08×10^{1}

arguments about searching on high-dimensional problems, we choose five recently developed novel composition functions, CEC benchmark functions. CEC will provide a set of test functions for competition every year. For example, CEC2013 LSGO benchmark suite is currently the latest proposed benchmark in the field of large-scale optimization. The topic of CEC2005 is single objective global optimization, and a set of benchmark functions is published. We chose CEC1–CEC5 as test functions to test these four algorithms. Details of constructions and properties of the composition functions can be found in [19].

The results of CSM-CFO are compared with the performance of the other three algorithms which are presented in Table 7, where the mean and standard deviation from 30 independent runs are listed except for CSM-CFO. On CEC1 to CEC5, CSM-CFO achieves better results compared to other algorithms. However, there is no significant difference for CSM-CFO and NM-PSO. From Table 7 we can see that the order of the search performance of these four algorithms is NM-PSO > CSM-CFO > DSSA > EP.

5. Conclusion

In this paper, a hybrid CSM-CFO algorithm is presented for locating the global optima of continuous unconstrained optimization problems. New algorithm preserves the main merits of central force optimization that the results will not change when initial population is unchanged. Clustering technique and good point set method are used to enhance the robustness of simplex method. The substantial improvements upon the effectiveness, efficiency, and accuracy are reported to justify the claim that the hybrid approach presents an excellent trade-off between exploitation in simplex method and exploration in central force optimization.

In order to evaluate proposed algorithm, it is tested on a set of standard benchmark problems. The results obtained by CSM-CFO in most cases provide superior results and in all cases are comparable with NM-PSO, EP, and DSSA. The application of CSM-CFO in the large-scale optimization problems and the improvement of the convergence speed need to be studied further and will be our ongoing work in the future.

Appendix

The 23 test functions we employed are given below. To define the test functions, we have adopted the following general format.

(Dimension): name of function

(a) function definition and search space;
(b) global optimum.

Test Functions

Function 1 (30-D): sphere:

(a) $f_1(x) = \sum_{i=1}^{n} x_i^2$, $-100 \le x_i \le 100$;
(b) global optimum with $f_1 = 0$ at $(0, 0, \ldots, 0)$.

Function 2 (30-D): Schwefel 2.22:

(a) $f_2(x) = \sum_{i=1}^{n} |x_i| + \prod_{i=1}^{n} |x_i|$, $-10 \le x_i \le 10$;
(b) global optimum with $f_2 = 0$ at $(0, 0, \ldots, 0)$.

Function 3 (30-D): Schwefel 1.2:

(a) $f_3(x) = \sum_{i=1}^{n} (\sum_{j=1}^{i} x_j)^2$, $-100 \le x_i \le 100$;
(b) global optimum with $f_3 = 0$ at $(0, 0, \ldots, 0)$.

Function 4 (30-D): Schwefel 2.21:

(a) $f_4(x) = \max_i \{|x_i|, 1 \le i \le n\}$, $-100 \le x_i \le 100$;
(b) global optimum with $f_4 = 0$ at $(0, 0, \ldots, 0)$.

Function 5 (30-D): Rosenbrock:

(a) $f_5(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$, $-30 \le x_i \le 30$;
(b) global optimum with $f_5 = 0$ at $(1, 1, \ldots, 1)$.

Function 6 (30-D): step function:

(a) $f_6(x) = \sum_{i=1}^{n} ([x_i - 1])^2$, $-100 \le x_i \le 100$;
(b) global optimum with $f_6 = 0$ at $(0, 0, \ldots, 0)$.

Function 7 (30-D): quartic function:

(a) $f_7(x) = \sum_{i=1}^{n} i x_i^4 + \text{random}[0, 1)$, $-1.28 \le x_i \le 1.28$;
(b) global optimum with $f_7 = 0$ at $(0, 0, \ldots, 0)$.

Function 8 (30-D): Schwefel 2.26:

(a) $f_8(x) = \sum_{i=1}^{n} -x_i \sin(\sqrt{|x_i|})$, $-500 \le x_i \le 500$;
(b) global optimum with $f_8 = -12569.5$ at $(420.9687, 420.9687, \ldots, 420.9687)$.

Function 9 (30-D): Rastrigin:

(a) $f_9(x) = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10]$, $-5.12 \le x_i \le 5.12$;
(b) global optimum with $f_9 = 0$ at $(0, 0, \ldots, 0)$.

Function 10 (30-D): Ackley:

(a) $f_{10}(x) = -20 \exp(-0.2\sqrt{(1/n) \sum_{i=1}^{n} x_i^2}) - \exp(\sum_{i=1}^{n} \cos(2\pi x_i)) + 20 + e$, $-32 \le x_i \le 32$;
(b) global optimum with $f_{10} = 0$ at $(0, 0, \ldots, 0)$.

Function 11 (30-D): Griewank:

(a) $f_{11}(x) = (1/4000) \sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos(x_i/\sqrt{i}) + 1$, $-600 \le x_i \le 600$;
(b) global optimum with $f_{11} = 0$ at $(0, 0, \ldots, 0)$.

Table 8: Coefficients of function f_{19}.

i	a_{ij}, $j = 1, 2, 3$			c_i	p_{ij}, $j = 1, 2, 3$		
1	3	10	30	1	0.3689	0.1170	0.2673
2	0.1	10	35	1.2	0.4669	0.4387	0.7470
3	3	10	30	3	0.1091	0.8732	0.5547
4	0.1	10	35	3.2	0.038150	0.5743	0.8828

Function 12 (30-D): generalized penalized function:

(a) $f_{12}(x) = (\pi/n)\{10 \sin(\pi y_1) + \sum_{i=1}^{n-1} (y_i - 1)^2 [1 + 10\sin^2(\pi y_{i+1})] + (y_n - 1)^2\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$, $-50 \le x_i \le 50$;
(b) global optimum with $f_{12} = 0$ at $(1, 1, \ldots, 1)$.

Function 13 (30-D): generalized penalized function:

(a) $f_{13}(x) = 0.1\{\sin^2(3\pi x_1) + \sum_{i=1}^{n} (x_i - 1)^2 [1 + \sin^2(3\pi x_i + 1)] + (x_n - 1)^2 [1 + \sin^2(2\pi x_n)]\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$, $-50 \le x_i \le 50$, where $y_i = 1 + (x_i + 1)/4$:

$$u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a \\ 0, & -a < x_i < a \\ k(-x_i - a)^m, & x_i < -a; \end{cases} \quad \text{(A.1)}$$

(b) global optimum with $f_{13} = 0$ at $(1, 1, \ldots, 1)$.

Function 14 (2-D): Shekel's foxholes:

(a) $f_{14}(x) = ((1/500) + \sum_{j=1}^{25} (j + \sum_{i=1}^{2} (x_i - a_{ij})^6)^{-1})^{-1}$, $-65.536 \le x_i \le 65.536$, where

$$(a_{ij}) = \begin{pmatrix} -32 & -16 & 0 & 16 & 32 & -32 & \cdots & 0 & 16 & 32 \\ -32 & -32 & -32 & -32 & -32 & -16 & \cdots & 32 & 32 & 32 \end{pmatrix}; \quad \text{(A.2)}$$

(b) global optimum with $f_{14} \approx 1$ at $(-32, -32)$.

Function 15 (4-D): Kowalik:

(a) $f_{15}(x) = \sum_{i=1}^{11} \{a_i - [x_1(b_i^2 + b_i x_2)][b_i^2 + b_i x_3 + x_4]^{-1}\}^2$, $-5 \le x_i \le 5$;
(b) global optimum with $f_{15} \approx 0.0003075$ at $(0.1928, 0.1908, 0.1231, 0.1358)$.

Function 16 (30-D): six-hump camel:

(a) $f_{16}(x) = 4x_1^2 - 2.1x_1^4 + (1/3)x_1^6 + x_1 x_2 - 4x_2^2 + 4x_2^4$, $-5 \le x_i \le 5$;
(b) global optimum with $f_{16} \approx -1.0316285$ at $(-0.08983, 0.71261)$ and $(0.08983, -0.71261)$.

Function 17 (30-D): Branin:

(a) $f_{17}(x) = (x_2 - (5.1/4\pi^2)x_1^2 + (5/\pi)x_1 - 6)^2 + 10(1 - 1/8\pi)\cos x_1 + 10$, $-5 \le x_1 \le 10$, $0 \le x_2 \le 15$;

Table 9: Coefficients of function f_{20}.

i	a_{ij}, $j = 1,2,3,4,5,6$						c_i	p_{ij}, $j = 1,2,3,4,5,6$					
1	10	3	17	3.5	1.7	8	1	0.1312	0.1696	0.5569	0.0124	0.8283	0.5886
2	0.05	10	17	0.1	8	14	1.2	0.2329	0.4135	0.8307	0.3736	0.1004	0.9991
3	3	3.5	1.7	10	17	8	3	0.2348	0.1415	0.3522	0.2883	0.3047	0.6650
4	17	8	0.05	10	0.1	14	3.2	0.4047	0.8828	0.8732	0.5743	0.1091	0.0381

Table 10: Coefficients of functions f_{21}, f_{22}, and f_{23}.

i	a_{ij}, $j = 1,2,3,4$				c_i
1	4	4	4	4	0.1
2	1	1	1	1	0.2
3	8	8	8	8	0.2
4	6	6	6	6	0.4
5	3	7	3	7	0.4
6	2	9	2	9	0.6
7	5	5	3	3	0.3
8	8	1	8	1	0.7
9	6	2	6	2	0.5
10	7	3.6	7	3.6	0.5

(b) global optimum with $f_{17} \approx 0.398$ at $(-3.142, 12.275)$ and $(9.425, 2.425)$.

Function 18 (2-D): Goldstein-Price:

(a) $f_{18}(x) = [1+(x_1+x_2+1)^2(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2)][30+(2x_1-3x_2)^2 \times (18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2)]$, $-2 \le x_i \le 2$;
(b) global optimum with $f_{18} = 3$ at $(0, -1)$.

Function 19 (3-D) and function 20 (6-D): Hartman's family:

(a) $f_{19}(x) = -\sum_{i=1}^{4} c_i \exp(-\sum_{j=1}^{m} a_{ij}(x_j - p_{ij})^2)$, $0 \le x_j \le 1$, with $m = 3, 6$ for $f_{19}(x)$ and $f_{20}(x)$; the coefficients are defined by Tables 8 and 9, respectively;
(b) global optimum with $f_{19} = -3.86$ at $(0.114, 0.556, 0.852)$; global optimum with $f_{20} = -3.32$ at $(0.201, 0.150, 0.477, 0.275, 0.311, 0.657)$.

Function 21 (3-D), function 22 (3-D), and function 23 (6-D): Shekel's family:

(a) $f_{21}(x) = -\sum_{i=1}^{m}[(x-a_i)(x-a_i)^T+c_i]^{-1}$, $0 \le x_j \le 10$, with $m = 5, 7$ and 10 for $f_{21}(x)$, $f_{22}(x)$, and $f_{23}(x)$, respectively;
(b) these functions have five, seven, and ten local minima for $f_{21}(x)$, $f_{22}(x)$, and $f_{23}(x)$, respectively; $x_{\text{local\_opt}} \approx a_i$, $f(x_{\text{local\_opt}}) \approx 1/c_i$ for $1 \le i \le m$; the coefficients are defined in Table 10.

Conflict of Interests

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

Acknowledgments

The authors are grateful to the anonymous referees for their careful reading of the paper and numerous suggestions for its improvement. This work was supported by the National Natural Science Foundation of China under Grants nos. 61272119, 11301414, and 11226173.

References

[1] Y. W. Leung and Y. Wang, "An orthogonal genetic algorithm with quantization for global numerical optimization," *IEEE Transactions on Evolutionary Computation*, vol. 5, no. 1, pp. 41–53, 2001.
[2] R. F. T. Neto and M. G. Filho, "An ant colony optimization approach to a permutational flowshop scheduling problem with outsourcing allowed," *Computers & Operations Research*, vol. 38, no. 9, pp. 1286–1293, 2011.
[3] R. C. Green II, L. Wang, and M. Alam, "Training neural networks using central force optimization and particle swarm optimization: insights and comparisons," *Expert Systems with Applications*, vol. 39, no. 1, pp. 555–563, 2012.
[4] S. Kirkpatrick, C. D. Gelatto, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.
[5] E. Rashedi, H. Nezamabadi-pour, and S. Saryazdi, "GSA: a Gravitational Search Algorithm," *Information Sciences*, vol. 179, no. 13, pp. 2232–2248, 2009.
[6] R. A. Formato, "Central force optimization: a new metaheuristic with applications in applied electromagnetics," *Progress in Electromagnetics Research*, vol. 77, pp. 425–491, 2007.
[7] R. A. Formato, "Central force optimization: a new nature inspired computational framework for multidimensional search and optimization," *Studies in Computational Intelligence*, vol. 129, pp. 221–238, 2008.
[8] R. A. Formato, "Central force optimization: a new deterministic gradient-like optimization metaheuristic," *Opsearch*, vol. 46, no. 1, pp. 25–51, 2009.
[9] R. A. Formato, "Improved cfo algorithm for antenna optimization," *Progress In Electromagnetics Research B*, no. 19, pp. 405–425, 2010.
[10] R. A. Formato, "Pseudorandomness in central force optimization," *British Journal of Mathematics & Computer Science*, vol. 3, no. 3, pp. 241–264, 2013.
[11] J. A. Nelder and R. Mead, "A simplex method for function minimization," *The Computer Journal*, vol. 7, no. 2, pp. 308–313, 1965.

[12] D. Ding, D. Qi, X. Luo, J. Chen, X. Wang, and P. Du, "Convergence analysis and performance of an extended central force optimization algorithm," *Applied Mathematics and Computation*, vol. 219, no. 4, pp. 2246–2259, 2012.

[13] R. Chelouah and P. Siarry, "A hybrid method combining continuous tabu search and Nelder-Mead simplex algorithms for the global optimization of multiminima functions," *European Journal of Operational Research*, vol. 161, no. 3, pp. 636–654, 2005.

[14] S. S. Fan and E. Zahara, "A hybrid simplex search and particle swarm optimization for unconstrained optimization," *European Journal of Operational Research*, vol. 181, no. 2, pp. 527–548, 2007.

[15] B. Y. Qu, J. J. Liang, and P. N. Suganthan, "Niching particle swarm optimization with local search for multi-modal optimization," *Information Sciences*, vol. 197, pp. 131–143, 2012.

[16] L. Zhang and B. Zhang, "Good point set based genetic algorithm," *Chinese Journal of Computers*, vol. 24, no. 9, pp. 917–922, 2001.

[17] C. Xiao, Z. Cai, and Y. Wang, "A good nodes set evolution strategy for constrained optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '07)*, pp. 943–950, Singapore, September 2007.

[18] X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 2, pp. 82–102, 1999.

[19] J. J. Liang, P. N. Suganthan, and K. Deb, "Novel composition test functions for numerical global optimization," in *Proceedings of the IEEE Swarm Intelligence Symposium (SIS '05)*, pp. 71–78, June 2005.

[20] A.-R. Hedar and M. Fukushima, "Hybrid simulated annealing and direct search method for nonlinear unconstrained global optimization," *Optimization Methods & Software*, vol. 17, no. 5, pp. 891–912, 2002.