Adaptive Central Force Optimization Algorithm Based on the Stability Analysis

Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2015, Article ID 914789, 10 pages
http://dx.doi.org/10.1155/2015/914789

Research Article

Adaptive Central Force Optimization Algorithm Based on the Stability Analysis

Weiyi Qian,[1] Bo Wang,[1] and Zhiguang Feng[2]

[1]*College of Mathematics and Physics, Bohai University, Jinzhou 121000, China*
[2]*College of Information Science and Technology, Bohai University, Jinzhou 121000, China*

Correspondence should be addressed to Weiyi Qian; qianweiyi2008@163.com

Received 15 October 2014; Revised 20 February 2015; Accepted 26 February 2015

Academic Editor: Farhang Daneshmand

In order to enhance the convergence capability of the central force optimization (CFO) algorithm, an adaptive central force optimization (ACFO) algorithm is presented by introducing an adaptive weight and defining an adaptive gravitational constant. The adaptive weight and gravitational constant are selected based on the stability theory of discrete time-varying dynamic systems. The convergence capability of ACFO algorithm is compared with the other improved CFO algorithm and evolutionary-based algorithm using 23 unimodal and multimodal benchmark functions. Experiments results show that ACFO substantially enhances the performance of CFO in terms of global optimality and solution accuracy.

1. Introduction

Consider the following global optimization problem:

$$\begin{aligned} &\max \quad f(x) \\ &\text{subject to} \quad x \in \Omega = \left\{ x \in R^{N_d} \mid R^{\min} \le x \le R^{\max} \right\}, \end{aligned} \tag{1}$$

where $f(x) : \Omega \subset R^{N_d} \to R$ is a real-valued bounded function and $R^{\min}$, $R^{\max}$, and x are N_d-dimensional continuous variable vectors. Such problem arises in many applications, for example, in risk management, applied sciences, and engineering design. The function of interest may be nonlinear and nonsmooth which makes the classical optimization algorithms easily fail to solve these problems. Over the last decades, many nature-inspired heuristic optimization algorithms without requiring much information about the function became the most widely used optimization methods such as genetic algorithms (GA) [1], particle swarm optimization (PSO) [2], ant colony optimization (ACO) [3], cuckoo search (CS) algorithm [4], group search optimizer (GSO) [5], and glowworm swarm optimization (GSO1) [6]. These search methods all simulate biological phenomena. Different from these algorithms, some heuristic optimization algorithms based on physical principles have been developed, for example, simulating annealing (SA) algorithm [7], electromagnetism-like mechanism (EM) algorithm [8], central force optimization (CFO) algorithm [9], gravitational search algorithm (GSA) [10], and charged system search (CSS) [11]. SA simulates solid material in the annealing process. EM is based on Coulomb's force law associated with electrical charge process. GSA and CFO utilize Newtonian mechanics law. CSS is based on Coulomb's force and Newtonian mechanics laws. Unlike other algorithms, CFO is a deterministic method. In other words, there is not any random nature in CFO, which attracts our attention on the CFO algorithm in this paper.

CFO, which was introduced by Formato in 2007 [9], is becoming a novel deterministic heuristic optimization algorithm based on gravitational kinematics. In order to improve the CFO algorithm, Formato and other researchers developed many versions of the CFO algorithm [12–23]. In [12, 13], Formato proposed PR-CFO (Pseudo-Random CFO) algorithm. The improved implementations are made in three areas: initial probe distribution, repositioning factor, and decision space adaptation. Formato presented an algorithm known as PF-CFO (Parameter Free CFO) in [14, 15]. PF-CFO algorithm improves and perfects the PR-CFO algorithm in the aspect of the selection of parameter. Mahmoud proposed an efficient global hybrid optimization algorithm combining the CFO algorithm and the Nelder-Mead (NM) method in

[16]. This hybrid method is called CFO-NM. An extended CFO (ECFO) algorithm was presented by Ding et al. by adding the historical information and defining an adaptive mass in [17] where the convergence of ECFO algorithm was proved based on the second order difference equation.

In aforementioned CFO algorithms, two updated equations were used: one for a probe's acceleration and the other for its position. In the probe's position updated equation which is established based on the laws of motion, the velocity is defined as zero. But the velocity has influence on the exploring ability of the CFO algorithm. Therefore, in this paper, we introduce the velocity in the probe's position updated equation, which leads us to build a velocity updated equation like the CSS algorithm. Since the weight which can balance the global and local search ability is an important parameter in many heuristic algorithms, we introduce weight in probe's position updated equation. If the value of weight is too large, then the probes may move erratically, going beyond a good solution. On the other hand, if weight is too small, then the probe's movement is limited and the optimal solution may not be reached. Therefore, an appropriate dynamically changing weight can improve the performance of the heuristic algorithm. However, in most of the heuristic algorithms, the changing weight was selected empirically according to the characteristics of the problems without theoretical analysis. The gravitational constant has the same effect on the weight in CFO algorithm. Hence, this paper will further investigate the weight and gravitational constant settings by employing the geometry-velocity stability theory of discrete time-varying dynamic systems. Based on the above discussion, an adaptive CFO (ACFO) algorithm is proposed in this paper.

To the best of our knowledge, there is no research on stability analysis of the CFO algorithm till now. In this paper, the stability of the ACFO algorithm is analyzed based on discrete time-varying dynamic systems theory. Based on the stability analysis of the proposed algorithm, we explore the weight and gravitational constant settings.

The rest of this paper is organized as follows. Section 2 will present the basics of the CFO algorithm and review the state of the art concerning the algorithm. In Section 3, we propose an adaptive central force optimization. Some numerical results are given to test the performance of the proposed algorithm in Section 4. Finally, we have some conclusions about the proposed algorithm.

2. Central Force Optimization

CFO solves problem (1) based on the movement of probes through the decision space (DS) along trajectories computation by utilizing the gravitational analogy. The DS is defined by $\Omega = \{(x_1, x_2, \ldots, x_{N_d}) \mid R_i^{\min} \le x_i \le R_i^{\max},\ 1 \le i \le N_d\}$. In CFO, a group of probes are represented as potential solutions, and each probe p is associated with position vector $R^p(t)$ and acceleration vector $A^p(t)$ at time step t. The position of each probe is initialized by a variable initial probe distribution formed by deploying N_p/N_d probes uniformly on each of the probe lines parallel to the coordinate axes and intersecting at a point along Ω's principal diagonal, where N_p is the total number of initial probes. The initial acceleration vectors are usually set to zero. During search process, the acceleration and position of probe p are updated as

$$A_i^p(t) = G \sum_{\substack{k=1 \\ k \ne p}}^{N_p} U\left(M^k(t) - M^p(t)\right)\left(M^k(t) - M^p(t)\right)^{\alpha} \cdot \frac{\left(R_i^k(t) - R_i^p(t)\right)}{\left\|R^k(t) - R^p(t)\right\|^{\beta}}, \tag{2}$$

$$R_i^p(t+1) = R_i^p(t) + \frac{1}{2} A_i^p(t)\, \Delta t^2, \tag{3}$$

where G is the gravitational constant; $M^j(t) = f(R^j(t))$ is the fitness value at probe j's ($j = 1, 2, \ldots, N_p$) position at time step t; α and β are the parameters; i ($i = 1, 2, \ldots, N_d$) is the coordinate number; $U(\cdot)$ is the unit step function;

$$U(z) = \begin{cases} 1, & z \ge 0 \\ 0, & \text{else;} \end{cases} \tag{4}$$

Δt is the unit time step increment; define $\|R^k(t) - R^p(t)\| = \sqrt{\sum_{i=1}^{N_d} [R_i^k(t) - R_i^p(t)]^2}$. The probe generated by (3) may be beyond the DS. If the coordinate $R_i^p(t+1)$ of the probe $R^p(t+1)$ is less than $R_i^{\min}$, then it is assigned to be

$$R_i^p(t+1) = R_i^{\min} + F_{\text{rep}} \left[R_i^p(t-1) - R_i^{\min}\right]. \tag{5}$$

If $R_i^p(t+1)$ is greater than $R_i^{\max}$, then

$$R_i^p(t+1) = R_i^{\max} - F_{\text{rep}} \left[R_i^{\min} - R_i^p(t-1)\right], \tag{6}$$

where $R_i^{\min}$ and $R_i^{\max}$ are the minimum and maximum values for ith component of the decision variable. F_{rep} is the reposition factor which starts at an arbitrary initial value $F_{\text{rep}}^{\text{initial}} < 1$ and is incremented by an arbitrary amount ΔF_{rep} at each iteration. If $F_{\text{rep}} \ge 1$, then it is reset to the starting value. In order to improve convergence speed, the DS size is adaptively reduced around the best probe R_{best}. The DS's boundary coordinates are reduced as follows:

$$\overline{R}_i^{\min} = R_i^{\min} + \frac{1}{2}\left[R_{\text{best}} - R_i^{\min}\right], \qquad \overline{R}_i^{\max} = R_i^{\max} - \frac{1}{2}\left[R_i^{\max} - R_{\text{best}}\right]. \tag{7}$$

The termination criterion is that iterations reach their maximum limit N_t. We also terminate the CFO algorithm early if the difference between the average best fitness over q steps (including the current step) and the current best fitness is less than 10^{-6}.

In order to improve the CFO algorithm, Formato proposed modifications to CFO algorithm, namely, PR-CFO [12, 13]. The steps of PR-CFO algorithm [13] are shown as follows;

For $N_p/N_d = (N_p/N_d)_{\text{start}}$ to $(N_p/N_d)_{\max}$ step size is 2.

For $\gamma = \gamma_{\text{start}}$ to γ_{stop} by $\Delta\gamma$

(a.1) compute initial probe distribution with distribution factor γ;

(a.2) compute initial fitness matrix; select the best probe fitness;

(a.3) assign initial probe acceleration;

(a.4) set initial $F_{\text{rep}} = F_{\text{rep}}^{\text{init}}$.

For $t = 0$ to N_t (or earlier termination criterion)

(b) update probe positions using (3);

(c) retrieve errant probe using (5) and (6);

(d) calculate fitness values; select the best probe fitness;

(e) compute accelerations using (2);

(f) increment F_{rep} by ΔF_{best};

if $F_{\text{rep}} > 1$ then $F_{\text{rep}} = F_{\text{rep}}^{\min}$;

End If

(g) if t MOD 10 = 0, then

shrink Ω around best probe using (7);

End If

Next t

(h) reset Ω's boundaries to their starting values before shrinking.

Next γ

Next N_p/N_d.

PR-CFO is further modified in order to create an algorithm known as PF-CFO (Parameter Free CFO) [14, 15]. This version is almost identical to PR-CFO and compensates for the number of parameters that must be chosen by fixing a wide array of internal parameters at specific values [19]. The values of parameters borrowed from [19] that are used in PF-CFO algorithm can be seen in Table 1.

Table 1: The values of parameters that are used in PF-CFO algorithm.

Parameter	N_d	Value
N_t		1000
$F_{\text{rep}}^{\text{init}}$		0.5
ΔF_{rep}		0.1
F_{rep}^{min}		0.05
G		2
α		1
β		2
$(N_p/N_d)_{max}$	$N_d \le 6$	14
	$7 \le N_d \le 10$	12
	$11 \le N_d \le 15$	10
	$16 \le N_d \le 20$	8
	$21 \le N_d \le 30$	6
	$N_d > 30$	4

3. Adaptive Central Force Optimization Algorithm

Qian and Zhang [23] proposed an adaptive central force optimization algorithm. In [23], the time Δt in (3) was updated based on the fitness value compared with the average fitness value. In this paper, we introduce adaptive weight in position update equation, adaptive gravitational constant in acceleration update equation and the velocity update formula. The weight and gravitational constant are updated based on this stability analysis of a discrete time-varying dynamic system. In ACFO algorithm, the position, acceleration, and velocity of probe p are updated as follows:

$$R_i^p(t+1) = R_i^p(t) + \omega^p(t) V_i^p(t) \Delta t + \frac{1}{2} A_i^p(t) \Delta t^2, \quad (8)$$

$$A_i^p(t+1) = G^p(t+1) \sum_{\substack{k=1 \\ k \neq p}}^{N_p} U\left(M^k(t+1) - M^p(t+1)\right) \cdot \left(M^k(t+1) - M^p(t+1)\right)^{\alpha} \cdot \frac{\left(R_i^k(t+1) - R_i^p(t+1)\right)}{\left(D_{pk}\right)^{\beta}}, \quad (9)$$

$$V_i^p(t+1) = \frac{R_i^p(t+1) - R_i^p(t)}{\Delta t}, \quad (10)$$

where $\omega^p(t)$ is the weight; $G^p(t)$ is a gravitational constant at probe p's position at iteration t; $\Delta t = 1$; α and β are the parameters; i $(i = 1, 2, \ldots, N_d)$ is the coordinate number; and D_{pk} is defined as follows:

$$D_{pk} = \begin{cases} d_{pk}, & \text{if } d_{pk} \ge a \\ a, & \text{if } d_{pk} < a, \end{cases} \quad (11)$$

where d_{pk} is the Euclidian distance between two probes, p and k, and a is a radius constant.

Let

$$\varphi_k(t) = \frac{U\left(M^k(t) - M^p(t)\right)\left(M^k(t) - M^p(t)\right)^{\alpha}}{\left(D_{pk}\right)^{\beta}}, \qquad \varphi(t) = \sum_{\substack{k=1 \\ k \neq p}}^{N_p} \varphi_k(t). \quad (12)$$

It is clear that $\varphi_k(t)$ and $\varphi(t)$ are nonnegative. By (8), (9), and (10), the position and velocity updated equations can be written as follows:

$$\begin{aligned} R_i^p(t+1) &= \left(1-\frac{1}{2}G^p(t)\varphi(t)\right)R_i^p(t)+\omega^p(t)V_i^p(t) \\ &\quad+\frac{1}{2}G^p(t)\sum_{\substack{k=1\\k\neq p}}^{N_p}\varphi_k(t)R_i^k(t), \\ V_i^p(t+1) &= -\frac{1}{2}G^p(t)\varphi(t)R_i^p(t)+\omega^p(t)V_i^p(t) \\ &\quad+\frac{1}{2}G^p(t)\sum_{\substack{k=1\\k\neq p}}^{N_p}\varphi_k(t)R_i^k(t). \end{aligned} \tag{13}$$

Equations (13) are written in matrix form as follows:

$$\begin{aligned} \begin{pmatrix} R_i^p(t+1) \\ V_i^p(t+1) \end{pmatrix} &= \begin{pmatrix} 1-\frac{1}{2}G^p(t)\varphi(t) & \omega^p(t) \\ -\frac{1}{2}G^p(t)\varphi(t) & \omega^p(t) \end{pmatrix}\begin{pmatrix} R_i^p(t) \\ V_i^p(t) \end{pmatrix} \\ &\quad+\frac{1}{2}G^p(t)\begin{pmatrix} \sum_{\substack{k=1\\k\neq p}}^{N_p}\varphi_k(t)R_i^k(t) \\ \sum_{\substack{k=1\\k\neq p}}^{N_p}\varphi_k(t)R_i^k(t) \end{pmatrix}. \end{aligned} \tag{14}$$

Let

$$\begin{aligned} x(t+1) &= \begin{pmatrix} R_i^p(t+1) \\ V_i^p(t+1) \end{pmatrix}, \\ g(t,x(t)) &= \begin{pmatrix} 1-\frac{1}{2}G^p(t)\varphi(t) & \omega^p(t) \\ -\frac{1}{2}G^p(t)\varphi(t) & \omega^p(t) \end{pmatrix}\begin{pmatrix} R_i^p(t) \\ V_i^p(t) \end{pmatrix} \\ &\quad+\frac{1}{2}G^p(t)\begin{pmatrix} \sum_{\substack{k=1\\k\neq p}}^{N_p}\varphi_k(t)R_i^k(t) \\ \sum_{\substack{k=1\\k\neq p}}^{N_p}\varphi_k(t)R_i^k(t) \end{pmatrix}. \end{aligned} \tag{15}$$

Equation (14) can be expressed as a discrete time-varying dynamic system as follows:

$$x(t+1) = g(t,x(t)), \quad t=1,2,\ldots. \tag{16}$$

Lemma 1 (see [24]). *Let $x(t+1)=g(t,x(t))$ be a discrete time-varying dynamic system; if $g(t,x(t))$ satisfies the condition*

$$\left\|g(t,x(t))\right\|_v \le \nu(t)\left\|x(t)\right\|_v + c, \tag{17}$$

under a certain vector norm $\|\cdot\|_v$, then the system is geometry-velocity stable in the bounded set $L=\{x \mid \|x\|_v < c/(1-\vartheta)\}$, where c is constant and $0 \le \nu(t) \le \vartheta < 1$.

Cui and Zen presented a selection of the parameters in PSO algorithm based on Lemma 1 in [24]. Now we analyze the stability of ACFO algorithm and give a selection of weight and gravitational constant based on Lemma 1.

If $\|\cdot\|_v$ in Lemma 1 is considered as an infinite norm, then we have

$$\begin{aligned} \left\|g(t,x(t))\right\|_\infty &\le \left\|\begin{pmatrix} 1-\frac{1}{2}G^p(t)\varphi(t) & \omega^p(t) \\ -\frac{1}{2}G^p(t)\varphi(t) & \omega^p(t) \end{pmatrix}\right\|_\infty \left\|x(t)\right\|_\infty \\ &\quad+\frac{1}{2}\left|G^p(t)\right|\left\|\begin{pmatrix} \sum_{\substack{k=1\\k\neq p}}^{N_p}\varphi_k(t)R_i^k(t) \\ \sum_{\substack{k=1\\k\neq p}}^{N_p}\varphi_k(t)R_i^k(t) \end{pmatrix}\right\|_\infty. \end{aligned} \tag{18}$$

Since $G^p(t)$ is a nonnegative finite number for any t and $f(\cdot)$ is a bounded function, we assume that $G^p(t) \le G$ and $\|f(\cdot)\| \le M = \max_{x\in\Omega}|f(x)|$. Thus, we can obtain

$$0 \le U\left(M^k(t)-M^p(t)\right)\left(M^k(t)-M^p(t)\right)^\alpha \le M^\alpha. \tag{19}$$

By (11), one has $D_{pk} \ge a$. Therefore, we have $\varphi_k(t) \le M^\alpha/a^\beta$. Thus,

$$\begin{aligned} \frac{1}{2}\left|G^p(t)\right|\left\|\begin{pmatrix} \sum_{\substack{k=1\\k\neq p}}^{N_p}\varphi_k(t)R_i^k(t) \\ \sum_{\substack{k=1\\k\neq p}}^{N_p}\varphi_k(t)R_i^k(t) \end{pmatrix}\right\|_\infty &\le \frac{1}{2}G\left|\sum_{\substack{k=1\\k\neq p}}^{N_p}\varphi_k(t)R_i^k(t)\right| \\ &\le \frac{1}{2}G\sum_{\substack{k=1\\k\neq p}}^{N_p}\varphi_k(t)\left|R_i^k(t)\right| \\ &\le \frac{\left(N_p-1\right)M^\alpha}{2a^\beta} \\ &\quad\cdot\max_{1\le i\le N_d}\left\{\max\left\{R_i^{\max}, -R_i^{\min}\right\}\right\}. \end{aligned} \tag{20}$$

In addition,

$$\begin{aligned} &\left\|\begin{pmatrix} 1-\frac{1}{2}G^p(t)\varphi(t) & \omega^p(t) \\ -\frac{1}{2}G^p(t)\varphi(t) & \omega^p(t) \end{pmatrix}\right\|_\infty \\ &\le \max\left\{\left|1-\frac{1}{2}G^p(t)\varphi(t)\right|+\omega^p(t), \left|-\frac{1}{2}G^p(t)\varphi(t)\right|+\omega^p(t)\right\}. \end{aligned} \tag{21}$$

Table 2: Comparison of ACFO with RP-CFO and RF-CFO.

F	N_d	F_{min}	PR-CFO		PF-CFO		ACFO	
			Best fitness	N_{eval}	Best fitness	N_{eva}	Best fitness	N_{eva}
Unimodal functions								
f_1	30	0	$-4.8438E-8$	507060	0*	222960	0*	159600
f_2	30	0	$-4E-8$	716400	0*	237540	0*	200340
f_3	30	0	$-6E-8$	1534260	−6.1861E − 5	397320	0*	187200
f_4	30	0	$-4.2E-7$	332340	0*	484260	0*	133200
f_5	30	0	$-1.09289E-3$	845640	−4.8623E − 5	436680	0*	156000
f_6	30	0	0*	350280	0*	176580	0*	216000*
f_7	30	0	$-4.29E-5$*	1983960	−1.2919E − 4	399960	−1.4232E − 4	220620
Multimodal functions, many local minima								
f_8	30	−12569.5	12569.487*	448800	12569.4865	415500	12569.4785	174000
f_9	30	0	$-2.05E-6$	680640	0*	397080	0*	145200
f_{10}	30	0	$-1.5E-7$	904980	−4.9905E − 18	518820	−8.8818E − 21*	111600
f_{11}	30	0	$-9.97293E-2$	489060	−1.7075E − 2	235800	0*	191040
f_{12}	30	0	$-2.067E-5$*	341400	−2.1541E − 5	292080	−9.87E − 2	192540
f_{13}	30	0	$-3.2853E-3$	679620	−1.8293E − 3*	360000	−9.5E − 3	174180
Multimodal functions, few local minima								
f_{14}	2	1	−0.9980*	141076	−0.9980*	78176	−0.9980*	37120
f_{15}	4	$-3.075E-5$	$-4.889E-4$	304664	−5.6967E − 4	143152	−3.886E − 4*	98400
f_{16}	2	−1.0316285	1.031626*	124340	1.03158	87240	1.0303	52640
f_{17}	2	0.398	−0.3979	108340	−0.3979	82096	−0.3980*	94520*
f_{18}	2	3	−3*	180472	−3*	100996	−3*	91320
f_{19}	3	−3.86	3.8627	200268	3.8628	160338	3.8627	49680
f_{20}	6	−3.32	3.32173	730212	3.3219*	457836	3.3218	87120
f_{21}	4	−10	10.1532*	336712	10.1532*	251648	10.1532*	128000
f_{22}	4	−10	10.4029*	386176	10.4029*	316096	10.4029*	143508
f_{23}	4	−10	10.5363	394320	10.5364*	304312	10.5363	141848

Set $c = ((N_p - 1)M^{\alpha}/2a^{\beta})\max_{1\le i\le N_d}\{\max\{R_i^{\max}, -R_i^{\min}\}\}$. If

$$\max\left\{\left|1-\frac{1}{2}G^p(t)\varphi(t)\right|+\omega^p(t), \left|-\frac{1}{2}G^p(t)\varphi(t)\right|+\omega^p(t)\right\} \le \theta < 1, \tag{22}$$

then by Lemma 1, system (16) is geometry-velocity stable in the bounded set $L = \{x \mid \|x\|_{\infty} \le c/(1-\theta)\}$.

By (22), one has $0 < (1/2)G^p(t)\varphi(t) < 1$. If $0 < (1/2)G^p(t)\varphi(t) < 0.5$, that is, $0 < G^p(t) < 1/\varphi(t)$, then $(1/2)G^p(t)\varphi(t) + \omega^p(t) < 1 - (1/2)G^p(t)\varphi(t) + \omega^p(t) \le \theta$; that is, $0 < \omega^p(t) \le (1/2)G^p(t)\varphi(t) + \theta - 1$. If $0.5 \le (1/2)G^p(t)\varphi(t) < 1$, that is, $1/\varphi(t) \le G^p(t) < 2/\varphi(t)$, then $1 - (1/2)G^p(t)\varphi(t) + \omega^p(t) \le (1/2)G^p(t)\varphi(t) + \omega^p(t) \le \theta$; that is, $0 < \omega^p(t) \le \theta - (1/2)G^p(t)\varphi(t)$.

From the above discussion, in order to guarantee the geometry-velocity stability of system (16), parameters $\omega^p(t)$ and $G^p(t)$ are selected as follows:

$$G^p(t) = \frac{2}{\varphi(t)}\text{rand}(0,1),$$
$$\omega^p(t) = \begin{cases} \left(\frac{1}{2}G^p(t)\varphi(t) + \theta - 1\right)\text{rand}(0,1), \\ \quad \text{if } 0 < G^p(t) < \frac{1}{\varphi(t)}, \\ \left(\theta - \frac{1}{2}G^p(t)\varphi(t)\right)\text{rand}(0,1), \\ \quad \text{if } \frac{1}{\varphi(t)} \le G^p(t) < \frac{2}{\varphi(t)}, \end{cases} \tag{23}$$

where rand$(0,1)$ is a random number in the interval $[0,1]$. However, CFO algorithm is a deterministic method which

should not contain any random nature. Therefore, we take parameters $\omega^p(t)$ and $G^p(t)$ as follows:

$$G^p(t) = \min\left\{2, \mu\frac{2}{\varphi(t)}\right\}, \tag{24}$$

$$\omega^p(t) = \begin{cases} \eta\left(\frac{1}{2}G^p(t)\varphi(t) - 0.1\right), \\ \quad \text{if } 0 < G^p(t) < \frac{1}{\varphi(t)}, \\ \eta\left(0.9 - \frac{1}{2}G^p(t)\varphi(t)\right), \\ \quad \text{if } \frac{1}{\varphi(t)} \le G^p(t) < \frac{2}{\varphi(t)}, \end{cases} \tag{25}$$

where μ and η are two constants between 0 and 1.

The specific iterative steps of ACFO algorithm are listed as follows.

For $N_p/N_d = 2$ to $(N_p/N_d)_{\max}$ step size is 2.

For $\gamma = \gamma_{\text{start}}$ to γ_{stop} by $\Delta\gamma$

(a.1) compute initial probe distribution with distribution factor γ;

(a.2) compute initial fitness matrix; select the best probe fitness;

(a.3) assign initial probe's accelerations and velocities;

(a.4) set initial $F_{\text{rep}} = F_{\text{rep}}^{\text{init}}$ and G^{init}.

For $t = 0$ to N_t (or earlier termination criterion)

(b) compute weights using (25);

(c) update probe positions using (8);

(d) retrieve errant probe using (5) and (6);

(e) calculate fitness values; select the best probe fitness;

(f) update gravitational constant using (24);

(g) compute accelerations using (9) and velocities using (10);

(h) increment F_{rep} by ΔF_{best};

if $F_{\text{rep}} > 1$ then $F_{\text{rep}} = F_{\text{rep}}^{\min}$;

End If

(i) if t MOD 10 = 0, then

shrink Ω around best probe using (7);

End If

Next t

(l) reset Ω's boundaries to their starting values before shrinking.

Next γ

Next N_p/N_d.

In ACFO algorithm, the initial acceleration and velocities vectors are set to zero and $\vec{e} = \sum_{k=1}^{N_d} 0.01\vec{e}_k$, where $\vec{e}_k$ is the unit vector along the k-axis.

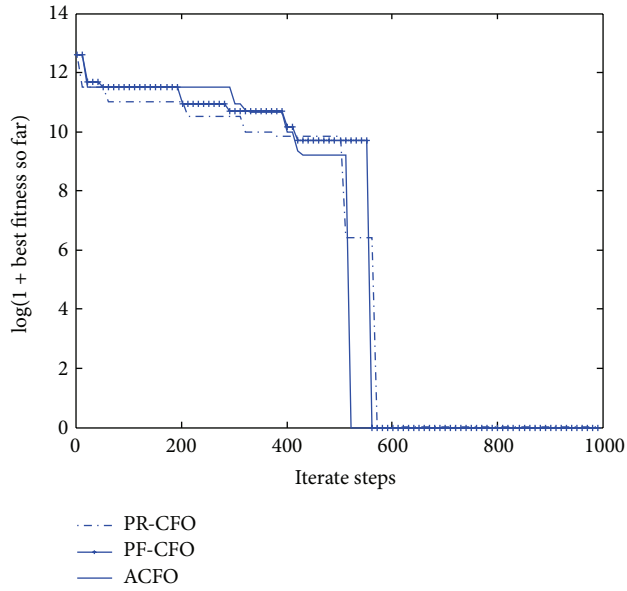

Figure 1: Convergence curves of three algorithms on f_1.

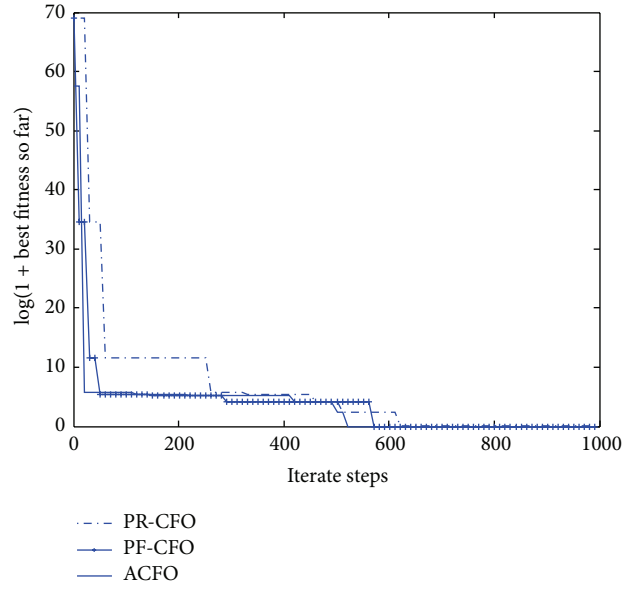

Figure 2: Convergence curves of three algorithms on f_2.

4. Numerical Experiments

In this section, the performance of ACFO algorithm is compared with the existing algorithms, GSO, GA, PSO, PR-CFO, PF-CFO, CFO-NM, and EPSO, using a suite of the former twenty-three benchmark functions provided in [25]. In ACFO algorithm, internal parameters $\gamma_{\text{start}} = 0$ and $\Delta\gamma = 0.1$. Other internal parameters are the same as the parameters of RF-CFO algorithm except parameter $N_t = 500$, which are listed in Table 1. We choose other parameters $\mu = 0.9$, $\eta = 1$, and $a = 0.01$. In our experiment, the codes were written in MATLAB 7.0 and run on PC with 2.00 GB RAM memory, 2.10 GHz CPU, and Windows 7 operation

Table 3: Comparison of ACFO with GA, PSO, GSO, CFO-NM, and ECFO.

Test function	GA	PSO	GSO	CFO-NM	ECFO	ACFO
			Unimodal functions			
f_1	3.711	$3.6927E-37$	$1.9481E-8$	$8.86597E-25$	$7.5137E-7$	0^*
f_2	0.5771	$2.9168E-24$	$3.7039E-5$	$9.46909E-5$	$4.8954E-7$	0^*
f_3	9749.9145	$1.1979E-3$	5.7829	$5.89019E-9$	$9.7713E-7$	0^*
f_4	7.9610	0.4123	$10.7E-2$	$6.40640E-3$	$9.7203E-7$	0^*
f_5	338.5616	37.3582	49.8359	$1.12593E-7$	$4.3433E-5$	0^*
f_6	3.69701	0.1460	$1.600E-2$	—	$2.2016E-7$	0^*
f_7	0.1045	$9.9024E-3$	$7.3773E-2$	—	$2.525E-5^*$	$1.4850E-4$
			Multimodal functions, many local maxima			
f_8	−12566.0977	−9659.6993	-12569.4882^*	−12569.486618	−837.9657	−12569.4785
f_9	$6.509E-1$	20.7863	1.0179	$6.89347E-23$	$7.5095E-5$	0^*
f_{10}	0.8678	$1.3404E-3$	$2.6548E-5$	$1.49081E-11$	$6.8426E-5$	$8.8818E-21^*$
f_{11}	1.0038	0.2323	$3.0792E-2$	$4.99600E-15$	$4.6279E-5$	0^*
f_{12}	$4.3572E-2$	$3.9503E-2$	$2.7648E-11$	$1.93296E-21^*$	$1.6471E-5$	$9.87E-2$
f_{13}	0.1681	$5.0519E-2$	$4.6948E-5^*$	—	$6.1817E-5$	$9.5E-3$
			Multimodal functions, few local maxima			
f_{14}	0.9989	1.0239	0.9980^*	—	—	0.9980^*
f_{15}	$7.0878E-3$	$3.8074E-4$	$3.7713E-4^*$	—	—	$3.886E-4$
f_{16}	−1.0298	−1.0160	−1.031628	-1.03163^*	—	−1.0303
f_{17}	0.4040	0.4040	0.3979	0.39700^*	—	0.3980
f_{18}	7.5027	3.0050	3^*	3.00000^*		3^*
f_{19}	−3.8624	−3.8582	−3.8628	—	—	−3.8627
f_{20}	−3.2627	−3.1846	−3.2697	—	—	-3.3218^*
f_{21}	−5.1653	−7.5439	−6.09	—	—	-10.1532^*
f_{22}	−5.4432	−8.3553	−6.5546	—	—	-10.4029^*
f_{23}	−4.9108	−8.9439	−7.4022	—	—	-10.5362^*

system. The stopping condition is that iterations reach their maximum limit N_t. We also early stop the ACFO algorithm if the difference between the average best fitness over 30 steps (including the current step) and the current best fitness is less than 10^{-6}.

In Table 2, F, N_d, $f_{\min}$, and N_{eval} stand for the test function, the dimension of decision space, the optimum minimum value for each function, and the total number of function evaluations, respectively. The statistical data in Table 2 for RP-CFO and RF-CFO are reproduced from [13] and [15], respectively. The best fitness is the optimum maximum (note the negative of each of the benchmark functions). The set of twenty-three benchmark functions are divided into unimodal functions (f_1 to f_7), multimodal functions (f_8 to f_{13}), and low-dimensional multimodal functions (f_{14} to f_{23}). Table 3 summarizes the obtained optimum minimum results which are compared with other optimization algorithms, such as GA, PSO, GSO, CFO-NM, and ECFO. The statistical data for CFO-NM and ECFO is from [16, 17] while the other statistical data is from [5]. In Tables 2 and 3, the star $*$ denotes that the numerical result is the best one among all the comparative algorithms. In Table 3, the symbol "—" means that the problem is not calculated in the original references.

From Table 2, it is clearly seen that the ACFO algorithm yields significantly better performance than PR-CFO algorithm on benchmark functions f_1–f_5. But the ACFO algorithm has a worse result on f_7 and same result on f_6 compared to PR-CFO algorithm. From the comparisons between ACFO algorithm and PF-CFO algorithm, we can find that ACFO algorithm performs better than PF-CFO algorithm on f_3 and f_5 and obtains the same results yielded by PF-CFO algorithm on f_1, f_2, f_4, f_6, and f_7. But it should be noted that both the ACFO and PF-CFO algorithm can obtain optimum minimum value of f_1, f_2, f_4, and f_6.

The set of benchmark functions f_8–f_{13} are multimodal functions with many local minima. From Table 2, we can see that the ACFO algorithm outperforms PR-CFO and PF-CFO algorithm except functions f_8, f_{12}, and f_{13}, but PR-CFO and PF-CFO algorithm are superior to ACFO algorithm on benchmark function f_8 by a very small percentage of $6.683E-07$ and $6.285E-07$, respectively.

The other set of benchmark functions f_{14}–f_{23} are low dimensions multimodal functions. From the comparison shown in Table 2, it can be obviously seen that the best fitness generated by ACFO, PR-CFO, and PF-CFO algorithm are almost the same on f_{14}–f_{23}.

From Table 2, we can also see that ACFO algorithm is superior to the PR-CFO and PF-CFO algorithms for the total number of function evaluations except functions f_6 and f_{17}.

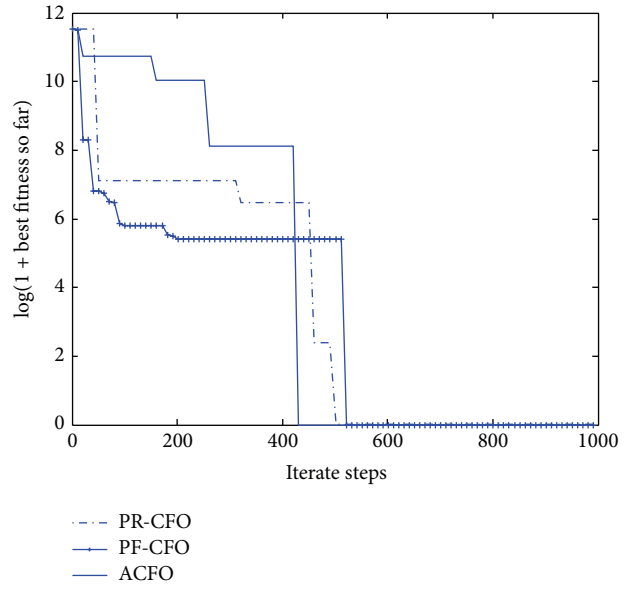

Figure 3: Convergence curves of three algorithms on f_3.

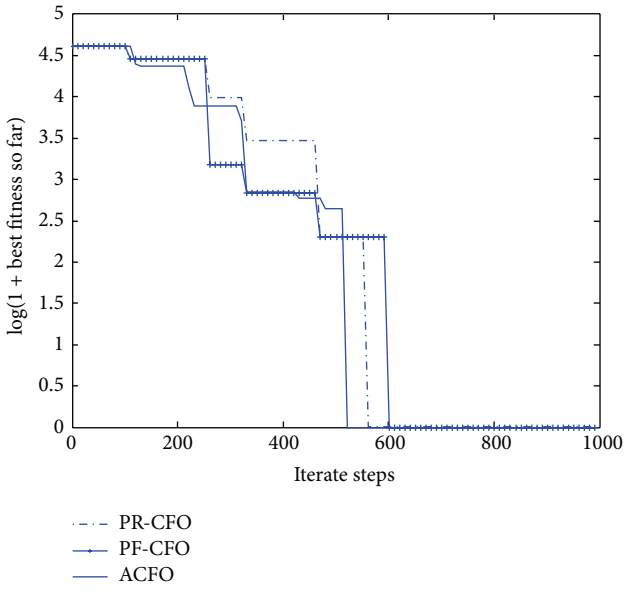

Figure 4: Convergence curves of three algorithms on f_4.

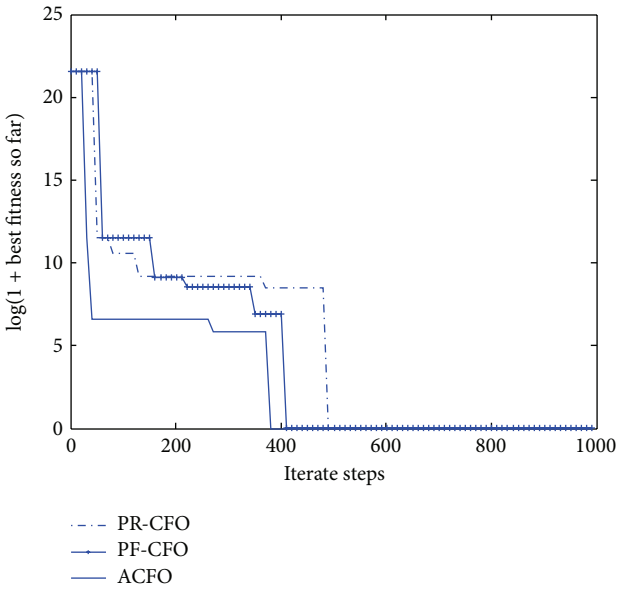

Figure 5: Convergence curves of three algorithms on f_5.

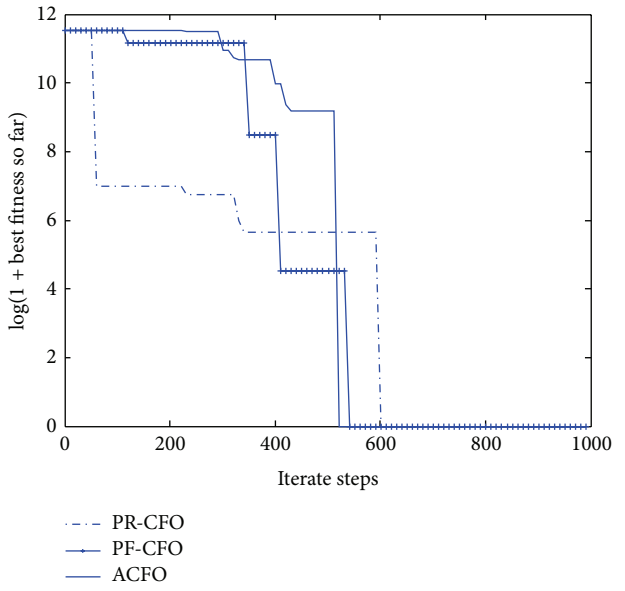

Figure 6: Convergence curves of three algorithms on f_6.

In Table 3, we can clearly see that the ACFO algorithm outperforms GA, PSO, GSO, CFO-MN, and ECFO algorithms on test functions f_1–f_7. The only exception is function f_7 in which the ECFO algorithm is superior to ACFO algorithm. It is seen that, for test functions f_8–f_{13}, ACFO algorithm performs better than GA and PSO algorithms except function f_{12}. In addition, ACFO algorithm outperforms GSO, CFO-NM, and ECFO on functions f_9–f_{11}. For functions f_{14}–f_{23}, we can also find that the ACFO algorithm generates better results than the GA and PSO. The only exception is function f_{15} in which the ACFO algorithm yields a similar result compared to PSO. From the comparisons between ACFO and GSO algorithm, we can see that the ACFO algorithm outperforms GSO algorithm on functions f_{20}–f_{23} and has similar results to GSO algorithm on functions f_{14}–f_{19}. In addition, ACFO algorithm has also similar result to CFO-NM algorithm on functions f_{16}–f_{18}. From the comparisons between ACFO and other algorithms, it is found that the ACFO algorithm performs better than the other algorithms.

Figures 1–7 show only convergence curves of PR-CFO, PF-CFO, and ACFO algorithms on f_1–f_7. The vertical axis is the logarithmic function value of (1 + best function value), and horizontal axis is the number of iterates. From Figures 1–6, we can obviously see that ACFO algorithm tends to find the global optimum faster than other algorithms and

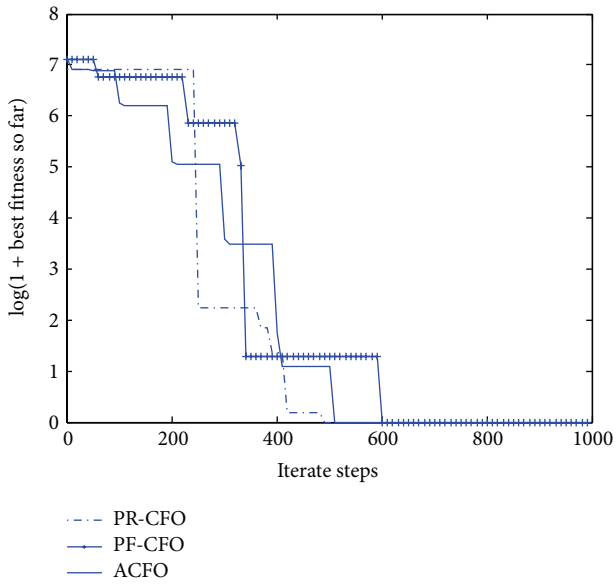

FIGURE 7: Convergence curves of three algorithms on f_7.

hence has a higher convergence rate. According to Figure 7, ACFO and PR-CFO algorithms have similar convergence rates, but ACFO algorithm has good convergence rate compared with PF-CFO algorithm.

5. Conclusion

This paper presents ACFO algorithm which enhances the convergence capability of the CFO algorithm. The ACFO algorithm introduces a weight and updates the equation that generates the probe's position. Based on the stability theory of discrete time-varying dynamic systems, we define adaptive weight and gravitational constant. In order to test ACFO algorithm performance, ACFO algorithm is compared with improved CFO algorithms and other state-of-the-art algorithms. The simulation results show that the ACFO is better than other algorithms.

Conflict of Interests

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

Acknowledgments

The authors would like to thank the editor and anonymous referees for their valuable comments and suggestions which improved this paper greatly. This work is partly supported by the National Natural Science Foundation of China (11371071) and Scientific Research Foundation of Liaoning Province Educational Department (L2013426).

References

[1] J. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, Mich, USA, 1975.

[2] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, vol. 4, pp. 1942–1948, Perth, Australia, December 1995.

[3] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *IEEE Computational Intelligence Magazine*, vol. 1, no. 4, pp. 28–39, 2006.

[4] X.-S. Yang and S. Deb, "Engineering optimisation by cuckoo search," *International Journal of Mathematical Modelling and Numerical Optimisation*, vol. 1, no. 4, pp. 330–343, 2010.

[5] S. He, Q. H. Wu, and J. R. Saunders, "Group search optimizer: an optimization algorithm inspired by animal searching behavior," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 973–990, 2009.

[6] K. N. Krishnanand and D. Ghose, "Glowworm swarm optimisation: a new method for optimising multi-modal functions," *International Journal of Computational Intelligence Studies*, vol. 1, no. 1, pp. 93–119, 2009.

[7] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of state calculations by fast computing machines," *The Journal of Chemical Physics*, vol. 21, no. 6, pp. 1087–1092, 1953.

[8] S. I. Birbil and S.-C. Fang, "An electromagnetism-like mechanism for global optimization," *Journal of Global Optimization*, vol. 25, no. 3, pp. 263–282, 2003.

[9] R. A. Formato, "Central force optimization: a new metaheuristic with applications in applied electromagnetics," *Progress in Electromagnetics Research*, vol. 77, pp. 425–491, 2007.

[10] E. Rashedi, H. Nezamabadi-Pour, and S. Saryazdi, "GSA: a gravitational search algorithm," *Information Sciences*, vol. 179, no. 13, pp. 2232–2248, 2009.

[11] A. Kaveh and S. Talatahari, "A novel heuristic optimization method: charged system search," *Acta Mechanica*, vol. 213, no. 3-4, pp. 267–289, 2010.

[12] R. A. Formato, "Pseudorandomness in central force optimization," Computing Research Repository, http://arxiv.org/abs/1001.0317.

[13] R. A. Formato, "Pseudorandomness in central force optimization," *British Journal of Mathematics & Computer Science*, vol. 3, no. 3, pp. 241–264, 2013.

[14] R. A. Formato, "Parameter-free deterministic global search with central force optimization," Computing Research Repository, http://arxiv.org/abs/1003.1039.

[15] R. A. Formato, "Parameter-free deterministic global search with simplified central force optimization," in *Advanced Intelligent Computing Theories and Applications*, vol. 6215 of *Lecture Notes in Computer Science*, pp. 309–318, Springer, Berlin, Germany, 2010.

[16] K. R. Mahmoud, "Central force optimization: Nelder-Mead hybrid algorithm for rectangular microstrip antenna design," *Electromagnetics*, vol. 31, no. 8, pp. 578–592, 2011.

[17] 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.

[18] R. A. Formato, "Improved cfo algorithm for antenna optimization," *Progress In Electromagnetics Research B*, vol. 19, pp. 405–425, 2010.

[19] R. C. Green II, L. Wang, M. Alam, and R. A. Formato, "Central force optimization on a GPU: a case study in high performance metaheuristics," *The Journal of Supercomputing*, vol. 62, no. 1, pp. 378–398, 2012.

[20] G. M. Qubati and N. I. Dib, "Microstrip patch antenna optimization using modified central force optimization," *Progress in Electromagnetics Research B*, vol. 21, pp. 281–298, 2010.

[21] R. A. Formato, "Central Force Optimization with variable initial probes and adaptive decision space," *Applied Mathematics and Computation*, vol. 217, no. 21, pp. 8866–8872, 2011.

[22] 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.

[23] W. Qian and T. Zhang, "Adaptive central force optimization algorithm," *Computer Science*, vol. 39, no. 6, pp. 207–209, 2012 (Chinese).

[24] Z. H. Cui and J. C. Zen, *Particle Swarm Optimization*, Science Press, Beijing, China, 2011.

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