Research Article

Gradient-Based Cuckoo Search for Global Optimization

Seif-Eddeen K. Fateen¹ and Adrián Bonilla-Petriciolet²

¹ Department of Chemical Engineering, Cairo University, Giza 12316, Egypt

² Department of Chemical Engineering, Aguascalientes Institute of Technology, 20256 Aguascalientes, AGS, Mexico

Correspondence should be addressed to Seif-Eddeen K. Fateen; fateen@engl.cu.edu.eg

Received 30 December 2013; Accepted 7 April 2014; Published 8 May 2014

Academic Editor: P. Karthigaikumar

Copyright © 2014 S.-E. K. Fateen and A. Bonilla-Petriciolet. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

One of the major advantages of stochastic global optimization methods is the lack of the need of the gradient of the objective function. However, in some cases, this gradient is readily available and can be used to improve the numerical performance of stochastic optimization methods specially the quality and precision of global optimal solution. In this study, we proposed a gradient-based modification to the cuckoo search algorithm, which is a nature-inspired swarm-based stochastic global optimization method. We introduced the gradient-based cuckoo search (GBCS) and evaluated its performance vis-à-vis the original algorithm in solving twenty-four benchmark functions. The use of GBCS improved reliability and effectiveness of the algorithm in all but four of the tested benchmark problems. GBCS proved to be a strong candidate for solving difficult optimization problems, for which the gradient of the objective function is readily available.

1. Introduction

The use of stochastic global optimization methods has gained popularity in a wide variety of scientific and engineering applications as those methods have some advantages over deterministic optimization methods [1]. Those advantages include the lack of the need for a good initial guess and the ability to handle multimodal and nonconvex objective functions without the assumptions of continuity and differentiability. In addition, one of the important advantages is the lack of the need for information about the gradient of the objective function. Gradient-free optimization methods can be either deterministic or stochastic, but their applications can be found in many disciplines [2]. In many applications, however, the gradient of the objective function is already available or easily obtainable. Yet, this valuable piece of information is entirely ignored by traditional stochastic optimization methods. However, for the functions whose gradient is available, the use of the gradient may improve the reliability and efficiency of the stochastic search algorithm. In particular, the quality and precision of solutions obtained with gradient-based optimization methods outperform those

obtained with traditional stochastic optimization methods. For a wide variety of engineering applications, it is needed to obtain solutions with a high precision and, therefore, the conventional stochastic optimization methods may fail to satisfy this requirement.

Until now, several stochastic methods have been proposed and investigated in challenging optimization problems using continuous variables and they include simulated annealing, genetic algorithms, differential evolution, particle swarm optimization, harmony search, and ant colony optimization. In general, these methods may show different numerical performances and, consequently, the search for more effective and reliable stochastic global optimization methods is currently an active area of research. In particular, the cuckoo search (CS) [3] is a novel nature-inspired stochastic optimization method. This relatively new method is gaining popularity in finding the global minimum of diverse science and engineering application problems [4–8]. For example, it was recently used for the design of integrated power systems [5] and solving reliability-redundancy allocation [6], phase equilibrium [7], and mobile-robot navigation [8] problems. CS was selected to test the concept of using the

gradient as a source of new information that guides cuckoos in their search.

Therefore, in this study, a simple modification was done to the CS algorithm to make use of the gradient information and enhance the reliability and efficiency of the algorithm. The aim of this work is to present a modification to the existing CS algorithm based on the gradient of the objective function and to evaluate its performance in comparison with the original algorithm. The remainder of this paper is divided as follows. Section 2 introduces the cuckoo search algorithm. Section 3 introduces the proposed modification and our new gradient-based cuckoo search (GBCS) algorithm. The numerical experiments performed to evaluate the modification are presented in Section 4. The results of the numerical experiments are presented and discussed in Section 5. Section 6 summarizes the conclusions of the work.

2. Cuckoo Search (CS) Algorithm

CS is a nature-inspired stochastic global optimization method that was developed by Yang and Deb [3, 9]. Its concept comes from the brood parasitism behavior of the cuckoo bird. Specifically, brood parasitism is a reproductive strategy followed by cuckoos in which they lay their eggs in the nests of other birds, which are usually other species. If these eggs are discovered by the host bird, it may abandon the nest completely or throw away the alien eggs. This natural phenomenon has led to the evolution of cuckoo eggs to mimic the egg appearance of local host birds. The following rules have been employed in the search algorithm to implement those concepts: (1) one egg is laid by each cuckoo in a random nest and it represents a set of solution coordinates, (2) the best eggs (i.e., solutions) are contained in a fraction of the nests and will carry over to the next generation, and (3) the number of nests is fixed and a host bird can find an alien egg with a specified probability $p_a \in [0, 1]$. If this condition occurs, the host bird can discard the egg or abandon the nest, and a new nest will be built elsewhere. For algorithm simplicity, this condition has been implemented in CS assuming that a fraction p_a of n nests is replaced by new nests. The pseudocode of CS is reported in Algorithm 1 and details of this metaheuristic are reported in [3]. Note that Lévy flights are used in CS for performing effectively both local and global searches in the solution domain. A Lévy flight is a random walk (i.e., a trajectory that consists of taking successive random steps) and is characterized by a sequence of sudden jumps, which are chosen from a probability density function that has a power law tail. In fact, Lévy flight is considered as the optimum random search pattern and has been useful in stochastic simulations of random natural phenomena including applications of astronomy, physics, and biology. To generate a new egg in CS, a Lévy flight is performed using the coordinates of an egg selected randomly. This step can be represented by

$$x_i^{t+1} = x_i^t + \alpha \oplus \text{Lévy}(\lambda), \quad (1)$$

where $\oplus$ denotes entry-wise multiplication, α is the step size, and Lévy (λ) is the Lévy distribution. Egg is displaced to this

new position if the objective function value is found better than another randomly selected egg. The step size α controls the scale of random search and depends on scales of the optimization problems under analysis.

A fraction $(1-p_a)$ of the nests selected at random is abandoned and replaced by new ones at new locations via local random walks. The local random walk can be written as

$$x_i^{t+1} = x_i^t + \alpha (x_j^t - x_k^t), \quad (2)$$

where x_j^t and x_k^t are two different solutions selected randomly by random permutation and α is a random number drawn from a uniform distribution. An advantage of CS over genetic algorithm, particle swarm optimization, and other stochastic optimization methods is that there is only one parameter to be tuned, namely, the fraction of nests to be abandoned $(1-p_a)$. However, Yang and Deb [3, 9] found that the results obtained for a variety of optimization problems were not so dependent on the value of p_a and suggested using $p_a = 0.25$.

3. Gradient-Based Cuckoo Search (GBCS) Algorithm

The purpose of this section is to introduce the simple modification of the original CS algorithm to incorporate information about the gradient of the objective function. Any modification to algorithm should not change its stochastic nature so as not to negatively affect its performance. A modification was made to the local random walk in which a fraction $(1-p_a)$ of the nests are replaced (2). In the original algorithm, when new nests are generated from the replaced nests via a random step, the magnitude and the direction of the step are both random. In the modified algorithm, the randomness of the magnitude of the step is reserved. However, the direction is determined based on the sign of the gradient of the function. If the gradient is negative, the step direction is made positive. If the gradient is positive, the step direction is made negative. Thus, new nests are generated randomly from the worse nests but in the direction of the minimum as seen from the point of view of the old nests. Thus, (2) is replaced by

$$\begin{aligned} \text{step}_i &= \alpha (x_j^t - x_k^t), \\ x_i^{t+1} &= x_i^t + \text{step}_i \otimes \text{sign}\left(-\frac{\text{step}_i}{df_i}\right), \end{aligned} \quad (3)$$

where sign function obtains the sign of its argument and df_i is the gradient of the objective function at each variable, that is, $\partial f / \partial x_i$.

This simple modification does not change the structure of the CS algorithm but makes important usage of the available information about the gradient of the objective function. No additional parameter is needed to implement this change.

4. Numerical Experiments

Twenty-four classical benchmark functions were used to evaluate the performance of GBCS as compared to the

```

begin
  Objective function  $f(x)$ 
  Generate initial population of  $n$  host nests  $x_i$ 
  while ( $t < \text{MaxGeneration}$ )
    Get a cuckoo randomly by Lévy flights
    Evaluate its quality/fitness
    Choose a nest among  $n$  (say,  $j$ ) randomly
    If ( $F_i > F_j$ )
      Replace  $j$  by the new solution
    end
    A fraction  $(1 - p_a)$  of nests are abandoned at random and new ones are built via random walk
    Keep the best solutions
    Rank the solutions and find the current best
  end while
  Postprocess results
end

```

ALGORITHM 1: Simplified algorithm of the cuckoo search algorithm.

original CS. These problems were chosen from amongst a list of forty-one functions. All forty-one functions were screened first by performing five optimization runs with both CS and GBCS. Functions for which both CS and GBCS performed extremely well with no significant difference in the results were deemed not suitable for comparison and excluded from the evaluation. Since CS is already a high-performance global optimizer, the excluded functions were not suitable for showing the differences between the two algorithms. Table 1 lists the twenty-four benchmark functions used for the evaluation of the two algorithms along with their derivatives and the search domain. The number of variables, the number of iterations used, and the value at the global minimum for each problem are shown with the results in Table 2.

Note that the benchmark functions used include three stochastic test functions. Most deterministic algorithms such as Nelder-Mead downhill simplex method would fail with those stochastic functions [9]. However, the inclusion of those stochastic functions in the present evaluation is to ensure that the modification of the stochastic method not only preserves its stochastic nature and ability to solve stochastic functions but also improves the performance of minimization algorithm. The stochastic functions were developed by turning three deterministic functions (Cube, Rosenbrock, and Griewank) into stochastic test functions by adding a stochastic vector ε drawn from a uniform distribution in $\in [0, 1]$ [9].

The twenty-four problems constitute comprehensive testing for the reliability and effectiveness of the suggested modification to the original CS. Eight functions have two variables only and their surface plots are shown in Figure 1.

Each test function was run for thirty times on MATLAB on an HP Pavilion dv6 laptop with Core i7 processor. The function value was recorded at every iteration and the average and standard deviation for the thirty runs were calculated at every iteration. The maximum number of iterations was 1000 for functions with 2 variables and 5000

for functions with 50 variables. The population size for the cuckoo swarm was 100. The only parameter used in the original CS, p_a , was kept constant at a value of 0.25. The MATLAB code for all the functions and their derivatives is available online at <http://dx.doi.org/10.1155/2014/493740> as supplementary material to this paper.

5. Results and Discussion

As stated, each of the numerical experiments was repeated 30 times with different random seeds for GBCS and for the original CS algorithm. The value at each iteration for each trial was recorded. The mean and the standard deviation of the function values were calculated at each iteration. The progress of the mean values is presented in Figures 2–4 for each function and a brief discussion of those results follows. The total CPU time for running the two algorithms for 30 times on each problem varied depending on the problem's number of variables and the number of iterations used. CPU time ranged from 38.27 min for two-variable problems to 91.03 min for fifty-variable problems.

The Ackley function has one minimum only, which was obtained using the two methods, as shown in Figure 2(a). However, GBCS is clearly more effective than CS in reaching the minimum at less number of iterations. The improvement in performance was also clear with the Beale function (Figure 2(b)). Beale has one minimum only, which was obtained satisfactorily by the two methods. This pattern of behavior was also observed for the booth function; GBCS was significantly more effective than the CS as it reached the global optimum within a tolerance of 10^{-32} in less than half of the number of iterations of CS, as shown in Figure 2(c). The first three functions are relatively easy to optimize; the global optima were easily obtained.

The cross-leg table function is a difficult one to minimize. Its value at the global minimum is -1 . Both algorithms have not been able to reach the global minimum. Yet,

TABLE 1: Test functions used for testing the performance of CS and GBCS.

Number	Name	Function and its derivative	Search Domain
1	Ackley	$f_1 = 20 \left(1 - e^{-0.5 \sqrt{0.5(x_1^2 + x_2^2)}} \right) - e^{0.5(\cos 2\pi x_1 + \cos 2\pi x_2)} + e^1$ $\frac{\partial f_1}{\partial x_k} = \frac{2x_k e^{-0.5 \sqrt{0.5(x_1^2 + x_2^2)}}}{\sqrt{0.5(x_1^2 + x_2^2)}} + \pi e^{0.5(\cos(2\pi x_1) + \cos(2\pi x_2))} \sin(2\pi x_k)$	$[-35, 35]$
2	Beale	$f_2 = (1.5 - x_1 + x_1 x_2)^2 + (2.25 - x_1 + x_1 x_2^2)^2 + (2.625 - x_1 + x_1 x_3^2)^2$ $\frac{\partial f_2}{\partial x_1} = 2(x_2^2 - 1)(x_1 x_2^2 - x_1 + 2.25) + 2(x_3^2 - 1)(x_1 x_3^2 - x_1 + 2.625) + 2(x_2 - 1)(x_1 x_2 - x_1 + 1.5)$ $\frac{\partial f_2}{\partial x_2} = 2x_1(x_1 x_2 - x_1 + 1.5) + 4x_1 x_2(x_1 x_2^2 - x_1 + 2.25) + 6x_1 x_2^2(x_1 x_3^2 - x_1 + 2.625)$	$[-4.5, 4.5]$
3	Booth	$f_3 = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$ $\frac{\partial f_3}{\partial x_1} = 10x_1 + 8x_2 - 34$ $\frac{\partial f_3}{\partial x_2} = 8x_1 + 10x_2 - 38$	$[-10, 10]$
4	Cross-leg table	$f_4 = - \left[\left \sin(x_1) \sin(x_2) e^{\left 100 - \sqrt{0.5(x_1^2 + x_2^2)} \right } \right + 1 \right]^{-0.1}$ $\frac{\partial f_4}{\partial x_1} = \frac{0.1 \sigma_1 (\sin(x_1) \sin(x_2)) \cos(x_1) \sin(x_2) + x_1 \sin(\sigma_2/\pi - 100) \sin(x_1) \sin(x_2) / \pi \sigma_2}{(\sigma_1 \sin(x_1) \sin(x_2) + 1)^{1.1}}$ $\frac{\partial f_4}{\partial x_2} = \frac{0.1 \sigma_1 (\sin(x_1) \sin(x_2)) \cos(x_2) \sin(x_1) + x_2 \sin(\sigma_2/\pi - 100) \sin(x_1) \sin(x_2) / \pi \sigma_2}{(\sigma_1 \sin(x_1) \sin(x_2) + 1)^{1.1}},$ <p>where $\sigma_1 = e^{(\sigma_2/n) - 100}$, $\sigma_2 = \sqrt{x_1^2 + x_2^2}$</p>	$[-10, 10]$
5	Himmelblau	$f_5 = (x_1^2 + x_2 - 11)^2 + (x_2^2 + x_1 - 7)^2$ $\frac{\partial f_5}{\partial x_1} = 2x_1 + 4x_1(x_1^2 + x_2 - 11) + 2x_2^2 - 14, \quad \frac{\partial f_5}{\partial x_2} = 2x_2 + 4x_2(x_2^2 + x_1 - 7) + 2x_1^2 - 22$	$[-5, 5]$
6	Levy 13	$f_6 = \sin^2(3\pi x_1) + (x_1 - 1)^2 [1 + \sin^2(3\pi x_2)] + (x_2 - 1)^2 [1 + \sin^2(2\pi x_2)]$ $\frac{\partial f_6}{\partial x_1} = (2x_1 - 2) (\sin(3\pi x_2)^2 + 1) + 6\pi \cos(3\pi x_1) \sin(3\pi x_1)$ $\frac{\partial f_6}{\partial x_2} = (2x_2 - 2) (\sin(2\pi x_2)^2 + 1) + 4\pi \cos(2\pi x_2) \sin(2\pi x_2) (x_2 - 1)^2 + 6\pi \cos(3\pi x_2) \sin(3\pi x_2) (x_1 - 1)^2$	$[-10, 10]$
7	Matyas	$f_7 = 0.26x_1^2 - 0.48x_1x_2 + 0.26x_2^2$ $\frac{\partial f_7}{\partial x_1} = 0.52x_1 - 0.48x_2, \quad \frac{\partial f_7}{\partial x_2} = 0.52x_2 - 0.48x_1$	$[-10, 10]$

TABLE 1: Continued.

Number	Name	Function and its derivative	Search Domain
8	Schaffer	$f_8 = 0.5 + \frac{\sin^2 \left[\sqrt{x_1^2 + x_2^2} \right] - 0.5}{\left[0.001 (x_1^2 + x_2^2) + 1 \right]^2}$ $\frac{\partial f_8}{\partial x_k} = -x_k \left(\frac{(0.004 \sin(\sigma)^2 - 0.002)}{(0.001 (x_1^2 + x_2^2) + 1)^3} - \frac{2 \cos(\sigma) \sin(\sigma)}{\sigma (0.001 (x_1^2 + x_2^2) + 1)^2} \right),$ <p>Where $\sigma = \sqrt{x_1^2 + x_2^2}$</p>	$[-100, 100]$
9	Powell	$f_9 = (x_1 + 10x_2)^2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^4 + 10(x_1 - x_4)^4$ $\frac{\partial f_9}{\partial x_1} = 2x_1 + 20x_2 + 40(x_1 - x_4)^3 \frac{\partial f_9}{\partial x_2} = 20x_1 + 200x_2 + 4(x_2 - 2x_3)^3$ $\frac{\partial f_9}{\partial x_3} = 10x_3 - 10x_4 - 8(x_2 - 2x_3)^3 \frac{\partial f_9}{\partial x_4} = 10x_4 - 10x_3 - 40(x_1 - x_4)^3$	$[-1000, 1000]$
10	Power sum	$f_{10} = \sum_{i=1}^m \left[\left(\sum_{j=1}^m x_j^i \right) - b_i \right]^2, b = [8, 18, 42, 114]$ $\frac{\partial f_{10}}{\partial x_k} = 2 \sum_{i=1}^m i x_k^{i-1} \left[\left(\sum_{j=1}^m x_j^i \right) - b_i \right]$	$[0, 4]$
11	Shekel 5	$f_{11} = - \sum_{i=1}^m \frac{1}{c_i + \sum_{j=1}^n (x_j - a_{ij})^2}, m = 10, \frac{\partial f_{11}}{\partial x_k} = \sum_{i=1}^m \frac{2(x_k - a_{ik})}{\left[c_i + \sum_{j=1}^n (x_j - a_{ij})^2 \right]^2}$	$[0, 10]$
12	Wood	$f_{12} = 100(x_1^2 - x_2)^2 + (x_1 - 1)^2 + (x_3 - 1)^2 + 90(x_3^2 - x_4)^2 + 10.1[(x_2 - 1)^2 + (x_4 - 1)^2] + 19.8(x_2 - 1)(x_4 - 1)$ $\frac{\partial f_{12}}{\partial x_1} = 2x_1 - 400x_1(x_2 - x_1^2) - 2, \frac{\partial f_{12}}{\partial x_2} = -200x_1^2 + 220.2x_2 + 19.8x_4 - 40$ $\frac{\partial f_{12}}{\partial x_3} = 2x_3 - 360x_3(x_4 - x_3^2) - 2, \frac{\partial f_{12}}{\partial x_4} = -180x_3^2 + 19.8x_2 + 200.2x_4 - 40$	$[-1000, 1000]$
13	Cube	$f_{13} = \sum_{i=1}^{m-1} 100(x_{i+1} - x_i^3)^2 + (1 - x_i)^2 \frac{\partial f_{13}}{\partial x_1} = 2x_1 - 600x_1^2(x_2 - x_1^3) - 2$ $\frac{\partial f_{13}}{\partial x_k} = 202x_k - 600x_k^2(x_{k+1} - x_k^3) - 200x_{k-1}^3 - 2, \frac{\partial f_{13}}{\partial x_m} = 200(x_m - x_{m-1}^3)$	$[-100, 100]$
14	Stochastic Cube	$f_{14} = \sum_{i=1}^{m-1} 100\epsilon_i(x_{i+1} - x_i^3)^2 + (1 - x_i)^2, \frac{\partial f_{13}}{\partial x_1} = 2x_1 - 600x_1^2\epsilon_1(x_2 - x_1^3) - 2$ $\frac{\partial f_{13}}{\partial x_k} = 2x_k - 600\epsilon_k x_k^2(x_{k+1} - x_k^3) + 200\epsilon_k(x_k - x_{k-1}^3) - 2, \frac{\partial f_{13}}{\partial x_m} = 200\epsilon_{m-1}(x_m - x_{m-1}^3)$	$[-100, 100]$
15	Sphere	$f_{15} = \sum_{i=1}^m x_i^2, \frac{\partial f_{15}}{\partial x_k} = 2x_k$	$[-100, 100]$
16	Hartmann	$f_{16} = - \sum_{i=1}^4 \alpha_i \exp \left(- \sum_{j=1}^6 A_{ij} (x_j - P_{ij})^2 \right) \frac{\partial f_{16}}{\partial x_k} = \sum_{i=1}^4 2\alpha_i A_{ij} (x_k - P_{ik}) \exp \left(- \sum_{j=1}^6 A_{ij} (x_j - P_{ij})^2 \right)$	$[0, 1]$

TABLE I: Continued.

Number	Name	Function and its derivative	Search Domain
17	Dixon-price	$f_{17} = (x_1 - 1)^2 + \sum_{i=2}^m i(2x_i^2 - x_{i-1})^2, \frac{\partial f_{17}}{\partial x_1} = -8x_2^2 + 6x_1 - 2,$ $\frac{\partial f_{17}}{\partial x_k} = (2x_k - 4x_{k+1}^2)(k+1) - 8kx_k(x_{k-1} - 2x_k^2), \frac{\partial f_{17}}{\partial x_m} = -8m x_m(x_{m-1} - 2x_m^2)$	$[-10, 10]$
18	Griewank	$f_{18} = \frac{1}{4000} \left[\sum_{i=1}^m (x_i - 100)^2 \right] - \left[\prod_{i=1}^m \cos\left(\frac{x_i - 100}{\sqrt{i}}\right) \right] + 1$ $\frac{\partial f_{18}}{\partial x_k} = \frac{(x_k - 100)}{2000} + \frac{1}{\sqrt{i}} \tan\left(\frac{x_k - 100}{\sqrt{i}}\right) \left[\prod_{i=1}^m \cos\left(\frac{x_i - 100}{\sqrt{i}}\right) \right]$	$[-600, 600]$
19	Griewank stochastic	$f_{19} = \frac{1}{4000} \left[\sum_{i=1}^m \varepsilon_i (x_i - 100)^2 \right] - \left[\prod_{i=1}^m \cos\left(\frac{x_i - 100}{\sqrt{i}}\right) \right] + 1$ $\frac{\partial f_{18}}{\partial x_k} = \frac{\varepsilon_k (x_k - 100)}{2000} + \frac{1}{\sqrt{i}} \tan\left(\frac{x_k - 100}{\sqrt{i}}\right) \left[\prod_{i=1}^m \cos\left(\frac{x_i - 100}{\sqrt{i}}\right) \right]$	$[-600, 600]$
20	Michaelwicz	$f_{20} = -\sum_{i=1}^m \sin(x_i) \sin\left(\frac{ix_i^2}{\pi}\right)$ $\frac{\partial f_{20}}{\partial x_k} = -\cos(k-x) \sin\left(\frac{kx_k^2}{\pi}\right) - \frac{40kx_k}{\pi} \cos\left(\frac{kx_k^2}{\pi}\right) \sin\left(\frac{kx_k^2}{\pi}\right) \sin(x_k)$	$[0, \pi]$
21	Rosenbrock	$f_{21} = \sum_{i=1}^{m-1} 100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2, \frac{\partial f_{21}}{\partial x_1} = 2(x_1 - 1) - 400x_1(x_2 - x_1^2)$ $\frac{\partial f_{21}}{\partial x_k} = -200x_{k-1}^2 + 202x_k - 400x_k(x_{k+1} - x_k^2) - 2, \frac{\partial f_{21}}{\partial x_m} = 200(x_m - x_{m-1}^2)$	$[-50, 50]$
22	StochasticRosenbrock	$f_{22} = \sum_{i=1}^{m-1} 100\varepsilon_i(x_{i+1} - x_i^2)^2 + (x_i - 1)^2, \frac{\partial f_{22}}{\partial x_1} = 2(x_1 - 1) - 400\varepsilon_1 x_1(x_2 - x_1^2)$ $\frac{\partial f_{22}}{\partial x_k} = 2x_k + 200\varepsilon_k(x_k - x_{k-1}^2) - 400\varepsilon_k x_k(x_{k+1} - x_k^2) - 2, \frac{\partial f_{22}}{\partial x_m} = 200\varepsilon_i(x_m - x_{m-1}^2)$	$[-50, 50]$
23	Trigonometric	$f_{23} = \sum_{i=1}^m \left[m + i(1 - \cos x_i) - \sin x_i - \sum_{j=1}^m \cos x_j \right]^2$ $\frac{\partial f_{23}}{\partial x_k} = 2 \left[m + k(1 - \cos x_i) - \sin x_i - \sum_{j=1}^m \cos x_j \right] [(i+1) \sin x_k - \cos x_k]$	$[-1000, 1000]$
24	Zacharov	$f_{24} = \sum_{i=1}^m x_i^2 + \left(\sum_{i=1}^m 0.5 i x_i \right)^2 + \left(\sum_{i=1}^m 0.5 i x_i \right)^4, \frac{\partial f_{24}}{\partial x_k} = 2x_k + k \sum_{i=1}^m 0.5 i x_i + 2k \left(\sum_{i=1}^m 0.5 i x_i \right)^3$	$[-5, 10]$

FIGURE 1: Surface plots of the two-variable benchmark functions used in this study: (a) Ackley, (b) Beale, (c) Booth, (d) cross-leg table, (e) Himmelblau, (f) Levy 13, (g) Matyas, and (h) Schaffer functions.

TABLE 2: Values of the mean minima and standard deviations obtained by the CS and GBCS algorithms compared with the value of the global minima of the twenty-four benchmark problems.

Number	Benchmark function	Number of variables	Global min.	Number of iterations	GBCS		CS	
					Mean	Std. Dev.	Mean	Std. Dev.
1	Ackley	2	0	1000	0	0	2.2204E-16	6.7752E-16
2	Beale	2	0	1000	0	0	5.7891E-30	1.2165E-29
3	Booth	2	0	1000	0	0	0	0
4	Cross-leg table	2	-1	1000	-1.1463E-2	7.672E-3	-6.2704E-3	3.6529E-3
5	Himmelblau	2	0	1000	1.7058E-28	2.836E-28	2.5958E-19	5.3451E-19
6	Levy 13	2	0	1000	1.3498E-31	6.6809E-47	1.3498E-31	6.6809E-47
7	Matyas	2	0	1000	2.7691E-54	4.728E-54	2.0407E-38	5.0616E-38
8	Schaffer	2	0	3000	0	0	7.4015E-18	1.9193E-17
9	Powell	4	0	1000	1.8694E-8	3.5848E-8	1.6296E-13	3.4802E-13
10	Power sum	4	0	1000	1.8328E-4	1.6761E-4	2.5432E-4	1.8167E-4
11	Shekel 5	4	-10.536	200	-10.536	1.6289E-5	-10.536	1.8421E-2
12	Wood	4	0	1000	2.3726	2.2208	0.40838	0.337
13	Cube	5	0	5000	1.2567	0.86542	5.782E-8	2.5596E-7
14	Stochastic cube	5	0	5000	7.7438	6.9815	6.4369	5.0292
15	Sphere	5	0	1000	2.5147E-38	5.1577E-38	1.1371E-21	1.2967E-21
16	Hartmann	6	-3.3224	200	-3.3224	4.3959E-10	-3.3215	6.0711E-4
17	Dixon-price	50	0	5000	4.7094E-2	1.6904E-1	6.6667E-1	2.6103E-6
18	Griewank	50	0	5000	0	0	3.3651E-10	9.4382E-10
19	Stochastic Griewank	50	0	5000	7.2758E-13	2.8579E-12	6.9263	2.0451
20	Michaelwicz	50		5000	-32.263	1.3729	-27.383	1.3551
21	Rosenbrock	50	0	5000	0.97368	0.5885	35.286	3.7012
22	Stochastic Rosenbrock	50	0	5000	58.599	19.122	4894.4	4678.3
23	Trigonometric	50	0	5000	5356.0	4536.0	19435	4033.7
24	Zacharov	50	0	5000	6.5769	1.3288	27.031	5.2748

GBCS performed significantly better than CS, as shown in Figure 2(d). On the other hand, both algorithms were able to identify the global minimum of the Himmelblau function. Figure 2(e) shows the evolution of the mean best values. GBCS performed more effectively than CS. Both algorithms were also able to identify the minimum of the Levy 13 function (Figure 2(f)). However, GBCS was significantly more effective than CS. This pattern was repeated with the Matyas function, as shown in Figure 2(g).

The Schaffer function is multimodal. Both GBCS and CS failed to converge to the global minimum within the 10^{-10} tolerance at 1000 iterations. Running both algorithms to 3000 iterations resulted in GBCS reaching the global optimum, while CS not, as shown in Figure 2(f). Schaffer function concludes the 2-variable functions. GBCS performed better than CS in all of them.

The Powell function has four variables. The evolution of the mean best values of CS and GBCS, plotted in Figure 3(a), showed that performance of CS was better than GBCS, although they were close to the global minimum. Powell is one of the few test functions, for which the use of the gradient did not improve the performance of the CS algorithm.

Minor improvements were observed with the power sum and the Shekel 5 functions. Power sum's global optimum was not obtained by the two algorithms, as shown in Figure 3(b), as both algorithms seem to be trapped in a local minimum. For the Shekel 5 function, Figure 3(c), the global optimum was easily optimized, with minor improvement in performance for the GBCS algorithm. The global optimum for the Wood function, which has four variables as well, was not achieved by both algorithms within 1000 iterations, as shown in Figure 3(d). When they were run for 5000, GBCS seemed to be trapped in a local minimum and was not able to reach the global optimum, which was obtained by CS. The Wood function is the only function for which GBCS performance was much worse than that of the original CS algorithm.

The performance of both algorithms for the Cube and the Stochastic Cube functions was peculiar. In both cases, GBCS outperformed CS at the early iterations. At larger iterations CS did better. For the two functions, as depicted in Figures 3(e) and 3(f), the global minimum was not obtained with 1000 iterations so both problems were run for 5000. CS was able to come close to the global minimum for the Cube function, while GBCS seemed to be trapped in a local minimum. For

FIGURE 2: Evolution of mean best values for GBCS and the original CS algorithm for (a) Ackley, (b) Beale, (c) Booth, (d) Cross-leg table, (e) Himmelblau, (f) Levy 13, (g) Matyas, and (h) Schaffer functions.

the Stochastic Cube function, CS did slightly better at larger iterations, but both algorithms failed to reach the global minimum after 5000 iterations.

The sphere function has five variables as the Cube function, but it is easier to solve. Both algorithms were able to identify the global optimum, as depicted in Figure 3(g), but GBCS considerably outperformed CS in terms of performance efficiency. The Hartmann function, which has 6

variables, was relatively easy to solve for GBCS algorithms as shown in Figure 3(h). GBCS arrived at the global minimum but CS could not do so within the tolerance. GBCS also outperformed CS in terms of performance efficiency in reaching the global optimum of the Hartmann function.

Figure 4 shows the performance of test functions with 50 variables. These are the most challenging problems due to the large domain space. Figure 4(a) depicts the performance of

FIGURE 3: Evolution of mean best values for GBCS and the original CS algorithm for (a) Powell, (b) Power Sum, (c) Shekel 5, (d) Wood, (e) Cube, (f) Stochastic Cube, (g) Sphere, and (h) Hartmann functions.

both algorithms for the Dixon-Price function. Clearly, GBCS performed better than CS in the early iterations. Even though both algorithms were not able to attain the global minimum, the minimum arrived at by GBCS at early iterations is orders-of-magnitude lower than that arrived at by CS.

GBCS outperformed CS in both the Griewank (Figure 4(b)) and the Stochastic Griewank (Figure 4(c)). For the Griewank function, both algorithms were able to identify the global minimum. However, GBCS arrived at the

global minimum at less than the half number of iterations compared to CS. For the Stochastic Griewank, CS was not able to identify the global minimum even after 5000 iterations. The result predicted by CS was more than 10 orders-of-magnitude higher than the result predicted by GBCS.

Figure 4(d) shows the results for the Michaelwics function. To the best of the authors' knowledge, the global minimum for the 50-variable Michaelwics is not known since

FIGURE 4: Evolution of mean best values for GBCS and the original CS algorithm for (a) Dixon-Price, (b) Griewank, (c) Stochastic Griewank, (d) Michaelwics, (e) Rosenbrock, (f) Stochastic Rosenbrock, (g) Trigonometric, and (h) Zacherov functions.

there is no published literature identifying it. GBCS identified a minimum that is lower than that identified by CS as depicted in Figure 4(d).

Although both algorithms were not able to identify the global optima of the Rosenbrock and the Stochastic Rosenbrock functions, as depicted in Figures 4(e) and 4(f), respectively, the results obtained by GBCS were orders-of-magnitude lower than those obtained by CS. The superiority of the GBCS is clearly demonstrated with those two functions.

Figure 4(g) shows the results for the Trigonometric function. For this difficult problem, both algorithms were not able to reach the global minimum within 5000 iterations. In this function, nonetheless, as in most cases studied, GBCS achieved a lower result than CS.

The last of the 50-variable functions tested was the Zacherov function. The results of Figure 4(h) also show the superiority of GBCS in reaching results that are orders-of-magnitude lower than those obtained by the original CS algorithm.

Table 2 shows a summary of the evaluation results for the twenty-four benchmark problems. GBCS was able to provide better solutions to all challenging problems. GBCS was able to achieve better performance in eighteen problems, equivalent performance in two problems, and worse performance in four out of the twenty-four problems. In one case, the Stochastic Griewank function, the global optimum was successfully obtained by GBCS for this 50-variable problem, while CS failed to find the global optimum even after 5000 iterations. On the other hand, the four functions for which CS performed better were the Powell, Wood, Cube, and Stochastic Cube functions.

It is interesting to note that GBCS outperformed CS in handling two of the three stochastic functions tested in this study. This result confirms that despite the use of the gradient as a guide upon which some moves are based, the stochastic nature of the algorithm remained unaffected. In fact, the additional information was used quite subtly. Cuckoos still search for nests randomly but with the help of some intelligence.

6. Conclusions

In this study, we use the gradient of the objective function, which could be available or easily obtainable for several objective functions in engineering calculations, to improve the performance of one of the most promising stochastic algorithms, the cuckoo search. The proposed modification was subtly implemented in the algorithm by changing the direction of the local random walk of cuckoos towards the direction of the minimum value from the point of view of the cuckoo's location. We evaluated this modification by attempting to find the global optimum of twenty-four benchmark functions. The newly developed GBCS algorithm led to improved reliability and effectiveness of the algorithm in all but four of tested benchmark problems, which included three stochastic functions. In some cases, the global minimum could not have been obtained via the original CS algorithm but was obtained via GBCS. Although the improvement in performance was not achieved with the entire set of benchmark problems, GBCS proved to be more reliable and efficient in the majority of the tested problems.

Conflict of Interests

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

References

- [1] C. A. Floudas and C. E. Gounaris, "A review of recent advances in global optimization," *Journal of Global Optimization*, vol. 45, no. 1, pp. 3–38, 2009.
- [2] L. M. Rios and N. V. Sahinidis, "Derivative-free optimization: a review of algorithms and comparison of software implementations," *Journal of Global Optimization*, vol. 56, pp. 1247–1293, 2012.
- [3] X. S. Yang and S. Deb, "Cuckoo search via Lévy flights," in *Proceedings of the World Congress on Nature & Biologically Inspired Computing (NABIC '09)*, pp. 210–214, IEEE, Coimbatore, India, December 2009.
- [4] X.-S. Yang and S. Deb, "Cuckoo search: recent advances and applications," *Neural Computing and Applications*, vol. 24, no. 1, pp. 169–174, 2014.
- [5] J. Piechocki, D. Ambroziak, A. Palkowski, and G. Redlarski, "Use of modified cuckoo search algorithm in the design process of integrated power systems for modern and energy self-sufficient farms," *Applied Energy*, vol. 114, pp. 901–908, 2014.
- [6] G. Kanagaraj, S. Ponnambalam, and N. Jawahar, "A hybrid cuckoo search and genetic algorithm for reliability-redundancy allocation problems," *Computers & Industrial Engineering*, vol. 66, no. 4, pp. 1115–1124, 2013.
- [7] V. Bhargava, S. Fateen, and A. Bonilla-Petriciolet, "Cuckoo search: a new nature-inspired optimization method for phase equilibrium calculations," *Fluid Phase Equilibria*, vol. 337, pp. 191–200, 2013.
- [8] P. K. Mohanty and D. R. Parhi, "Cuckoo search algorithm for the mobile robot navigation," in *Swarm, Evolutionary, and Memetic Computing*, pp. 527–536, Springer, New York, NY, USA, 2013.
- [9] 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.