An Enhanced Cuckoo Search Algorithm for Solving Optimization Problems

Ammar Mansoor Kamoona
University of Kufa
Najaf, Iraq
ammarm.kamoona@uokufa.edu.iq

Jagdish Chandra Patra
Swinburne University of Technology
Melbourne, Australia
jpatra@swin.edu.au

Alex Stojcevski
Swinburne University of Technology
Melbourne, Australia
astojcevski@swin.edu.au

***Abstract*—In this paper, we present an enhanced cuckoo search (ECS) algorithm based on Gaussian diffusion random walks and greedy selection approach. Despite wide applications of cuckoo search (CS) algorithm, it suffers from low convergence speed and lacks balance between local and global search. To overcome these limitations, ECS algorithm is proposed. It employs Gaussian diffusion random walks instead of Lévy flights random walks to enhance the local search. In addition, the greedy selection approach is employed to ensure that ECS reaches the optimum solution. Twenty one IEEE benchmark functions are used to evaluate the performance of the proposed ECS algorithm against CS and a recent adaptive cuckoo search (ACS) algorithms. The ECS shows excellent performance in reaching optimum values with a high convergence speed compared to CS and ACS algorithms. In addition, with several experiments, the effects of population size and the abandon probability P_a are carried out for the three algorithms and experiments shown that ECS is more robust than CS and ACS algorithms. Furthermore, superior performance of ECS algorithm against four other competitive algorithms has also been shown.**

Keywords— Enhanced cuckoo search; Optimization; Hybrid cuckoo search algorithm.

I. INTRODUCTION

In recent years, several metaheuristic nature-inspired algorithms for solving complex engineering optimization problems have been proposed [1] [2]. Metaheuristic algorithms are basically a part of stochastic optimization algorithms [3] that depend on randomization in generating possible solutions in search space. Therefore, these algorithms perform better by escaping local optima compared with deterministic algorithms [4]. In addition, metaheuristic algorithms are inspired by natural evolutionary behavior of some species, in most cases. Due to the robustness of metaheuristic optimization algorithms, these are used in solving complex problems in different scientific fields, e.g., optimization of FIR filter design [5], image processing [6], task scheduling [7], mechanical optimal design problem [8], data mining applications [9], dynamic optimization [10], and economic dispatch problems [11].

Some of the nature inspired optimization algorithms proposed in the literature are genetic algorithm (GA) [12], particle swarm optimization (PSO) [13], artificial bee colony (ABC) [14], ant colony optimization (ACO) [15] and cuckoo search (CS) algorithms [16][17]. GA [12] is inspired by the crossover and mutation principle of genetics, and PSO algorithm [13] is inspired by the social behavior of bird flocking in searching for food sources. The ABC algorithm [14] is inspired by the natural behavior of honey bee swarm in searching for food sources. ACO [15] is inspired by ants' behavior in seeking food sources, and CS algorithm [16][17] is inspired by the aggressive breeding behavior of cuckoo birds. The CS algorithm is one of the popular metaheuristic algorithms and it has been applied in different optimization problems, for instance, in data clustering [18], maximum power point tracking of solar panels [19] and system identification [20]. The CS algorithm suffers from a low convergence speed, since it uses a fixed step size over generations. To overcome this, Naik *et al.* [21] proposed an adaptive cuckoo search algorithm (ACS) with adaptive step size. The ACS converges to a near optimum solution faster than CS algorithm. However, the problem of ACS algorithm is, that it does not achieve better solutions compared to CS, in most cases. Two factors that play a significant role in optimization algorithms are exploration and exploitation. Exploration means searching at global scale while exploitation means searching at local scale. A good algorithm should find a balance between these two factors [22].

In this paper, we propose an enhanced cuckoo search (ECS) algorithm with a goal to reach better solutions with higher convergence speed compared to CS and ACS. The proposed algorithm utilizes both Gaussian diffusion random walks and greedy selection approach to enhance exploitation ability of CS algorithm.

II. CUCKOO SEARCH ALGORITHM

The CS algorithm [16], proposed by Yang and Deb, is one of the population based metaheuristic optimization algorithms inspired by the nature of brood parasitism of some cuckoo species. In addition, it is enhanced by utilizing the Lévy flights random walks instead of isotropic random walks which makes it more efficient than other metaheuristic algorithms, such as, GA and PSO. Another advantage of CS algorithm compared to PSO and GA is that it uses less number of parameters to be tuned. Therefore, CS algorithm is more adaptable and efficient. Certain species, e.g., the *ani* and *Guira*, cuckoos birds use a brood parasitism breeding mechanism as a reproduction strategy, where they lay their eggs in other birds' nests and

they may remove the host eggs to increase their survival chance. However, there is a chance that host birds directly engage in a combat with intruding cuckoos. If the host bird discovers the intruder cuckoo's eggs, she might destroy the cuckoo's eggs by throwing alien eggs away or she abandons the nest and build a new nest. This is represented by probability $P_a \in [0,1]$ [23]. The following assumptions are made for modeling the behavior of cuckoos [16]: one egg is laid by each cuckoo at time into a random chosen nest; the best nests with high quality eggs is carried over into the next generation; there is a fixed number of host nests; the chance that cuckoo eggs being discovered by the host birds is given by the probability P_a. This probability reflects the effect of replacing cuckoo eggs (discovered eggs by host bird) with new eggs at each generation. Note that an egg represents a solution.

These assumptions ensure that the best solutions will survive from generation to next generation as a selection process for the optimization algorithm. Thus, the goal of the CS algorithm is to replace solutions in the nests with new solutions that have better quality. A new solution X_i^{t+1} for cuckoo i is given by [16]:

$$X_i^{t+1} = X_i^t + \alpha \otimes L\acute{e}vy(\lambda) \tag{1}$$

$$\alpha = \alpha_0 \otimes (X_j^t - X_i^t) \tag{2}$$

where α is the step size (α > 0) with dimension equal to the dimension of the problem; the product ⊗ represents entry-wise multiplications; X_j^t is a random selected solution; and the $L\acute{e}vy(\lambda)$ is Lévy flights random walks. The parameter α_0 is chosen equal to 0.01, as recommend in [22], to enhance the search capability. Lévy flights is one of the random walks where its step length is drawn from *Lévy* distribution. This distribution is characterized by a series of instantaneous jumps generated by a probability density function that has a power law tail given by [23]:

$$L\acute{e}vy(\lambda) \approx S = t^{-\lambda}, (1 < \lambda \le 3), \tag{3}$$

The step length S of Lévy flights is drawn from a uniform distribution that obeys *Lévy* distribution. However, Mantegna algorithm is used as *Lévy* stable distribution in most problems to decide the step length. Thus, the step length S from Mantegna algorithm can be written as follows [23]:

$$S = u * |v|^{-1/\lambda} \tag{4}$$

where u and v are samples drawn from Gaussian normal distribution given by:

$$u \approx Norm(0, \sigma_u^2); v \approx Norm(0,1) \tag{5}$$

where σ_u^2 is the variance of the distributions given by [23]:

$$\sigma_u^2 = \left\{ \frac{\Gamma(1+\lambda)\sin(\pi\lambda/2)}{\Gamma\left[\frac{(1+\lambda)}{2}\right]\lambda 2^{(\lambda-1)/2}} \right\}^{1/\lambda} \tag{6}$$

where Γ is the gamma function and $1 < \lambda \le 3$.

For minimization problems, the basic steps of the CS algorithm are shown in the following Pseudo code. The initialization process of nests with solutions is carried out as follows:

$$X_i^t = LB + (UB - LB) \otimes rand(D) \tag{7}$$

where *LB* and *UB* are the lower and upper bounds of the optimization problem. The fraction P_a of worse solutions are generated as follows [23]:

$$X_i^{t+1} = X_i^t + \alpha_0 \otimes H(P_a - r) \otimes (X_j^t - X_k^t) \tag{8}$$

where X_j^t and X_k^t are two random solutions chosen by random permutation; H is a Heaviside function; and r is a random number drawn from a normal distribution.

Pseudo code: Cuckoo search algorithm

Begin
1 Fitness function $F(X), X = [x_1, x_2, ..., x_D]$, where D is the dimension of problem
2 Generation $t = 1$
3 Initialize population of N solutions $X_i (i = 1,2,..., N)$
4 **for** $i = 1$ to N
5 Initialize the nest X_i with solution randomly using (7)
6 Evaluate the fitness $F_i = F(X_i)$
7 **end for**
8 Determine the best nest X_{max} and the best fitness F_{max}
9 **while** (Maximum number of generation not reached) **do**
10 Choose a random solution i from all solutions
11 Generate a new solution by Lévy flights using (1)
12 Evaluate the fitness for the cuckoo F_i
13 Choose randomly a nest among N (say j)
14 **if** ($F_i < F_j$)
15 $X_j = X_i$
16 **end if**
17 A fraction P_a of worse nests are abandoned and new nests are built using (8)
18 Keep the best solution
19 Rank the solutions and find the current best F_{new}
20 **if** $F_{new} < F_{max}$
21 $F_{max} = F_{new}$ and $X_{max} = X_{new}$
22 **end if**
23 Increment the generation number $t = t + 1$
24 **end while**
End

III. PROPOSED ENHANCED CUCKOO SEARCH

Exploration and exploitation of optimization algorithms have a significant role in enhancing performance of the optimization algorithms. Exploration ensures searching the search space globally and this prevents algorithms from trapping in local optima. Whereas, exploitation ensures searching around current good solutions to find the best optimum values. Finding a good balance between exploration and exploitation plays a major role in the success of the algorithm [23]. From this perspective, the proposed ECS algorithm takes into account these two factors. The ECS

utilizes the Gaussian random walk for virus diffusion instead of the *Lévy* flights in exploitation of the search space to avoid the fixed step size used in the CS algorithm. Gaussian diffusion random walk has a promising performance in reaching the global optimum as it describes the behavior of random diffusion. Gaussian random walk for virus diffusing is introduced in [24] to model the behavior of virus diffusion in host cell. Here, the new solution is generated as follows [24]:

$$X_i^{t+1} = Gaussian(Best^t, \sigma) + (r_1 * Best^t - r_2 * X_i^t) \qquad (9)$$

where X_i^{t+1} represents the new solution and X_i^t is *i*th solution in generation *t*, where $i = (1, 2, \ldots, N)$ and *N* is the population size. $Best^t$ refers to the best solution in generation *t* and both r_1 and r_2 are random numbers between [0,1]. In Gaussian distribution, the mean is assigned to its best nest and the standard deviation σ is chosen as $(\log(t)/t) * (X_i^t - Best^t)$. The term $(\log(t)/t)$ is used to decrease the size of Gaussian steps as generation increases. This helps in reaching the optimum value at the end of the generation, because the searching is around the current solution and $(X_i^t - Best^t)$ is used to preserve the best solutions. The term $(r_1 * Best^t - r_2 * X_i^t)$ is used to adjust the search direction of Gaussian [24].

The performance of exploitation of ECS algorithm is enhanced further by utilizing greedy selection approach as used in ABC algorithm [25]. The ABC algorithm is a swarm based metaheuristic algorithm, which is inspired by the natural behavior of artificial bee swarm [14]. The population of ABC is divided into three sub-groups: employed bees, onlookers bees and scouts bees. In ABC, the number of solutions is equal to the number of employed bees and the number of onlookers bees. In generation *t*, the employed bees search around the current food source X_i^t, where $X_i^t = [x_{i1}, x_{i2}, \ldots, x_{iD}]$, to find a new food source V_i^t using greedy selection approach. The new food source V_i^t is generated as follows [25]:

$$V_i^t = X_i^t, \quad \text{where } V_i^t = [v_{i1}, v_{i2}, \ldots, v_{iD}]$$
$$v_{ij}^t = x_{ij}^t + r_3 * (x_{ij}^t - x_{kj}^t) \qquad (10)$$

where V_i^t is the new solution, j is a random chosen parameter index $1 \geq j \leq D$, r_3 is a random number between [-1, 1], x_{kj}^t is a food source selected randomly from food sources and k is a random number between [1, N]. After finding the new food source V_i^t, the fitness of new source (nectar amount) is evaluated. If the fitness of new food source is better than the old food source, then the employed bee will abandon the old food source and move to the new food source [25]. This process ensures searching around the new solution as an attempt to find optimal solution. The proposed ECS utilizes this approach to enhance exploitation to find the optimum value. Thus, in the ECS algorithm, in each iteration, the functions is evaluated twice. Firstly, by Gaussian diffusion random walks and secondly, by greedy selection approach as shown in the following pseudo code:

Pseudo code: The proposed ECS algorithm

Begin
1 Fitness function $F(X)$, $X = [x_1, x_2, \ldots, x_D]$, where D is the dimension of problem
2 Generation $t = 1$
3 Initialize population of N solutions X_i ($i = 1, 2. \ldots, N$)
4 **for** $i = 1$ to N **Do**
5 Initialize nest X_i with solution randomly using (7)
6 Evaluate the fitness $F_i = F(X_i)$
7 **end for**
8 Determine the best nest X_{max} and the best fitness F_{max}
9 **while** (Maximum number of generation not reached) **do**
10 Choose a random solution i from all solutions
11 Generate a new solution by Gaussian diffusion random walks using (9)
12 Evaluate the fitness for the cuckoo F_i
13 Choose a nest among N (say j) randomly
14 **if** ($F_i < F_j$)
15 Replace X_j by the new solution X_i
16 **end if**
17 Generate a new solution by greedy selection approach using (10)
18 Calculate the fitness of new solution $F_i' = F(V_i)$
19 Assume the old solution X_i with its fitness as F_i
20 **if** $F_i' < F_i$
21 Replace X_i by V_i
22 **end if**
23 A fraction P_a of worse nests are abandoned and new nests are built using (8)
24 Keep the best solution
25 Rank the solutions and find the current best F_{new}
26 **if** $F_{new} < F_{max}$
27 $F_{max} = F_{new}$ and $X_{max} = X_{new}$
28 **end if**
29 Increment the generation number $t = t+1$
30 **end while**
End

IV. Results and Discussions

In this Section, performance of the proposed ECS algorithm is evaluated using 21 IEEE benchmark functions [26] as a minimization problem. These benchmark functions, have different features, such as unimodal and multimodal, are widely used by researchers to evaluate the performance of different global optimization algorithms [16] [17] [21] [27].

A. Benchmark functions

The 21 benchmark functions can be divided into three categories: unimodal functions (F_1 - F_7), multimodal functions with variable dimension (F_8 - F_{11}) and multimodal functions with fixed dimensions (F_{12} - F_{21}).

The definition of unimodal functions are: F_1 - sphere function, F_2 - Schwefel's problem 2.22, F_3 - Schwefel's problem 1.2, F_4 - Schwefel's problem 2.21, F_5 - generalized Rosenbrock's function, F_6 - a step function and F_7 - a quartic function. The definition of multimodal functions with variable dimension are: F_8 - generalized Schwefel's problem 2.26, F_9 - generalized Rastrigin's function, F_{10} - Ackley's function, F_{11} -

generalized Griewank function. The definition of the multimodal functions with fixed dimension D and m number of local minima are: F_{12} - Shekel's foxholes function with $D = 2$, F_{13} - Kowalik's function with $D = 4$, F_{14} - a Six-hump Camel-Back function with $D = 2$, F_{15} - a Branin function with $D = 2$, F_{16} - a Goldstein-price function with $D = 2$, F_{17} and F_{18} are Hartman family functions with $D = 3$ and 6, respectively. $F_{19} – F_{21}$ are Shekel's family functions with $D = 4$, and $m = 5, 7$ and 10, respectively. The mathematical definition of these functions can be found in [26].

B. Experimental setup and results

ECS, CS and ACS algorithms were implemented using MATLAB 2014b in a PC Intel 2.4 GHz processor with 4 GB RAM. As recommend in [23], the parameter settings used in experiments are as follows: population size $N = 25$, abandon probability $P_a = 0.25$, and the Lévy flights settings, $\lambda = 1.5$ and $\alpha_0 = 0.01$. The maximum number of generation is set to 2000. We set the dimension D of both unimodal and multimodal functions with variable dimension to 30. The CS, ACS and proposed ECS algorithms are evaluated using benchmark functions with a goal to find the minimum optimum solution.

The three algorithms were executed with 100 independent runs, and the mean (μ), standard deviation (σ) of the solutions and average execution time were reordered. Table I shows results of the solution obtained for the three algorithms where the best mean among the three algorithms is shown in bold. The optimal value of the function (F_{min}) is shown in Col. 2. Among the seven unimodal functions F_1 - F_7, the CS and ACS algorithms reach quite close to the optimal solution, i.e., (0.00), except for F_5. Whereas, the ECS algorithm reaches the exact solution, in all cases. In case of F_5, the ECS algorithm achieves the best solution among the three algorithms.

TABLE I. PERFORMANCE COMPARISON BETWEEN THE PROPOSED ECS, ACS [21] AND CS [16] ALGORITHMS USING 21 BENCHMARK FUNCTIONS.

#	F_{min}	Algo.	μ	σ	Time (s)
F_1	0.0000E+00	**ECS**	**0.0000E+00**	0.00E+00	7.04
		ACS	1.9094E-09	2.08E-09	2.96
		CS	4.5078E-10	3.66E-10	4.01
F_2	0.0000E+00	**ECS**	**0.0000E+00**	0.00E+00	7.36
		ACS	6.3969E-04	3.59E-04	3.13
		CS	7.6169E-05	4.33E-05	4.23
F_3	0.0000E+00	**ECS**	**0.0000E+00**	0.00E+00	20.47
		ACS	1.1894E+01	4.59E+00	9.64
		CS	1.2976E+01	6.02E+00	10.70
F_4	0.0000E+00	ECS	**0.0000E+00**	0.00E+00	6.91
		ACS	2.2496E-01	9.27E-02	2.95
		CS	5.1471E+00	2.12E+00	4.06
F_5	0.0000E+00	**ECS**	**1.4034E+01**	6.94E+01	8.49
		ACS	2.4086E+01	6.52E+00	3.86
		CS	2.5311E+01	1.43E+01	4.97
F_6	0.0000E+00	ECS	0.0000E+00	0.00E+00	7.14
		ACS	0.0000E+00	0.00E+00	3.50
		CS	0.0000E+00	0.00E+00	4.15
F_7	0.0000E+00	**ECS**	**3.0101E-05**	2.50E-05	8.56
		ACS	1.5053E-02	5.16E-03	3.70
		CS	3.1697E-02	1.24E-02	4.76
F_8	-1.2569E+04	**ECS**	**-1.2569E+04**	7.61E-01	8.14
		ACS	-9.3646E+03	3.85E+02	3.61
		CS	-9.2887E+03	3.14E+02	4.67
F_9	0.0000E+00	**ECS**	**0.0000E+00**	0.00E+00	7.80
		ACS	8.3580E+01	1.25E+01	3.48
		CS	5.4419E+01	9.82E+00	4.57
F_{10}	0.0000E+00	**ECS**	**8.8818E-16**	0.00E+00	7.78
		ACS	1.5749E-03	4.74E-03	3.39
		CS	5.8399E-01	7.59E-01	4.48
F_{11}	0.0000E+00	**ECS**	**0.0000E+00**	0.00E+00	9.11
		ACS	8.3113E-06	3.63E-05	3.82
		CS	4.4054E-04	1.84E-03	4.91
F_{12}	1.0000E+00	ECS	9.9800E-01	2.57E-15	28.68
		ACS	9.9800E-01	2.57E-15	15.12
		CS	9.9800E-01	2.57E-15	15.82
F_{13}	3.0750E-04	ECS	3.0749E-04	2.42E-19	6.37
		ACS	3.0749E-04	5.97E-17	3.20
		CS	3.0749E-04	1.83E-18	3.94
F_{14}	-1.0316E+00	ECS	-1.0316E+00	1.56E-15	6.01
		ACS	-1.0316E+00	1.56E-15	2.54
		CS	-1.0316E+00	1.56E-15	3.24
F_{15}	3.9800E-01	ECS	3.9789E-01	1.06E-15	5.92
		ACS	3.9789E-01	1.06E-15	2.47
		CS	3.9789E-01	1.06E-15	3.20
F_{16}	3.0000E+00	ECS	3.0000E+00	8.85E-16	5.96
		ACS	3.0000E+00	1.03E-15	2.51
		CS	3.0000E+00	8.78E-16	3.22
F_{17}	-3.8600E+00	**ECS**	**-3.8600E+00**	6.25E-15	8.32
		ACS	-3.8628E+00	6.25E-15	3.71
		CS	-3.8628E+00	6.25E-15	4.41
F_{18}	-3.3200E+00	**ECS**	**-3.2970E+00**	4.86E-02	8.34
		ACS	-3.3220E+00	2.22E-15	3.75
		CS	-3.3220E+00	2.22E-15	4.48
F_{19}	-1.0000E+01	ECS	-1.0153E+01	1.79E-14	8.66
		ACS	-1.0153E+01	1.83E-14	4.01
		CS	-1.0153E+01	1.79E-14	4.76
F_{20}	-1.0000E+01	ECS	-1.0403E+01	1.58E-14	9.45
		ACS	-1.0403E+01	1.48E-14	4.50
		CS	-1.0403E+01	1.54E-14	5.26
F_{21}	-1.0000E+01	ECS	-1.0536E+01	1.42E-14	10.62
		ACS	-1.0536E+01	1.46E-14	5.29
		CS	-1.0536E+01	1.43E-14	6.03

For the five multimodal functions with variable dimensions ($F_8 - F_{12}$), the ECS algorithm reaches the optimum solution for three function (F_8, F_9 and F_{11}) and it shows best results for F_{10} compared to ACS and CS algorithms. The three algorithms show similar performance for F_{12}. Finally, in case of the nine multimodal functions with fixed dimension (F_{13} - F_{21}), all the three algorithms show similar performance. Thus it can be observed from Table I that the ECS algorithm outperforms the other two algorithms in reaching the optimum solution. However, the execution time of ECS algorithm is longer than other two algorithms.

C. Convergence characteristic

The convergence characteristic is an important factor that provides the rate at which the algorithm reaches the optimum solution. The convergence characteristics of six benchmark functions are shown in Figs. 1-6. Due to the rapid convergence of ECS algorithm, only 150 out of 2000 iterations are displayed in Figs. 3-6. From these figures, it can be seen that the ECS algorithm gives a high convergence rate compared to CS and ACS algorithms. Similar convergence characteristics were also observed for other benchmark functions.

D. Effect of Gaussain difussion

Here, we analyze the effect of Gaussian diffusion random walks by removing the greedy selection approach from ECS algorithm. We call this the new algorithm as ECS2. Here, in each generation, the function is evaluated only once using Gaussian diffusion random walks using (9). The pseudo code of ECS2 algorithm is the same as ECS algorithm except that we skip the steps 17-22 in the pseudo code in Section III. For the 21 benchmark functions, ECS2 reaches the same mean fitness values as ECS algorithm shown in Table I except for the functions shown in Table II. It is clear that ECS2 shows good results even by just using Gaussian diffusion random walks. However, this is not true for functions F_5, F_7, F_8, F_{13} and F_{17}, where ECS2 fails to reach equal or better solutions compared to ECS. This clearly shows how greedy selection approach enhances the exploration ability of proposed ECS algorithm.

E. Effect of parameter P_a

To investigate the effect of parameter P_a on the performance of CS, ACS and ECS algorithms, experiments with different values of P_a were carried. Three benchmark functions were selected namely, F_1 (Sphere), F_5 (generalized Rosenbrock) and F_9 (generalized Rastrigin). The parameter settings for CS, ACS and ECS used in experiments are as given in Section IV-B. Seventeen values for P_a = 0.1, 0.15, ..., 0.19 were used. The average fitness values over 100 runs were recorded and are shown in Figs. 7, 8 and 9 for the functions F_1, F_5 and F_9, respectively. From these Figures, it can be seen that the performance of the CS and ACS algorithms vary substantially with variation of P_a. Whereas, the ECS algorithm shows stable performance against variation of P_a, indicating that it is least sensitive to P_a .

F. Effect of population size

To show the behavior of ECS, CS and ACS algorithms with different population size on average fitness values, experiments with four different population sizes, N = 15, 25, 50, and 100 were conducted. The same parameter settings given in Section IV-B were used. The average fitness value and the average execution time over 100 independent runs are shown in Tables III, IV and V for the functions F_1, F_5 and F_9, respectively. It is clear that ECS is more robust against changes in population size compared to CS and ACS algorithms. With N = 25 the performance of the three algorithms show good performance. Increasing the population size beyond 25 increases the execution time but does not give substantial improvement in performance. Therefore, N=25 was selected in all the experiments. In addition, it is observed that the performance of both CS and ACS algorithms decrease when N increases beyond 25.

G. Comparison with other competitive algorithms

In this section, performance of ECS algorithm is compared with other four competitive optimization algorithms, namely, PSO [13], ABC [14], gray wolf optimization (GWO) [28] and whale optimization algorithm (WOA) [29]. The parameters settings for PSO algorithm is as follows: $C_1 = C_2 = 2.0$, w_{max} = 0.9, w_{min} = 0.2 and v_{max} = 6.0. In ABC, the number of employed bees plus the number of onlooker bees is set to 50. The population size (N) and maximum number of generations are set to 25 and 2000, respectively, for all algorithms. All algorithms are executed for 100 independent runs. The mean (μ) and standard deviation (σ) of twenty one benchmark functions are recorded in Table VI. It is clear that the proposed ECS algorithm achieves better results than the other four competitive algorithms in most of the benchmark functions.

TABLE II. PERFORMANCE COMPARISON BETWEEN THE PROPOSED ECS, ECS2 AND CS [16] ALGORITHMS USING FIVE BENCHMARK FUNCTIONS.

#	F_{min}	Algo.	μ	σ	Time (s)
F_5	0.0000E+00	**ECS**	**1.4034E+01**	6.94E+01	8.49
		ECS2	1.8439E+01	7.31E-01	6.78
		CS	2.5311E+01	1.43E+01	4.97
F_7	0.0000E+00	ECS	3.0101E-05	2.50E-05	8.56
		ECS2	2.4023E-05	2.07E-05	6.00
		CS	3.1697E-02	1.24E-02	4.76
F_8	-1.2569E+04	**ECS**	**-1.2569E+04**	7.61E-01	8.14
		ECS2	-1.0289E+04	5.63E+02	6.10
		CS	-9.2887E+03	3.14E+02	4.67
F_{13}	3.0750E-04	ECS	**3.0749E-04**	2.42E-19	6.37
		ECS2	3.3983E-03	7.17E-03	5.95
		CS	**3.0749E-04**	1.83E-18	3.94
F_{17}	-3.8600E+00	**ECS**	**-3.8600E+00**	6.25E-15	8.32
		ECS2	-3.8623E+00	6.25E-15	6.83
		CS	-3.8628E+00	6.25E-15	4.41

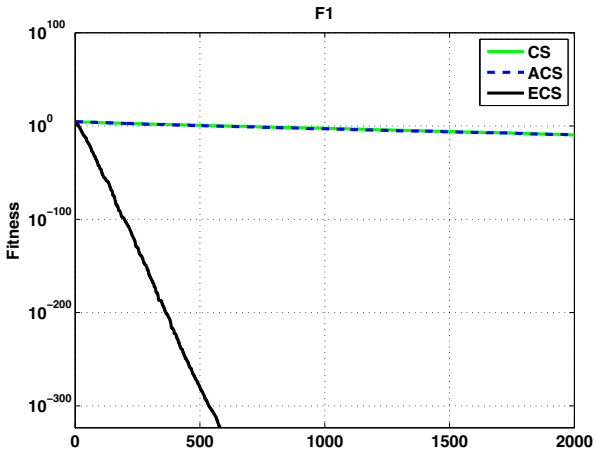

Fig. 1. Convergence characteristics of the CS, ACS and ECS algorithms for unimodal function F_1.

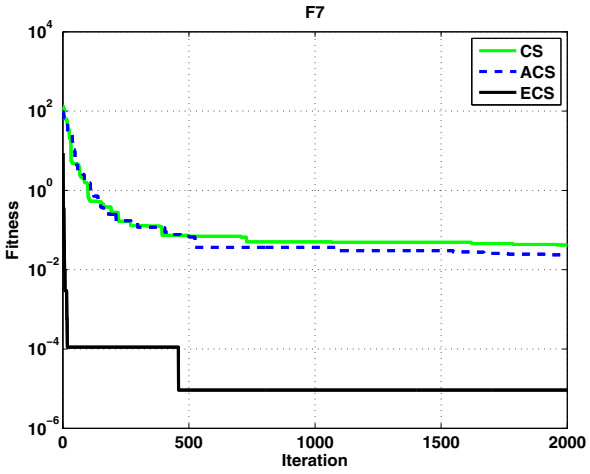

Fig. 2. Convergence characteristics of the CS, ACS and ECS algorithms for unimodal function F_7.

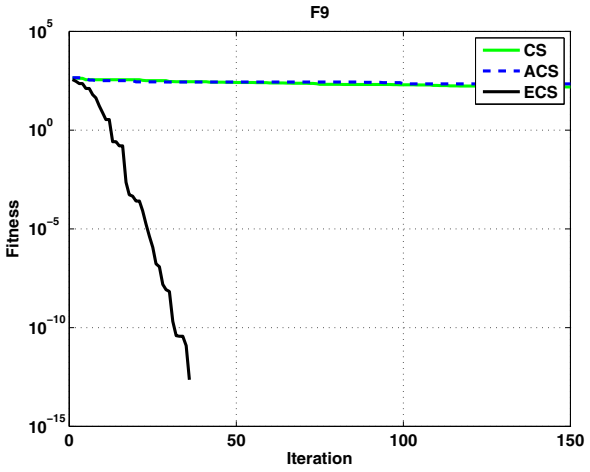

Fig. 3. Convergence characteristics of the CS, ACS and ECS algorithms for multimodal function F_9.

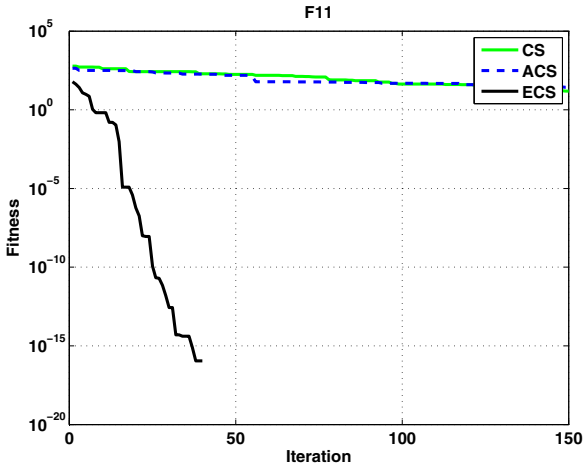

Fig. 4. Convergence characteristics of the CS, ACS and ECS algorithms for multimodal function F_{11}.

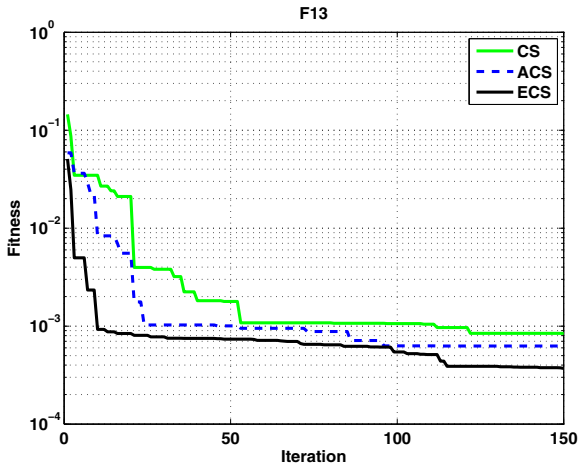

Fig. 5. Convergence characteristics of the CS, ACS and ECS algorithms for multimodal function F_{13}.

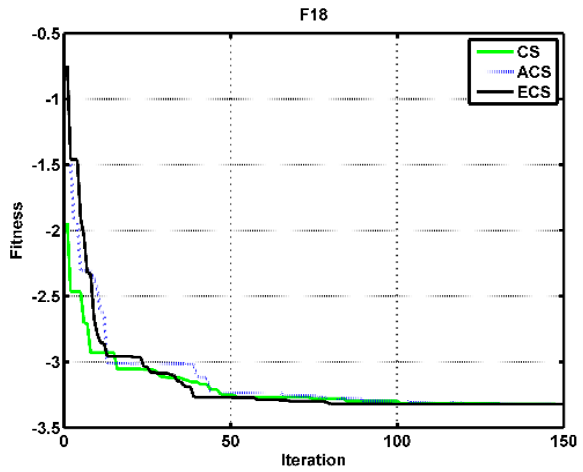

Fig. 6. Convergence characteristics of the CS, ACS and ECS algorithms for multimodal function F_{18}.

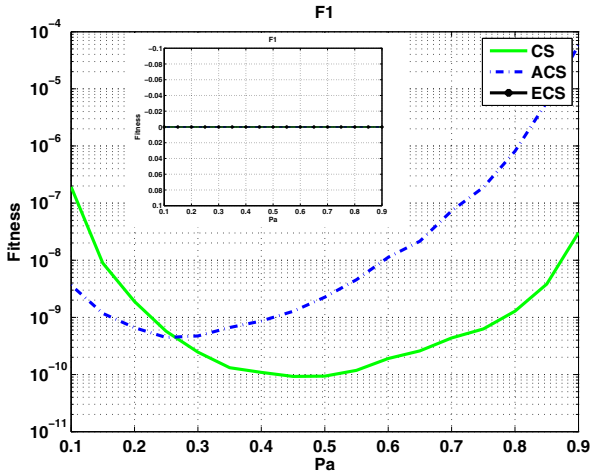

Fig. 7. Effect of P_a on the fitness of F_1.

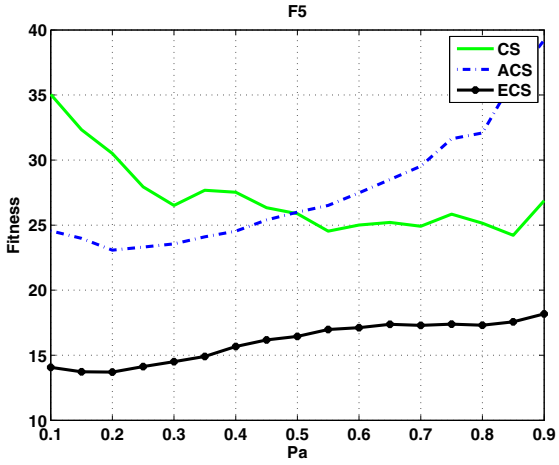

Fig. 8. Effect of P_a on the fitness of F_5.

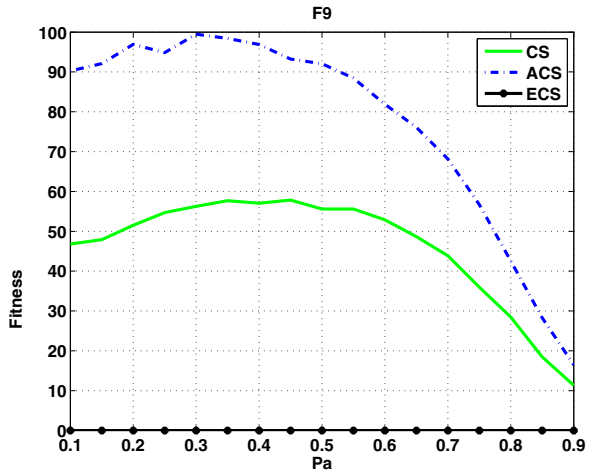

Fig. 9. Effect of P_a on the fitness *of* F_9.

TABLE III. EFFECT OF POPULATION SIZE N ON F_1 AVERAGE FITNESS.

Parameter	CS	ACS	ECS
N=15	7.3500E-08	9.0425E-10	**0.0000E+00**
Time (s)	2.49	1.84	4.45
N=25	5.1921E-10	4.8463E-10	**0.0000E+00**
Time (s)	4.10	2.94	7.04
N=50	3.3117E-09	7.7808E-08	**0.0000E+00**
Time (s)	8.65	5.78	10.60
N=100	2.1082E-07	1.5454E-05	**0.0000E+00**
Time (s)	15.66	11.03	28.35

TABLE IV. EFFECT OF POPULATION SIZE N ON F_5 AVERAGE FITNESS.

Parameter	CS	ACS	ECS
N=15	5.8282E+01	2.7760E+01	**1.7880E+01**
Time (s)	3.41	2.60	5.49
N=25	3.0439E+01	2.3100E+01	**1.4034E+01**
Time (s)	4.86	3.86	8.49
N=50	2.2610E+01	2.4714E+01	**1.0997E+01**
Time (s)	9.38	7.55	17.68
N=100	2.3264E+01	2.7701E+01	**8.6874E+00**
Time (s)	19.01	15.99	36.46

TABLE V. EFFECT OF POPULATION SIZE N ON F_9 AVERAGE FITNESS.

Parameter	CS	ACS	ECS
N=15	4.2981E+01	6.6360E+01	**0.0000E+00**
Time (s)	2.85	2.42	5.80
N=25	5.3250E+01	9.5295E+01	**0.0000E+00**
Time (s)	4.47	3.35	7.79
N=50	6.4725E+01	1.2841E+02	**0.0000E+00**
Time (s)	8.41	6.69	15.10
N=100	6.8157E+01	1.4713E+02	**0.0000E+00**
Time (s)	17.16	13.12	30.54

V. CONCLUSION

In order to enhance the exploitation capability of CS algorithm to reach the optimum solution, we proposed a new enhanced cuckoo search (ECS) algorithm by utilizing Gaussian diffusion random walks and greedy selection approach. The performance of the proposed ECS algorithm is evaluated using twenty-one IEEE benchmark functions and its superior performance over CS and ACS algorithms is shown. We have shown that in most cases the ECS reaches the optimum solution compared to CS and ACS. In addition, the ECS algorithm provides a high convergence rate when compared to the CS and ACS algorithms. Furthermore, by conducting sensitivity analysis against population size and abandon probability we have shown that the ECS shows more robustness than other two algorithms. We also showed the effects of Gaussian diffusion random walks and greedy selection approach on the performance of ECS, where we showed that greedy selection approach was essential to enhance the performance of ECS. In conclusion, the ECS algorithm outperforms CS and ACS algorithms in term of reaching optimum solution, speed of convergence and robustness. Furthermore, the ECS algorithm has shown superior performance compared to a few other competitive algorithms, such as, PSO, ABC, GWO and WOA.

ACKNOWLEDGMENT

This work is supported by higher committee for education development in Iraq (HCED) and University of Kufa, Iraq.

TABLE VI. PERFORMANCE COMPARISION BETWEEN ECS AND OTHER COMPETITIVE ALGORITHMS USING 21 BENCHMARK FUNCTIONS.

#	μ/σ	PSO	ABC	GWO	WOA	ECS
F_1	μ	7.83E-13	1.67E-15	1.36E-113	1.16E-285	**0.00E+00**
	σ	4.28E-12	1.14E-15	1.30E-65	0.00E+00	0.00E+00
F_2	μ	1.08E-04	1.62E-14	5.27E-66	4.13E-205	**0.00E+00**
	σ	1.03E-03	1.14E-14	1.30E-65	0.00E+00	0.00E+00
F_3	μ	1.73E+00	8.64E-01	1.60E-27	7.60E+03	**0.00E+00**
	σ	1.11E+00	5.53E-01	1.57E-26	6.08E+03	0.00E+00
F_4	μ	3.23E-01	5.79E+01	1.07E-27	2.39E+01	**0.00E+00**
	σ	1.17E-01	5.09E+00	2.72E-27	2.54E+01	0.00E+00
F_5	μ	5.51E+01	**1.74E+00**	2.69E+01	2.66E+01	1.40E+01
	σ	3.93E+01	1.92E+00	7.87E-01	5.20E-01	6.94E+01
F_6	μ	1.42E-13	1.81E-15	6.89E-01	2.43E-02	**0.00E+00**
	σ	5.57E-13	9.77E-16	3.16E-01	6.71E-02	0.00E+00
F_7	μ	3.77E-02	1.48E-01	4.41E-04	9.58E-04	**3.01E-05**
	σ	1.56E-02	3.56E-02	2.56E-04	1.22E-03	2.50E-05
F_8	μ	-6.37E+03	-1.25E+04	-6.08E+03	-1.15E+04	**1.25E+04**
	σ	1.44E+03	6.07E+01	7.56E+02	1.38E+03	7.61E-01
F_9	μ	4.27E+01	3.70E-08	1.08E-01	0.00E+00	**0.00E+00**
	σ	1.10E+01	2.23E-07	6.27E-01	0.00E+00	0.00E+00
F_{10}	μ	1.15E-02	4.91E-13	9.98E-15	3.77E-15	**8.88E-16**
	σ	1.16E-01	2.11E-13	2.87E-15	2.24E-15	0.00E+00
F_{11}	μ	1.15E-02	3.75E-04	1.40E-03	2.80E-03	**0.00E+00**
	σ	1.10E-02	1.83E-03	4.40E-03	1.06E-02	0.00E+00
F_{12}	μ	3.36E+00	**9.98E-01**	5.03E+00	2.19E+00	**9.98E-01**
	σ	1.78E-02	1.36E-16	1.98E-02	2.66E+00	2.57E-15
F_{13}	μ	7.38E-04	5.70E-04	4.96E-03	6.31E-04	**3.075E-04**
	σ	3.00E-04	1.00E-04	8.46E-03	3.93E-04	2.42E-19
F_{14}	μ	**-1.032E+00**	**-1.032E+00**	**-1.032E+00**	**-1.032E+00**	**-1.032E+00**
	σ	1.56E-15	5.83E-16	2.40E-09	3.20E-11	1.56E-15
F_{15}	μ	**3.98E-01**	**3.98E-01**	**3.98E-01**	**3.98E-01**	**3.98E-01**
	σ	1.06E-15	1.50E-05	2.98E-05	2.99E-07	1.06E-15
F_{16}	μ	**3.00E+00**	**3.00E+00**	**3.00E+00**	**3.00E+00**	**3.00E+00**
	σ	6.20E-16	5.33E-04	4.51E-06	1.78E-05	8.85E-16
F_{17}	μ	**-3.86E+00**	**-3.86E+00**	**-3.86E+00**	**-3.86E+00**	**-3.86E+00**
	σ	6.25E-15	5.02E-15	2.69E-03	2.33E-03	6.25E-15
F_{18}	μ	-3.28E+00	**-3.32E+00**	-3.27E+00	-3.24E+00	-3.30E+00
	σ	5.62E-02	1.54E-15	7.61E-02	1.29E-01	4.86E-02
F_{19}	μ	-8.60E+00	**-1.01E+01**	-9.50E+00	-9.43E+00	**-1.01E+01**
	σ	2.60E+00	1.35E-02	1.78E+00	1.87E+00	1.79E-14
F_{20}	μ	-9.35E+00	**-1.04E+01**	-1.02E+01	-9.39E+00	**-1.04E+01**
	σ	2.20E+00	3.56E-02	9.09E-01	2.18E+00	1.58E-14
F_{21}	μ	**-1.01E+01**	-1.05E+01	-1.03E+01	-9.35E+00	-1.05E+01
	σ	1.53E+00	2.24E-02	1.10E+00	2.42E+00	1.42E-14

REFERENCES

[1] C. Blum, J . Puchinger, G.R. Raidl and A. Roli, "Hybrid metaheuristics in combinatorial optimization: a survey," Appl. Soft Comput., vol. 11, pp. 4135–4151, 2011.

[2] I. Boussaïd, J. Lepagnot and P. Siarry, "A survey on optimization metaheuristics," Information Sciences, vol. 237, pp. 82–117, 2013.

[3] J. C. Spall, Introduction to stochastic search and optimization: estimation, simulation, and control, vol. 65, Hoboken:Wiley & Sons, 2005.

[4] A. R. Simpson, G. C. Dandy and L. J. Murphy, "Genetic algorithms compared to othertechniques for pipe optimization," Journal Water Resource Plannning and Management, vol. 120, pp. 423–43, 1994.

[5] S. K. Sarangi, R. Panda, P. K Das and A. Abraham, "Design of optimal high pass and band stop FIR filters using adaptive Cuckoo search algorithm," Engineering Applications of Artificial Intelligence, vol. 70, pp. 67-80, 2018.

[6] V. N. Manju and A. L. Fred, "AC coefficient and K-means cuckoo optimisation algorithm-based segmentation and compression of compound images," IET Image Processing, vol. 12, pp. 218-225, 2018.

[7] K. Li, and J. Wang, "Multi-objective Optimization for cloud task scheduling based on the ANP model, " Chinese Journal of Electronics, vol. 26, pp. 889-898, 2017.

[8] R. V. Rao, V. J. Savsani and D. Vakharia, "Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems," Computer-Aided Design, vol. 43, pp. 303-315, 2011.

[9] R. S. Parpinelli, H. S. Lopes and A. A. Freitas, "Data mining with an ant colony optimization algorithm," IEEE Trans. Evol. Comput., vol. 6, pp. 321-332, 2002.

[10] K. Deb and S. Jain, "Running performance metrics for evolutionary multi-objective optimization," Kanpur: Indian Institute of Technology Kanpur, India, (KanCAL Report No. 2002004), 2002.

[11] L. T. Al Bahrani and J. C. Patra, "Orthogonal PSO algorithm for economic dispatch of thermal generating units under various power constraints in smart power grid," Applied Soft Computing, vol. 58, pp. 404-426, 2017.

[12] M. Mitchell, An introduction to genetic algorithms, MIT press, 1998.

[13] R. C. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in Proc. of the sixth international symposium on micro machine and human science, 1995, pp. 39-43.

[14] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm," Journal of global optimization, vol. 39, pp. 459-471, 2007.

[15] C. Alberto, D. Marco and M. Vittorio, "Distributed optimization by ant colonies," in Proceedings of the first European conference on artificial life, 1991. pp. 134–42.

[16] X-S. Yang and S. Dev, "Cuckoo Search via Levy flights," in Proc. World Congr. Nature Biol. Inspired Compu., NaBIC 2009, Coimbatore, India, Dec. 2009, pp. 210-214.

[17] Y. Umenai, et al., "A modified cuckoo search algorithm for dynamic optimization problems," in Proc. World Congr. Evolu. Computation, CEC 2016, Vancouver, BC, Canada, July 2016, pp. 1757-1764.

[18] S. I. Boushaki, N. Kamel and O. Bendjeghaba, "A new quantum chaotic cuckoo search algorithm for data clustering", Expert Systems with Applications, vol. 96, pp. 358-372, 2018.

[19] J. Ahmed and Z. Salam, "A maximum power point tracking (MPPT) for PV system using cuckoo search with partial shading capability," Applied Energy, vol. 119, pp. 118-130, 2014.

[20] A. P. Patwardhan, R. Patidar and N. V. George, "On a cuckoo search optimization approach towards feedback system identification," Digital Signal Processing, vol. 32, pp. 156-163, 2014.

[21] M. Naik, M. R. Nath, A. Wunnava, S. Sahany and R. Panda, "A new adaptive Cuckoo search algorithm," in Proc. 2nd Int. Conf. Recent Trends in Information Systems, (ReTIS), 2015, pp. 1-5.

[22] X-S. Yang and M.Karamanoglu, "Swam intelligence and bio-inspired computation: An overview, " in Swarm intellgience and Bio-inspired computation theory and applications, Elsevier, pp. 3-23, 2013.

[23] X-S. Yang, Nature-Inspired Optimization Algorithms, Elsevier, pp. 45-65, 2014.

[24] M. D. Li, H. Zhao, X. W. Weng and T. Han, "A novel nature-inspired algorithm for optimization: virus colony search," Advances in Engineering Software, vol. 92, pp. 65-88, 2016.

[25] A. Draa and A. Bouaziz, "An artificial bee colony algorithm for image contrast enhancement," Swarm and Evolutionary Computation, vol. 16, pp. 69-84, 2014.

[26] Y. Xin, L. Yong and L. Guangming, "Evolutionary programming made faster," IEEE Trans. Evol. Comput., vol. 3, pp. 82-102, 1990.

[27] H. Rakhshani and A. Rahati, "Snap-drift cuckoo search: a novel cuckoo search optimization algorithm," Applied Soft Computing, vol. 52, pp. 771-794, 2017.

[28] S. Mirjalili and A. Lewis, "Grey wolf optimizer," Adv. Eng. Softw., vol. 69, pp. 46-61, 2014.

[29] S. Mirjalili and A. Lewis, "The whale optimization algorithm," Adv. Eng. Softw., vol. 95, pp. 51-67, 2016.