IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 19, 2019, accepted August 13, 2019, date of publication August 19, 2019, date of current version September 6, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2936360

An Innovative Damped Cuckoo Search Algorithm With a Comparative Study Against Other Adaptive Variants

MOHAMED REDA[1], AMIRA Y. HAIKAL[1], MOSTAFA A. ELHOSSEINI[1,2], AND MAHMOUD BADAWY[1]

[1]Computers Engineering and Control Systems Department, Faculty of Engineering, Mansoura University, Mansoura 35516, Egypt
[2]College of Computer Science and Engineering in Yanbu, Taibah University, Medina 30012, Saudi Arabia

Corresponding author: Mohamed Reda (mohamed.reda@mans.edu.eg)

ABSTRACT This paper aims to find the best variant of Cuckoo Search Algorithm that makes the step size of Lévy flight adaptive. For this reason, we introduce a new variant of CSA called Damped Cuckoo Search (DCS) in which the step size of Lévy flight is adaptive via the concept of damped oscillations that exist in most second order control systems. Moreover, we propose two other methods that tune the step size of Lévy flight based on chaotic maps. Then a deep comparative study is conducted among almost all variants of CSA that appeared in the last decade that modifies the step size of Lévy flight. All these variants are tested on CEC2017 benchmark functions. Statistical analyses are performed using the Friedman test followed by four post-hoc procedures to hold paired comparisons between the proposed DCS and the other CSA variants. Also, graphical statistical analyses are conducted on all variants via Box Plots. Finally, convergence graphs for all the variants are illustrated as well to show the speed of solution improvement over generations. Simulation results prove that the proposed DCS outperforms all other variants with a large degree of significance. Moreover, DCS increases the speed of convergence in comparison with the other variants. The box plot graphs prove that DCS has the most compact distribution for all results obtained in all runs on most functions.

INDEX TERMS Adaptive step size, box plots, CEC2017 benchmark functions, damped cuckoo search, damped oscillations, Friedman test, global optimization, Lévy flight, multiple chaotic cuckoo search, post-hoc procedures.

I. INTRODUCTION

optimization is mainly concerned with finding the optimal values for several decision variables to form a solution to an optimization problem. It is widely used in different fields such as energy, computer science, engineering, economics, medical and Engineering. An optimization algorithm is a set of steps used to solve the optimization problem and the most common are those developed based on nature-inspired ideas that deal with selecting the best alternative of the given objective functions.

Optimization algorithms can be a heuristic or metaheuristic approach [1]. Heuristic approaches are problem-based approaches in which each optimization problem has its own heuristic methods that can deal with it but can't deal with other kinds of optimization problems.

The metaheuristic-based algorithm is a general solver template that can manipulate various kinds of optimization problems including NP-hard problems such as traveling salesman problem [2], [3]. All metaheuristic algorithms have two major characteristics: the first one is called intensification (exploitation) which aims to search around the current best solutions and select the best candidates in the current search space as we get closer to the optimal solution. The second one is called diversification (exploration) and it is concerned with exploring the search space efficiently to avoid getting stuck in a local minimum [4].

Metaheuristic will be successful in solving a given optimization problem if it can provide a proper balance between exploitation and exploration, which will guarantee to find

The associate editor coordinating the review of this article and approving it for publication was Kai Li.

a high-quality solution for a given problem without being trapped in a local minimum. Metaheuristic optimization algorithms can be categorized into three main types: evolutionary algorithms (EAs), Trajectory-based algorithms, and swarm-based algorithms [4].

Evolutionary algorithms imitate the principle of survival of the fittest. It begins with a population of a set of individuals. At each generation, the EA recombines the promising characteristics of the current population in order to obtain a new population with better characteristics than previous generations. EAs include genetic algorithms (GAs) [5], genetic programming (GP) [6], differential evolution (DE) [7], and a harmony search (HS) algorithm [8].

Trajectory-based algorithms begin with a single candidate solution. At each iteration, this solution will move to its neighboring solution, which exists in the same search space region, using a specific neighborhood structure. Examples of trajectory-based algorithms include tabu search (TS) [1], simulated annealing (SA) [9], hill climbing [10], and β-hill climbing [11].

Swarm-based algorithms, on the other hand, mimic the behavior of a group of animals when searching for food. At each iteration, the solutions are normally generated based on information gained by previous generations [12]. Swarm-based algorithms include the artificial bee colony (ABC) [13], the particle swarm optimization (PSO) [14], the firefly algorithm (FA) [15], and the cuckoo search algorithm (CSA) [2]. The classification of these meta-heuristic algorithms is shown in Fig. 1.

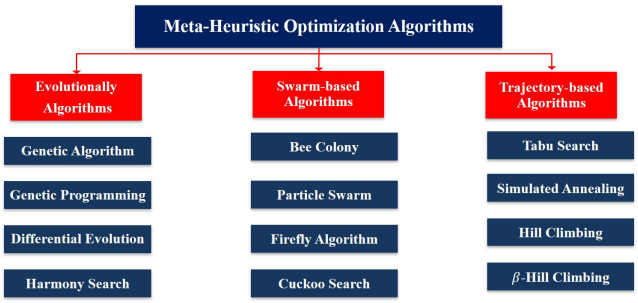

FIGURE 1. Optimization Algorithms [3].

In this paper, we are interested in Cuckoo Search Algorithm (CSA) which was developed by Yang and Deb in 2009 to imitate the brood parasitism behavior which is the primary mechanism of cuckoos. This bird lays its eggs in another birds' nest which is called the host nest. Cuckoo bird carefully mimics the color and pattern of the hosts' eggs to make their eggs carefully matching the host bird's eggs. If the host recognizes the cuckoo eggs in its own nest, it can throw the cuckoo eggs out of their nest or simply leave its nest and build a new one. For this reason, a cuckoo must be careful and accurate in mimicking the host eggs. From the optimization point of view, each egg in the nest represents a solution, and the cuckoo eggs represent a new candidate solution. The goal is to replace a not-so-good solution that exists in the nest with the new and potentially better solutions. The fraction P_a with a probability [0,1] checks if a host discovers the cuckoo eggs which are not it's own [2], [16].

Cuckoo search has significant advantages over all other meta-heuristic algorithms. The first one is that CSA has a smaller number of parameters to be tuned than PSO and GA, this makes it more generic to adopt more classes of optimization problems and also inexperienced users can easily deal with it. CSA combines two concepts where, it is a population-based similar to GA and PSO, and it also uses some sort of elitism similar to harmony search. The most important merit is that CSA has the strength of Trajectory-based Algorithms (TAs) in exploitation via a random walk and the strength of Evolutionary Algorithms (EAs) in exploration via Lévy Flight. Therefore, it has the ability of an efficient balance between local search strategy (exploitation) and the whole space search strategy (exploration). Another advantage of CSA is the randomization which is more effective because the step size of Lévy Flight is heavy-tailed, and any large step is possible [17].

Due to these merits, CSA has been successfully applied to various optimization problems such as image processing [18], [19], in the medical field [20], [21], data mining and clustering [22], economic dispatch problems [23], [24], engineering design [25], [26], and power and energy field [27], [28]. Taking into consideration the step size of Lévy Flight, which is heavy-tailed (any large step is possible), making the randomization in the exploration phase more effective. For the sake of improving the step size of Lévy Flight, many pieces of research aim to make Lévy Flight's step size adaptive such that it is inversely proportional to the generation number. At the beginning of iterations, the step size will be large enough to explore the search space (exploration), while when moving near to last iterations, the step size gets smaller to increase the efficiency of searching around the best solution (exploitation) [29]. The adaptive step size of Lévy Flight introduces a new source of balance between exploration and exploitation.

The first trial for the adaptivity was conducted by [30] by introducing a Modified Cuckoo Search (MCS) in which the step is inversely proportional to the square root of the iteration number. In the same year, Valian et al. proposed a new important variant of the adaptive cuckoo search called Improved Cuckoo Search ICS where the step size is exponentially decreasing with the iteration number [31]. After three years, Zhang and Chen proposed a new Improved Cuckoo Search ICS2 in which the step size is decreased exponentially with the fourth root of the iteration number [32]. Chaos Theory was first introduced into Cuckoo Search by Wang *et al.* [33], they used the chaotic sequence to make the step size of Lévy Flight adaptive. Wang et al. proposed a new variant of Cuckoo Search with a variable scaling factor called VSF [34]. Another study by Ong et al. introduced adaptive cuckoo search with two-parent cross over where the step size depends on the best solution obtained so far and is inversely proportional to the square root of iteration number [35]. In 2016,

Wang et al. proposed the formal Chaotic Cuckoo search CCS that uses the chaotic sequence for the adaptive step size [36]. Chi et al. proposed another variant of an Adaptive Cuckoo search ACS where the step size is exponentially decreasing with the iteration number [37]. A recent study by Cheng et al. involved a cuckoo search algorithm with dynamic feedback information [38].

The main contributions of this research are as follows:

- A novel variant of a cuckoo search algorithm is proposed named as the Damped Cuckoo Search (DCS). The Damped Cuckoo Search adapts the step size of Lévy Flight based on a proposed equation based on the concept of damped oscillations. This equation can be controlled by two parameters in order to change the speed of convergence and the decay speed of the step size as required.
- Introducing another variant of Cuckoo Search Algorithm that uses 10 chaotic maps to adapt the step size of Lévy Flight instead of using only one chaotic map. This method is inspired by multiple chaotic cuckoo search introduced in [39], however, we reuse this concept in adapting the step size of Lévy Flight instead of the chaotic search radius parameter in the chaotic local search equation. These 10 chaotic maps are applied to the step size of the Lévy Flight by two different methods. The first one is based on a parallel manner and the second one is based on the success rate for each map.
- A comparative analysis of 12 different variants of CSA which are using the concept of adaptive step size of Lévy Flight and our proposed algorithms is conducted as well. Our concern is spotted on the adaptive step size modification only, keeping the rest of the CSA parameters fixed to the standard CSA for all of the 12 variants presented in this paper. All of these algorithms are tested on CEC2017 benchmark functions with 30 dimensions.
- Statistical analysis is performed using the Friedman test followed by paired comparisons between DCS and the other variants using 4 different post-hoc procedures (Bonferroni procedure - Holm procedure - Holland procedure - Finner procedure).
- A graphical statistical analysis via Box plot graphs is performed to show the distribution of results for all algorithms on all functions for all runs.
- Convergence graphs are also supported in this paper for all 30 benchmark functions for all 12 algorithms to show graphically the evolution of the population overall the generations.

The rest of this paper is organized as follows: the next section covers the standard cuckoo search algorithm. Section III discusses various CSA variants through a literature review. Section IV is concerned with our proposed damped cuckoo search and the multiple chaotic cuckoo search applied to the step size of a Lévy Flight. Section V is about the parameter setting for all the algorithm and CEC2017 benchmark functions. Section VI contains all the simulation results, statistical analysis, box plots and convergence graphs for all the results. Finally, the conclusion gives a summary of our results and the recommendations based on our results and future work.

II. STANDARD CUCKOO SEARCH

A. INSPIRATION

1) CUCKOO BEHAVIOR

More than 1,000 species of birds exist in nature [16]. All mother birds for most of these species lay eggs in their nest that is built by them in secure places to keep their eggs safe from predators [40]. Some species that are called brood parasites birds. These kinds of birds do not build their own nests, but they lay their eggs in another species' nest, leaving the host bird to care for its eggs. The most famous species of the brood parasites is the cuckoo [41]. There are three types of brood parasitism, the first type is intraspecific brood parasitism, the second one is cooperative breeding and the last one is called nest takeover [2]. The Cuckoo strategy has some amazing characteristics, it starts by replacing one egg laid by the host bird with her own to increase the hatching probability of their egg in the nest of the host bird. Next, it mimics the color and pattern of the host eggs to reduce the probability of their eggs being discovered and abandoned. Also, the timing of egg-laying is amazing that it selects a nest in which the host bird just laid its eggs, so the cuckoo egg will hatch before the host eggs. The first action taken by the young cuckoo is to evict the host eggs out of the nest by blind propelling to increase its chance of food provided by the host bird [41]. Also, the young cuckoos mimic the call of host chicks to gain more access to the food [42]. On the other hand, if the host recognizes the cuckoo's egg in its nest, they throw the strange egg out of its nest or simply build a new nest and abandon the old one.

2) LÉVY FLIGHT

Many studies have proved that the flight behavior of a lot of animals and insects can be demonstrated by what so-called Lévy Flight [43]–[46]. Lévy Flight can be noticed when some animals and insects follow a long path with sudden turns combined with random, short movements [46]. A possible path of Lévy Flight can be shown in Fig. 2. Lévy Flight has been applied to optimization field, and results show its promising capability [44], [46]–[48]. Lévy Flight is a type of random walk that has a power law step length distribution with a heavy tail.

B. CUCKOO SEARCH ALGORITHM

Cuckoo Search Algorithm CSA is a recent meta-heuristic swarm-based optimization algorithm proposed by Yang and Deb in 2009. This algorithm combines the obligate brood parasitic behavior which is found in some cuckoo species with the Lévy Flight that exists in some birds and fruit flies [2]. CSA shows its efficiency and balances between the exploitation represented in the local search strategy and exploration

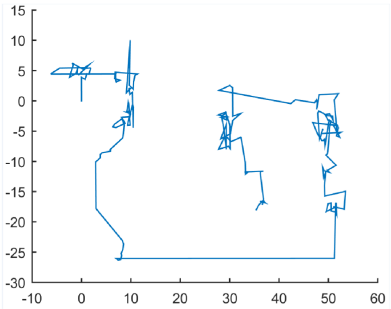

FIGURE 2. Possible Lévy flight path.

represented in the whole space search, in the search space of the problem [17].

CSA is based on three idealized rules as follows [2]:

- A cuckoo lays only one egg at a time and put it in a nest chosen randomly.
- The eggs that will pass to the next generation are the eggs with high quality.
- The number of available host nests is constant, and a cuckoo egg can be discovered by the host bird with probability $P_a \in [0, 1]$. If the cuckoo egg is discovered by the host, the host bird can throw it out of the nest or it simply leaves it and build another nest. For simplicity, this assumption is approximated by a fraction of the n nests are replaced by new random nests.

Based on the above three rules, the steps of CSA are illustrated in the pseudo code shown in Algorithm 1.

C. MATHEMATICAL REPRESENTATION OF STANDARD CUCKOO SEARCH

Based on Algorithm 1, CSA has three main components [34], [49]:

1) Exploitation using Lévy Flight Random Walk (LFRW).
2) Exploration using Biased/Selective Random Walk (BSRW).
3) Elitism scheme via Greedy Selection.

1) EXPLOITATION USING LÉVY FLIGHT RANDOM WALK (LFRW)

In the exploitation phase, new solutions are generated around the best solution obtained so far and this will increase the speed of local search. The exploitation in CSA is done via Lévy Flight Random Walk (LFRW) which is described as a general form in (1) in which the step size is drawn from the Lévy distribution.

$$X_{i,G+1} = X_{i,G} + \alpha \oplus Lévy(\beta) \tag{1}$$

where $X_{i,G}$ is the i^{th} nest in the G_{th} generation and $X_{i,G+1}$ is the new nest generated by Lévy flight. The product $\oplus$ means entry-wise multiplications. α is the step size where $\alpha > 0$ and is obtained by (2). This formula ensures that the newly generated solution will be close to the best-obtained solution so far.

$$\alpha = \alpha_0 \times (X_{i,G} - X_{best}) \tag{2}$$

where X_{best} represents the best-obtained solution so far.α_0 is a scaling factor and it is set to 0.01 in standard CSA [2], [50].

The term Lévy(β) is a random number, which is generated from a Lévy distribution and it is calculated from (3).

$$Lévy(\beta) \sim \frac{\phi \times \mu}{|\upsilon|^{\frac{1}{\beta}}} \tag{3}$$

where β is constant and is set at 1.5 as preferred by Yang and Deb [2] in standard CSA. μ and ν are random numbers drawn from a normal distribution with a mean of zero and standard deviation equals 1. The parameter ϕ is calculated as follows in (4).

$$\phi = \left(\frac{\Gamma(1+\beta) \times sin(\frac{\pi \times \beta}{2})}{\Gamma\left(\frac{1+\beta}{2} \times \beta \times 2^{\frac{\beta-1}{2}}\right)} \right)^{\frac{1}{\beta}} \tag{4}$$

where Γ is a gamma function. To sum up, the above equations can be combined to obtain the final form for LFRW as shown in (5).

$$X_{i,G+1} = X_{i,G} + \alpha_0 \frac{\phi \times \mu}{|\upsilon|^{\frac{1}{\beta}}} (X_{i,G} - X_{best}) \tag{5}$$

2) EXPLORATION USING BIASED/SELECTIVE RANDOM WALK (BSRW)

In the exploration phase, new solutions should be generated randomly in locations far from the current best solution, this will ensure the system will not be trapped in a local optimum and provide good diversity and exploration in the whole search space. The exploration step in CSA is performed using

Algorithm 1 Standard Cuckoo Search Algorithm (CSA)

```
initialize P_a = 0.25;
Objective function f(x), x = (x_1, ..., x_d)^d where d is number of dimensions;
Generate initial population of n hosts X_i (i = 1,2,...,n);
while g ≤ g_max or stopping criteria do
    Get a new cuckoo via Lévy flight;
    Evaluate its fitness F_i;
    Choose a nest randomly among n (say j );
    if F_i > F_j then
        Replace j by the new solution ;
    end
    A fraction P_a of worst nests are abandoned and new ones are generated;
    Keep the best solutions (or nests with quality solutions);
    Rank the solutions and keep the current best;
end
Display the results;
```

Biased/Selective Random Walk (BSRW) which is more efficient in exploring the whole search space especially when it is large because the step size in Lévy Flight is much longer in the long run [49]. BSRW is used to find new solutions far from the current best solution. First, a trial solution is generated using the mutation of the current solution and a differential step size from two randomly selected solutions. Then the new solution is generated by a crossover operator from the current and the trial solutions. BSRW can be formulated as follows in (6) [34].

$$x_{i,G+1} = \begin{cases} x_{i,G} + r \times (x_{m,j,G} - x_{n,j,G}) & with\ P_a \\ x_{i,G} & with\ remaining\ p_a \end{cases} \tag{6}$$

where m and n are random indexes, r is a random number on the range [0,1], is the probability of discovery (best value = 0.25) [2].

3) ELITISM SCHEME VIA GREEDY SELECTION

After each random walk process, CS selects the solutions with better fitness value using the greedy strategy to pass to the next generation. This will ensure that good solutions will not be lost. Any algorithm that has a good compromise of the above three components can lead to an efficient algorithm. Cuckoo search has a very good balance between the above three components. As previously explained, our main objective of this paper is to make the step size of Lévy Flight adaptive and hold a comparative study among all the modifications made on that aspect. The main parameter that controls the step size of Lévy Flight in (5) is the scaling factor α_0. The next section will discuss in details various modification that has been made on the scaling factor α_0 over the past decade.

III. RELATED WORK

This section explores all variants of cuckoo search that focus on adapting the step size in flight. Various pieces of research that are studying the concept of adaptive step size are concerned with making the value of α adaptive. This can be done by making the scaling factor α_0 inversely proportional to the current iteration (generation) number. So, in the early generations, the value of α will be large to explore more areas in search space. While, when the number of iterations increases, the value of the step size will be smaller and this will allow more exploitation in the search space as we get closer to the global optimal value. Through making the step size adaptive, this will introduce a suitable additional balance between exploration and exploitation. Over the past decade, most of the research on cuckoo search aims to make the step size of Lévy flight adaptive using different methods. This section will discuss these variants focusing on how they modify the step size of Lévy flight to be adaptive. This paper focuses only on various modifications concerning the step size. For a fair comparison, other modifications are ignored keeping all other parameters the same as the standard cuckoo search. This is due to our objective of tracking the optimal method for making an effective and powerful adaptive step size. Initially, Yang and Deb introduced the standard CSA in 2009 setting the step size of Lévy flight as a constant of $\alpha_0 = 0.01$ based on a study in 2014 [2], [50]. In 2011, Walton et al. modified the standard CSA and proposed a variant called Modified Cuckoo Search (MCS) [30]. They made two modifications, the first one was making the Lévy flight step size adaptive and the second modification was the addition of information exchange between eggs. We will focus on how to make the step size adaptive using (7).

$$\alpha_0 = A/\sqrt{G} \tag{7}$$

where G is the generation number and A is the initial value of Lévy step size and is set to 1. In 2013, Nasa-ngium et al. combined the MCS by Walton with a chaotic sequence and introduced an improved MCS with Chaotic Sequence algorithm (ICMCS). In this algorithm, the step size in a Lévy flight was adaptive using (7) [51]. One study by Valian et al. introduced a new variant of Cuckoo search called Improved Cuckoo search (ICS) in which the parameters α and P_a are adaptive and change their values with the current iteration number [31]. Valian made the Lévy flight step size adaptive using (8) and (9).

$$\alpha_0 = \alpha_{max}\ exp(c.gn) \tag{8}$$

$$c = \frac{1}{NI} Ln\left(\frac{\alpha_{min}}{\alpha_{max}}\right) \tag{9}$$

where α_{min} and α_{max} were the lower and the upper bounds of α respectively, NI was the maximum number of iterations and gn was the current iteration number. Zhang and Chen proposed a new Improved Cuckoo Search Algorithm with an adaptive method [32]. It is also called Improved Cuckoo Search (ICS2). They made one modification to the value of α_0 which is calculated from (10).

$$\alpha_0 = \frac{exp\left(\frac{ln\alpha_{min} - ln\alpha_{max}}{N_{max}} \times N\right)}{\sqrt[4]{N}} \tag{10}$$

where N is the current iteration number and it initially equals 1. N_{max} is the maximum number of iterations (generations).

Wang et al. proposed Cuckoo Search with varied scaling factor (VSF) [34]. VSF sets the value of α_0 to a random number drawn from the uniform distribution on the range [0,U] and they preferred to set U to 1. This random number is generated in each time a new nest is obtained via Lévy flight. This method can be formulated as shown in (11).

$$\alpha_0 = rand \tag{11}$$

In 2015, Ong et al. proposed an adaptive cuckoo search algorithm with two-parent crossover (ACSAC) [35]. They made two modifications, the first one was using two-parent cross over operator to allow the exchange of information between good solutions. While the second one was making the step size adaptive using (12).

$$\alpha_0 = \alpha_L(1 + \alpha_i\ tanh(\gamma/F^t_{best})/\sqrt{t}) \tag{12}$$

where α_L is the predefined minimum step size. α_0 is the initial step size. t the current iteration number. γ is the best fitness value in the initial population. *Fbest* is the best fitness in the current iteration.

In 2016, Chi et al. proposed an Adaptive cuckoo search (ACS) [37]. They made two modifications. The first one was making the discovery probability adaptive. The second one was making the step size of Lévy Flight adaptive based on (13).

$$\alpha_0 = (\alpha_{up} - \alpha_{low}) \times exp\left[\frac{t}{T} Ln\left(\frac{\alpha_{low}}{\alpha_{up}}\right)\right] + \alpha_{low} \quad (13)$$

where α_{low} and α_{up} were the lower and the upper bounds of α_0 respectively. T was the maximum number of iterations (generations) and t was the current iteration number. In the analysis of adaptive cuckoo search, Wang et al. proposed a novel cuckoo search with Chaos Theory and Elitism Scheme (CCS) [36]. They introduced chaotic maps to adapt the value of α_0.They used 12 chaotic maps and they proposed 12 variants of CCS which is illustrated in Table 1.

TABLE 1. Twelve chaotic cuckoo search variants with 12 chaotic map.

Variant Name	Map No.	Name
CCS1	M01	Chebyshev map
CCS2	M02	Circle map
CCS3	M03	Gaussian map
CCS4	M04	Intermittency map
CCS5	M05	Iterative map
CCS6	M06	Liebovitch map
CCS7	M07	Logistic map
CCS8	M08	Piecewise map
CCS9	M09	Sine map
CCS10	M10	Singer map
CCS11	M11	Sinusoidal map
CCS12	M12	Tent map

Each map will be normalized and their variations are always in the range [0,2], then it will be able to tune the step size α_0. CCS uses an Elitism scheme to protect the best-found solution. The experimental results in that research showed that $M11$ (Sinusoidal map) performed more effectively than the others. In 2016, Haung et al. enhanced the CCS by introducing the chaotic maps in three positions inside standard cuckoo search [52]. The first one was using chaotic maps in the initial population, the second one was using chaotic maps in handling the boundary and the third one was using chaotic maps to make the step size of Lévy flight adaptive in a similar manner like [36]. In 2017, Pandey et al. made a new variant based on the ICS which is called Hybrid Step Size Based Cuckoo Search (HSCS) [53]. In this algorithm, they used the property of Lévy flight and Gaussian distribution and they made the step size adaptive using (8) and (9).

In a recent study by Cheng et al., they proposed a new variant of Cuckoo Search called Cuckoo Search Algorithm with dynamic feedback Information (DFCS) [38]. They introduced two modifications to the standard cuckoo search. The first one was a dynamic selection between two selection schemes based on dynamic switching probability. The second one was a dynamic step size of Lévy flight in which the value of α_0 was dynamically tuned using the randomness and stability trend of the cloud model which is described in Algorithm 2.

Algorithm 2 Procedure of Obtaining α_0 in DFCS

```
for t ← 1 to maxIterations do
    calculate f^t_avg1, f^t_avg2 and f^t_best;
    for i ← 1 to N do
        if f^t_i < f^t_avg1 then
            α0 = αmin;
        end
        else if f^t_avg1 < f^t_i < f^t_avg2 then
            α0 is obtained from (14);
        end
        else
            α0 = αmax;
        end
    end
end
```

As indicated in Algorithm 2, N denotes the population size, f_i^t is the fitness value of the i^{th} individual in the t^{th} generation. f_{best}^t is the best fitness value obtained so far. f_{avg}^t is the mean fitness of the entire population, f_{avg1}^t is the mean fitness of solutions with fitness better than f_{avg}^t. f_{avg2}^t is the mean fitness of solutions with fitness worse than f_{avg}^t. Equations (14-18) are used to calculate α_0 as follows.

$$\alpha_0 = \alpha_{max} - (\alpha_{max} - \alpha_{min}) \times exp\left(\frac{(f_i^t - E_x)^2}{2(E_n')^2}\right) \quad (14)$$

$$E_x = f_{best}^t \quad (15)$$

$$E_n = (f_{avg1}^t - f_{best}^t)/c_1 \quad (16)$$

$$H_e = E_n/c_2 \quad (17)$$

$$E_n' = normrnd(E_n, H_e) \quad (18)$$

where E_x, E_n, and H_e are called Expectation, Entropy, and Hyper Entropy respectively.C_1 and C_2 are adjustment parameters which are set to 6 and 2 respectively. As previously pointed, our concern is focused on different variants of adaptive step size for CSA. Therefore, we implement only the modifications made to α_0 for each of the variants illustrated before, then a comparative study is conducted against our propped methods. The modifications done on the step size can be summarized in Table 2.

IV. THE PROPOSED VARIANTS OF CUCKOO SEARCH

A. DAMPED CUCKOO SEARCH

The main objective of this paper is to search for the most promising strategy of making an adaptive step size in Lévy flight. This can be achieved by decreasing the value of α_0 when the number of generations (iterations) increases. In the early generations, it is preferable for the step size to be large to explore more areas of the search space which allows more

TABLE 2. Summary of all the modifications done on Step Size of Lévy flight.

Variant Name	Equation
CSA [2]	$\alpha_0 = 0.01$
MCS [30]	$\alpha_0 = A/\sqrt{G}$
ICS [31]	$\alpha_0 = \alpha_{max}\ exp(c.gn)\ ,\ c = \frac{1}{NI} Ln\left(\frac{\alpha_{min}}{\alpha_{max}}\right)$
ICS2 [32]	$\alpha_0 = \frac{exp\left(\frac{ln\alpha_{min} - ln\alpha_{max}}{N_{max}} \times N\right)}{\sqrt[4]{N}}$
VSF [34]	$\alpha_0 = rand$
ACSAC [35]	$\alpha_0 = \alpha_L(1 + \alpha_i\ tanh(\gamma/F^t_{best})/\sqrt{t})$
ACS [37]	$\alpha_0 = (\alpha_{up} - \alpha_{low}) \times exp\left[\frac{t}{T} Ln\left(\frac{\alpha_{low}}{\alpha_{up}}\right)\right] + \alpha_{low}$
CCS [36]	α_0 is generated from sinusoidal chaotic map (M11)
DFCS [38]	$\alpha_0 = \alpha_{max} - (\alpha_{max} - \alpha_{min}) \times exp\left(\frac{(f^t_i - E_x)^2}{2(E'_n)^2}\right)$

exploration. However, by moving on through generations, the solutions inside the population are improving and become near to the optimal solution. Therefore, it is preferred to make the step size small in order to exploit the search space. This concept introduces a simple and good balance between exploration and exploitation. The proposed variant of CSA is inspired by the concept of damped oscillations having amplitude decreasing with time. The common Mass-Damper-Spring system motivates the initiation of the new variant of CSA through the solution of its differential equation as represented in (19).

$$x = e^{-\zeta\omega_n t}\left[\frac{\dot{x}_0 + \zeta\omega_n x_0}{\omega_d}\sin\omega_d t + x_0\cos\omega_d t\right] \quad (19)$$

where ζ is the damping ratio, ω_n is the natural undamped frequency, x_0 is the initial displacement, $\dot{x}_0$ is the initial velocity of the mass and $\omega_d = \omega_n.\sqrt{1-\zeta^2}$.

If the value of ζ equals zero it is called the undamped case, when the value of ζ lies between 0 and 1 this is called the underdamped case, and when $\zeta = 1$ it is called the critically damped case, when $\zeta > 1$ this is called the overdamped case. We adopt the concept of the damped oscillations to the parameter α_0 by introducing the proposed equation (20). Where α_{min}, α_{max} in (20) are the lower and upper bounds of α_0 respectively, τ represents the time constant, ω is the damped frequency, g is the current generation (iteration), and g_{max} is the maximum number of generations (iterations). Equation (21) introduces another equivalent form of (20).

$$\alpha_0 = \alpha_{min} + (\alpha_{max} - \alpha_{min})e^{-\left(\frac{\tau\ g}{gmax}\right)}\left[cos\left(\frac{\omega\ g}{gmax}\right) + \alpha_{min}\sin\left(\frac{\omega\ g}{gmax}\right)\right] \quad (20)$$

$$\alpha_0 = \alpha_{min} + (\alpha_{max} - \alpha_{min})e^{-\left(\frac{\tau\ g}{gmax}\right)}\left[cos\left(\frac{\omega\ g}{gmax}\right) - tan^{-1}(\alpha_{min})\right] \quad (21)$$

Different behaviors for the damped α_0 can be obtained by varying the values of τ and, ω. Setting the value of $\tau = 5$ and the values of ω as 1, 2, 3, 4, 5, 30 results in different curves of α_0 as shown in Fig. 3 for $gmax = 6000$.

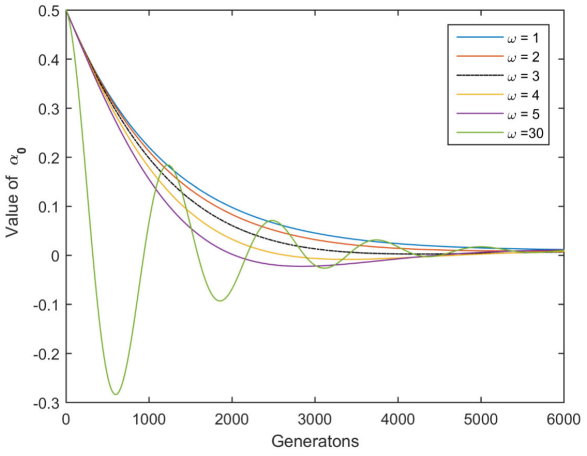

FIGURE 3. Variation of α_0 for 6000 generations under different values of ω using the proposed equation (20).

As shown in Fig. 3 the value of ω has a great effect on the behavior of the proposed equation. When the value of ω is very large (i.e 30), the value of α_0 is changed in an under-damped oscillation manner. For small values of ω (i.e $1, 2, \ldots, 5$) the value of α_0 is gradually decreasing in a smooth manner without oscillations, and the larger the value of ω the faster decay of the value of α_0. In the proposed DCS algorithm, we set the value of $\tau = 5$ and the value of $\omega = 3$ to avoid any possibility of oscillating behavior as well as a possible fast decay in the value of α_0. Algorithm 3 illustrates the proposed DCS Algorithm. The values of α_{min} and α_{max}

Algorithm 3 Damped Cuckoo Search

```
initialize α_min = 0.01, α_max = 0.5, P_a = 0.25, τ = 5, ω = 3 ;
Objective function f(x), x = (x_1, ..., x_d)^d where d is number of dimensions;
Generate initial population of n hosts X_i (i = 1,2,...,n);
while g ≤ g_max or stopping criteria do
    obtain value of α_0 using the proposed equation(20);
    Get a new cuckoo via Lévy flight using the calculated α_0 using (5);
    Evaluate its fitness F_i;
    Choose a nest randomly among n (say j );
    if F_i > F_j then
        Replace j by the new solution ;
    end
    P_a of worst nests are abandoned and new ones are generated using (6);
    Keep best solution;
end
Display results;
```

TABLE 3. Maps that are used in the proposed MCCSA-P and MCCSA-S.

Map No.	Map Name
M1	Chebyshev
M2	Circle
M3	Gaussian
M4	Iterative
M5	Logistic
M6	Piecewise
M7	Sine
M8	Singer
M9	Sinusoidal
M10	Tent

are recommended to be at 0.01 and 0.5 respectively [31]. The value of P_a is preferred to be 0.25 [2].

B. MULTIPLE CHAOTIC CUCKOO SEARCH FOR ADAPTIVE STEP SIZE (PARALLEL-BASED) (MCCSA-P)

Ten chaotic maps as shown in Table 3 are used to get 10 different values of α_0, then Lévy Flight is used to generate 10 candidate solutions. Thereafter we will choose the best one of these 10 candidate solutions with the best fitness value. This can be formulated in (22) and (23).

$$X^j_{i,G+1} = X_{i,G} + \alpha^j_0 \frac{\phi \times \mu}{|\upsilon|^{\frac{1}{\beta}}}(X_{i,G} - X_{best}), \quad where\ j = 1, 2, \ldots, 10. \tag{22}$$

$$X_{i,G+1} = argmin\{f(X^j_{i,G+1})\} \tag{23}$$

$X^j_{i,G+1}$ in (22) is the candidate solution generated by Lévy flight via j^{th} chaotic map, α^j_0 is the value of α_0 generated by the j^{th} chaotic map. $X_{i,G+1}$ in (23) is the best candidate solution with the best fitness value. This variant is inspired by [39]. However, it is adapted to the α_0 parameter instead of the chaotic search radius parameter in the chaotic local search equation.

C. MULTIPLE CHAOTIC CUCKOO SEARCH FOR ADAPTIVE STEP SIZE (SUCCESS RATE-BASED) (MCCSA-S)

Based on the previous chaotic maps illustrated in Table 3, only one chaotic map from the 10 maps is to be selected based on the success and failure rates. The steps of MCCSA-S are illustrated in pseudo-code in Algorithm 4. The probability of selecting the chaotic map in the generation is calculated from (24). The success rate of the i^{th} chaotic map in the g^{th} generation is obtained from (25) in which ϵ is set to 0.01 to avoid the possible null success rates.

$$p_{g,i} = \frac{S_{g,i}}{\sum_{i=1}^{10} S_{g,i}} \tag{24}$$

$$S_{g,i} = \frac{\sum_{t=g-LP}^{t=g-1} ns(g,i)}{\sum_{t=g-LP}^{t=g-1} ns(g,i) + \sum_{t=g-LP}^{t=g-1} nf(g,i)} + \epsilon \tag{25}$$

where $i = 1, 2, .., 10$; $g > LP$

Algorithm 4 Multiple Chaotic Cuckoo Search Algorithm - Success Rate Based MCCSA-S

```
initialize LP = 50, Pa = 0.25;
Objective function f(x), x = (x_1, ..., x_d)^d where d is number of dimensions;
Generate initial population of n hosts X_i (i = 1,2,...,n);
define two arrays ns_{LP*10} and nf_{LP*10};
while g ≤ g_max or stopping criteria do
    if g ≤ LP then
        for i ← 1 to 10 do
            generate a new population of n solutions using the i^th chaotic map using (22);
            find the best fitness f_new of the new n solutions generated in the previous step;
            if f_new < f_min then
                ns(g,i) = 1;
                nf(g,i) = 0;
            end
            else
                ns(g,i) = 0;
                nf(g,i) = 1;
            end
        end
    end
    else
        calculate the success rate for each chaotic map using (25);
        calculate the selection probability for each chaotic map using (24);
        select one of the 10 maps based on the calculated probability, say map k;
        generate n solutions using the k^th selected map using (22);
        find the best fitness f_new of the new n solutions generated in the previous step;
        if f_new < f_min then
            ns(g % LP,k) = 1;
            nf(g % LP,k) = 0;
        end
        else
            ns(g % LP,k) = 0;
            nf(g % LP,k) = 1;
        end
    end
    Pa of worst nests are abandoned and new ones are generated from (6);
    Keep best solution and best fitness f_min;
end
Display results;
```

V. PERFORMANCE MEASUREMENT AND PARAMETER SETTINGS

A. PARAMETER SETTING FOR ALGORITHMS

As previously illustrated, our objective is to track the most promising variant of the CSA based on different adaptive step

size strategies. Keeping other parameters of the CSA variants fixed to the standard while varying only the step size of Lévy flight is our main idea. Nine different variants of CSA are evaluated and compared to the three proposed variants of CSA discussed previously in section IV. For all algorithms, the population size (n) is set to 25 and the probability of discovery P_a is set to 0.25. The other parameters for each algorithm are set according to the recommendations made by the authors for each algorithm as summarized in Table 4.

TABLE 4. The parameter settings for all algorithms.

Variant Name	Parameter Settings
CSA [2]	$\alpha_0 = 0.01$
MCS [30]	$A = 1$
CCS [36]	α_0 is initialized randomly, rescale sinusoidal map to the range of [0,2].
ICS [31]	$\alpha_{min} = 0.01, \alpha_{max} = 0.5, NI = 6000$
VSF [34]	No special parameters.
ICS2 [32]	$\alpha_{min} = 0.1, \alpha_{max} = 1, N_{max} = 6000$
ACSAC [35]	$\alpha_L = 0.1, \alpha_i = 1$
ACS [37]	$\alpha_{low} = 0.01, \alpha_{up} = 0.5, T = 6000$
MCCSA-P	Initial α_0 randomly, All maps are rescaled to the range of [0,2].
MCCSA-S	$LP = 50$, α_0 is initialized randomly, All maps are rescaled to the range of [0,2].
DCS	$\alpha_{min} = 0.1, \alpha_{max} = 0.5, \omega = 3, \tau = 5, gmax = 6000$

B. BENCHMARK FUNCTIONS CEC2017

All the algorithms in Table 4 are tested on CEC2017 benchmark functions. According to [54], CEC2017 contains 30 minimization problems which consist of 3 unimodal functions (F1-F3), 7 simple multimodal functions (F4-F10), 10 hybrid functions (F11-F20), and 10 composition functions (F21-F30). These functions can be summarized in Table 5. The objective function can be formulated as a minimization problem in which the error function is calculated from (26).

$$error = f_i(x) - f_i(x^*) \tag{26}$$

where $f_i(x)$ is the value of the i^{th} function obtained by the algorithm and $f_i(x^*)$ is the optimal value of the i^{th} function. The number of dimensions nd for these functions can be 10, 30, 50 and 100. The dimensions are set at 30 in all of the implemented analysis. The algorithms are conducted for 51 runs. All problems have the global optimum in the bounds $[-100, 100]^{nd}$ of and there is no need to search outside this given range The algorithm terminates when it reaches the maximum function evaluations or the error value (fitness value) is smaller than $1.0E-08$. Maximum function evaluations ($MaxFEs$) $= 10000 * nd$ (i.e. $MaxFEs = 300000$.). The parameter settings for these functions are illustrated in Table 6.

C. PERFORMANCE METRICS

We calculate the mean, standard deviation, best, worst, median values for the best fitness obtained in the 51 runs. Error values smaller than $1.0E-08$ will be considered as zero. Then the Friedman test is performed on the obtained results to compare all algorithms and to decide which one is the most promising in the adaptation of the step size. Then four post-hoc procedures are applied to find the adjusted p-values and perform paired comparisons between the proposed DCS and all other algorithms. The Box Plots for each function are illustrated as well to observe the distribution of the best fitness

TABLE 5. Summary of CEC2017 benchmark functions [54].

Type	Number	Function Name	$f_i(x^*)$
Unimodal	1	Shifted and Rotated Bent Cigar Function	100
	2	Shifted and Rotated Sum of Different Power Function*	200
	3	Shifted and Rotated Zakharov Function	300
Multimodal	4	Shifted and Rotated Rosenbrock's Function	400
	5	Shifted and Rotated Rastrigin's Function	500
	6	Shifted and Rotated Expanded Scaffer's F6 Function	600
	7	Shifted and Rotated Lunacek Bi_Rastrigin Function	700
	8	Shifted and Rotated Non-Continuous Rastrigin's Function	800
	9	Shifted and Rotated Levy Function	900
	10	Shifted and Rotated Schwefel's Function	1000
Hybrid	11	Hybrid Function 1 (N=3)	1100
	12	Hybrid Function 2 (N=3)	1200
	13	Hybrid Function 3 (N=3)	1300
	14	Hybrid Function 4 (N=4)	1400
	15	Hybrid Function 5 (N=4)	1500
	16	Hybrid Function 6 (N=4)	1600
	17	Hybrid Function 6 (N=5)	1700
	18	Hybrid Function 6 (N=5)	1800
	19	Hybrid Function 6 (N=5)	1900
	20	Hybrid Function 6 (N=6)	2000
Composition	21	Composition Function 1 (N=3)	2100
	22	Composition Function 2 (N=3)	2200
	23	Composition Function 3 (N=4)	2300
	24	Composition Function 4 (N=4)	2400
	25	Composition Function 5 (N=5)	2500
	26	Composition Function 6 (N=5)	2600
	27	Composition Function 7 (N=6)	2700
	28	Composition Function 8 (N=6)	2800
	29	Composition Function 9 (N=3)	2900
	30	Composition Function 10 (N=3)	3000

TABLE 6. Summary of parameter settings for CEC2017 benchmark functions.

Parameter	Value
Objective Function	$error = f_i(x) - f_i(x^*)$
Number of runs	51
Number of dimensions (nd)	30
Search Range	$[-100, 100]^{nd}$
Maximum Function Evaluations (MaxFEs)	$10000 * nd = 300000$
Tolerance	$1.0E-08$

values obtained for the 51 runs. Finally, the convergence graphs for each function are plotted. All these statistical tests and graphs are described in detail in the following sections.

VI. SIMULATION RESULTS AND STATISTICAL ANALYSIS

A. SIMULATION RESULTS

As previously mentioned, 51 runs are conducted on each of the 30 functions of CEC2017 for each algorithm to record the best fitness value in each run. Table 7 shows the results after performing the 13 variants of CSA that modify α_0 with different strategies including the proposed methods. We notice that all algorithms reach a total number of generations of 6000, this is because two function evaluations are performed for each nest in each iteration (one in Lévy 's flight phase and the other in the biased random walk phase). Therefore, for the assumption of our population size (25 nests), the total number of function evaluations in each run equals 50, dividing the 300000 maximum function evaluations by 50 we obtain 6000 generations (iterations). This is true for all algorithms except MCCSA-P which performs 10 function evaluations for each nest in Lévy's flight phase and only one function evaluation in the biased random walk so each nest face 11 function evaluations in each iteration. Therefore, the total number of function evaluations for all the nests equals 25 * 11 = 275, resulting in the number of total generations generated in MCCSA-P equals 300000/275 = 1090 generations which is much lower compared to the other

TABLE 7. Best, worst, median, mean, and standard deviation results of all algorithms for 51 runs for all CEC2017 benchmark functions.

Beginning of Table VII						
Fn.	Algorithm	best	worst	median	mean	STD
F1	CSA [2]	1.0000E+10	1.0000E+10	1.0000E+10	1.0000E+10	0.0000E+00
	MCS [30]	1.0000E+10	1.0000E+10	1.0000E+10	1.0000E+10	0.0000E+00
	CCS [36]	1.0000E+10	1.0000E+10	1.0000E+10	1.0000E+10	0.0000E+00
	ICS [31]	1.0000E+10	1.0000E+10	1.0000E+10	1.0000E+10	0.0000E+00
	VSF [34]	1.0000E+10	1.0000E+10	1.0000E+10	1.0000E+10	0.0000E+00
	ICS2 [32]	1.0000E+10	1.0000E+10	1.0000E+10	1.0000E+10	0.0000E+00
	ACSAC [35]	1.0000E+10	1.0000E+10	1.0000E+10	1.0000E+10	0.0000E+00
	ACS [37]	1.0000E+10	1.0000E+10	1.0000E+10	1.0000E+10	0.0000E+00
	DFCS [38]	1.0000E+10	1.0000E+10	1.0000E+10	1.0000E+10	0.0000E+00
	MCCSA-P1	1.0000E+10	1.0000E+10	1.0000E+10	1.0000E+10	0.0000E+00
	MCCSA-P2	1.0000E+10	1.0000E+10	1.0000E+10	1.0000E+10	0.0000E+00
	MCCSA-S	1.0000E+10	1.0000E+10	1.0000E+10	1.0000E+10	0.0000E+00
	DCS	1.0000E+10	1.0000E+10	1.0000E+10	1.0000E+10	0.0000E+00
F2	CSA [2]	1.0000E+10	1.0000E+10	1.0000E+10	1.0000E+10	0.0000E+00
	MCS [30]	1.0000E+10	1.0000E+10	1.0000E+10	1.0000E+10	0.0000E+00
	CCS [36]	1.0000E+10	1.0000E+10	1.0000E+10	1.0000E+10	0.0000E+00
	ICS [31]	1.0000E+10	1.0000E+10	1.0000E+10	1.0000E+10	0.0000E+00
	VSF [34]	1.0000E+10	1.0000E+10	1.0000E+10	1.0000E+10	0.0000E+00
	ICS2 [32]	1.0000E+10	1.0000E+10	1.0000E+10	1.0000E+10	0.0000E+00
	ACSAC [35]	1.0000E+10	1.0000E+10	1.0000E+10	1.0000E+10	0.0000E+00
	ACS [37]	1.0000E+10	1.0000E+10	1.0000E+10	1.0000E+10	0.0000E+00
	DFCS [38]	1.0000E+10	1.0000E+10	1.0000E+10	1.0000E+10	0.0000E+00
	MCCSA-P1	1.0000E+10	1.0000E+10	1.0000E+10	1.0000E+10	0.0000E+00
	MCCSA-P2	1.0000E+10	1.0000E+10	1.0000E+10	1.0000E+10	0.0000E+00
	MCCSA-S	1.0000E+10	1.0000E+10	1.0000E+10	1.0000E+10	0.0000E+00
	DCS	1.0000E+10	1.0000E+10	1.0000E+10	1.0000E+10	0.0000E+00
F3	CSA [2]	4.3180E+03	2.1528E+04	7.7651E+03	9.0644E+03	3.8696E+03
	MCS [30]	1.6404E+03	9.5730E+03	4.9436E+03	5.1842E+03	1.8550E+03
	CCS [36]	2.8861E+04	8.1854E+04	5.0126E+04	5.1921E+04	1.2736E+04
	ICS [31]	9.7509E+02	7.7742E+03	3.7753E+03	3.9011E+03	1.7283E+03
	VSF [34]	3.1639E+03	1.2576E+04	6.9419E+03	7.0451E+03	2.1885E+03
	ICS2 [32]	1.2056E+03	7.2985E+03	3.6880E+03	3.7202E+03	1.2623E+03
	ACSAC [35]	9.8717E+02	5.7232E+03	2.7118E+03	2.9001E+03	1.2142E+03
	ACS [37]	9.9438E+02	7.7032E+03	3.2915E+03	3.5822E+03	1.5514E+03
	DFCS [38]	1.7787E+03	1.1807E+04	6.0353E+03	6.5575E+03	2.6230E+03
	MCCSA-P1	5.2639E+04	1.2823E+05	8.7731E+04	8.7834E+04	1.5645E+04
	MCCSA-P2	**9.0469E+02**	**5.5516E+03**	**2.3857E+03**	**2.5320E+03**	**9.6360E+02**
	MCCSA-S	1.3673E+04	5.4644E+04	3.2827E+04	3.2905E+04	9.7086E+03
	DCS	1.5471E+03	1.3920E+04	5.4832E+03	5.6340E+03	2.3203E+03
F4	CSA [2]	**7.7098E-05**	7.4966E+01	2.1376E+01	3.0809E+01	2.5626E+01
	MCS [30]	3.6360E-02	7.5914E+01	1.9210E+01	2.3440E+01	1.9053E+01
	CCS [36]	2.4046E-01	7.7944E+01	1.7483E+01	2.5753E+01	2.1434E+01
	ICS [31]	5.9678E+00	7.3477E+01	1.6379E+01	1.9324E+01	1.1448E+01
	VSF [34]	4.0064E+00	7.7010E+01	1.8054E+01	2.0306E+01	1.2813E+01
	ICS2 [32]	1.2288E+01	7.3891E+01	1.7927E+01	2.4338E+01	1.7820E+01
	ACSAC [35]	1.2434E-04	7.1977E+01	1.8436E+01	2.1067E+01	2.0878E+01
	ACS [37]	2.2643E+00	7.1943E+01	1.6561E+01	2.1831E+01	1.6794E+01
	DFCS [38]	1.7302E-02	7.6186E+01	2.2226E+01	3.2967E+01	2.7278E+01
	MCCSA-P1	2.2173E+01	8.4173E+01	2.7437E+01	3.2992E+01	1.6868E+01
	MCCSA-P2	3.2729E-01	**2.6482E+01**	**1.5099E+01**	1.5669E+01	3.4839E+00
	MCCSA-S	6.6531E+00	7.1774E+01	1.6537E+01	**1.7518E+01**	**8.4751E+00**
	DCS	8.4099E-03	7.1043E+01	1.6359E+01	1.8216E+01	9.4432E+00
F5	CSA [2]	9.7732E+01	2.2594E+02	1.4532E+02	1.4532E+02	2.6327E+01
	MCS [30]	3.3927E+01	1.3485E+02	8.2691E+01	8.1520E+01	1.5066E+01
	CCS [36]	6.3472E+01	1.5425E+02	1.1403E+02	1.1408E+02	2.0463E+01
	ICS [31]	**3.0076E+01**	7.8918E+01	5.6669E+01	5.6511E+01	1.1353E+01
	VSF [34]	4.3730E+01	1.1912E+02	7.8782E+01	7.8944E+01	1.5892E+01
	ICS2 [32]	3.7011E+01	**7.2250E+01**	**5.4736E+01**	5.4770E+01	**9.0356E+00**
	ACSAC [35]	4.6772E+01	1.4027E+02	9.9773E+01	9.9234E+01	2.0882E+01
	ACS [37]	3.0457E+01	7.6610E+01	5.5777E+01	5.5864E+01	1.0032E+01
	DFCS [38]	8.6257E+01	1.6350E+02	1.2446E+02	1.2258E+02	1.7249E+01
	MCCSA-P1	7.0062E+01	1.7063E+02	1.3609E+02	1.3638E+02	1.8155E+01
	MCCSA-P2	4.5312E+01	9.8722E+01	7.0287E+01	7.1709E+01	1.2976E+01
	MCCSA-S	5.6162E+01	1.1276E+02	8.7265E+01	8.4929E+01	1.5867E+01
	DCS	3.3210E+01	7.5851E+01	5.5126E+01	**5.4507E+01**	1.0143E+01

TABLE 7. *(Continued.)* **Best, worst, median, mean, and standard deviation results of all algorithms for 51 runs for all CEC2017 benchmark functions.**

Continuation of Table VII						
Fn.	Algorithm	best	worst	median	mean	STD
F6	CSA [2]	1.8455E+01	5.5089E+01	3.2923E+01	3.4713E+01	9.9242E+00
	MCS [30]	1.0659E+00	1.4984E+01	5.6332E+00	6.0369E+00	3.0968E+00
	CCS [36]	6.9706E-03	5.2049E+00	1.8508E-01	6.8678E-01	1.0985E+00
	ICS [31]	1.8593E-07	4.2158E-04	5.5201E-06	2.5154E-05	7.1941E-05
	VSF [34]	**7.3493E-09**	1.4895E-02	9.8566E-09	2.9228E-04	2.0857E-03
	ICS2 [32]	4.6684E-04	1.3379E-01	1.6328E-02	2.1944E-02	2.3883E-02
	ACSAC [35]	3.6271E-06	2.8979E-02	8.6207E-04	3.0178E-03	5.7170E-03
	ACS [37]	9.2549E-09	4.8539E-04	4.9349E-07	1.8897E-05	7.2120E-05
	DFCS [38]	1.0768E+01	4.7104E+01	1.9926E+01	2.1191E+01	7.8753E+00
	MCCSA-P1	3.9802E-01	2.5884E+00	1.0360E+00	1.0428E+00	4.6496E-01
	MCCSA-P2	7.8320E-09	**9.9926E-09**	**9.6701E-09**	**9.3822E-09**	**6.1921E-10**
	MCCSA-S	7.6791E-04	7.0949E-01	4.0553E-02	9.7691E-02	1.3622E-01
	DCS	2.5348E-04	4.6628E-02	7.6633E-03	1.1470E-02	1.1665E-02
F7	CSA [2]	1.0786E+02	2.6811E+02	1.9858E+02	2.0322E+02	3.1261E+01
	MCS [30]	8.1499E+01	1.5345E+02	1.1687E+02	1.1884E+02	1.6186E+01
	CCS [36]	1.2728E+02	2.2166E+02	1.6997E+02	1.6870E+02	1.9619E+01
	ICS [31]	7.0331E+01	1.1630E+02	9.0075E+01	8.9436E+01	1.0190E+01
	VSF [34]	9.2384E+01	1.6678E+02	1.2805E+02	1.2758E+02	1.5343E+01
	ICS2 [32]	6.8598E+01	**1.0863E+02**	8.5196E+01	8.7172E+01	1.0213E+01
	ACSAC [35]	9.0041E+01	1.7685E+02	1.3468E+02	1.3644E+02	1.8958E+01
	ACS [37]	**5.8271E+01**	1.2976E+02	9.1697E+01	9.1174E+01	1.3143E+01
	DFCS [38]	1.2303E+02	2.1365E+02	1.6290E+02	1.6096E+02	2.1350E+01
	MCCSA-P1	1.1192E+02	2.1782E+02	1.8511E+02	1.8345E+02	2.2463E+01
	MCCSA-P2	7.8882E+01	1.4587E+02	1.1050E+02	1.1349E+02	1.6077E+01
	MCCSA-S	1.0034E+02	1.7333E+02	1.3721E+02	1.3943E+02	1.5871E+01
	DCS	6.2622E+01	1.1087E+02	**8.1304E+01**	**8.2739E+01**	**8.5062E+00**
F8	CSA [2]	8.2528E+01	1.9505E+02	1.4325E+02	1.3483E+02	2.2439E+01
	MCS [30]	5.7568E+01	1.0910E+02	8.1219E+01	8.2768E+01	1.2023E+01
	CCS [36]	6.9560E+01	1.4313E+02	1.0901E+02	1.0882E+02	1.9173E+01
	ICS [31]	3.4965E+01	**7.8233E+01**	5.6941E+01	5.7187E+01	1.0273E+01
	VSF [34]	4.6354E+01	1.1608E+02	8.2210E+01	8.0737E+01	1.5236E+01
	ICS2 [32]	3.6996E+01	9.1142E+01	5.5314E+01	5.7538E+01	1.0393E+01
	ACSAC [35]	5.2394E+01	1.6403E+02	9.1017E+01	9.6627E+01	2.2494E+01
	ACS [37]	3.1448E+01	9.0225E+01	5.8458E+01	5.8181E+01	1.2539E+01
	DFCS [38]	6.3700E+01	1.5654E+02	1.1450E+02	1.1313E+02	1.8524E+01
	MCCSA-P1	1.0240E+02	1.6862E+02	1.3672E+02	1.3641E+02	1.6718E+01
	MCCSA-P2	4.2206E+01	1.0725E+02	7.5044E+01	7.4514E+01	1.4140E+01
	MCCSA-S	6.4074E+01	1.1749E+02	8.7775E+01	8.9396E+01	1.3169E+01
	DCS	**2.8954E+01**	8.0101E+01	**5.1891E+01**	**5.2763E+01**	**1.0124E+01**
F9	CSA [2]	1.2135E+03	8.1015E+03	3.2976E+03	3.6769E+03	1.5542E+03
	MCS [30]	2.1640E+02	1.6852E+03	7.5752E+02	8.4029E+02	4.1703E+02
	CCS [36]	8.8608E+00	1.7203E+03	2.5685E+02	4.0092E+02	4.4960E+02
	ICS [31]	8.4576E-09	1.1673E+02	5.4865E+00	1.6969E+01	2.8399E+01
	VSF [34]	9.1063E-09	3.0296E+02	1.8657E+01	3.4446E+01	5.2178E+01
	ICS2 [32]	1.4120E+01	4.6232E+02	8.3560E+01	1.1043E+02	9.3086E+01
	ACSAC [35]	8.0882E+00	2.2573E+03	1.4827E+02	3.1423E+02	4.0898E+02
	ACS [37]	8.2671E-09	**3.8518E+01**	**4.5443E-01**	**4.6052E+00**	**8.3346E+00**
	DFCS [38]	8.3211E+02	5.7168E+03	2.7207E+03	2.7987E+03	1.1598E+03
	MCCSA-P1	1.0294E+01	9.8317E+02	1.8111E+02	2.2958E+02	1.9856E+02
	MCCSA-P2	**8.2487E-09**	1.9567E+02	2.8802E+00	2.6789E+01	4.7113E+01
	MCCSA-S	1.3268E-06	9.7148E+02	5.7903E+01	1.2103E+02	1.6636E+02
	DCS	9.8744E-09	1.5446E+02	1.3954E+01	2.2412E+01	2.8313E+01
F10	CSA [2]	3.3057E+03	4.5092E+03	3.8800E+03	3.8654E+03	2.5224E+02
	MCS [30]	2.9238E+03	3.9059E+03	3.4177E+03	3.4131E+03	**2.3536E+02**
	CCS [36]	4.3623E+03	6.3147E+03	5.7138E+03	5.5592E+03	5.1447E+02
	ICS [31]	2.7733E+03	4.5612E+03	3.8162E+03	3.8031E+03	3.7451E+02
	VSF [34]	3.8232E+03	5.2111E+03	4.5323E+03	4.5631E+03	3.2341E+02
	ICS2 [32]	3.0650E+03	4.3448E+03	3.7247E+03	3.7509E+03	2.9750E+02
	ACSAC [35]	3.3061E+03	5.3529E+03	4.7035E+03	4.7178E+03	3.3746E+02
	ACS [37]	3.4878E+03	5.1650E+03	4.3405E+03	4.2975E+03	3.6634E+02
	DFCS [38]	3.0062E+03	4.2130E+03	3.7224E+03	3.6940E+03	3.2135E+02
	MCCSA-P1	2.3288E+03	6.0904E+03	4.4810E+03	4.2981E+03	1.0314E+03
	MCCSA-P2	**1.2571E+03**	4.8241E+03	3.4060E+03	3.3479E+03	7.7739E+02
	MCCSA-S	1.9454E+03	5.5573E+03	4.1921E+03	4.1073E+03	9.0210E+02
	DCS	2.4416E+03	**3.8194E+03**	**3.2097E+03**	**3.1927E+03**	2.9725E+02

TABLE 7. ***(Continued.)*** **Best, worst, median, mean, and standard deviation results of all algorithms for 51 runs for all CEC2017 benchmark functions.**

Continuation of Table VII						
Fn.	Algorithm	best	worst	median	mean	STD
F11	CSA [2]	2.7511E+01	1.5058E+02	7.8439E+01	7.5732E+01	2.3744E+01
	MCS [30]	3.4699E+01	1.1007E+02	6.4702E+01	6.8303E+01	1.9657E+01
	CCS [36]	3.7816E+01	1.6161E+02	7.3262E+01	7.7596E+01	2.3209E+01
	ICS [31]	1.7963E+01	**8.7008E+01**	4.4059E+01	4.9971E+01	2.1991E+01
	VSF [34]	1.2052E+01	1.2393E+02	8.1129E+01	7.2648E+01	2.9864E+01
	ICS2 [32]	1.9895E+01	1.1326E+02	**3.8919E+01**	5.0093E+01	2.2878E+01
	ACSAC [35]	1.4874E+01	1.3187E+02	7.9982E+01	7.7102E+01	2.8963E+01
	ACS [37]	**1.0816E+01**	9.6901E+01	4.2375E+01	4.8584E+01	2.4281E+01
	DFCS [38]	2.9657E+01	1.2067E+02	6.5805E+01	6.6823E+01	2.0053E+01
	MCCSA-P1	4.2111E+01	1.7947E+02	1.2317E+02	1.2481E+02	2.5087E+01
	MCCSA-P2	1.4632E+01	1.2868E+02	7.8542E+01	6.7369E+01	3.1764E+01
	MCCSA-S	3.3371E+01	1.3093E+02	9.1768E+01	8.7415E+01	2.5679E+01
	DCS	1.8174E+01	9.9879E+01	4.3647E+01	**4.8416E+01**	**1.9515E+01**
F12	CSA [2]	6.6693E+03	1.0000E+10	1.0000E+10	5.9107E+09	4.9384E+09
	MCS [30]	8.7941E+03	1.0000E+10	1.0000E+10	5.7304E+09	4.9554E+09
	CCS [36]	1.9197E+04	1.0000E+10	1.0000E+10	6.7790E+09	4.6670E+09
	ICS [31]	**3.9448E+03**	1.0000E+10	1.0000E+10	5.2964E+09	5.0386E+09
	VSF [34]	6.3186E+03	1.0000E+10	6.2222E+08	4.5911E+09	4.9661E+09
	ICS2 [32]	5.6818E+03	1.0000E+10	1.0000E+10	6.0804E+09	4.9285E+09
	ACSAC [35]	5.3509E+03	1.0000E+10	1.0000E+10	7.1431E+09	4.4461E+09
	ACS [37]	5.1048E+03	1.0000E+10	1.0000E+10	5.1161E+09	5.0318E+09
	DFCS [38]	1.7371E+04	1.0000E+10	1.0000E+10	7.0975E+09	4.5107E+09
	MCCSA-P1	1.0288E+06	1.0000E+10	1.0000E+10	5.5673E+09	4.9522E+09
	MCCSA-P2	9.4711E+03	1.0000E+10	**6.0900E+04**	**1.5696E+09**	**3.6725E+09**
	MCCSA-S	3.8416E+04	1.0000E+10	1.0000E+10	7.0692E+09	4.5860E+09
	DCS	5.0426E+03	1.0000E+10	1.0000E+10	5.7499E+09	4.9251E+09
F13	CSA [2]	1.3663E+02	3.6095E+08	3.2908E+02	7.0886E+06	5.0541E+07
	MCS [30]	1.1542E+02	1.0000E+10	3.5288E+02	1.3726E+09	3.4754E+09
	CCS [36]	4.0542E+02	1.0000E+10	2.0191E+03	3.9268E+08	1.9603E+09
	ICS [31]	7.8079E+01	1.0000E+10	3.1638E+02	2.0053E+08	1.3998E+09
	VSF [34]	1.1075E+02	**1.8703E+04**	2.7876E+02	**1.1021E+03**	**2.8294E+03**
	ICS2 [32]	1.4538E+02	1.0000E+10	3.1439E+02	6.0128E+08	2.3749E+09
	ACSAC [35]	1.3777E+02	1.0000E+10	3.5146E+02	5.8824E+08	2.3764E+09
	ACS [37]	**5.4894E+01**	1.0000E+10	2.8726E+02	2.3867E+08	1.4270E+09
	DFCS [38]	1.2695E+02	1.0000E+10	2.6465E+02	1.1765E+09	3.2540E+09
	MCCSA-P1	4.1916E+02	2.9442E+09	1.5811E+04	1.0484E+08	5.2623E+08
	MCCSA-P2	1.2012E+02	2.9346E+04	**2.5616E+02**	2.0532E+03	5.5616E+03
	MCCSA-S	2.0791E+02	1.0000E+10	1.2800E+03	5.9011E+08	2.3759E+09
	DCS	1.1521E+02	1.0000E+10	3.1630E+02	1.9608E+08	1.4003E+09
F14	CSA [2]	3.6181E+01	9.5107E+01	6.2855E+01	6.5191E+01	1.1690E+01
	MCS [30]	4.3791E+01	8.9485E+01	6.0650E+01	6.1558E+01	1.1489E+01
	CCS [36]	4.3484E+01	7.9089E+01	5.7914E+01	5.8433E+01	7.4144E+00
	ICS [31]	**3.1192E+01**	**6.1108E+01**	**4.8975E+01**	**4.9077E+01**	6.7485E+00
	VSF [34]	3.9449E+01	7.8515E+01	5.8264E+01	5.8914E+01	1.0314E+01
	ICS2 [32]	3.5975E+01	6.7936E+01	5.3062E+01	5.3142E+01	7.5124E+00
	ACSAC [35]	4.0644E+01	7.5960E+01	5.6672E+01	5.7447E+01	9.0361E+00
	ACS [37]	3.8755E+01	6.6520E+01	4.9802E+01	5.0645E+01	**6.2620E+00**
	DFCS [38]	3.9803E+01	1.0399E+02	5.9448E+01	6.0302E+01	1.0945E+01
	MCCSA-P1	9.5497E+01	1.7623E+02	1.3460E+02	1.3618E+02	1.8775E+01
	MCCSA-P2	3.6148E+01	6.9772E+01	5.5556E+01	5.5377E+01	7.5877E+00
	MCCSA-S	3.8525E+01	7.7192E+01	5.7937E+01	5.8082E+01	8.4062E+00
	DCS	3.6038E+01	7.1604E+01	5.1824E+01	5.1762E+01	7.6492E+00
F15	CSA [2]	2.9581E+01	1.3151E+02	6.7895E+01	7.2090E+01	2.2636E+01
	MCS [30]	2.3397E+01	1.1831E+02	6.3478E+01	**6.4917E+01**	2.2710E+01
	CCS [36]	5.5122E+01	2.0687E+02	1.0292E+02	1.0640E+02	3.1751E+01
	ICS [31]	2.5630E+01	1.5159E+02	6.1205E+01	6.5936E+01	2.4360E+01
	VSF [34]	3.2033E+01	1.5116E+02	8.7769E+01	8.8842E+01	2.5119E+01
	ICS2 [32]	2.7123E+01	1.6887E+02	**5.9834E+01**	6.8129E+01	2.6737E+01
	ACSAC [35]	2.4727E+01	1.2452E+02	6.9767E+01	7.1961E+01	2.5578E+01
	ACS [37]	2.6944E+01	1.5049E+02	7.4936E+01	7.4476E+01	2.8272E+01
	DFCS [38]	2.3188E+01	1.1444E+02	6.7153E+01	6.9345E+01	**2.1383E+01**
	MCCSA-P1	1.8243E+02	8.8572E+02	4.0273E+02	4.4234E+02	1.4509E+02
	MCCSA-P2	2.2892E+01	1.4454E+02	7.7230E+01	8.0647E+01	3.0061E+01
	MCCSA-S	4.6940E+01	2.0792E+02	8.9394E+01	9.8690E+01	3.1470E+01
	DCS	**1.7470E+01**	**1.1073E+02**	6.8591E+01	6.9820E+01	2.4267E+01

TABLE 7. *(Continued.)* **Best, worst, median, mean, and standard deviation results of all algorithms for 51 runs for all CEC2017 benchmark functions.**

Continuation of Table VII						
Fn.	Algorithm	best	worst	median	mean	STD
F16	CSA [2]	2.9594E+02	1.2504E+03	9.0323E+02	8.9076E+02	1.9699E+02
	MCS [30]	4.8076E+02	1.1100E+03	7.6508E+02	7.8465E+02	1.5944E+02
	CCS [36]	6.6990E+02	1.6352E+03	1.1333E+03	1.1319E+03	1.9081E+02
	ICS [31]	2.5543E+02	1.0562E+03	7.1043E+02	7.1240E+02	1.5406E+02
	VSF [34]	**1.4738E+02**	1.2512E+03	7.6554E+02	7.7448E+02	2.5048E+02
	ICS2 [32]	3.7393E+02	1.0255E+03	7.0875E+02	7.0150E+02	1.5147E+02
	ACSAC [35]	3.7847E+02	1.1245E+03	7.6786E+02	7.7346E+02	1.9966E+02
	ACS [37]	2.5994E+02	1.0719E+03	6.9133E+02	6.9321E+02	1.7146E+02
	DFCS [38]	4.3415E+02	1.2105E+03	8.7975E+02	8.5536E+02	1.5016E+02
	MCCSA-P1	4.1390E+02	1.6485E+03	1.1245E+03	1.0792E+03	2.4395E+02
	MCCSA-P2	2.8941E+02	1.2056E+03	**6.0127E+02**	**6.3133E+02**	2.1183E+02
	MCCSA-S	4.9056E+02	1.2094E+03	8.7984E+02	8.8460E+02	1.9005E+02
	DCS	3.4908E+02	**9.5851E+02**	6.4772E+02	6.5680E+02	**1.4197E+02**
F17	CSA [2]	9.5149E+01	5.3497E+02	2.8496E+02	2.7964E+02	1.2425E+02
	MCS [30]	5.9934E+01	4.4524E+02	1.9774E+02	2.1675E+02	8.4970E+01
	CCS [36]	1.2402E+02	6.9713E+02	3.8267E+02	3.7681E+02	1.1715E+02
	ICS [31]	7.4067E+01	3.6312E+02	1.9588E+02	1.8165E+02	**6.7034E+01**
	VSF [34]	8.4785E+01	3.5365E+02	1.9062E+02	1.8823E+02	7.1849E+01
	ICS2 [32]	8.8880E+01	3.7147E+02	1.7277E+02	1.9379E+02	7.0521E+01
	ACSAC [35]	6.3816E+01	4.7169E+02	1.9693E+02	2.0117E+02	8.7144E+01
	ACS [37]	7.3425E+01	4.3655E+02	2.0064E+02	1.9802E+02	7.1235E+01
	DFCS [38]	7.5354E+01	4.6366E+02	2.3715E+02	2.3647E+02	8.4382E+01
	MCCSA-P1	1.6630E+02	5.8248E+02	3.4475E+02	3.5054E+02	9.0394E+01
	MCCSA-P2	**5.6557E+01**	4.3558E+02	**1.6256E+02**	**1.7464E+02**	7.6359E+01
	MCCSA-S	1.4333E+02	4.6706E+02	2.4969E+02	2.6473E+02	7.8339E+01
	DCS	6.4083E+01	**3.1564E+02**	2.0544E+02	1.8815E+02	6.8397E+01
F18	CSA [2]	2.2379E+02	1.0526E+03	4.1044E+02	4.6391E+02	1.9396E+02
	MCS [30]	**1.5055E+02**	1.3031E+03	**3.7856E+02**	**4.2061E+02**	2.0645E+02
	CCS [36]	8.4157E+02	1.2323E+04	3.0889E+03	3.3971E+03	1.9595E+03
	ICS [31]	2.4725E+02	1.9723E+03	4.9761E+02	6.2451E+02	3.3876E+02
	VSF [34]	2.4244E+02	2.1889E+03	5.3276E+02	6.5414E+02	3.6903E+02
	ICS2 [32]	2.8065E+02	1.1866E+03	5.1351E+02	5.4064E+02	2.0558E+02
	ACSAC [35]	2.0469E+02	2.6153E+03	6.6309E+02	6.9999E+02	3.8005E+02
	ACS [37]	2.4471E+02	1.2235E+03	6.0393E+02	6.0497E+02	1.9374E+02
	DFCS [38]	2.3275E+02	8.7802E+02	3.9674E+02	4.2843E+02	1.4469E+02
	MCCSA-P1	4.7349E+04	3.9366E+05	1.3391E+05	1.5422E+05	7.8246E+04
	MCCSA-P2	2.1576E+02	1.1826E+03	4.0098E+02	4.7396E+02	2.1672E+02
	MCCSA-S	6.5013E+02	6.5804E+03	1.5497E+03	1.8667E+03	1.1825E+03
	DCS	2.4400E+02	**7.0288E+02**	4.3713E+02	4.4338E+02	**1.2983E+02**
F19	CSA [2]	1.7974E+01	4.8197E+01	2.9554E+01	3.0135E+01	6.8005E+00
	MCS [30]	1.8042E+01	4.4543E+01	2.6090E+01	2.7389E+01	5.8511E+00
	CCS [36]	1.9709E+01	3.8993E+01	2.8218E+01	2.9327E+01	**4.1317E+00**
	ICS [31]	1.4584E+01	3.9601E+01	2.5415E+01	2.5753E+01	5.0607E+00
	VSF [34]	1.8317E+01	4.9152E+01	2.8320E+01	2.8881E+01	5.8708E+00
	ICS2 [32]	1.5659E+01	3.9469E+01	2.4007E+01	2.5233E+01	5.1320E+00
	ACSAC [35]	1.8568E+01	4.2405E+01	2.9260E+01	2.8973E+01	5.7707E+00
	ACS [37]	**1.3709E+01**	4.1301E+01	**2.3323E+01**	**2.4128E+01**	5.4385E+00
	DFCS [38]	2.0176E+01	3.9529E+01	2.7288E+01	2.7500E+01	4.5310E+00
	MCCSA-P1	4.8384E+01	2.8559E+02	1.1863E+02	1.3014E+02	4.7141E+01
	MCCSA-P2	1.6083E+01	4.0214E+01	2.5671E+01	2.5972E+01	5.1541E+00
	MCCSA-S	2.0810E+01	4.4016E+01	2.9227E+01	2.9924E+01	5.0104E+00
	DCS	1.6776E+01	**3.5434E+01**	2.4085E+01	2.4903E+01	4.5851E+00
F20	CSA [2]	1.1843E+02	5.9906E+02	3.6400E+02	3.6671E+02	1.1806E+02
	MCS [30]	1.2521E+02	5.6236E+02	3.1572E+02	3.0913E+02	1.0128E+02
	CCS [36]	1.2330E+02	6.5812E+02	4.0503E+02	3.9649E+02	1.2204E+02
	ICS [31]	6.4825E+01	4.0402E+02	2.4828E+02	2.3904E+02	8.7000E+01
	VSF [34]	5.8449E+01	5.0660E+02	2.3486E+02	2.5660E+02	1.1919E+02
	ICS2 [32]	7.8984E+01	4.7163E+02	2.7309E+02	2.6496E+02	9.7019E+01
	ACSAC [35]	8.4188E+01	4.8203E+02	2.8153E+02	2.9084E+02	1.0196E+02
	ACS [37]	7.0010E+01	4.3030E+02	2.1827E+02	2.4208E+02	9.5213E+01
	DFCS [38]	1.0189E+02	5.7170E+02	3.6833E+02	3.6661E+02	9.9703E+01
	MCCSA-P1	5.6677E+01	5.8035E+02	4.0483E+02	3.8452E+02	1.1758E+02
	MCCSA-P2	**3.5074E+01**	**3.9237E+02**	**2.0461E+02**	**2.0271E+02**	8.5861E+01
	MCCSA-S	1.5483E+02	4.9189E+02	3.3293E+02	3.2816E+02	**7.6470E+01**
	DCS	5.8348E+01	4.8207E+02	2.7385E+02	2.6715E+02	8.6505E+01

TABLE 7. ***(Continued.)*** **Best, worst, median, mean, and standard deviation results of all algorithms for 51 runs for all CEC2017 benchmark functions.**

Continuation of Table VII						
Fn.	Algorithm	best	worst	median	mean	STD
F21	CSA [2]	1.1027E+02	4.0539E+02	3.2806E+02	3.2418E+02	5.5201E+01
	MCS [30]	2.5392E+02	3.1719E+02	2.7830E+02	2.8077E+02	1.3022E+01
	CCS [36]	2.7516E+02	3.3155E+02	3.0199E+02	3.0284E+02	1.4253E+01
	ICS [31]	2.3762E+02	**2.8208E+02**	**2.5567E+02**	2.5680E+02	**9.0058E+00**
	VSF [34]	2.3220E+02	3.1211E+02	2.8312E+02	2.8149E+02	1.6991E+01
	ICS2 [32]	2.3649E+02	2.9388E+02	2.6017E+02	2.6003E+02	1.2021E+01
	ACSAC [35]	1.1253E+02	3.4048E+02	2.9361E+02	2.8439E+02	4.6586E+01
	ACS [37]	2.0646E+02	3.0136E+02	2.6098E+02	2.5850E+02	1.5752E+01
	DFCS [38]	**1.0947E+02**	3.6422E+02	3.0322E+02	3.0010E+02	4.2963E+01
	MCCSA-P1	3.0097E+02	3.6546E+02	3.3567E+02	3.3632E+02	1.5876E+01
	MCCSA-P2	1.8888E+02	2.9679E+02	2.7887E+02	2.7580E+02	1.7506E+01
	MCCSA-S	2.6562E+02	3.3105E+02	2.9207E+02	2.9216E+02	1.4507E+01
	DCS	1.7347E+02	2.8846E+02	2.5638E+02	**2.5490E+02**	1.6016E+01
F22	CSA [2]	1.0000E+02	4.8714E+03	4.2475E+03	3.2552E+03	1.8817E+03
	MCS [30]	1.0000E+02	4.5305E+03	1.0000E+02	1.8516E+03	1.8916E+03
	CCS [36]	1.0009E+02	6.4828E+03	5.7975E+03	4.9514E+03	2.1073E+03
	ICS [31]	1.0000E+02	4.6746E+03	3.1906E+03	2.1273E+03	2.0263E+03
	VSF [34]	1.0000E+02	5.4033E+03	1.0000E+02	2.4230E+03	2.4001E+03
	ICS2 [32]	1.0000E+02	**4.3843E+03**	3.6919E+03	2.4808E+03	1.8718E+03
	ACSAC [35]	1.0000E+02	5.4870E+03	**1.0000E+02**	1.9385E+03	2.3254E+03
	ACS [37]	1.0000E+02	5.2836E+03	**1.0000E+02**	2.0523E+03	2.2888E+03
	DFCS [38]	1.0000E+02	4.6037E+03	3.6939E+03	2.3712E+03	2.0095E+03
	MCCSA-P1	1.0017E+02	6.2791E+03	5.4857E+03	4.7538E+03	1.8163E+03
	MCCSA-P2	1.0000E+02	5.1565E+03	4.2389E+03	3.7672E+03	**1.3410E+03**
	MCCSA-S	1.0000E+02	5.9706E+03	4.9195E+03	4.4179E+03	1.5883E+03
	DCS	1.0000E+02	4.5039E+03	**1.0000E+02**	**1.6319E+03**	1.7940E+03
F23	CSA [2]	4.3188E+02	5.1058E+02	4.6996E+02	4.7326E+02	2.0230E+01
	MCS [30]	4.1020E+02	4.7254E+02	4.3529E+02	4.3796E+02	1.4299E+01
	CCS [36]	4.3097E+02	4.9974E+02	4.6347E+02	4.6555E+02	1.7498E+01
	ICS [31]	3.8144E+02	4.3266E+02	4.0932E+02	4.0911E+02	1.0813E+01
	VSF [34]	4.0054E+02	4.5987E+02	4.3441E+02	4.3204E+02	1.5076E+01
	ICS2 [32]	3.8504E+02	4.3681E+02	4.0962E+02	4.1114E+02	1.0435E+01
	ACSAC [35]	3.9780E+02	4.7462E+02	4.3956E+02	4.3935E+02	1.6815E+01
	ACS [37]	**3.8047E+02**	4.4199E+02	4.1570E+02	4.1464E+02	1.3210E+01
	DFCS [38]	4.2237E+02	5.0547E+02	4.6161E+02	4.5884E+02	1.9251E+01
	MCCSA-P1	4.1171E+02	5.1083E+02	4.7638E+02	4.7597E+02	1.8820E+01
	MCCSA-P2	4.0162E+02	4.5877E+02	4.3360E+02	4.3195E+02	1.4185E+01
	MCCSA-S	4.0381E+02	4.8844E+02	4.4842E+02	4.4738E+02	1.6103E+01
	DCS	3.8988E+02	**4.2438E+02**	**4.0768E+02**	**4.0812E+02**	**9.3490E+00**
F24	CSA [2]	4.9494E+02	5.7866E+02	5.3371E+02	5.3325E+02	1.8889E+01
	MCS [30]	4.7314E+02	5.6015E+02	5.1302E+02	5.1389E+02	1.7910E+01
	CCS [36]	5.0372E+02	5.9208E+02	5.3360E+02	5.3588E+02	1.8958E+01
	ICS [31]	4.6427E+02	5.2215E+02	4.8960E+02	4.8854E+02	**1.1342E+01**
	VSF [34]	4.7093E+02	5.3593E+02	5.0529E+02	5.0501E+02	1.5006E+01
	ICS2 [32]	**4.6002E+02**	5.1572E+02	4.8468E+02	4.8660E+02	1.1902E+01
	ACSAC [35]	4.7728E+02	5.5388E+02	5.2086E+02	5.2129E+02	1.7868E+01
	ACS [37]	4.6337E+02	5.1112E+02	4.9036E+02	4.8956E+02	1.1767E+01
	DFCS [38]	4.9711E+02	5.5163E+02	5.2219E+02	5.2127E+02	1.3039E+01
	MCCSA-P1	5.3400E+02	5.9224E+02	5.5903E+02	5.6077E+02	1.3399E+01
	MCCSA-P2	4.7914E+02	5.3444E+02	5.0993E+02	5.0933E+02	1.3537E+01
	MCCSA-S	4.8770E+02	5.5968E+02	5.2510E+02	5.2611E+02	1.6568E+01
	DCS	4.6287E+02	**5.0790E+02**	**4.8166E+02**	**4.8177E+02**	1.1407E+01
F25	CSA [2]	3.7509E+02	3.7877E+02	3.7837E+02	3.7793E+02	1.0797E+00
	MCS [30]	**3.7505E+02**	3.7868E+02	3.7833E+02	3.7770E+02	1.2580E+00
	CCS [36]	3.7508E+02	3.7864E+02	3.7837E+02	3.7788E+02	1.2067E+00
	ICS [31]	3.7514E+02	3.8676E+02	3.7836E+02	3.7836E+02	1.3650E+00
	VSF [34]	3.7511E+02	**3.7863E+02**	3.7836E+02	3.7811E+02	8.4762E-01
	ICS2 [32]	3.7511E+02	3.7884E+02	3.7833E+02	3.7810E+02	8.8515E-01
	ACSAC [35]	3.7505E+02	3.7863E+02	**3.7832E+02**	3.7754E+02	1.3775E+00
	ACS [37]	3.7546E+02	3.7875E+02	3.7838E+02	3.7834E+02	**4.2172E-01**
	DFCS [38]	3.7509E+02	3.7899E+02	3.7835E+02	**3.7738E+02**	1.5030E+00
	MCCSA-P1	3.7853E+02	3.8362E+02	3.7871E+02	3.7908E+02	1.0812E+00
	MCCSA-P2	3.7575E+02	3.7869E+02	3.7834E+02	3.7827E+02	4.7420E-01
	MCCSA-S	3.7528E+02	3.7885E+02	3.7838E+02	3.7823E+02	7.4152E-01
	DCS	3.7509E+02	3.7892E+02	3.7837E+02	3.7808E+02	9.0367E-01

TABLE 7. *(Continued.)* **Best, worst, median, mean, and standard deviation results of all algorithms for 51 runs for all CEC2017 benchmark functions.**

Continuation of Table VII						
Fn.	Algorithm	best	worst	median	mean	STD
F26	CSA [2]	**2.0000E+02**	3.0167E+03	1.6286E+03	1.4973E+03	9.2055E+02
	MCS [30]	2.0021E+02	2.3406E+03	1.9345E+03	1.6696E+03	5.8028E+02
	CCS [36]	1.6328E+03	3.0592E+03	2.2006E+03	2.2412E+03	2.7618E+02
	ICS [31]	3.0000E+02	2.0733E+03	1.6804E+03	1.6087E+03	3.2973E+02
	VSF [34]	2.2542E+02	2.2892E+03	1.8521E+03	1.7782E+03	3.9991E+02
	ICS2 [32]	2.2478E+02	**1.9848E+03**	1.6785E+03	1.6000E+03	3.4513E+02
	ACSAC [35]	**2.0000E+02**	2.4507E+03	1.6123E+03	**1.2374E+03**	8.5471E+02
	ACS [37]	6.3166E+02	2.0460E+03	1.6748E+03	1.6569E+03	**2.2144E+02**
	DFCS [38]	2.0123E+02	2.6691E+03	**1.2417E+03**	1.3754E+03	7.8969E+02
	MCCSA-P1	1.8178E+03	3.0589E+03	2.4213E+03	2.4260E+03	2.7031E+02
	MCCSA-P2	1.1698E+03	2.2258E+03	1.9231E+03	1.8989E+03	2.2513E+02
	MCCSA-S	1.3200E+03	2.6252E+03	2.0179E+03	2.0502E+03	2.3436E+02
	DCS	6.5914E+02	2.0851E+03	1.6280E+03	1.5819E+03	2.3710E+02
F27	CSA [2]	5.0001E+02	5.0001E+02	5.0001E+02	5.0001E+02	**4.5940E-05**
	MCS [30]	5.0001E+02	5.0001E+02	5.0001E+02	5.0001E+02	6.3571E-05
	CCS [36]	5.0001E+02	5.0001E+02	5.0001E+02	5.0001E+02	6.8012E-05
	ICS [31]	5.0001E+02	5.0001E+02	5.0001E+02	5.0001E+02	5.7491E-05
	VSF [34]	5.0001E+02	5.0001E+02	5.0001E+02	5.0001E+02	6.4794E-05
	ICS2 [32]	5.0001E+02	5.0001E+02	5.0001E+02	5.0001E+02	6.2676E-05
	ACSAC [35]	5.0001E+02	5.0001E+02	5.0001E+02	5.0001E+02	7.1879E-05
	ACS [37]	5.0001E+02	5.0001E+02	5.0001E+02	5.0001E+02	7.4091E-05
	DFCS [38]	5.0001E+02	5.0001E+02	5.0001E+02	5.0001E+02	5.2517E-05
	MCCSA-P1	5.0001E+02	5.0001E+02	5.0001E+02	5.0001E+02	7.3505E-05
	MCCSA-P2	5.0001E+02	5.0001E+02	5.0001E+02	5.0001E+02	6.7631E-05
	MCCSA-S	5.0001E+02	5.0001E+02	5.0001E+02	5.0001E+02	5.2773E-05
	DCS	5.0001E+02	5.0001E+02	5.0001E+02	5.0001E+02	6.6565E-05
F28	CSA [2]	5.0001E+02	5.0001E+02	5.0001E+02	5.0001E+02	**4.8812E-05**
	MCS [30]	**3.0000E+02**	5.0001E+02	5.0001E+02	**4.9609E+02**	2.8007E+01
	CCS [36]	5.0001E+02	5.0001E+02	5.0001E+02	5.0001E+02	5.4226E-05
	ICS [31]	5.0001E+02	5.0001E+02	5.0001E+02	5.0001E+02	5.4636E-05
	VSF [34]	5.0001E+02	5.0001E+02	5.0001E+02	5.0001E+02	6.1882E-05
	ICS2 [32]	4.9791E+02	5.0001E+02	5.0001E+02	4.9997E+02	2.9389E-01
	ACSAC [35]	5.0001E+02	5.0001E+02	5.0001E+02	5.0001E+02	7.3677E-05
	ACS [37]	4.9710E+02	5.0001E+02	5.0001E+02	4.9995E+02	4.0647E-01
	DFCS [38]	4.9688E+02	5.0001E+02	5.0001E+02	4.9995E+02	4.3774E-01
	MCCSA-P1	5.0001E+02	5.0001E+02	5.0001E+02	5.0001E+02	5.9637E-05
	MCCSA-P2	5.0001E+02	5.0001E+02	5.0001E+02	5.0001E+02	6.6771E-05
	MCCSA-S	5.0001E+02	5.0001E+02	5.0001E+02	5.0001E+02	7.3295E-05
	DCS	5.0001E+02	5.0001E+02	5.0001E+02	5.0001E+02	5.8477E-05
F29	CSA [2]	5.3498E+02	1.2374E+03	9.1547E+02	9.0385E+02	1.5502E+02
	MCS [30]	5.3715E+02	1.0603E+03	8.4481E+02	8.3214E+02	1.0795E+02
	CCS [36]	6.6699E+02	1.3111E+03	1.0689E+03	1.0350E+03	1.4571E+02
	ICS [31]	5.6912E+02	**9.7205E+02**	7.9499E+02	**7.9099E+02**	**1.0045E+02**
	VSF [34]	5.9996E+02	1.1705E+03	8.7504E+02	8.8234E+02	1.3421E+02
	ICS2 [32]	5.3063E+02	1.0545E+03	7.9399E+02	7.9785E+02	1.0465E+02
	ACSAC [35]	4.8702E+02	1.0806E+03	8.3360E+02	8.4022E+02	1.4174E+02
	ACS [37]	4.8466E+02	1.1431E+03	7.9716E+02	8.0989E+02	1.2345E+02
	DFCS [38]	7.1889E+02	1.3453E+03	9.4795E+02	9.5964E+02	1.5126E+02
	MCCSA-P1	6.5472E+02	1.4812E+03	1.1595E+03	1.1500E+03	1.7508E+02
	MCCSA-P2	4.9017E+02	1.1708E+03	8.8963E+02	8.7763E+02	1.4525E+02
	MCCSA-S	6.0005E+02	1.2018E+03	9.3406E+02	9.2632E+02	1.4889E+02
	DCS	**4.7451E+02**	1.1552E+03	**7.9151E+02**	8.0121E+02	1.3329E+02
F30	CSA [2]	**2.1971E+02**	2.6900E+02	2.3533E+02	2.3646E+02	8.3672E+00
	MCS [30]	2.2292E+02	2.7548E+02	2.3938E+02	2.4120E+02	1.1023E+01
	CCS [36]	2.2199E+02	2.6083E+02	2.3629E+02	2.3629E+02	7.5290E+00
	ICS [31]	2.2699E+02	2.6468E+02	2.3851E+02	2.3933E+02	8.2269E+00
	VSF [34]	2.2339E+02	2.6640E+02	2.3750E+02	2.3901E+02	8.9335E+00
	ICS2 [32]	2.2281E+02	2.7398E+02	2.3619E+02	2.3983E+02	9.7890E+00
	ACSAC [35]	2.2383E+02	2.8147E+02	2.4095E+02	2.4296E+02	1.2004E+01
	ACS [37]	2.2092E+02	2.8168E+02	2.3737E+02	2.4014E+02	1.1185E+01
	DFCS [38]	2.2373E+02	**2.5510E+02**	**2.3518E+02**	**2.3577E+02**	**6.4407E+00**
	MCCSA-P1	5.0518E+02	4.0479E+03	1.4344E+03	1.6726E+03	8.4382E+02
	MCCSA-P2	2.2618E+02	2.8093E+02	2.3802E+02	2.4039E+02	9.8210E+00
	MCCSA-S	2.2560E+02	2.8007E+02	2.3856E+02	2.4048E+02	1.1855E+01
	DCS	2.2628E+02	2.7402E+02	2.3658E+02	2.3856E+02	9.4211E+00

algorithms. From this perspective this algorithm is executed twice, one terminates when the algorithm reaches the maximum function evaluations (300000) marked in Table 7 as MCCSA-P1, and the other terminates when the algorithm reaches 6000 iterations (generations), which is represented in Tables 7 as MCCSA-P2.

For each algorithm, the mean value of the best fitness obtained in 51 runs is recorded as well as the standard deviation, best, worst and the median value for all runs. The best result for each metric in each function is marked in bold. As seen from Table 7, mean, standard deviation, best, worst, and median values for the functions F1 and F2 are the same for all the algorithms, and these values are similar to the results obtained by a variant of CSA which was also applied on 30 dimensions CEC2017 Benchmark functions under the same conditions [55]. Considering the mean value results shown in Table 7, the proposed DCS gives the best results on 9 benchmark functions (F5, F7, F8, F10, F11, F21, F22, F23, and F24), while the other proposed MCCSA-P2 gives the best results in 6 benchmark functions (F3, F4, F6, F12, F17, and F20). MCS algorithm gives the best results in 3 functions (F15, F18, and F28). The ICS algorithm has the advantage only on 2 functions (F14 and F29), moreover, the ACS algorithm has the best mean value on 2 functions as well (F9 and F19). Considering the standard deviation results shown in Table 7, DCS gives the best results on 5 benchmark functions (F7, F8, F11, F18, and F23), while the other proposed MCCSA-P2 gives the best results in 5 benchmark functions as well (F3, F4, F6, F12, and F22). Also, the ICS algorithm gives the best results in 5 functions (F5, F17, F21, F24, and F29). ACS gives the best results in 4 functions (F9, F14, F25, and F26).

B. STATISTICAL ANALYSIS (SIGN TEST)

Table 8 shows paired sign test comparisons between DCS and all the other algorithms considering the mean value results. The "+" sign in Table 8 indicates that the DCS has better performance than the other algorithm, while the "−" sign indicates that DCS has worse mean value than the other algorithm, the "=" sign shows a draw between the two algorithms. As shown in Table 8, DCS outperforms all other algorithms as it has the largest number of wins, and the P-value in most cases is less than a level of significance of 0.05. However, the p-value for the ICS case equals 0.17 and this value is more than 0.05 but DCS still has the best performance as it has 17 wins, while ICS has only 9 wins. Generally, DCS has the best performance based on the sign test results. However, this is not sufficient to judge the performance between algorithms based only on the best-obtained value and counting how many times the algorithm wins. Therefore, a deep statistical analysis is performed using the Friedman test to show which algorithm gives the best performance using its average rank, followed by 4 post-hoc procedures to compare the paired results of the proposed algorithms with all other variants. Also, graphical statistical analysis is performed via Box plots to show the distribution of the best fitness values all over the 51 runs. All the statistical analysis is described in details in the next section.

TABLE 8. paired sign test for DCS vs all other algorithms considering the mean value.

DCS vs	+/=/-	P-Value
CSA [2]	22/4/4	0.001
MCS [30]	21/3/6	0.007
CCS [36]	24/4/2	0
ICS [31]	17/4/9	0.17
VSF [34]	22/4/4	0.001
ICS2 [32]	22/3/5	0.002
ACSAC [35]	22/4/4	0.001
ACS [37]	19/3/8	0.054
DFCS [38]	21/3/6	0.007
MCCSA-P1	26/2/2	0
MCCSA-P2	18/4/8	0.078
MCCSA-S	25/4/1	0

C. STATISTICAL ANALYSIS (FRIEDMAN TEST AND POST-HOC PROCEDURES)

This section aims to find which algorithm gives the best results as well as the degree of significance of these results. The statistical analysis is conducted in two stages. A Friedman test will be performed in the first stage followed by a post-hoc paired test in the second stage. For both stages, the null hypothesis H_0 is assumed such that there is no significant difference between the algorithms under the comparison, and we will assume the alternative hypothesis H_1 that represents the fact that there is a significant difference between the algorithms we are comparing. In the first stage (Friedman test), a level of significance of $\alpha = 0.05$ is set and the degree of freedom $df = k - 1$, where k is the number of algorithms in the comparison so the degree of freedom is $13 - 1 = 12$. From the chi-square distribution, the decision rule threshold is obtained using $df = 12$ and $\alpha = 0.05$, hence we get 21.03 where this value is used to judge our hypotheses by comparing it with the Friedman statistic value Fr obtained from (27).

$$F_r = \frac{12}{nk(k+1)} \sum_{j=1}^{k} R_j^2 - 3n(k+1) \tag{27}$$

where k is the number of algorithms and n is the number of benchmark problems and R_j is the summation of all ranks for the j^{th} algorithm. If $F_r > 21.03$, The null hypothesis is to be rejected due to the significant difference between the algorithms. In case of the rejection of the null hypothesis in the first stage, a post-hoc paired test is performed in the second stage to compare the best algorithm with the minimum rank with each one of the other 12 algorithms. Hence 12 paired comparisons are conducted for the 13 algorithms. For each comparison, Z-value is calculated as in (28).

$$z = \frac{R_i - R_j}{\sqrt{\frac{k(k+1)}{6n}}} \tag{28}$$

where R_i and R_j are the average rank of the i^{th} and j^{th} algorithms. The value of z represents the difference between

TABLE 9. The average rank for each algorithm and the APV for paired comparisons between DCS and all other algorithms on mean value results using 4 post-hoc procedures.

Algorithm	avg rank	z-value	p-value	Bonf. APV	Holm APV	Holland APV	Finner APV
CSA [2]	9.1000	5.4034	0.000001	0.000009	0.000007	0.000007	0.000003
MCS [30]	6.6167	2.9337	0.003300	0.039600	0.019200	0.019047	0.005479
CCS [36]	10.0333	6.3316	0.000000	0.000000	0.000000	0.000000	0.000000
ICS [31]	4.4000	0.7293	0.465800	**1.000000**	**0.560800**	**0.465800**	**0.465800**
VSF [34]	6.6333	2.9503	0.003200	0.038400	0.019200	0.019047	0.005479
ICS2 [32]	5.1500	1.4752	0.140200	**1.000000**	**0.560800**	**0.453501**	**0.182421**
ACSAC [35]	7.3667	3.6796	0.000234	0.002800	0.001635	0.001634	0.000467
ACS [37]	4.7833	1.1105	0.266800	**1.000000**	**0.560800**	**0.462418**	**0.287196**
DFCS [38]	7.7167	4.0277	0.000057	0.000684	0.000456	0.000456	0.000137
MCCSA-P1	11.4667	7.7570	0.000000	0.000000	0.000000	0.000000	0.000000
MCCSA-P2	5.0333	1.3591	0.174100	**1.000000**	**0.560800**	**0.453501**	**0.205099**
MCCSA-S	9.0333	5.3371	0.000001	0.000013	0.000009	0.000009	0.000003
DCS	**3.6667**	N/A	N/A	N/A	N/A	N/A	N/A

the ranks. Next, the two-tailed p-value is calculated using this z value from a normal distribution with mean zero and a standard deviation equals 1. But this p-value is called the unadjusted p-value because it considers only the probability error of a certain single comparison but it doesn't take into account the remaining comparisons. So if a level of significance $\alpha = 0.05$ is used for each single paired comparison, then the family-wise error rate for the 12 comparisons is computed as the following $\alpha_{FW} = 1 - (1 - \alpha)^{k-1}$ which will be 0.4596 and this is a very large value. So to deal with this problem the adjusted p-value (APV) is calculated using 4 different methods as follows:

- The first simplest method is the Bonferroni-Dunn procedure that is given in (29) in which p_i is the unadjusted p-value for the i^{th} comparison, and k is the number of algorithms. Where all the unadjusted p-values are multiplied by $k - 1$.
- The second method is called Holm procedure that depends on ordering the p-values in ascending order so that ($p1 \leq p2 \leq P3 \leq \ldots..\leq pk - 1$), and the adjusted p values are calculated as in (30).
- The third method is called Holland procedure that orders the p-values in ascending order like Holm method and the adjusted p values are obtained by (31).
- The fourth method is called Finner procedure that also orders the p-values in ascending order and calculate the adjusted p values by (32).

$$BonferroniAPV_i = min(\upsilon, 1), \quad where\ \upsilon = (k-1)p_i \tag{29}$$

$$HolmAPV_i = min(\upsilon, 1), \quad where\ \upsilon = max\{(k-1)p_j : 1 \leq j \leq i\} \tag{30}$$

$$HollandAPV_i = min(\upsilon, 1), \quad where\ \upsilon = max\{1 - (1 - p_j)^{(k-j)} : 1 \leq j \leq i\} \tag{31}$$

$$FinnerAPV_i = min(\upsilon, 1), \quad where\ \upsilon = max\left\{1 - (1 - p_j)^{\frac{(k-1)}{j}} : 1 \leq j \leq i\right\} \tag{32}$$

Hence, if APV is less than α, the null hypothesis is rejected and the difference between the two algorithms is significant. If the opposite, the null hypothesis is accepted and we can say that the two algorithms have almost the same performance with a slight advantage to the one with the higher average rank [56]. These two stages are performed on the mean and standard deviation obtained for the 13 algorithms for 30 benchmark functions in Table 7.

1) MEAN VALUE ANALYSIS

For each function, all algorithms are ranked based on the mean fitness value such that the algorithm which has the smallest mean value will take rank equals 1 and the one with the largest mean value will take rank equals 13. If two algorithms have the same mean value, both of them will take the same rank which will be the average rank between them. The first column in Table 9 shows the average ranks for the 13 algorithms considering the mean value of the 51 runs for the 30 functions. By applying (27) on the summation of the ranks (average rank multiplied by 30), $Fr = 135.8923$ which is greater than 21.03, so the null hypothesis is rejected and we can say that there is a significant difference between the mean values of the 13 algorithms. The proposed MCCSA-P gives the worst results among all the algorithms when it terminates at MaxFEs (300000) as it has the largest average rank. However, it comes at the fourth place when it is given enough time and terminates after 6000 iterations. The results of the proposed MCCSA-S also are not promising as it comes at the 10^{th} place. The proposed algorithm DCS outperforms all the other algorithms as it has the lowest average rank. In spite of the promising results of DCS, a post-hoc paired test needs to be performed between DCS and each one of the other algorithms to validate the significant performance of DCS overall other algorithms. It is clear that from Table 9 that the Finner procedure exhibits the most powerful behavior, by reaching the lowest p-values among the comparisons. the results can be summarized by the following, the proposed algorithm (DCS) has a better significant performance than the standard CSA, MCS, CCS, VSF, ACSAC, DFCS, MCCSA-P1, and MCCSA-S. However, compared to ICS, ICS2, ACS, and MCCSA-P2 the null hypothesis can't be rejected, where these algorithms have similar performance considering the mean fitness values but with a slight advantage to the proposed DCS as it has the best average rank in the Friedman

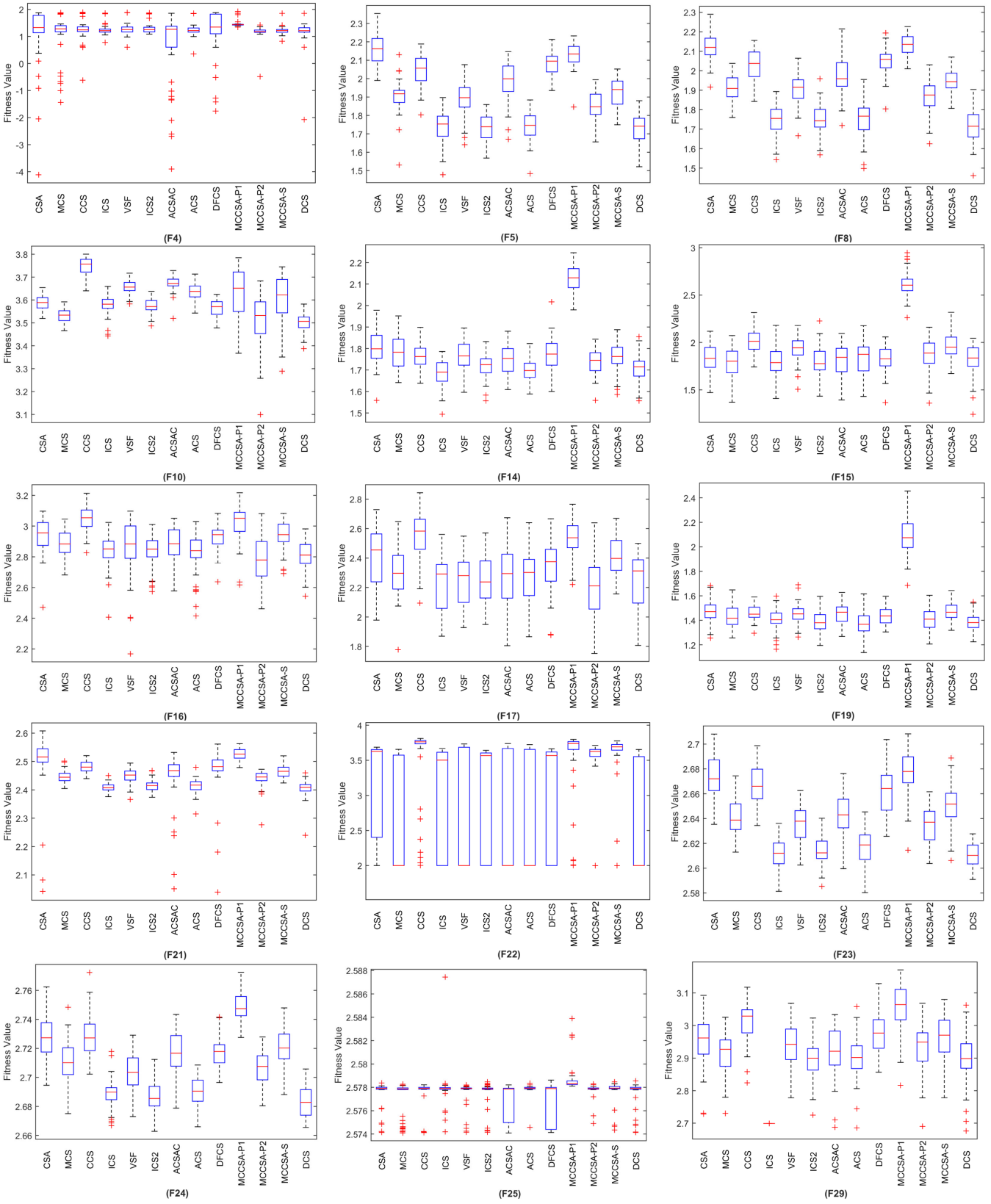

FIGURE 4. Box plots for best fitness values over 51 runs obtained by CSA, MCS, CCS, ICS, VSF, ICS2, ACSAC, ACS, DFCS, MCCSA-P1, MCCSA-P2, MCCSA-S and DCS (F4, F5,F7, F8, F10, F14, F15, F16, F17, F19, F21, F22, F23, F24, F25, F29, F30).

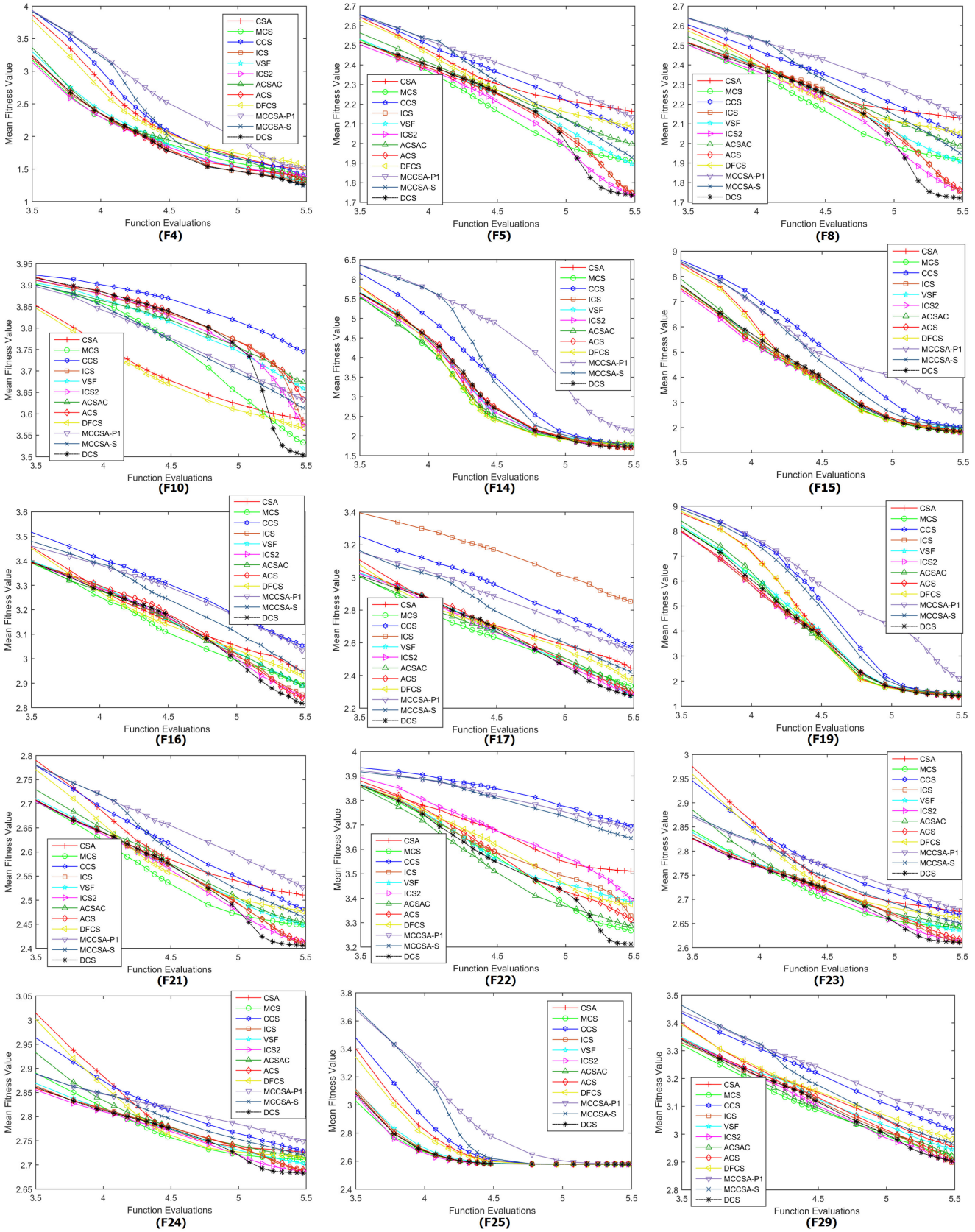

FIGURE 5. Convergence graphs that show the improvement of mean fitness values for the 51 runs against function evaluations for all algorithms (F4, F5, F8, F10, F14, F15, F16, F17, F19, F21, F22, F23, F24, F25, F29).

TABLE 10. The average rank for each algorithm and the APV for paired comparisons between DCS and all other algorithms on STD value results using 4 post-hoc procedures.

Algorithm	avg rank	z-value	p-value	Bonf. APV	Holm APV	Holland APV	Finn. APV
CSA [2]	8.8333	4.7736	0.0000181	0.0002170	0.0001810	0.0001810	0.0000723
MCS [30]	7.4667	3.4144	0.0006393	0.0076700	0.0039600	0.0039533	0.0011311
CCS [36]	8.5000	4.4421	0.0000209	0.0002510	0.0001880	0.0001880	0.0000723
ICS [31]	4.6333	0.5967	0.5507126	**1.0000000**	**0.5507126**	**0.5507126**	**0.5507126**
VSF [34]	7.5000	3.4476	0.0005657	0.0067900	0.0039600	0.0039533	0.0011311
ICS2 [32]	5.1667	1.1271	0.2597068	**1.0000000**	**0.5194135**	**0.4519659**	**0.2796702**
ACSAC [35]	9.1333	5.0719	0.0000042	0.0000498	0.0000457	0.0000457	0.0000249
ACS [37]	5.8000	1.7569	0.0789294	**0.9470000**	**0.3157178**	**0.2802667**	**0.1038297**
DFCS [38]	7.7000	3.6465	0.0002659	0.0031900	0.0021273	0.0021253	0.0006381
MCCSA-P1	9.5333	5.4697	0.0000005	0.0000061	0.0000061	0.0000061	0.0000061
MCCSA-P2	5.8000	1.7569	0.0789294	**0.9470000**	**0.3157178**	**0.2802667**	**0.1038297**
MCCSA-S	6.9000	2.8509	0.0043601	0.0523000	0.0218000	0.0216000	0.0065300
DCS	**4.0333**	N/A	N/A	N/A	N/A	N/A	N/A

test. In other words, based on the Finner APV, DCS has better performance than ACS by 71.28% considering the mean fitness calculations. Also, DCS is better than ICS by 53.42% and is similar to MCCSA-P1 by 20.5%. Also, DCS has an advantage over ICS2 by 71.28%. Although DCS gives the best results over these algorithms by a specific percentage, we can't admit that the difference in results is significant because these percentages do not meet the 5% significance level which is very small.

2) STANDARD DEVIATION ANALYSIS

The average rank for each algorithm based on the standard deviation is calculated in the same manner performed for the mean value. Table 10 shows the average rank for the standard deviation of all algorithms applied to the 30 benchmark functions of 51 runs. When applying (27) on the summation of the ranks (average rank multiplied by 30), $Fr = 75.5429$ which is greater than 21.03. Therefore, the null hypothesis is rejected to admit the significant difference between the standard deviation of the fitness values of the 13 algorithms. As seen from Table 10, The proposed MCCSA-P has the largest average rank among all the algorithms when it terminates at MaxFEs (300000). However, it comes at the fourth place when it is given enough time and terminates after 6000 iterations. From Table 10, It is clear that the proposed algorithm DCS has the lowest average rank which seems to be the best one of all other algorithms (minimum STD), however, a post-hoc paired test needs to be performed to ensure the significant performance of the DCS overall other algorithms. From Table 10, it can be seen that the Finner procedure gives the lowest p-values among comparisons. the results are summarized as following, the proposed algorithm DCS has a better significant performance than the standard CSA, MCS, CCS, VSF, ACSAC, DFCS, MCCSA-P1, and MCCSA-S. However, compared to ICS, ICS2, ACS, MCCSA-P2 the null hypothesis is accepted and these algorithms have similar performance considering the STD in fitness values but with a slight advantage to the proposed DCS based on the average rank from the Friedman test where DCS has the best rank in STD. In other words, based on the Finner APV, DCS and ACS have similar performances by 10.83% considering the STD fitness calculations. Also, DCS is similar to ICS by 55.07% and is similar to MCCSA-P1 by 10.83%. Also, DCS and ICS2 have similar performances by 27.97%.

D. BOX PLOTS GRAPHS

Box plot is a graphical representation that shows the distribution of results for each algorithm. The box plots for each function in CEC2017 are shown in Fig. 4. The horizontal axis represents the algorithm and the vertical axis represents the fitness value on a logarithmic scale. The best fitness value for each run is collected to sketch the box plot for these best values over the 51 runs for each algorithm. The aim is to visualize the distribution of the best fitness values all over the 51 runs for each algorithm on each function. The length of the box represents the variance. When the length is shorter and compact this indicates that the deviation between results is small. When the plot is lower, this means that the results are near to the optimal fitness value which is zero according to (26). The red line inside the box represents the median of the 51 results which is identical to the median obtained in Table 7. Therefore, the best box plot among all the algorithms is the most compact and lower one. The proposed DCS shows the best distribution in all the functions represented in Fig. 4 except for F4, F22, and F25 where the distribution of the results is equivalent for all algorithms. In summary, DCS shows a very consistent and compact distribution for most benchmark functions.

E. CONVERGENCE GRAPHS

Convergence graphs are plotted as well for some functions to compare the average convergence speed for all algorithms against each other. In each run, the best fitness values obtained are stored after reaching certain percentages of the MaxFEs (0.01 MaxFEs, 0.02 MaxFEs,..., 0.1 MaxFEs, 0.2MaxFEs, ..., MaxFEs). These values are recommended by CEC2017 in collecting the results [54]. Then the average value is calculated for each percentage of the MaxFEs over the 51 runs. These calculated average values are used to construct the average convergence graphs for each algorithm. As shown in Fig.5, the proposed DCS shows an amazing convergence behavior in almost all functions. In early function

evaluations, DCS has large fitness values compared to other algorithms. But in the final function evaluations, DCS reaches the smallest fitness values among other algorithms. Starting from a large value of α_0 increases the step size to allow more exploration in the search space in the early generations. Through generations' progression, the value of α_0 is decreased in order to allow more exploitation of the search space. It is obvious that the DCS succeeded in achieving the previously proposed concept of a damped step size of Lévy Flight in the CSA represented in the α_0 value. It is important to notice that MCCSA-P1 is the same as MCCSA-P2 but at different termination conditions, the first one terminates at MaxFEs of 300000 and the second one terminates at MaxFEs of 1650000. Therefore, we only sketch the convergence graphs of MCCSA-P1. MCCSA-P2 has the same convergence graphs however, it is too slow compared to MCCSA-P1 and all other algorithms.

VII. CONCLUSION AND FUTURE WORK

In this paper, three different variants of CSA are proposed, analyzed, and statistically validated. The proposed variants are based on making the α_0 parameter of the step size adaptive. One variant is using ten chaotic maps to adapt the step size in a parallel way. The other variant uses a success rate measure for each map to decide the best choice. The third and the most promising variant is the Damped Cuckoo Search (DCS) as a new adaptive variant of Cuckoo Search Algorithm which is based on the damped oscillation concept that exists in the second order systems. DCS makes the step size of Lévy flight adaptive as well which improves the convergence rate and accuracy. The performance of the DCS is investigated on 30 benchmark functions of CEC2017 with 30 dimensions and then it is compared against almost all other variants that also modify Lévy flight's step size. Then a nonparametric statistical Friedman test is executed to show the superiority of DCS. Moreover, 4 post-hoc procedures are performed to hold paired comparisons between DCS and all other variants to prove the significant performance of DCS against each one of the other variants. Box plots are presented as well to show the distribution of the results overall the 51 runs for all the variants. Convergence graphs are also supported to show how fast each algorithm can improve over the generations. The numerical results clearly prove the superiority of the proposed DCS where it outperforms all other variants in terms of accuracy. Also, convergence graphs illustrate the improvement of the quality of solutions in DCS away faster than other variants for most benchmark problems. In the future, we will continue studying the other parameters of CSA to discover the best way of setting these parameters. Also, DCS can be combined with other modifications to produce new efficient variants of CSA to solve real-world problems.

REFERENCES

[1] F. Glover, "Heuristics for integer programming using surrogate constraints," *Decis. Sci.*, vol. 8, no. 1, pp. 156–166, 1977.
[2] X.-S. Yang and S. Deb, "Cuckoo Search via Lévy flights," in *Proc. World Congr. Nature Biologically Inspired Comput. (NaBIC)*, Dec. 2009, pp. 210–214.
[3] M. Shehab, A. T. Khader, and M. A. Al-Betar, "A survey on applications and variants of the cuckoo search algorithm," *Appl. Soft Comput.*, vol. 61, pp. 1041–1059, Dec. 2017.
[4] C. Blum and A. Roli, "Metaheuristics in combinatorial optimization: Overview and conceptual comparison," *ACM Comput. Surv.*, vol. 35, no. 3, pp. 268–308, Sep. 2003.
[5] J. Holland, "Adaptation in natural and artificial systems : An introductory analysis with application to biology," *Control Artif. Intell.*, to be published.
[6] J. R. Koza, *Genetic Programming II: Automatic Discovery of Reusable Programs*, vol. 13, no. 8. Cambridge, MA, USA: MIT Press, 1994, p. 32.
[7] R. Storn and K. Price, "Minimizing the real functions of the ICEC'96 contest by differential evolution," in *Proc. IEEE Int. Conf. Evol. Comput.*, May 1996, pp. 842–844.
[8] Z. W. Geem, J. H. Kim, and G. V. Loganathan, "A new heuristic optimization algorithm: Harmony search," *J. Simul.*, vol. 76, no. 2, pp. 60–68, Feb. 2001.
[9] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.
[10] S. Koziel and X.-S. Yang, *Computational Optimization, Methods Algorithms*, vol. 356. Berlin, Germany: Springer-Verlag, 2011.
[11] M. A. Al-Betar, "β-Hill climbing: An exploratory local search," *Neural Comput. Appl.*, vol. 28, no. 1, pp. 153–168, 2017.
[12] A. L. Bolaji, M. A. Al-Betar, M. A. Awadallah, A. T. Khader, and L. M. Abualigah, "A comprehensive review: Krill herd algorithm (KH) and its applications," *Appl. Soft Comput.*, vol. 49, pp. 437–446, Dec. 2016.
[13] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," Fac. Eng. Comput., Erciyes Univ., Kayseri, Turkey, Tech. Rep. TR06, 2005.
[14] J. Kennedy and R. Eberhart, "Particle swarm optimization (PSO)," in *Proc. IEEE Int. Conf. Neural Netw.*, Perth, WA, Australia, Nov. 1995, pp. 1942–1948.
[15] X.-S. Yang and N.-I. M. Algorithms, *Nature-Inspired Metaheuristic Algorithms*. Beckington, U.K.: Luniver Press, 2008. pp. 242–246.
[16] R. Rajabioun, "Cuckoo optimization algorithm," *Appl. Soft Comput.*, vol. 11, no. 8, pp. 5508–5518, Aug. 2011.
[17] S. Roy and S. S. Chaudhuri, "Cuckoo search algorithm using Lévy flight: A review," *Int. J. Mod. Educ. Comput. Sci.*, vol. 5, no. 12, pp. 10–15, 2013.
[18] S. Pare, A. Kumar, V. Bajaj, and G. K. Singh, "A multilevel color image segmentation technique based on cuckoo search algorithm and energy curve," *Appl. Soft Comput.*, vol. 47, pp. 76–102, Oct. 2016.
[19] S. Agrawal, R. Panda, S. Bhuyan, and B. K. Panigrahi, "Tsallis entropy based optimal multilevel thresholding using cuckoo search algorithm," *Swarm Evol. Comput.*, vol. 11, pp. 16–30, Aug. 2013.
[20] X. Liu and H. Fu, "PSO-based support vector machine with cuckoo search technique for clinical disease diagnoses," *Sci. World J.*, vol. 2014, May 2014, Art. no. 548483.
[21] D. Giveki, H. Salimi, G. Bahmanyar, and Y. Khademian, "Automatic detection of diabetes diagnosis using feature weighted support vector machines based on mutual information and modified Cuckoo search," 2012, *arXiv:1201.2173*. [Online]. Available: https://arxiv.org/abs/1201.2173
[22] S. Goel, A. Sharma, and P. Bedi, "Cuckoo Search Clustering Algorithm: A novel strategy of biomimicry," in *Proc. World Congr. Inf. Commun. Technol.*, Dec. 2011, pp. 916–921.
[23] C. D. Tran, T. T. Dao, V. S. Vo, and T. T. Nguyen, "Economic load dispatch with multiple fuel options and valve point effect using cuckoo search algorithm with different distributions," *Int. J. Hybrid Inf. Technol.*, vol. 8, no. 1, pp. 305–316, 2015.
[24] P. Sekhar and S. Mohanty, "An enhanced cuckoo search algorithm based contingency constrained economic load dispatch for security enhancement," *Int. J. Elect. Power Energy Syst.*, vol. 75, pp. 303–310, Feb. 2016.
[25] V. Bhargava, S.-E. K. 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, Jan. 2013.
[26] A. H. Gandomi, X.-S. Yang, and A. H. Alavi, "Cuckoo search algorithm: A metaheuristic approach to solve structural optimization problems," *Eng. Comput.*, vol. 29, no. 1, pp. 17–35, 2011.
[27] W. Buaklee and K. Hongesombut, "Optimal DG allocation in a smart distribution grid using cuckoo Search algorithm," in *Proc. 10th Int. Conf. Elect. Eng./Electron. Comput. Telecommun. Inf. Technol.*, May 2013, pp. 1–6.

[28] K. R. Devabalaji, T. Yuvaraj, and K. Ravi, "An efficient method for solving the optimal sitting and sizing problem of capacitor banks based on cuckoo search algorithm," *Ain Shams Eng. J.*, vol. 9, no. 4, pp. 589–597, 2018.
[29] E. Valian, S. Tavakoli, S. Mohanna, and A. Haghi, "Improved cuckoo search for reliability optimization problems," *Comput. Ind. Eng.*, vol. 64, no. 1, pp. 459–468, 2013.
[30] S. Walton, O. Hassan, K. Morgan, and M. Brown, "Modified cuckoo search: A new gradient free optimisation algorithm," *Chaos, Solitons Fractals*, vol. 44, no. 9, pp. 710–718, 2011.
[31] E. Valian, S. Mohanna, and S. Tavakoli, "Improved cuckoo search algorithm for global optimization," *Int. J. Commun. Inf. Technol.*, vol. 1, no. 1, pp. 31–44, 2011.
[32] Z. Zhang and Y. Chen, "An improved cuckoo search algorithm with adaptive method," in *Proc. 7th Int. Joint Conf. Comput. Sci. Optim.*, Jul. 2014, pp. 204–207.
[33] G. G. Wang, S. Deb, A. H. Gandomi, Z. Zhang, and A. H. Alavi, "A novel cuckoo search with chaos theory and elitism scheme," in *Proc. Int. Conf. Soft Comput. Mach. Intell.*, Sep. 2014, pp. 64–69.
[34] L. Wang, Y. Yin, and Y. Zhong, "Cuckoo search with varied scaling factor," *Frontiers Comput. Sci.*, vol. 9, no. 4, pp. 623–635, 2015.
[35] P. Ong, Z. Zainuddin, C. K. Sia, and B. A. M. Zain, "Adaptive cuckoo search algorithm with two-parent crossover for solving optimization problems," in *Proc. Int. Conf. Swarm Intell.* Cham, Switzerland: Springer, 2015, pp. 427–435.
[36] G.-G. Wang, S. Deb, A. H. Gandomi, Z. Zhang, and A. H. Alavi, "Chaotic cuckoo search," *Soft Comput.*, vol. 20, no. 9, pp. 3349–3362, 2016.
[37] R. Chi, Y.-X. Su, D.-H. Zhang, and X.-X. Chi, "Adaptive cuckoo search algorithm for continuous function optimization problems," in *Proc. 12th World Congr. Intell. Control Automat. (WCICA)*, Jun. 2016, pp. 672–676.
[38] J. Cheng, L. Wang, Q. Jiang, Z. Cao, and Y. Xiong, "Cuckoo search algorithm with dynamic feedback information," *Future Gener. Comput. Syst.*, vol. 89, pp. 317–334, Dec. 2018.
[39] S. Wang, S. Song, Y. Yu, Z. Xu, H. Yachi, and S. Gao, "Multiple chaotic cuckoo search algorithm," in *Proc. Int. Conf. Swarm Intell.* Cham, Switzerland: Springer, 2017, pp. 531–542.
[40] G. H. Davies, *The Life of Birds*. Berkeley, CA, USA: Parenthood, 1970.
[41] K. Khan and A. Sahai, "Neural-based cuckoo search of employee health and safety (HS)," in *Int. J. Intell. Syst. Appl.*, vol. 5, no. 2, pp. 76–83, 2013.
[42] X.-S. Yang, *Nature-Inspired Metaheuristic Algorithms*, 2nd ed. Frome, U.K.: Luniver Press, 2010.
[43] C. T. Brown, L. S. Liebovitch, and R. Glendon, "Lévy flights in dobe Ju/'hoansi foraging patterns," *Hum. Ecol.*, vol. 35, no. 1, pp. 129–138, 2007.
[44] I. Pavlyukevich, "Lévy flights, non-local search and simulated annealing," *J. Comput. Phys.*, vol. 226, no. 2, pp. 1830–1844, 2007.
[45] I. Pavlyukevich, "Cooling down Lévy flights," *J. Phys. A, Math. Theor.*, vol. 40, no. 41, p. 12299, 2007.
[46] A. M. Reynolds and M. A. Frye, "Free-flight odor tracking in Drosophila is consistent with an optimal intermittent scale-free search," *PLoS ONE*, vol. 2, no. 4, p. e354, 2007.
[47] M. F. Shlesinger, G. M. Zaslavsky, and U. Frisch, *Lévy Flights Related Topics in Physics*. Berlin, Germany: Springer, 1995.
[48] M. F. Shlesinger, "Mathematical physics Search research," *Nature*, vol. 443, no. 7109, pp. 281–282, 2006.
[49] X.-S. Yang and S. Deb, "Engineering optimisation by cuckoo search," 2010, *arXiv:1005.2908*. [Online]. Available: https://arxiv.org/abs/1005.2908
[50] X.-S. Yang and S. Deb, "Cuckoo search: Recent advances and applications," *Neural Comput. Appl.*, vol. 24, no. 1, pp. 169–174, Jan. 2014.
[51] P. Nasa-ngium, K. Sunat, and S. Chiewchanwattana, "Enhancing modified cuckoo search by using mantegna Lévy flights and chaotic sequences," in *Proc. 10th Int. Joint Conf. Comput. Sci. Softw. Eng. (JCSSE)*, May 2013, pp. 53–57.
[52] L. Huang, S. Ding, S. Yu, J. Wang, and K. Lu, "Chaos-enhanced Cuckoo search optimization algorithms for global optimization," *Appl. Math. Model.*, vol. 40, nos. 5–6, pp. 3860–3875, 2016.
[53] A. C. Pandey, D. S. Rajpoot, and M. Saraswat, "Hybrid step size based cuckoo search," in *Proc. 10th Int. Conf. Contemp. Comput. (IC3)*, Aug. 2017, pp. 1–6.
[54] N. Awad, M. Ali, J. Liang, B. Qu, and P. Suganthan, "Problem definitions and evaluation criteria for the CEC 2017 special session and competition on single objective real-parameter numerical optimization," Nanyang Technol. Univ., Singapore, Tech. Rep., 2016.
[55] R. Salgotra, U. Singh, and S. Saha, "Improved cuckoo search with better search capabilities for solving CEC2017 benchmark problems," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jul. 2018, pp. 1–7.
[56] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm Evol. Comput.*, vol. 1, no. 1, pp. 3–18, 2011.

MOHAMED REDA received the B.Sc. degree from the Computers Engineering and Control Systems Department, Faculty of Engineering, Mansoura University, Egypt. He is currently pursuing the M.Sc. degree. He is also a Teaching Assistant with the Computers Engineering and Control Systems Department, Faculty of Engineering, Mansoura University. His major research interests include artificial intelligence and optimization techniques, such as differential evolution, artificial bee colony, cuckoo search, and genetic algorithms. He is also interested in the application of AI in machine learning, neural networks, fuzzy logic, modern control theory, embedded systems, and robotics.

AMIRA Y. HAIKAL received the B.Sc., M.Sc., and Ph.D. degrees from the Computers Engineering and Control Systems Department, Mansoura University, Egypt. She is also an Associate Professor with the Computers Engineering and Control Systems Department, Faculty of Engineering, Mansoura University. Her major research interests include artificial intelligence, such as genetic algorithms, neural networks, particle swarm optimization, simulated annealing, and fuzzy logic.

MOSTAFA A. ELHOSSEINI received the B.Sc. degree from the Electronics Engineering Department, Mansoura University, Egypt, and the M.Sc. and Ph.D. degrees from the Computers and Systems Engineering Department, Mansoura University. He is currently an Associate Professor with the Computer Science Department, College of Computer Science and Engineering, Taibah University, Yanbu, Saudi Arabia. He is also an Associate Professor with the Computer Engineering and Control Systems Department, Faculty of Engineering, Mansoura University. His major research interests include artificial intelligence (AI), such as genetic algorithms, neural networks, particle swarm optimization, simulated annealing, and fuzzy logic. He is also interested in the application of machine learning in image processing, access control, and the optimization and applications of computational intelligence CI and soft computing tool in bioinformatics. He has served as a member of the international program committees of numerous international conferences.

MAHMOUD BADAWY received the M.Sc. and Ph.D. degrees in computer engineering and control systems from Mansoura University, Egypt, where he is currently an Assistant Professor with the Computer and Systems Engineering Department, Faculty of Engineering. His research interests include computer networking, distributed control systems, database management systems, mobile robots, the Internet of Things, and cloud computing.