Improving Cuckoo Search: Incorporating Changes for CEC 2017 and CEC 2020 Benchmark Problems

Rohit Salgotra, Urvinder Singh
Dept. of Electronics and Communication
Thapar Institute of Engineering and Technology
Patiala, India
Email: r.03dec@gmail.com, urvinder@thapar.edu

Sriparna Saha
Dept. of Computer Science
Indian Institute of Technology
Patna, India
sriparna@iitp.ac.in

Amir H Gandomi
Faculty of Engg. and IT
University of Technology
Sydney, Australia
gandomi@uts.edu.au

***Abstract*—Cuckoo search (CS) is a highly competitive single objective optimization technique. The algorithm has been widely applied in various diverse application domains and has been found to be efficient in solving various real-life problems. In the present work, we have proposed a new enhanced version of CS algorithm and tested its performance on recently proposed CEC 2017 and CEC 2020 benchmark test problems. The proposed algorithm has been named as CSsin and it employs four major modifications, $i)$ new techniques for global and local search are devised, $ii)$ dual search strategy is followed to enhance exploration and exploitation properties of CS algorithm, $iii)$ a linearly decreasing switch probability has been used to add a balance between local and global search, and $iv)$ linearly decreasing population size is used to reduce the computational burden. Apart from these modifications, the division of iterations has been employed as a further modification. The CSsin algorithm has been tested on IEEE CEC 2017 and CEC 2020 benchmark test problems having various dimension sizes and a comparative study has been performed with respect to state-of-the-art optimization algorithms for single objective bound constraint optimization problems. The results of statistical significance test affirm the competitiveness of the proposed algorithm with respect to state-of-the-art techniques. *Index Terms*—Cuckoo Search, Self-adpatation, cuckoo version 1.0, CEC2017, numerical optimization.**

I. Introduction

Nature has always served as the main source of inspiration for solving various real world problem. Instead we can say that, it has been solving various problems from the past millions of years. There are numerous examples where nature has served as a source of inspiration. These include immune system to fight against various diseases, perception based systems for recognizing various patterns, learning process of the brain to train neural networks, decision making and reflex action for making robots and others. All these systems have a common feature of high level computing and numerous algorithms have been introduced in this context to solve various optimization problems from varying fields such as business management, feature selection, classification problems, travelling salesmen problems and others. The main reason for the popularity of these algorithms is that they are simple in structure, flexible and have minimal set of training parameters. Further these algorithms do not require any guess in the initial stages for solving problems and hence have a better chance of solving problems at hand when compared to traditional optimization approaches.

Rohit Salgotra acknowledges the support by DST Govt. of India under (IF-160215) INSPIRE fellowship

Dr. Sriparna Saha would like to acknowledge the support of Early Career Research Award of Science and Engineering Research Board (SERB) of Department of Science and Technology, India to carry out this research.

Cuckoo search (CS) is one such recently introduced nature inspired algorithm, based on the obligate brood parasitic behavior of cuckoo birds found in nature [1]. By using obligate brood parasitic behavior, the cuckoos lay their egg in the nests of crows which consider these eggs as their own and help them in hatching. The newly hatched cuckoo birds, consume all the food gathered by host bird and also kick the host eggs out of the nest. In some cases, the foreign eggs are identified by the host bird and they throw the foreign eggs out of their nests. This danger has motivated the cuckoo species to evolve continuously in such a way that they depict the foraging patterns of host species and change their color, so that the host treats them as own and help them in hatching. The CS algorithm is inspired from this behavior of cuckoos and is based on three basic rules, i) each cuckoo lays only one egg in any random nest, ii) the nest with the best fitness will allow the cuckoo species to survive and carry on to the next generation, iii) there is a probability that the eggs can be recognized by the host birds and based on this probability the host bird either abandons the nest or throws the cuckoo species out of the nest. Based on these concepts, the CS algorithm has been derived and is given by Lévy flight based global search and simple random walk based local search. The global search equation is given by equation (1)

$$x_i^{t+1} = x_i^t + \alpha \times Levy(\lambda)(x_{best} - x_i^t) \tag{1}$$

In case of local search, random walk by using two search equation if followed and is given by equation (2)

$$x_i^{t+1} = x_i^t + \alpha \times H(p - \epsilon) \times (x_j^t - x_k^t) \tag{2}$$

where x_i^t is the solution of the previous generation, ϵ is a random number in the range of [0, 1], p is the probability switch which helps in controlling the extent of global and local search, x_{best} is the best solution in the current generation, x_j^t and x_k^t are two randomly selected solutions in the t^{th} generation.

Overall there are three phase's namely global phase, local phase and the probability, which decides the better performance of CS algorithm. Both the global search phase and local search phases further corresponds to exploration and exploitation phases. Broadly, exploration or diversification tends to explore whole of the search space where as exploitation or intensification refers to searching of specified areas of search space. Both these phenomena require a proper balance in order to find an optimal solution [2]. This balance in CS is brought by the use of switching probability. So there are three major properties of CS which decides its proper functioning. The global search is further controlled by using Lévy flights component where as local search by uniformly distributed random solutions [1]. Thus it can be said that CS satisfies all the requirements of global convergence and hence a global solution is guaranteed for most of the cases [3]. The main reason for this is that CS is able to move toward the global solution or has the ability to find global minima without falling or getting trapped in some local optima. Many studies have proved the efficiency of CS in solving diverse set of optimization problems, including multimodality, high dimensionality, ill-conditioning and others [4].

Further, CS is very much affected by the choice of parameters used and is highly dependent on them [5]. The major parameters used are the Lévy flight component or scaling factor (L), a uniform random number and the switching probability (p). Based on this fact, a large number of CS algorithms have been proposed in the recent past to enhance the performance of CS and improve its working capability [4]. Most of the studies have improved either the Lévy flight component or switching probability but less work has been done to enhance local search phase. The idea of adaptive Lévy flight component was suggested in [6]. The authors further enhanced the local search phase by employing golden ratio with 0.5 probability. Other important hybrid versions include opposition based CS for improving the accuracy [7], improved CS for enhancing the scaling factor and probability switch [12], self-adaptive CS to enhance its exploration capability [8], orthogonal based learning for enhancing CS [9], chaos based CS [10] and others. The algorithm has also been applied to a larger set of application domains namely distribution networks [13], feature selection [11], load dispatch problems [15], signal watermarking in electrocardiogram [14], web clustering [19], modelling [17], infrastructure [18] and others [20], [21], [22].

CEC benchmark are the most widely used benchmark problems and have been used by a large number of research scientists for testing their algorithms. Even for CS, many researchers have continuously focused on applying their algorithms to CEC benchmarks in order to prove their effectiveness. Cuckoo version 1.0 (CV 1.0) is one such recently introduced algorithm which has been applied to CEC 2005 and CEC 2015 benchmarks for real parameter optimization [3]. CV 1.0 is proven to be very powerful algorithm and has proven its worth in line with differential evolution (DE) [24], grey wolf optimization (GWO) [25], and others. The algorithm used Cauchy and Normal distribution instead of simple Lévy flight based component of exploration and uniformly distributed exploitation random numbers respectively. The algorithm though used a standard value for p but still was capable of providing better results. This algorithm also used concepts of division of population and generations to achieve a proper balance between the extent of exploration and exploitation. But from the inference of CV 1.0, it can be found that there is still scope of modification. So in [12], the authors have modified the original CV 1.0 algorithm by employing parameter adaptation and population reduction with respect to generations, to design a new algorithm namely CVnew. The proposed approach uses Cauchy based scaling factor along with equation modification using GWO in the exploration phase and dual division strategy along with sinusoidal adaptive decreasing adjustment as used in LSHADE-EpSin [26] for enhancing the exploitation phase. The third parameter that is probability was also adapted by using exponentially decreasing distribution [12]. The Cauchy based scaling and GWO equations because of their heavy tailed nature, tend to potentially explore the search space whereas dual division strategy and ensemble of sinusoidal waves is used to adapt the parameters in an efficient manner. Both Cauchy based random number for global phase and dual division strategy for local phase is followed for first half of the iterations whereas for second half Cauchy based adaptation along with GWO equations for exploration and sinusoidal based adaptation for exploitation is followed [12].

In present work, the CS algorithm has been modified to test its performance on the CEC 2017 [23] and CEC 2020 [34] problems. The newly proposed algorithm has been named as CSsin algorithm and uses dual population division in both local and the global searching phases. The major reason for the use of dual division of population is because of the extra computational burden on the algorithm in solving two more equations for finding the final solutions. Since most of the work aims at providing the better performance at minimal computational cost, so here CSsin algorithm aims at providing the same when compared to its counterpart CS. The same point can be validated from [18], where three versions of CS were designed and it was found that there were marginal variation of results in all the proposed algorithm for different population divisions. So a slightly better solution can be compensated if the computational burden can reduce significantly. Another modification which has been proposed in the CSsin algorithm, is the use of linear decreasing probability instead of constant probability. The reason for the use of linearly decreasing probability is that it decrease the probability marginally with respect to iterations and there is not much variation in the probability over the course of iterations and hence limit growth of algorithm from exploration to exploitation gradually with time. But in case of exponentially decreasing function, the variation is slow at the start of iterations but as the solution approaches final stages it converges at a faster pace, making the algorithm getting stuck in some local minima. The third modification which has been added in the proposed version is the change in the total population with respect to iterations. That is, with increase in iterations or generations, the population size is reduced in order to restrict the maximum function evaluations. The remainder of the paper is organized into following sections. The proposed CSsin is presented in section 2. The experimental setup and simulation results are given in section 3 and finally conclusions are summarized in section 4.

II. Proposed Approach

In this section, the proposed CSsin algorithm is elaborated in detail. The algorithm firstly starts with initialization of a random population of *N* cuckoo birds with respect to the search range of the problem under consideration. After the initialization step, two concepts based on population division and iterations are employed. The basic idea for these concepts have been motivated from [22] and it has been found that both these conditions have helped the algorithm in successfully finding the global optimal solution. The population division has been followed to add diversity among the positions of search agents during initial steps and converge towards the final steps. On the other hand, iterative division is added for achieving a balanced exploration and exploitation. This modification is followed in both local and global search phase and detailed discussion has been presented as follows.

For the first iterative half, dual population division is employed for both global and local search. Here Cauchy distributed global search is followed during the first half of the population and for the second half equation (2) is modified using concepts of GWO algorithm [25] to generate the new solution. The Cauchy based solution is generated by using equation (3)

$$f_{Cauchy(0,g)}(\delta) = \frac{1}{\pi}\frac{g}{(g^2+\delta^2)} \tag{3}$$

$$y = \frac{1}{2} + \frac{1}{\pi}arctan(\frac{\delta}{g}) \tag{4}$$

$$\delta = tan(\pi(y-\frac{1}{2})) \tag{5}$$

$$x_i^{t+1} = x_i^t + \alpha \otimes Cauchy(\delta)(x_{best} - x_i^t) \tag{6}$$

where scaling factor g is equal to 1, δ is the Cauchy random number in the range of [0, 1].The main reason for using Cauchy based distribution is its fatter tail generating larger steps for exploring search space in a much better way. Also in CS, the final solution is guided by the best solution of the current iteration, so there are chances that it may fall in some local minima, ultimately leading to premature convergence. Thus Cauchy based random number will help the algorithm in performing extensive exploration. For second half of the population, three different solutions from the search space are taken to find the new solution. These three different solutions are generated by using current solution and equations regarding the same are given by (7)

$$\begin{cases} x_1 = x_i - A_1(C_1.x_{new} - x_i^t); x_2 = x_i - A_2(C_2.x_{new} - x_i^t); \\ x_3 = x_i - A_3(C_3.x_{new} - x_i^t); x_i^{t+1} = \frac{x_1+x_2+x_3}{3} \end{cases} \tag{7}$$

where A_1, A_2, A_3 and C_1, C_2, C_3 are given by $A = 2a.r_1 - a$; $C = 2.r_2$, a is linearly decreasing whereas $r_1 and r_2$ are uniform distributed random numbers.This search equation because of the presence of three different solution helps in providing more intensive global search and ultimately paves way for better performance of the proposed algorithm.

In the local search phase, population division is controlled by using a new parameter namley *F* and this search equation has been derived from the search equation used by CV1.0 [22] and is given by (10)

$$F_i^{t+1} = \frac{1}{2}\times(sin(2\pi\times freq\times t+\pi)\times\frac{t_{max-t}}{t_{max}}+1);\ if\, r_1 > 0.5 \tag{8}$$

$$F_i^{t+1} = \frac{1}{2}\times(sin(2\pi\times freq\times t)\times\frac{t_{max-t}}{t_{max}}+1);\ if\, r_1 < 0.5 \tag{9}$$

$$x_i^{t+1} = x_i^t + F.((x_j^t - x_k^t) + (x_l^t - x_m^t)) \tag{10}$$

where *F* is the scaling factor, $x_j^t, x_k^t, x_l^t, x_m^t$ are four random solutions. The scaling factor *F* has been derived from [26]. Here because of the use of a single scaling factor, the search equation becomes efficient for smaller sections of search space and hence make the algorithm perform better exploitation. Also, instead of using a simple random *F*, the local search phase uses a new ensemble of parameters for adaptation by using sinusoidal decreasing *F* as used by LSHADE-cnEpSin [26]. The frequency in case of LSHADE-cnEpSin is generated using Cauchy distribution whereas for present case uniform distribution is followed to generate the same. All these modifications have been added for the second iterative half, whereas for the first half, equation (2) has been used.

Apart from these modifications, the probability parameter p which controls the extent of exploration and exploitation is also adapted by linearly decreasing its value from a high value of $p_{max} = 0.75$ to $p_{min} = 0.25$ by using equation (11)

$$p = p_{init} - \frac{(p_{max} - p_{min})}{t_{max}} \tag{11}$$

Here p_{init} is the initial value of p and is taken to be 0.75, whereas t_{max} is the maximum iteration size. This equation has been inspired form the perturbation rate equation used by modified spider monkey optimization algorithm [24]. The reason for the use of this adaptation is because of the requirement of less extensive exploitation towards the start of the iterations and more intensive local search towards the end. A higher value of p allows the algorithm in performing more extensive local search whereas lower values leads to lesser intensive local search. But an adapted value of p may help in balancing the two search phases and hence provide proper exploitation operation toward the end of the iterations. Furthermore, adaptive linear population reduction [28] is also followed to reduce the total computational burden. Note that this population is firstly sorted and only the worst members of the population are eliminated. The general equation for population reduction is given by equation 12

$$N(g+1) = round[(\frac{N_{min} - N_{max}}{FEs_{max}}).FEs + N_{max}] \tag{12}$$

where values for N_{max} and N_{min} are set to $18 \times D$ and 4, respectively, FEs_{max} are maximum number of function evaluations. The minimum population size is kept 4 because only 4 individuals are required at the minimum to perform global and local search operations. This modification helps in creating a self adaptive population step and hence reducing the computational complexity of the algorithm. In the next section, detailed discussion about the experimental results is presented.

III. Results and Discussion

In this section, detailed discussion related to the simulation results of newly proposed CSsin algorithm are presented. For performance evaluation, simulations are performed on Windows 10 having Matlab 2017a, E5-2630 2.20GHz Intel Xeon Processor with 32GB RAM. Here CEC2017 numerical benchmark problems are used and comparison has been performed with respect to SaDE [29], JADE [30], SHADE [28], mean-variance mapping optimization (MVMO) [31], CV1.0 [22] and CVnew [12]. The results for SaDE, JADE, MVMO and SHADE are taken from [26] whereas for CV1.0 and CVnew are taken from their respective papers. The CEC2017 test suite on the other hand consists of 30 real challenging benchmark problems with 1-3 unimodal, 4-10 multimodal, 11-20 hybrid and 21-30 composite functions. This data set is the most recent and highly complex one, consisting of all major types of optimization problems. A general discussion about these benchmark problems and their definitions can be had from [32]. As far as parameter settings are concerned, the proposed CSsin algorithm consists of very few parameters when compared to its counter parts. The first parameter is the initial population and is set to 50 where as other major parameters consist of the upper and lower limits of probability that is p_{min} and p_{max} and are taken as 0.75 and 0.25. The final parameter is the *freq* which is set to 0.5. Apart from this, all other parameters are chosen as such as used by original CS algorithm. The stopping criteria was taken as $10,000 \times D$ total number of function evaluations with 51 runs performed for each test problem. The results are calculated as error values and are presented in terms of best, worst, mean and standard deviation. The error values are calculated by finding the difference expected and the desired solution and if the difference becomes less than 10^{-8}, the error is considered as zero. The results in this section are divided into two subsections. In the first subsection, the results of CSsin algorithm are presented for $D = 10$, $D = 30$ and $D = 50$ where D is the dimension size.In the second subsection, results with respect to other state-of-the-art algorithms is presented. Further Wilcoxon's rank-sum test [33] has also been done to prove the significance of CSsin algorithm statistically.

A. Algorithm Complexity

In this section, the runtime complexity of the CSsin algorithm in terms of run time t_0 is calculated by using the code given as: $for\ \ i = 1 : 1000000$
$x = 0.55 + double(i);\ \ x = x + x;\ \ x = x/2;\ \ x = x \times x$
$x = sqrt(x);\ \ x = log(x);\ \ x = exp(x);\ \ x = x/(x + 2)$
end
The complexity of algorithm is also shown in Table I. Here the notations T_0 corresponds to the computing time for the code given above and T_1 is the computing time for F_{18} function from the Test suite given in the next section, with a total number of function evaluations of 200,000. T_2 is the mean run time for the same function with same number of function evaluations for a total number of five runs.

TABLE I
Complexity of CSsin algorithm

D	T_0	T_1	T_2	$(T_2 - T_1)/T_0$
10	0.1562	4.094	3.964	0.834
30		20.76	20.148	4.08
50		55.581	58.134	15.54

B. Statistical Results

In this section, results of CSsin algorithm for $D = 10$, $D = 30$ and $D = 50$ are presented. The results are taken as the best, worst, mean and standard deviation values, found by the difference of desired and the expected solution. These results are presented in Table II for $10D$, Table III for $30D$ and Table IV for $50D$.

TABLE II
Statistical Results for $10D$

Function	Best	Worst	Mean	Std dev.
F_1	8.45E-02	8.83E+00	1.91E+00	1.87E+00
F_2	0.00E+00	1.00E+10	1.96E+09	4.00E+09
F_3	0.00E+00	4.80E-08	4.53E-09	1.08E-08
F_4	1.86E-05	2.34E-01	1.22E-02	3.41E-02
F_5	3.97E+00	2.48E+01	1.38E+01	5.63E+00
F_6	2.63E-05	8.94E-04	2.73E-04	1.85E-04
F_7	1.49E+01	4.66E+01	2.90E+01	7.32E+00
F_8	2.98E+00	2.28E+01	1.37E+01	3.75E+00
F_9	0.00E+00	0.00E+00	0.00E+00	0.00E+00
F_{10}	1.24E-01	4.24E+02	2.33E+02	1.09E+02
F_{11}	1.48E-04	5.13E+00	1.42E+00	1.02E+00
F_{12}	1.06E+03	5.71E+03	2.30E+03	8.49E+02
F_{13}	7.96E+00	2.34E+01	1.45E+01	3.12E+00
F_{14}	2.48E+00	1.00E+01	5.59E+00	1.81E+00
F_{15}	6.53E-01	2.51E+00	1.40E+00	4.96E-01
F_{16}	4.01E-01	2.38E+02	5.88E+01	7.18E+01
F_{17}	4.43E-01	1.96E+01	8.12E+00	7.20E+00
F_{18}	3.56E+00	2.29E+01	1.47E+01	5.13E+00
F_{19}	7.86E-01	2.42E+00	1.56E+00	3.71E-01
F_{20}	5.52E-06	7.41E-01	1.30E-01	2.09E-01
F_{21}	0.00E+00	1.02E+02	9.42E+01	2.38E+01
F_{22}	1.51E+01	1.01E+01	8.94E+01	2.46E+01
F_{23}	0.00E+00	3.19E+02	2.18E+02	1.33E+02
F_{24}	5.27E-06	1.00E+02	9.41E+01	2.37E+01
F_{25}	1.00E+02	3.99E+02	3.91E+02	4.18E+01
F_{26}	0.00E+00	2.00E+02	3.92E+01	8.01E+01
F_{27}	3.87E+02	3.92E+02	3.89E+02	7.67E-01
F_{28}	1.00E+02	3.00E+02	2.96E+02	2.80E+01
F_{29}	1.58E+02	2.77E+02	2.48E+02	1.65E+01
F_{30}	4.56E+02	7.26E+03	1.97E+03	1.42E+03

IV. Comparison with other algorithms

In this section, SaDE, JADE, SHADE, MVMO, CV1.0 and CVnew algorithms have been used for comparison with the newly proposed CSsin algorithm. The last row of the table gives the values of Wilcoxon rank-sum test [33]. These results are calculated at 0.5 significance level with comparison performed in terms of $w(+win)$ $/l(-loss)$ and $/t(= tie)$. It should be noted that " + " signifies that the algorithm under consideration is better than the proposed algorithm, " − " means the opposite and " = " means that there is either no relevance or are having same statistical results and are

TABLE III
STATISTICAL RESULTS FOR $30D$

Function	Best	Worst	Mean	Std dev.
F_1	1.00E+10	1.00E+10	1.00E+10	0.00E+00
F_2	1.00E+10	1.00E+10	1.00E+10	0.00E+00
F_3	2.34E+03	9.02E+03	4.58E+03	1.51E+03
F_4	1.07E-02	8.81E+01	3.30E+01	3.40E+01
F_5	1.32E+02	2.16E+02	1.68E+02	1.85E+01
F_6	5.92E-02	1.61E+00	5.17E-01	4.21E-01
F_7	1.03E+02	3.12E+02	2.00E+02	4.83E+01
F_8	8.45E+01	1.43E+02	1.16E+02	1.25E+01
F_9	1.15E+03	3.96E+02	3.17E+03	5.68E+02
F_{10}	1.57E+03	3.17E+03	2.48E+02	3.33E+02
F_{11}	6.53E+00	9.77E+01	3.61E+01	1.86E+01
F_{12}	4.14E+04	1.00E+10	2.56E+09	4.39E+09
F_{13}	9.80E+02	6.14E+04	1.23E+04	1.45E+04
F_{14}	1.61E+02	1.02E+03	3.36E+02	1.42E+02
F_{15}	2.88E+02	7.44E+02	4.72E+02	1.03E+02
F_{16}	1.37E+02	5.54E+02	3.97E+02	1.03E+02
F_{17}	3.17E+01	1.65E+02	7.27E+01	2.53E+01
F_{18}	1.93E+04	7.59E+04	3.91E+04	1.16E+04
F_{19}	1.05E+02	5.87E+02	2.54E+02	8.41E+01
F_{20}	4.21E+01	2.00E+02	1.58E+02	3.84E+01
F_{21}	1.00E+02	1.04E+02	1.00E+02	7.70E-01
F_{22}	1.00E+02	1.00E+02	1.00E+02	3.09E-02
F_{23}	1.00E+02	4.17E+02	2.82E+02	1.30E+02
F_{24}	1.00E+02	2.00E+02	1.66E+02	4.76E+01
F_{25}	3.83E+02	3.87E+02	3.84E+02	1.61E+00
F_{26}	2.00E+02	2.16E+02	2.02E+02	2.58E+00
F_{27}	4.77E+02	5.09E+02	4.97E+02	8.51E+00
F_{28}	3.00E+02	4.09E+02	3.48E+02	4.43E+01
F_{29}	4.10E+02	5.91E+02	5.11E+02	4.23E+01
F_{30}	9.78E+03	3.16E+04	1.84E+04	5.34E+03

TABLE IV
STATISTICAL RESULTS FOR $50D$

Function	Best	Worst	Mean	Std dev.
F_1	1.00E+10	1.00E+10	1.00E+10	0.00E+00
F_2	1.00E+10	1.00E+10	1.00E+10	0.00E+00
F_3	1.59E+03	4.75E+04	1.07E+04	6.68E+03
F_4	2.74E-01	1.50E+02	1.88E+01	2.45E+01
F_5	2.65E+02	3.53E+02	3.09E+02	2.10E+01
F_6	1.46E+00	2.16E+01	1.00E+01	5.28E+00
F_7	5.15E+01	9.31E+02	1.39E+02	9.71E+01
F_8	2.69E+02	3.63E+02	3.17E+02	2.43E+01
F_9	7.81E+03	1.33E+04	1.11E+04	1.00E+03
F_{10}	3.66E+03	6.72E+03	4.97E+03	5.83E+03
F_{11}	6.52E+01	1.80E+02	1.17E+02	2.91E+01
F_{12}	1.00E+10	1.00E+10	1.00E+10	0.00E+00
F_{13}	1.00E+10	1.00E+10	1.00E+10	0.00E+00
F_{14}	1.70E+03	7.53E+04	2.23E+04	1.72E+04
F_{15}	1.64E+03	2.06E+04	1.13E+04	6.02E+03
F_{16}	3.75E+02	1.07E+03	7.23E+02	1.79E+02
F_{17}	2.86E+02	9.28E+02	6.50E+02	1.16E+02
F_{18}	4.75E+04	4.31E+04	1.73E+05	7.91E+04
F_{19}	9.06E+02	1.32E=04	5.84E+03	3.15E+03
F_{20}	2.80E+01	6.94E+02	2.31E+01	9.73E+01
F_{21}	1.08E+02	4.17E+02	1.57E+02	9.74E+01
F_{22}	1.0E+02	1.02E+02	1.00E+02	3.91E-01
F_{23}	1.30E+02	6.82E+02	3.51E+02	7.88E+01
F_{24}	5.84E+02	7.40E+02	6.87E+02	3.57E+01
F_{25}	3.61E+02	5.70E+02	4.26E+02	2.08E+01
F_{26}	3.00E+02	3.00E+02	3.00E+02	4.57E-02
F_{27}	5.39E+02	6.51E+02	5.97E+02	3.25E+01
F_{28}	2.59E+02	5.15E+02	4.13E+02	1.83E+01
F_{29}	4.98E+02	1.03E+03	8.03E+02	1.24E+02
F_{30}	8.39E+04	3.45E+06	1.64E+05	6.29E+05

equivalent to each other. From the results of last row of Table V, it is imperative that here also the proposed algorithm is better than CV1.0, SaDE, CVnew, JADE and SHADE and is highly competitive when compared to MVMO. Thus we can say that the proposed algorithm is highly competitive and future modification in the same approach may lead to much better results. Also here CS can be considered as equally competitive in line with DE and exceptional performance can also be had from this algorithm also if serious efforts are put forth for its modifications.

V. RESULTS FOR CEC 2020 BENCHMARK PROBLEMS

The CEC 2020 benchmark set is the most recently introduced dataset in the field of numerical optimization [34]. This dataset consists of ten highly challenging optimization problems with one unimodal function, three multi-modal functions, three hybrid functions and three composite functions.

TABLE V
STATISTICAL RESULTS OF PROPOSED ALGORITHM IN COMPARISON TO THE STATE-OF-THE-ART ALGORITHMS

	SaDE	JADE	SHADE	MVMO	CV1.0	CV_{new}	CSsin
F_1	1.21E+03 (1.97E+03) +	5.23E-14 (2.51E-14) +	0.00E+00 (0.00E+00) +	1.33E-05 (5.60E-06) +	1.00E+10 (0.00E+00) =	1.00E+10 (0.00E+00) =	1.00E+10 (0.00E+00)
F_2	9.27E+01 (4.12E+01) +	1.31E+13 (8.53E+13) -	1.08E+12 (4.39E+12) -	1.80E+17 (1.27E+18) -	1.00E+10 (0.00E+00) =	1.00E+10 (0.00E+00) =	1.00E+10 (0.00E+00)
F_3	2.71E+02 (8.28E+02) +	1.77E+04 (3.70E+04) -	0.00E+00 (0.00E+00) +	5.30E-07 (1.09E-07) +	1.95E+04 (6.27E+03) -	8.71E+04 (4.08E+03) -	1.07E+04 (6.68E+03)
F_4	8.92E+01 (4.21E+01) -	4.96E+01 (4.71E+01) -	5.68E+01 (8.80E+00) -	3.58E+01 (3.66E+01) -	1.16E+02 (6.27E+03) +	2.67E+01 (5.92E+00) -	1.88E+01 (3.45E+01)
F_5	9.23E+01 (1.86E+01) +	5.42E+01 (8.80E+00) +	3.28E+01 (5.03E+00) +	8.07E+01 (1.64E+01) +	3.41E+02 (8.02E+01) -	2.39E+02 (3.80E+01) +	3.09E+02 (2.10E+01)
F_6	7.43E-03 (2.35E-02) +	1.44E-13 (9.11E-14) +	8.38E-04 (1.01E-03) +	5.43E-03 (3.30E-03) +	4.85E+01 4.85E+01 -	4.07E+01 (8.14E+00) -	1.00E+01 (5.20E+00)
F_7	1.40E+02 (1.97E+01) -	1.01E+02 (6.48E+00) +	8.09E+01 (3.78E+00) +	1.23E+02 (1.27E+01) +	2.74E+02 (7.29E+01) -	2.22E+02 (3.49E+01) -	1.39E+02 (9.71E+01)
F_8	9.42E+01 (1.77E+01) +	5.52E+01 (7.76E+00) +	3.23E+01 (3.82E+00) +	7.59E+01 (1.61E+01) +	3.29E+02 (7.29E+01) +	2.50E+02 (4.51E+01) +	3.17E+02 (2.43E+01)
F_9	4.83E+01 (6.29E+01) +	1.17E+00 (1.31E+00) +	1.11E+00 (9.37E-01) +	7.38E+00 (5.77E+00) +	1.00E+04 (2.90E+03) =	1.06E+04 (3.10E+03)	1.11E+04 (1.00E+03)
F_{10}	6.60E+03 (1.63E+03) -	3.75E+03 (2.54E+02) +	3.34E+03 (2.94E+02) +	3.49E+03 (4.31E+02) +	7.10E+03 (5.34E+02) -	6.09E+03 (3.55E+02) -	4.97E+03 (5.83E+02)
F_{11}	1.09E+02 (3.54E+01) -	1.36E+02 (3.39E+01) -	1.20E+02 (2.93E+01) -	4.74E+01 (8.72E+00) -	1.66E+02 (3.38E+01) -	1.18E+02 (1.91E+01) -	1.17E+01 (2.91E+01)
F_{12}	1.11E+05 (6.20E+04) +	5.14E+03 (3.32E+03) +	5.13E+03 (2.87E+03) +	1.29E+03 (2.79E+02) +	1.00E+10 (0.00E+00) =	1.00E+10 (0.00E+00) =	1.00E+10 (0.00E+00)
F_{13}	1.21E+03 (1.45E+03) +	3.03E+02 (2.69E+02) +	2.65E+02 (1.49E+02) +	4.37E+01 (1.76E+01) +	1.00E+10 (0.00E+00) =	9.80E+09 (1.40E+09) +	1.10E+10 (0.00E+00)
F_{14}	2.18E+03 (2.20E+03) +	1.05E+04 (3.11E+04) +	2.15E+02 (7.29E+01) +	4.85E+01 (1.21E+01) +	2.05E+02 (2.13E+01) +	3.98E+01 (1.62E+01) +	2.23E+04 (1.72E+04)
F_{15}	3.35E+03 (2.79E+03) +	3.49E+02 (4.42E+02) +	3.22E+02 (1.42E+02) +	4.46E+01 (1.12E+01) +	1.37E+09 (3.47E+09) +	2.85E+02 (3.54E+02) +	1.13E+04 (6.02E+03)
F_{16}	8.17E+02 (2.34E+02) -	8.56E+02 (1.75E+02) -	7.33E+02 (1.88E+02) -	8.40E+02 (1.93E+02) -	1.53E+03 (2.74E+02) -	1.44E+03 (2.10E+02) -	7.23E+02 (1.79E+02)
F_{17}	5.08E+02 (1.53E+02) -	6.00E+02 (1.21E+02) -	5.16E+02 (1.11E+02) -	5.19E+02 (1.33E+02) -	1.25E+03 (1.85E+02) -	1.13E+02 (1.92E+02) +	1.50E+02 (1.16E+02)
F_{18}	3.24E+04 (1.68E+04) +	1.89E+02 (1.25E+02) +	1.89E+02 (1.03E+02) +	4.17E+01 (1.94E+01) +	5.21E+02 (1.19E+02) +	1.51E+02 (4.43E+01) +	1.73E+05 (7.91E+04)
F_{19}	1.13E+04 (1.68E+04) -	3.24E+02 (1.25E+03) +	1.59E+02 (568E+01) +	1.73E+01 (5.13E+00) +	1.73E+02 (4.17E+02) +	5.57E+01 (1.10E+01) +	5.84E+03 (3.15E+03)
F_{20}	3.52E+02 (1.50E+02) -	4.38E+02 (1.33E+02) -	3.33E+02 (1.20E+02) -	3.29E+02 (1.47E+02) -	1.05E+03 (2.14E+02) -	2.81E+02 (1.65E+02) -	2.31E+02 (9.73E+01)
F_{21}	2.87E+02 (1.36E+01) -	2.51E+02 (9.63E+00) -	2.33E+02 (5.11E+00) -	2.77E+02 (1.60E+01) -	5.41E+02 (6.27E+01) -	1.18E+02 (8.77E+01) +	1.57E+02 (9.74E+01)
F_{22}	2.92E+03 (3.24E+03) -	3.33E+03 (1.80E+03) -	3.17E+03 (1.55E+03) -	3.26E+03 (1.71E+03) -	7.33E+03 (1.99E+03) -	5.77E+03 (3.64E+02) -	1.00E+02 (3.91E-01)
F_{23}	5.22E+02 (2.05E+01) -	4.79E+02 (1.17E+01) -	4.59E+02 (8.75E+00) -	5.04E+02 (1.71E+03) +	7.74E+02 (8.06E+01) +	1.87E+02 (5.11E+01) -	4.51E+02 (7.88E+01)
F_{24}	5.89E+02 (1.86E+01) +	5.31E+02 (7.62E+00) +	5.31E+02 (7.45E+00) +	5.83E+02 (1.69E+01) +	8.32E+02 (1.21E+01) -	3.25E+02 (8,95E+01) +	6.87E+02 (3.57E+01)
F_{25}	5.71E+02 (3.05E+01) -	5.19E+02 (3.48E+01) -	5.06E+02 (3.64E+01) -	5.09E+02 (3.12E+01) -	5.43E+02 (1.51E+01) -	4.70E+02 (2.26E+01) -	4.26E+02 (2.08E+01)
F_{26}	2.52E+03 (3.37E+02) -	1.61E+03 (1.21E+02) -	1.41E+03 (9.78E+01) -	1.93E+03 (2.86E+02) -	2.48E+03 (1.88E+03) -	1.16E+03 (1.56E+03) -	3.00E+02 (4.57E-02)
F_{27}	7.10E+02 (6.65E+01) -	5.50E+02 (2.34E+01) +	5.49E+02 (2.78E+01) +	5.43E+02 (1.75E+01) +	7.38E+02 (8.21E+01) -	4.53E+02 (7.17E+01) -	5.97E+02 (3.22E+01)
F_{28}	4.99E+02 (1.53E+01) -	4.91E+02 (2.08E+01) -	4.79E+02 (2.41E+01) -	4.64E+02 (1.50E+01) -	4.94E+02 (1.93E+01) -	4.58E+02 (2.33E-01) -	4.13E+02 (1.83E+01)
F_{29}	5.11E+02 (1.37E+02) +	4.77E+02 (8.06E+01) +	4.87E+02 (1.05E+02) +	4.89E+02 (1.40E+01) +	1.69E+03 (2.29E+02) -	1.45E+03 (1.68E+02) -	8.03E+02 (1.24E+02)
F_{30}	8.07E+05 (8.33E+04) -	6.68E+05 (9.25E+04) -	6.82E+05 (8.51E+04) -	5.81E+05 (1.02E+04) -	4.64E+06 (8.59E+06) -	6.02E+05 (2.99E+04) -	1.64E+05 (6.25E+05)
$w/t/l$	13/0/17	14/0/16	14/0/16	14/0/16	8/3/19	10/3/17	

TABLE VI
STATISTICAL RESULTS OF PROPOSED ALGORITHM FOR CEC 2020 BENCHMARK PROBLEMS

Dimension	Function	Best	Worst	Mean	Median	Std
D=5	F1	0.00E+00	1.36E-06	2.81E-07	2.06E-07	3.01E-07
	F2	1.08E-04	1.19E+02	8.75E+00	2.49E-01	2.81E+01
	F3	5.14E+00	7.58E+00	5.65E+00	5.68E+00	4.74E-01
	F4	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	F5	9.45E-06	3.80E-03	3.94E-04	1.67E-04	6.52E-04
	F6	6.67E-05	1.90E-03	5.77E-04	4.68E-04	4.14E-04
	F7	3.90E-07	1.49E-05	4.18E-06	2.29E-06	3.94E-06
	F8	0.00E+00	1.00E+02	3.67E+01	4.87E-06	4.77E+01
	F9	2.74E-05	1.00E+02	1.17E+01	1.98E-03	3.25E+01
	F10	1.00E+02	3.47E+02	2.88E+02	3.00E+02	7.00E+01
D=10	F1	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	F2	4.37E-01	4.02E+01	1.58E+01	1.12E+01	1.21E+01
	F3	1.11E+01	1.71E+01	1.39E+01	1.40E+01	1.71E+00
	F4	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	F5	1.99E+00	9.12E+00	4.43E+00	4.33E+00	1.82E+00
	F6	3.07E-02	3.85E-01	1.70E-01	1.50E-01	1.20E-01
	F7	3.65E-02	4.61E-01	1.55E-01	1.22E-01	9.95E-02
	F8	1.84E+01	1.00E+02	8.01E+01	1.00E+02	3.18E+01
	F9	1.00E+02	1.00E+02	1.00E+02	1.00E+02	1.38E-13
	F10	1.00E+02	3.97E+02	3.77E+02	3.97E+02	7.55E+01
D=15	F1	0.00E+00	1.00E+10	3.33E+08	0.00E+00	1.82E+09
	F2	4.72E+00	2.41E+02	7.25E+01	4.82E+01	5.99E+01
	F3	1.64E+01	2.11E+01	1.80E+01	1.76E+01	1.25E+00
	F4	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	F5	1.24E+00	2.57E+01	1.30E+01	1.34E+01	6.19E+00
	F6	5.23E-02	8.70E+00	1.44E+00	3.83E-01	2.85E+00
	F7	4.93E-01	1.24E+00	8.85E-01	8.98E-01	1.96E-01
	F8	0.00E+00	1.00E+02	8.75E+01	1.00E+02	2.76E+01
	F9	1.00E+02	1.00E+02	1.00E+02	1.00E+02	2.30E-13
	F10	4.00E+02	4.00E+02	4.00E+02	4.00E+02	0.00E+00
D=20	F1	0.00E+00	1.00E+10	9.33E+09	1.00E+10	2.53E+09
	F2	3.54E+00	2.75E+02	9.83E+01	1.27E+02	8.33E+01
	F3	2.12E+01	3.17E+01	2.55E+01	2.53E+01	2.27E+00
	F4	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	F5	1.96E+01	2.22E+02	1.16E+02	1.38E+02	6.34E+01
	F6	2.87E-01	1.42E+00	6.72E-01	6.58E-01	82.23E-01
	F7	7.90E-01	1.03E+01	2.62E+00	1.96E+00	2.26E+00
	F8	6.93E+01	1.00E+02	9.89E+01	1.00E+02	5.59E+00
	F9	1.00E+02	2.00E+02	1.03E+02	1.00E+02	1.82E+01
	F10	3.99E+02	4.00E+02	3.99E+02	3.99E+02	2.44E-01

The experimental setting used for testing the proposed CSsin algorithm is similar as given by [34]. For the 10 minimization problems, the experiments are performed 30 times for a variable dimension (D) size of 5, 10, 15 and 20. The maximum number of function evaluations for $D = 5$ is $50,000$, for $D = 10$ is $1,000,000$, for $D = 15$ is $3,000,000$ and for $D = 20$ is $10,000,000$ with a search range of $[-100, 100]^D$ for all the test functions under consideration [34]. This is the first termination criteria while the second being the error value smaller than 10^{-8}. The authors have used the same parameter setting for CEC 2019 as used for evaluating the performance of proposed CSsin algorithm for CEC 2017 benchmarks. From the experimental results, it has been found that CSsin algorithm performs better for $D = 5$, 10 whereas provides competitive results for $D = 15$ and 20.

VI. CONCLUSIONS

This paper presents a new version of cuckoo search algorithm namely CSsin algorithm and its application to CEC2017 and CEC2020 benchmark problems. The new algorithm employs adaptive parameters including dual search strategy to enhance exploration and exploitation properties, linearly decreasing switch probability to balance between local and global search, and linearly decreasing population size to reduce the computational burden. For performance evaluation, the CSsin algorithm is applied to CEC2017 benchmark problems and compared with SaDE, JADE, SHADE, MVMO, CV1.0 and CVnew algorithms. All these algorithms are highly efficient and comparitive study show that the newly proposed CSsin algorithm is highly competitve and hence can be used for real world optimization problems.

REFERENCES

[1] Yang, Xin-She, and Suash Deb. "Cuckoo search via Lévy flights." 2009 World Congress on Nature & Biologically Inspired Computing (NaBIC). IEEE, 2009.

[2] Črepinšek, Matej, Shih-Hsi Liu, and Marjan Mernik. "Exploration and exploitation in evolutionary algorithms: A survey." ACM Computing Surveys (CSUR) 45.3 (2013): 35.

[3] Salgotra, Rohit, and Urvinder Singh. "Application of mutation operators to flower pollination algorithm." Expert Systems with Applications 79 (2017): 112-129.

[4] Fister, Iztok, Xin-She Yang, and Dušan Fister. "Cuckoo search: a brief literature review." Cuckoo search and firefly algorithm. Springer, Cham, 2014. 49-62.

[5] Yang, X. S., Cui, Z., Xiao, R., Gandomi, A. H., and Karamanoglu, M. (Eds.). (2013). Swarm intelligence and bio-inspired computation: theory and applications. Newnes.

[6] Walton, S., et al. "Modified cuckoo search: a new gradient free optimisation algorithm." Chaos, Solitons & Fractals 44.9 (2011): 710-718.

[7] Zhao, Pengjun, and Huirong Li. "Opposition-based Cuckoo search algorithm for optimization problems." 2012 Fifth International Symposium on Computational Intelligence and Design. Vol. 1. IEEE, 2012.

[8] Li, Xiangtao, and Minghao Yin. "Modified cuckoo search algorithm with self adaptive parameter method." Information Sciences 298 (2015): 80-97.

[9] Li, Xiangtao, Jianan Wang, and Minghao Yin. "Enhancing the performance of cuckoo search algorithm using orthogonal learning method." Neural Computing and Applications 24.6 (2014): 1233-1247.

[10] Huang, Li, et al. "Chaos-enhanced Cuckoo search optimization algorithms for global optimization." Applied Mathematical Modelling 40.5-6 (2016): 3860-3875.

[11] Wang, Jie-sheng, et al. "Features extraction of flotation froth images and BP neural network soft-sensor model of concentrate grade optimized by shuffled cuckoo searching algorithm." The Scientific World Journal 2014 (2014).

[12] Salgotra, Rohit, Urvinder Singh, and Sriparna Saha. "Improved Cuckoo Search with Better Search Capabilities for Solving CEC2017 Benchmark Problems." 2018 IEEE Congress on Evolutionary Computation (CEC). IEEE, 2018.

[13] Nguyen, Thuan Thanh, Anh Viet Truong, and Tuan Anh Phung. "A novel method based on adaptive cuckoo search for optimal network reconfiguration and distributed generation allocation in distribution network." International Journal of Electrical Power & Energy Systems 78 (2016): 801-815.

[14] Dey, Nilanjan, et al. "Optimisation of scaling factors in electrocardiogram signal watermarking using cuckoo search." International Journal of Bio-Inspired Computation 5.5 (2013): 315-326.

[15] Bindu, A. Hima, and M. Damodar Reddy. "Economic load dispatch using cuckoo search algorithm." Int. Journal Of Engineering Research and Apllications 3.4 (2013): 498-502.

[16] Kumar, Manjeet, and Tarun Kumar Rawat. "Optimal design of FIR fractional order differentiator using cuckoo search algorithm." Expert Systems with Applications 42.7 (2015): 3433-3449.

[17] Gandomi, A. H., Yang, X. S., Talatahari, S., a Alavi, A. H. (2013). Metaheuristic algorithms in modeling and optimization. Metaheuristic applications in structures and infrastructures, 1-24.

[18] Gandomi, A. H., Yang, X. S., and Alavi, A. H. (2013). Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems. Engineering with computers, 29(1), 17-35.

[19] Cobos, Carlos, et al. "Clustering of web search results based on the cuckoo search algorithm and Balanced Bayesian Information Criterion." Information Sciences 281 (2014): 248-264.

[20] Wang, Gai-Ge, et al. "Hybridizing harmony search algorithm with cuckoo search for global numerical optimization." Soft Computing 20.1 (2016): 273-285.

[21] Salgotra, Rohit, and Urvinder Singh. "The naked mole-rat algorithm." Neural Computing and Applications 31.12 (2019): 8837-8857.

[22] Salgotra, Rohit, Urvinder Singh, and Sriparna Saha. "New cuckoo search algorithms with enhanced exploration and exploitation properties." Expert Systems with Applications 95 (2018): 384-420.

[23] Wu, Guohua, R. Mallipeddi, and P. N. Suganthan. ”Problem definitions and evaluation criteria for the CEC 2017 competition on constrained real-parameter optimization.” National University of Defense Technology, Changsha, Hunan, PR China and Kyungpook National University, Daegu, South Korea and Nanyang Technological University, Singapore, Technical Report (2017).

[24] Storn, Rainer, and Kenneth Price. ”Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces.” Journal of global optimization 11.4 (1997): 341-359.

[25] Mirjalili, Seyedali, Seyed Mohammad Mirjalili, and Andrew Lewis. ”Grey wolf optimizer.” Advances in engineering software 69 (2014): 46-61.

[26] Awad, Noor H., Mostafa Z. Ali, and Ponnuthurai N. Suganthan. ”Ensemble sinusoidal differential covariance matrix adaptation with Euclidean neighborhood for solving CEC2017 benchmark problems.” 2017 IEEE Congress on Evolutionary Computation (CEC). IEEE, 2017.

[27] Yao, Xin, Yong Liu, and Guangming Lin. ”Evolutionary programming made faster.” IEEE Transactions on Evolutionary computation 3.2 (1999): 82-102.

[28] Tanabe, Ryoji, and Alex S. Fukunaga. ”Improving the search performance of SHADE using linear population size reduction.” 2014 IEEE congress on evolutionary computation (CEC). IEEE, 2014.

[29] Qin, A. Kai, Vicky Ling Huang, and Ponnuthurai N. Suganthan. ”Differential evolution algorithm with strategy adaptation for global numerical optimization.” IEEE transactions on Evolutionary Computation 13.2 (2008): 398-417.

[30] Zhang, Jingqiao, and Arthur C. Sanderson. ”JADE: adaptive differential evolution with optional external archive.” IEEE Transactions on evolutionary computation 13.5 (2009): 945-958.

[31] Erlich, István, et al. ”Evaluating the mean-variance mapping optimization on the IEEE-CEC 2014 test suite.” 2014 IEEE congress on evolutionary computation (CEC). IEEE, 2014.

[32] Liang, J. J., B. Y. Qu, and P. N. Suganthan. ”Problem definitions and evaluation criteria for the CEC 2014 special session and competition on single objective real-parameter numerical optimization.” Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore 635 (2013).

[33] Derrac, Joaquín, et al. ”A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms.” Swarm and Evolutionary Computation 1.1 (2011): 3-18.

[34] C. T. Yue, K. V. Price, P. N. Suganthan, J. J. Liang, M. Z. Ali, B. Y. Qu, N. H. Awad, and Partha P Biswas , Problem Definitions and Evaluation Criteria for the CEC 2020 Special Session and Competition on Single Objective Bound Constrained Numerical Optimization, Technical Report 2019 11, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China And Technical Report, Nanyang Technological University, Singapore