ISSN: 2229-6956 (ONLINE)
DOI: 10.21917/ijsc.2023.0402

META HEURISTIC OPTIMIZATION APPROACH FOR CMOS BASED ANALOG CIRCUIT DESIGN AND PERFORMANCE EVALUATION OF EVOLUTIONARY ALGORITHMS

Sureshbhai L. Bharvad[1], Pankaj P. Prajapati[2] and Anilkumar J. Kshatriya[3]
[1]*Department of Electronics and Communication Engineering, Gujarat Technological University, India*
[2,3]*Department of Electronics and Communication Engineering, L D College of Engineering, India*

Abstract

Manual design of Complementary Metal Oxide Semiconductor (CMOS) based analog circuit design becomes more challenging and tedious task due to very complex physical models and variation in the fabrication process as technology scale down. In this continuously changing era, the demand of mixed signal System on Chip (SoC) increasing day by day which digital and analog circuits integrated on same silicon chip. For the digital circuit design, many mature computer based automated tools have been established and limited research efforts made towards automization of the analog circuit design. This gap opens the ample research space for the researcher in the field of analog circuit design. Automization of analog circuit design makes the mixed signal SoC is the best approach to cope up with this problem, cost considerations and the time to market. This motivates the analog circuit designer to explore more automated computer aided tools in the field of analog circuit design. In this review paper, performance evaluation of various evolutionary algorithms is compared. The comparison includes most used Differential Evolution (DE) algorithm, Cuckoo Search (CS) algorithm, Particle Swarm Optimization (PSO) algorithm, hybrid CSPSO algorithm. The performance evaluations of these algorithms are compared with the different standard benchmark functions and the convergence graphs of these standard benchmark functions are compared to test number of runs with respect to number of iterations.

Keywords:

Automation of Analog Circuit Design, Optimization, DE Algorithm, PSO Algorithm, Hybrid CSPSO

1. INTRODUCTION

The real world is analog in nature and modern digital systems are requires an integrated analog part built on the same chip to improve speed and reduce power dissipation. The integration of digital and analog systems is more common in the modern System-on-Chips (SoCs) and nearly 75 % of SoCs contain an analog part to communicate with the outside world [1]. Complementary Metal Oxide Semiconductor (CMOS) based design of analog circuit is a more challenging task to meet the desired specifications. Determination of the values of the design parameters of a circuit is known as circuit sizing (schematic-level design) which controls the overall performance of the circuit. In the schematic level design flow, numbers of design parameters are requiring to be adjusted by a circuit designer to obtain the desired specifications for a given circuit [2]. Conventional analog circuit is designed by converting device behaviour in the general analytic equations and these equations need to be simulated to check whether the circuit works as per estimation or not. This is an iterative, time-consuming, and tedious job and there is no assurance about optimum circuit design in terms of channel length, gain, bandwidth, slew rate, power consumption etc. One more aspect that designer needs to be considered is, given analog circuit requires to meet more than one design metrics and most of the design metrics are trade-off with each other which turns the analog circuit design task into multi-objective optimization problem. The Fig.1 shows an analog design octagon which illustrates the trade-off with typical design metrics.

Nowadays much research is being spent to develop a novel methodology that automates the analog circuit design flow. Automization of analog circuit can save a vast portion of the overall design cycle time and reduces the cost of IC. Many research institutions and researchers across the world are working on this research problem and have been trying to develop a Computer Aided Design (CAD) tool which can handle all design problems in short period of time.

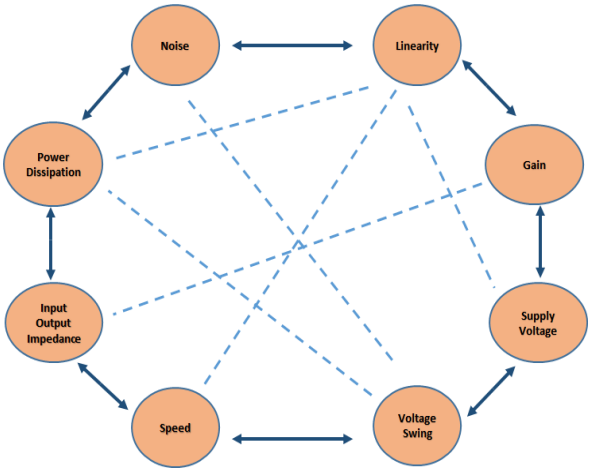

Fig.1. Analog circuit design octagon [3]

For the automated design of analog circuit at schematic level many techniques have presented in the literature. These techniques mainly classify in to two categories. (1) The Knowledge based approach and (2) The Optimization based approach. In the knowledge-based approach, the circuit designer generates synthesis rules based on the expertise and experience in the field of the analog circuit design and these rules are incorporated into an algorithm to find the solution of design problem [4]. This approach makes the analog circuit design tedious, time consuming and limited to few circuit topologies. The optimization-based approach uses a search algorithm that forces the value of design parameters towards a solution specific provided by the user [5]. The optimization-based approach is highly reliable and more accurate for the analog circuit design which is independent of topologies of the circuit.

1.1 OPTIMIZATION OF ANALOG CIRCUIT DESIGN

Optimization is a process of finding the best possible solution for a given design problem based on the given parameters. It is the field of computational science which deals with the finding out the best possible solution for the given design problem through trial-and-error method [6]. This optimization method applied on the analog circuit design to meets the design specifications by optimizing the design parameters through optimization algorithm. The main design parameters of Complementary Metal Oxide Semiconductor (CMOS) based analog circuits may be the width (*W*), length (*L*), resistor (*R*), Capacitor (*C*) and biasing current (I_{bias}). Optimization is an iterative process where the values of design parameters are updated until the optimal result obtained [7]. Desired specifications for a particular circuit topology can be represented by Eq.(1),

$$Spec_i = f(X,Y),\ i=1,2,3,\ldots N \tag{1}$$

The Eq.(1) shows the desired specifications, represented by $Spec_i$, vector *X* indicates the design parameters values, vector *Y* indicates supply voltage, desired temperature, process information, etc., *N* represent the number of desired specifications of given problem.

Let, consider a circuit which has *p* number of Metal Oxide Semiconductor (MOS) transistors, *q* number of resistors, *r* number of capacitors, and reference current source I_{ref}. The parameters values are allowed within a specified search space such that it satisfy the constraints limit of the given vector *Y*. The Eq.(2) represents the analog circuit design problem [7]. Which find the values of W_i and L_i of transistor, R_j, C_k, and I_{ref}.

$$\begin{pmatrix} spec_1 \le value_1 \text{ or } spec_1 \ge value_1 \\ spec_2 \le value_2 \text{ or } spec_2 \ge value_2 \\ spec_3 \le value_3 \text{ or } spec_3 \ge value_3 \\ . \\ . \\ . \\ spec_N \le value_N \text{ or } spec_N \ge value_N \end{pmatrix} \tag{2}$$

Subjected to

$$\begin{pmatrix} L_{\min} \le L_i \le L_{\max}, i = 1,2,3,\ldots, p \\ W_{\min} \le W_i \le W_{\max}, i = 1,2,3,\ldots, p \\ R_{\min} \le R_j \le R_{\max}, j = 1,2,3,\ldots, q \\ C_{\min} \le C_k \le C_{\max}, k = 1,2,3,\ldots, r \\ I_{\min} \le I_{ref} \le I_{\max} \end{pmatrix} \tag{3}$$

For the given vector *Y*, many optimization methods are explored for the atomization of circuit sizing in the past and recent time also. In the literature many optimization methods have been proposed and mainly classified as: gradient-based optimization (also known as classical optimization) techniques, evolutionary optimization (also known as meta-heuristic) optimization techniques, and convex optimization techniques [8]. These techniques require computation of the derivatives in each iteration and good preliminary estimate for the given design variable. These methods require good initial guess close to global optimum solution, otherwise stick to a local optimal solution for a Non deterministic Polynomial (NP) hard type problem, with non-linear objective functions and the large number of variables. Sequential quadratic programming [9], steepest descent method [10], Levenberg-Marquardt method [11], and phase I-II-III feasible directions method [12], are the examples of gradient-based optimization techniques.

The convex optimization techniques give the globally optimal solution but require good knowledge of MOSFET device physics in which behaviour of device represented by very complex quadratic mathematical equations. This would be very tough reviewing the current state-of-the-art complex modern MOSFET device models [13]. Geometric programming [14], interior-point algorithm [15] are the examples of convex optimization techniques.

The evolutionary optimization-based techniques used to solve multimodal, multi objective optimization design problem and explore the solution space very robustly [16]. It also not requires the familiarity of analog circuit design and complex physical models, as it requires in the convex and gradient-based optimization. Also not require computing complex mathematical calculations and it give the global optimum solution. Many evolutionary algorithms proposed and implemented in the literature, Artificial Bee Colony (ABC), Ant Colony Optimization (ACO), Genetic Algorithm (GA), Differential Evolution (DE), Harmony Search (HS), Particle Swarm Optimization (PSO), Tabu Search (TS), Simulated Annealing (SA), Cuckoo Search (CS) are the popular evolutionary algorithm in the field of engineering application and other field. As stated in the No Free LUNCH (NFL) theorem, one single algorithm cannot give best suitable result for the optimization problem, for all the design specifications [17].

Genetic Algorithm (GA), the popular optimization algorithm has been tested for analog circuit design by many researchers [18]. The ACO algorithm is more constituent than the GA with higher convergence speed makes it suitable for transistor sizing problem, so it can replace the GA [19]. The GA, TS, and SA algorithm require more computational time as complexity increases. The ABC algorithm performs well at exploration but poor at exploitation results in less convergence speed for the unimodal problem and trap in local minima for solving the complex multimodal optimization problem as concluded by the authors in [20]. The Cuckoo Search algorithm (CS) is a population-based optimization algorithm developed by Yang and Deb and fewer parameters need to be adjusted. It gives more efficient randomization result compared to PSO and DE algorithms. The performance of PSO, DE, ABC and CS are analysed and compared in [26], and the authors has concluded that the CS algorithm gives more precise and robust results than the ABC and PSO algorithms.

Modern CMOS based analog circuit design uses evolutionary algorithm based optimization approach to solve analog circuit design problem and much research work has been devoted in the field of automated analog circuit design [14]. Three stage CMOS based Miller op-amp and an ultra-low power CMOS based Miller OTA are optimized using PSO, Hierarchical PSO (HPSO) and GA algorithm [33]. Authors have concluded that the Hierarchical PSO (HPSO) algorithm converges better compared to all other evolutionary algorithm. The DE, PSO and ABC algorithms have

been explored to optimize the CMOS based Miller OTA by the authors in [2] and concluded that DE algorithm performed better compared to the PSO, and the ABC algorithm does not meet the require goals. The PSO, DE, ABC and HS algorithms were applied to optimize n^{th} order filters and authors have concluded that the HS performs faster among all but having the maximum fault. The other algorithms converged superiorly [22]. The Modified PSO (MPSO), ABC and standard PSO algorithm has been compared and tested to optimize the two-stage CMOS op-amp and bulk driven OTA. The authors concluded that the Modified PSO (MPSO) performs better compared to other algorithms [23].

2. EVOLUTIONARY ALGORITHMS

The Evolutionary Algorithms are nature inspired and are more efficient, flexible, goal oriented and independent to problem model. Researcher's community working on these optimization techniques and many improved and modified versions are frequently invented in the field of scientific research. This section describes the DE, PSO, CS and modified CSPSO evolutionary algorithms in brief.

2.1 DE ALGORITHM

The Differential Evolution (DE) is an evolutionary algorithm developed by Storn and Price [27]. In this algorithm the formation of a new candidate solution is carried out by calculating a weighted difference between two randomly selected population numbers added to a third randomly selected population number. It uses the crossover, selection, and mutation strategies of used in the GA. For the DE algorithm there are ten mutation techniques are given as per the literature [27].

DE\best\1\exp
DE\rand\1\exp
DE\rand-to-best\1\exp
DE\best2\exp
DE\rand2\exp
DE\best\1\bin
DE\rand\1\bin
DE\rand-to-best\1\bin
DE\best\2\bin
DE\rand\2\bin

For the optimization problem, DE/rand/1/bin is widely used to find the optimum global solution [28].

The DE algorithm consists of two arrays with population size N and dimension D. The N array holds the vector population for the next generation and D array holds the current vector population. N competitions are to be carried out for each generation to obtain the resultant composition of the consecutive generation. The differential vector $(X_{r0} - X_{r1})$ is a pair of randomly chosen vectors X_{r0} and X_{r1} and weighted difference $(X_{r0} - X_{r1})$ is multiplied with weighted factor F, which is further added to another randomly chosen vector X_{r2}. The mathematical representation of this process is represented by the Eq.(4) [28].

$$V = X_{r2} + F*(X_{r0} - X_{r1}) \quad (4)$$

The Eq.(4) represent mutant vector V, target vector X, r_0, $r_1$and r_2 are the random numbers consist range of (1,2,…, N). The weighting factor or mutation scaling factor F is a constant and specified by the user in the range of (0.5, 1.0) which controls the amplification of the vector $(X_{r0} - X_{r1})$ [28].

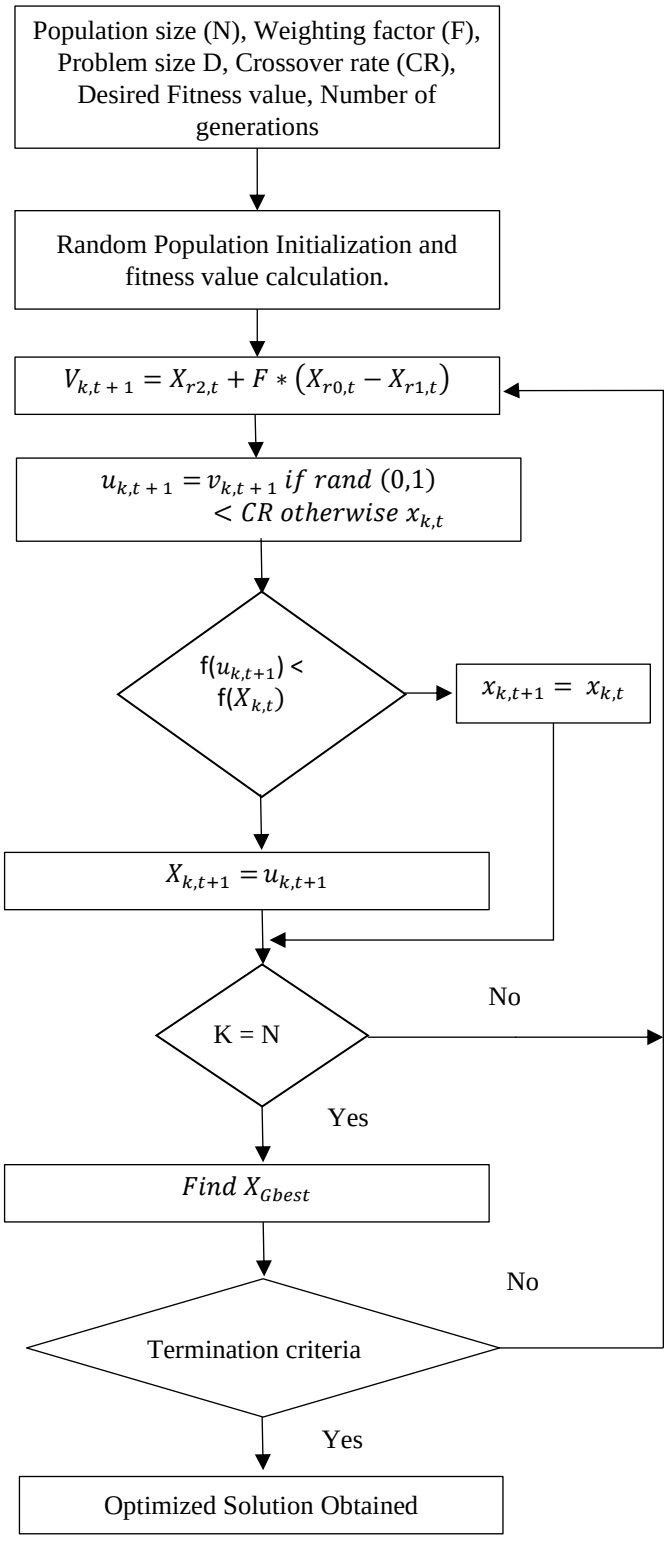

Fig.2. DE algorithm Flowchart [28]

The Fig.2 illustrates the flow diagram of the DE algorithm where X_{Gbest} is generation best vector, u is trial vector, t is a generation number, and k = 1,2,3,…,N. The termination criterion decided by either minimum value of the fitness function or a

maximum number of the generations. The Crossover Rate (CR), weighted factor *F*, and number *N* are the main control parameters of DE algorithm are [28].

2.2 PSO ALGORITHM

The Particle Swarm Optimization (PSO) algorithm is swarm intelligence based evolutionary algorithm and developed by Kennedy and Eberhart [29]. This algorithm gets the best optimum result using a set of birds that fly with different velocities and different positions. According to their past performance and their neighbor in the exploration search space the velocities of all these birds are adjusted. Each solution bird in group is identified as particle. PSO algorithm works in an iterative way and finds the best solution. Suppose *N* number of particles in swam with *D* dimension. The velocity and position of kth particle are described by $V_k = [V_k^1, V_k^2,...,V_k^D]$ and $X_k = [X_k^1, X_k^2,...,X_k^D]$.

The velocity of the k^{th} particle updated after every iteration according to the Eq.(5) [30].

$$V_{kd}^{i+1} = \omega * V_{kd}^{i} + C_1 * rand_1 * \left(pbest_{kd}^{i} - X_{kd}^{i} \right) + C_2 * rand_2 * \left(gbest_{d}^{i} - X_{kd}^{i} \right) \quad (5)$$

In the Eq.(5), range of vector *k* is {1,2,3...,*N*}, range of vector *d* is {1,2,3…,*D*}, range of vector *i* is {1,2,3.... maximum iteration number}, X_{kd}^{i} and V_{kd}^{i} are the position and velocity of k^{th} particle in d^{th} dimension for the i^{th} iteration respectively, $pbest_{kd}^{i}$ is the personal best position of k^{th} particle in d^{th} dimension for the i^{th} iteration, $gbest_{d}^{i}$ is the d^{th} dimension of the global best particle in the swarm for the i^{th} iteration, $rand_1$ and $rand_2$ are uniformly distributed random numbers between 0 and 1, C_1 and C_2 are constants known as acceleration coefficients, ω is known as inertia weight. The value of ω is chosen less than one initially and then with each iteration it reduced linearly. ω controls the influence of the previous direction of displacement. The position of the k^{th} particle in $N \times D$ dimension of the search space can be updated using a Markov chain property as per the Eq.(6), [31].

$$X_{kd}^{i+1} = X_{kd}^{i} + V_{kd}^{i+1} \quad (6)$$

The flow diagram of PSO algorithm is illustrated with steps in Fig.3.

2.3 CS ALGORITHM PSO

The Cuckoo Search (CS) algorithm is a bio-inspired meta-heuristic algorithm developed by Yang and Deb [21]. The CS algorithm is robust and generic compared to PSO, GA, DE, and other meta-heuristic algorithms for global optimization problem due to less number of control parameters and have a fine balance of exploration and exploitation [34].

CS algorithm motivated by the incomparable lifestyle of cuckoo species. Cuckoo breeding behavior and Lévy flight behavior combined to find all optima solution in a search space. CS algorithm starts with a random initial population like other meta-heuristic algorithms, at the same time it explores some kind of selection and/or elitisam as that of HS algorithm [35].

The flow diagram of CS algorithm illustrated in Fig.4 [36]. CS algorithm, each pattern relates to a nest and each individual element of pattern relates to a cuckoo egg. The CS algorithm represented by the Eq. (7) [21].

$$X_{t+1;i} = X_{t;i} + \alpha \otimes Lévy(\lambda) \quad (7)$$

Here, $X_{t;i}$ is the current solution of the i^{th} cuckoo, $X_{t+1;i}$ is the next solution generated for the i^{th} cuckoo with Lévy flight, *t* represents a number of the present generation and the product $\otimes$ indicates the entry-wise multiplication, and $\alpha>0$ is a scaling factor of the step size that depends on scales of the given problem of interest. For small dimension problem, $\alpha=O(L/10)$ is suitable and for large dimension problem, $\alpha=O(L/100)$ is more appropriate [38]. Big O notation is used to represent the time complexity of the algorithm. The characteristics of the scale *L* depend on the problem to be solved. In addition, Lévy(λ) is a random movement using Lévy flight that is comparatively more effective in the long run than a random walk used in other algorithms such as the DE, ABC, PSO etc. This is a global walk intended for exploration or diversification of the search space.

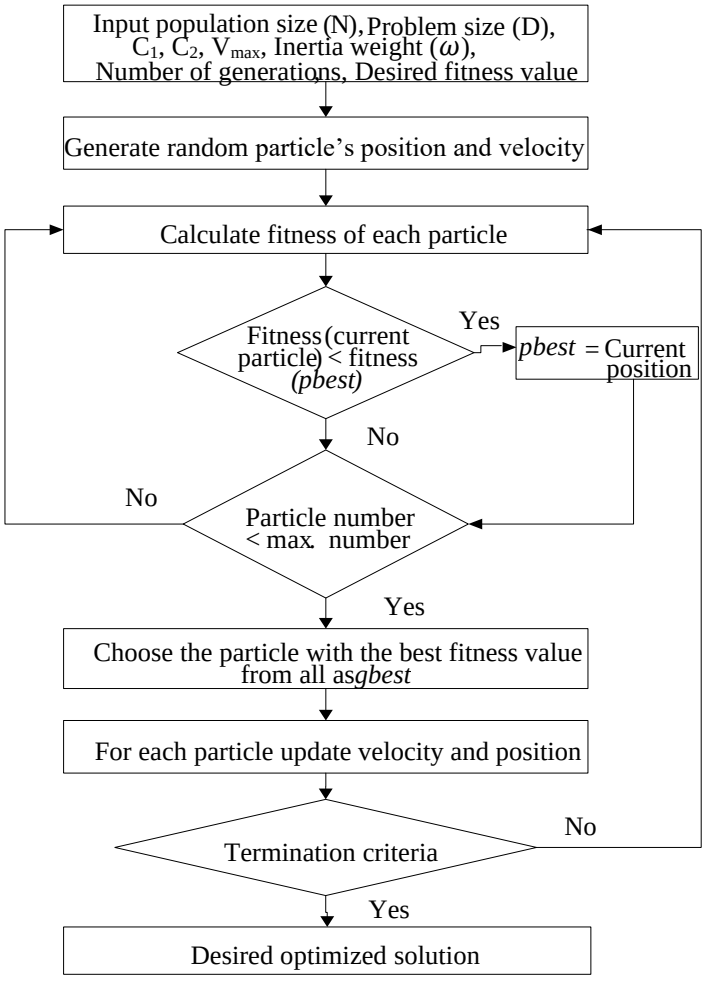

Fig.3. Flowchart of PSO algorithm [29]

Lévy(λ) is derived from a Lévy distribution which has an infinite variance with an infinite mean using the Eq.(8) [21].

$$\text{Lévy} \sim u = t^{-\lambda};\ 1<\lambda\leq 3 \quad (8)$$

In the Eq.8, '∼' means random numbers drawn from the Lévy distribution wherein the step size follows a random walk process with a power-law distribution with heavy-tailed as shown in

Fig.2.4 and λ is the power coefficient [39]. In Fig.5, $P(x)$ is a probability density function of a random variable x.

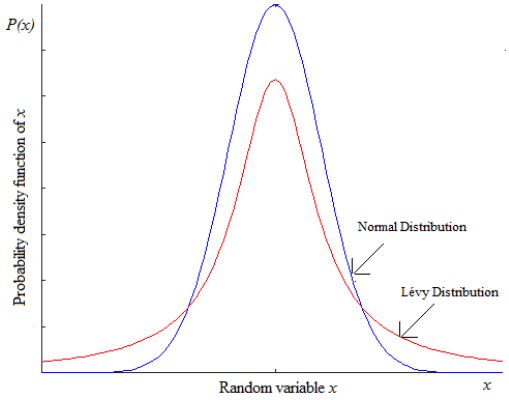

Fig.5. Lévy distribution [35]

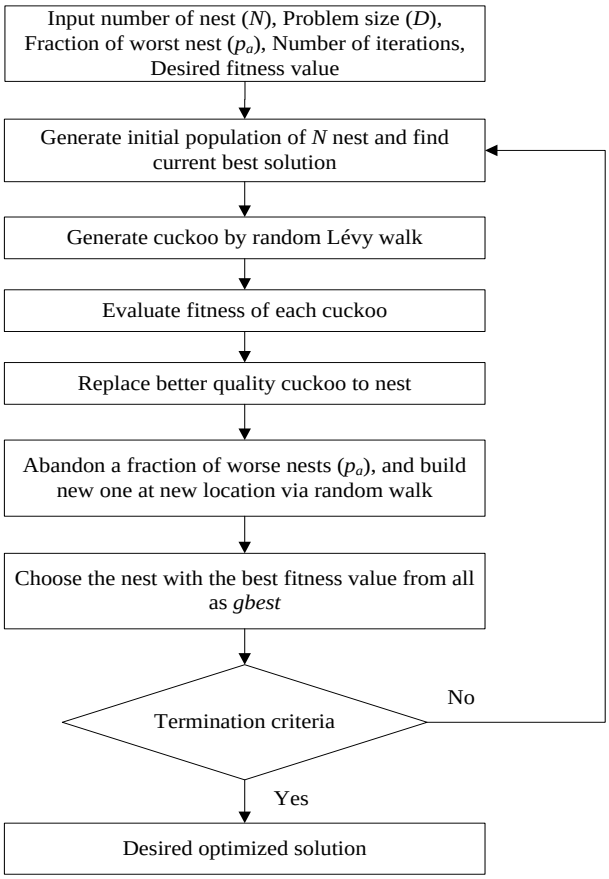

Fig.4. Flowchart of the CS algorithm [21]

Lévy flight has a large coverage range in a search space of the variables and new numerous solutions can be generated by Lévy walk near the best solution and improve the speed of local exploration [40]. To ensure that the CS algorithm will not be trapped in a local optimum solution, a substantial part of the new solutions must be generated through far-field randomization, so that location would be sufficiently far from the current obtained best solution. Thus, steps generated by Lévy walk can have both small and large components, which enable the cuckoo search to do both large-scale explorations and local exploitation [41]. There are a few methods to generate Lévy distribution, but one of the most efficient and yet a straightforward method is the Mantegna algorithm proposed by Mantegna in [42]. This method generates random numbers according to symmetric Lévy stable distribution. The simple way to generate a new solution using the Lévy walk is mathematically expressed by the Eq.(9) [26].

$$X_{t+1} = X_t + step\ size \otimes N(0,1) \quad (9)$$

The Eq.(9) represents the Lévy random walk calculated by Mantegna's algorithm.

$$Step\ size = 0.01*(u/|v|^{\beta}) \otimes (X_t - X_{best}) \quad (10)$$

The Eq.(10) represents calculation of the step size. Where, 0.01 is a factor that control step size of cuckoo walks. X_{best} is the best global solution, X_t is the current solution, step size is the length of walk step, $\otimes$ is entry-wise product, u and v are normally distributed stochastic variables generated from $u \sim N(0, \sigma^2)$ and $v \sim N(0, 1)$, and σ^2 is the variance given by the Eq.(11) [25],

In the Eq.(11) Γ is gamma function which is an extension of the factorial function of positive number, and β is the variable which controls distribution by $0 \leq \beta \leq 2$.

The value of β equal to 1.5 is suggested in [21]. The Eq.(11) represents the calculation of the variance σ^2 as,

$$\sigma^2 = \left\{ \frac{\Gamma(1+\beta)\sin\left(\frac{\pi\beta}{2}\right)}{\Gamma\left(\frac{1+\beta}{2}\right)\beta * 2\left(\frac{\beta-1}{2}\right)} \right\}^{\frac{1}{\beta}} \quad (11)$$

The local random walk intended for exploitation or intensification of the search space is based on the Eq.(12),

$$X_{t+1} = X_t + r \otimes H(p_a - r) \otimes (X_j - X_k) \quad (12)$$

In the Eq.(12), X_j and X_k are two different randomly selected solutions, $H(u)$ is a Heaviside function ($H(u) = 1$ if $u > 0$ and $H(u) = 0$ if $u < 0$)), p_a is the discovery probability of a cuckoo egg, by a host bird which is $\in [0,1]$, r is a random number obtained from uniform distribution in [0,1].

The CS algorithm finds the desired solutions very efficiently for many global optimization problems as presented in the literature and requires smaller number of control parameters. The CS algorithm has a fine balance of exploitation and exploration compared to other algorithms like TS, PSO, SA, GA, ACO, ABC and DE.

3. PERFORMANCE ESTIMATION OF EVOLUTIONARY ALGORITHMS

3.1 INTRODUCTION

It is very difficult to optimize all optimization problems with single algorithm. And to resolve this problem, two or more algorithms can be merged to get the global optimum solution for a broad range of optimization problems [39].This section covers the concept of a hybrid algorithm. Then, benchmark functions which are generally used to test the performance of the optimization algorithms and the parameter settings of each algorithm are reviewed. At the end of this section, the performance of different Evolutionary Algorithms such as the CS, PSO, hybrid CSPSO and DE algorithms are compared by comparing the set of commonly used benchmark functions.

3.2 HYBRID CSPSO ALGORITHM

A hybrid algorithm is a merger of two or more algorithms that run together and complement each other to produce a profitable synergy from their integration [45]. Generally, the result of hybridization makes improvements in form of either accuracy or computational speed [46]. The hybridization aims to combine the advantages of each algorithm, while at the same time trying to minimize any considerable drawbacks [39].

The PSO algorithm is one amongst the most competent optimization algorithms. But the problem is it converges very fast, therefore it has generally an early convergence in complex problems [26].The CS algorithm is applied to solve a number of complex problems and it outperforms other optimization algorithms [38].The CS algorithm may converge slightly slower but it has enhanced explorative skill. Consequently, there is some trade-off between convergence and accuracy. These trade-off leads to the solution diversity that an algorithm can produce in the searching process as illustrated in Fig.6 [47].

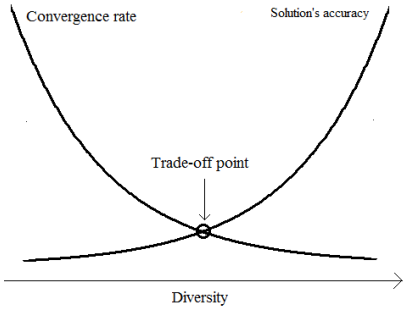

Fig.6. Trade-off between accuracy and convergence [47]

The objective of hybridization is to inform each cuckoo about their position and helps each cuckoo to move to a better position. Fig.7. illustrates the Pseudo code for the CSPSO. In this algorithm, each cuckoo updates its position and velocity according to the PSO algorithm. The hybrid CSPSO algorithm's main step are demonstrated by the flow diagram shown in Fig.8.

The Hybrid CSPSO Algorithm

Input: Nest size (N), Problem dimension (D), Fraction of worst nest (p_a), No. of generations, Desired fitness function value, C_1, C_2, V_{max}, ω;

Output: Optimized solution;

Begin

Objective function f(x),x=(x_1, x_2, x_3…x_D);

Generate initial a population of N host nests x_i(i=1,2,3, ….,N);

While (t<*Max.generation*) or (*stop criteria*);

Get a cuckoo (say i) randomly by Lévy flights;

Evaluate its quality/fitness F_i;

Choose a nest among N (say j) randomly;

If($F_i > F_j$) **then**

Replace j by the new solution;

End If

For each cuckoo update velocity and position according to the PSO algorithm;

Abandon a fraction (p_a) of worse nests and build new ones at new locations;

Keep the best solutions;

Rank the solutions and find the current best;

End while

End

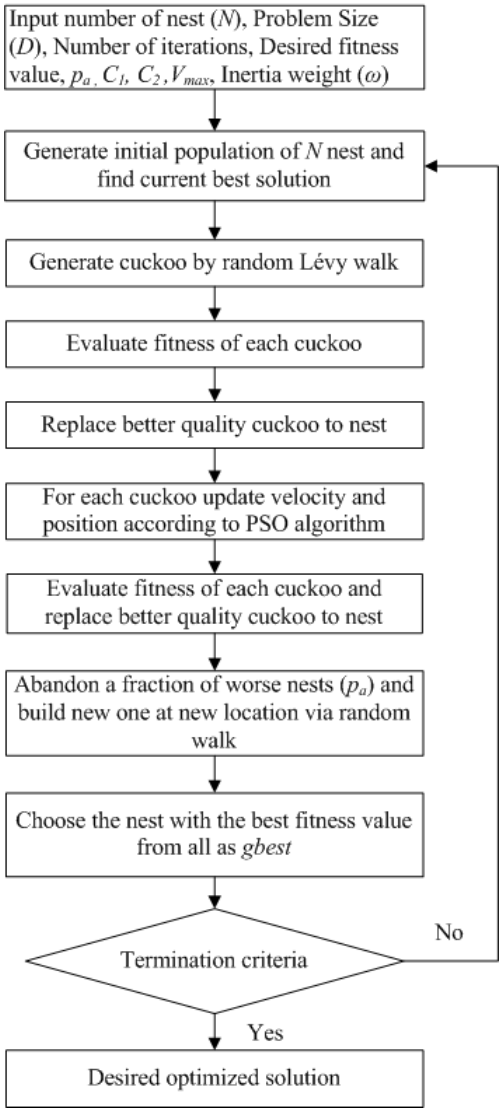

Fig.8. Flowchart of the hybrid CSPSO algorithm [48]

3.3 BENCHMARK FUNCTION

Once the optimization algorithm is implemented, it is assessed through different benchmark functions, which is one of the best primary ways to evaluate and compare the performance of implemented algorithm [49]. If an implemented algorithm produces a satisfactory solution with the benchmark functions, then it is proven to solve the global optimization problems in majority of the cases. There are several different types of such test functions available in the various literatures. The commonly used functions have been selected to estimate the performance of the CS and hybrid CSPSO algorithms. The benchmark functions used for review work are listed with their equation, type, algorithms also have been compared with the PSO and DE algorithms. The

search space of $\boldsymbol{x}$, the global optimum solution of the function (x^*), and the global minimum value of the function $f(x^*)$ in Table.1 [50].

3.4 EXPERIMENTAL SETTINGS AND RESULTS

The CS, CSPSO, PSO, and DE algorithms are implemented in the C programming language which is comparatively faster and use the least memory. These algorithms are repeatedly executed on the GNU Compiler Collection (GCC) to solve the required value for each benchmark function.

The performance of all these Evolutionary Algorithms is evaluated using multimodal as well as unimodal benchmark functions with different problem dimensions (D). Author has performed these experiments on a system having specification of: Intel® core™ i5, 2.40 GHz processor, Internal RAM of 8 GB and Operating System (OS) as an Ubuntu [51].

Every algorithm has its own parameters that affect its performance in terms of processing time and result quality. Depending on the nature of the optimization problem and search, different algorithms have different values for their parameters. In the DE algorithm, parameters CR, F and N are relatively tough to set and some benchmark functions are very sensitive to appropriate settings of these parameters [28].

For the PSO algorithm, more parameters need to be set where in the CS algorithm have fewer parameters need to be set. For the comparative study author has taken, the size of population (N) for the DE algorithm, the number of particles (N) for the PSO algorithm, the number of nests (N) for the CS algorithm, and the number of nests (N) for the CSPSO algorithm have been considered equal, the values of the remaining parameters for each algorithm are also considered the same for all benchmark functions [51]. However, depending on the nature of the optimization problem or the objective function, solution quality can be improved by changing the values of the algorithm's parameters.

The parameters for the different Evolutionary Algorithms are set as follows: For the DE algorithm, $F \in [0,2]$ and the $CR \in [0,1]$ as suggested in [27]. If $F<0.4$ or $F>1.0$, then the DE algorithm is rarely effective as mentioned in [27].

If the population converges prematurely, then F and/or N should be increased. The larger value of CR increases the speed of convergence. Ali and Törn have obtained an optimal value for $CR = 0.5$ empirically, halfway between the two parents [52]. This approach ensures that each parent's component has a 50% chance of being selected to produce a new point. In this literature survey, $F = 0.8$ and $CR = 0.5$ are selected for the DE algorithm as suggested in [52].

The PSO algorithm comparatively has more tuning parameters which greatly influence its performance. As mentioned in [29], a recommended value for C_1 and C_2 is 2. Clerc has reported that C_1 and C_2 are important factors to ensure convergence of the PSO algorithm in [54]. Trelea has tested the different set of parameters for different standard benchmark functions for the PSO algorithm [55]. From his empirical study, he suggested optimum value of $C_1=C_2= 1.49$ gives a higher success rate for the PSO algorithm. Shi and Eberhart have analysed the impact of ω and V_{max} on the performance of the PSO algorithm in [56]. The choice of $V_{max} = x_{max}$ and linear variation of ω from 0.9 to 0.4 with iterations provide good performance on the tested benchmark functions as mentioned in [57].

Yang and Deb have tested CS algorithm for different benchmark functions with different values of population size (N) and p_a. From their empirical study, suggested that N=15 to 40 and $p_a= 0.25$ are sufficient for most of the optimization problems. This literature survey carried out for, N= 30 and $p_a= 0.25$ as suggested in [21].

For the hybrid CSPSO algorithm, Number of nests $N = 30$, $p_a = 0.25$, $C_1 = 1.49$, $C_2 = 1.49$, $V_{max} = 0.1*x_{max}$, $V_{min} = 0.1*x_{min}$, and ω varies linearly from 0.9 to 0.4 with iterations are chosen.

The solution obtained by any Evolutionary Algorithm to solve an objective function $f(x)$ with the solution search space of a variable $[x_{j,min}, x_{j,max}]_{j=1,2,\ldots,D}$ is represented by a $x=(x_1, x_2,\ldots,x_D)$. Here, x_(j,min)is a minimum value of a variable x_j and $x_{j,max}$ is a maximum value of a variable x_j. In the initialization phase, each algorithm randomly generates a solution vector which is sampled from the search space $[x_{j,min}, x_{j,max}]$. The initial values of the j^{th} attributes of the i^{th} pattern have been generated as suggested in the Eq.(13) [59].

$$x_{i,j} = x_{i,j,min} + r * (x_{i,j,max} - x_{i,j,min}) \quad i = 1,2,\ldots,N \tag{13}$$

In the Eq.(13), r represents uniformly distributed random variable with the range (0,1). This randomly generated solution vector r is given to a test function. Based on the results obtains from the test function, the algorithm will update the value of the solution vector. This iterative process continues until termination criteria are not fulfilled. The termination criteria are either a maximum number of iterations or a minimum value of the function. These criteria are selected based on the complexity problem and requirement of quality of solutions. The value of tolerance for the variation of function is considered $\leq 1e^{-6}$. The maximum number of iterations for the given benchmark function testing are considered 30000.

Being a stochastic search process, a statistical study is required to compare the consistency of the different Evolutionary Algorithms and their quality of solutions. Each benchmark function is solved by the CS, PSO, CSPSO and DE algorithms for 100 independent runs with different random seeds. The performance of each algorithm is evaluated based on different performance criteria such as a standard deviation of a function value, success rate, simulation time, an average number of function evaluations and an average number of iterations in the experimental setup [51].

Standard deviation of a function value: The best value of a function $f(x)$ that each algorithm can find is recorded during each run. The maximum, minimum, mean, and standard deviation of the best function values are calculated. The minimum, maximum, mean, and standard deviation of the best function values are denoted as 'Max_f', 'Min_f', '$Mean_f$', and 'SD_f' respectively.

Average number of iterations: The number of iterations is also recorded in different runs when each algorithm finds the targeted value of the function within the maximum number of iterations. Then, the average number of iterations is calculated based on the number of runs. It is denoted as '$Iter_{avg}$'.

Average number of function evaluations: The number of function evaluations is also recorded in different runs when each algorithm finds the targeted value of the function within the termination criteria. Then, the average number of function

evaluations is calculated based on the number of runs. It is denoted as 'FE_{avg}'.

Success rate: The number of successful runs is recorded when the function value reaches the targeted value within a maximum number of iterations. It is denoted as 'S_{rate}'.

Simulation time: The total simulation time is recorded for all runs of the function taken by each algorithm in the respective experiment. It is denoted as 'T_{sim}'.

The performance criteria of each EA for different benchmark functions are tabulated in Table.2. In which, Beale $f_2(x)$, Drop-Wave $f_3(x)$, Easom $f_4(x)$ and Schaffer $f_9(x)$ functions are 2-D optimization problems and Ackley $f_1(x)$, Griewank $f_5(x)$, Lévy $f_6(x)$, Rastrigin $f_7(x)$, Rosenbrock $f_8(x)$, and Sphere $f_{10}(x)$ functions are 10-D optimization problems. In this table, the optimum values of different criteria obtained by the EA are shown in bold letters. Ackley, Griewank, Lévy, Rastrigin, Rosenbrock, and Sphere functions are also solved using each EA independently with 20-D as well as 30-D optimization problems. The performance criteria of each EA for 20-D benchmark functions are tabulated in Table.3.

For 20-D optimization problems, the size of population (N) for the DE algorithm, the number of particles (N) for the PSO algorithm, the number of nests (N) of the CS algorithm, and the number of nests (N) for the CSPSO algorithm are changed to 40 in place of 30 which was set previously in the case of 10-D benchmark functions presented in Table.2. Other parameters of all algorithms are unaltered.

The performance criteria of each EA for 30-D benchmark functions are tabulated in Table.4. For the 30-D optimization problems, the value of N for each algorithm is changed to 60 in place of 40 set previously in the case of 20-D benchmark functions. Other parameters of all algorithms are unchanged [51].

From Table.2-Table.4., it can be concluded that the average number of iterations ($Iter_{avg}$) taken by the CS and CSPSO algorithms to reach the targeted function value for most of the benchmark functions are less compared to the PSO and DE algorithms.The CS and CSPSO algorithms also acquire a lower standard deviation of the function value (SD_f) and achieve a higher success rate (S_{rate}) compared to the DE and PSO algorithms for most of the test functions [51].

Table.1. Standard benchmark functions

Function	**Equation**	**Type**	**Search Space**	**Global Optimum Solution (x^*)**	**$f(x^*)$**
Ackley	$f_1(x)=-20e^{-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D}x_i^2}}-e^{\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)}+20+e$	Multi-Modal	(-32, 32)	(0, …., 0)	0
Beale	$f_2(x)=\left(1.5-x_0+x_0x_1\right)^2+\left(2.25-x_0+x_0x_1^2\right)^2+\left(2.625-x_0+x_0x_1^3\right)^2$	Uni-Modal	(-4.5, 4.5)	(3, 0.5)	0
Drop-Wave	$f_3(x)=-\frac{1+\cos\left(12\sqrt{x_1^2+x_2^2}\right)}{0.5\left(x_1^2+x_2^2\right)+2}$	Multi-Modal	(-5.12, 5.12)	(0, 0)	-1
Easom	$f_4(x)=-\cos(x_0)\cos(x_1)\exp\left(-\left(x_0-\pi\right)^2-\left(x_1-\pi\right)^2\right)$	Uni-Modal	(-100, 100)	(π, π)	-1
Griewank	$f_5(x)=\frac{1}{4000}\sum_{i=1}^{D}x_i^2-\prod_{i=1}^{D}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	Multi-Modal	(-600, 600)	(0, …., 0)	0
Lévy	$f_6(x)=\sin^2(\pi\omega_1)+\sum_{i=1}^{D-1}(\omega_i-1)^2\left[1+10\sin^2(\pi\omega_i+1)\right]+(\omega_D-1)^2\left[1+\sin^2(2\pi\omega_D)\right]$ where $\omega_i=1+\frac{x_i-1}{4}$	Multi-Modal	(-10, 10)	(1, …., 1)	0
Rastrigin	$f_7(x)=10D+\sum_{i=1}^{D}\left[x_i^2-10\cos(2\pi x_i)\right]$	Multi-Modal	(-5.12, 5.12)	(0, …., 0)	0
Rosenbrock	$f_8(x)=\sum_{i=1}^{D}\left[100\left(x_{i+1}-x_i^2\right)^2-\left(x_i-1\right)^2\right]$	Uni-Modal	(-30, 30)	(1, …., 1)	0
Schaffer	$f_9(x)=0.5+\frac{\left(\sin\left[\sqrt{x_i^2-x_2^2}\right]\right)-0.5}{\left(1+10^{-3}\left[x_1^2+x_2^2\right]\right)^2}$	Multi-Modal	(-100, 100)	(0, 0)	0
Sphere	$f_{10}(x)=\sum_{i=1}^{D}x_i^2$	Uni-Modal	(-100, 100)	(0, …., 0)	0

Table.2. Performance evaluation of different EAs with different benchmark functions

Algorithm	**Function**	***D***	Min_f	Max_f	$Mean_f$	SD_f	$Iter_{avg}$	FE_{avg}	S_{rate}	$T_{sim}(s)$
DE	$f_1(x)$	10	9.1e-7	1.7e-5	1.1e-6	2.0e-6	21419	642608	99	110.77
PSO			6.5e-7	19.93	3.9e-1	2.79	4234	127025	98	**24.27**
CS			6.9e-7	1.0e-6	**9.1e-7**	**0.0**	1318	**79121**	**100**	24.39
CSPSO			5.7e-7	1.0e-6	**9.1e-7**	**0.0**	**1090**	98202	**100**	36.70
DE	$f_2(x)$	2	1.0e-6	2.0e-1	9.3e-3	2.4e-2	30000	900030	0	33.86
PSO			2.3e-8	7.6e-1	7.6e-3	7.5e-2	580	17400	99	0.83
CS			1.2e-8	9.9e-7	**5.1e-7**	**0.0**	152	9165	**100**	0.60
CSPSO			1.2e-8	9.9e-7	5.5e-7	**0.0**	**83**	**7524**	**100**	**0.37**
DE	$f_3(x)$	2	1.7e-9	9.8e-7	3.1e-7	0.0	76	2319	100	0.08
PSO			0.0	1.0e-6	3.3e-7	0.0	43	1286	100	**0.04**
CS			3.1e-10	1.0e-6	**3.0e-7**	0.0	9	**576**	100	0.05
CSPSO			2.3e-11	1.0e-6	**3.0e-7**	0.0	**7**	669	100	**0.04**
DE	$f_4(x)$	2	8.9e-6	1.3e-1	7.0e-3	1.5e-2	30000	900030	0	32.04
PSO			5.6e-9	1.0e-6	5.5e-7	**0.0**	**583**	**17491**	**100**	**0.92**
CS			2.0e-8	1.0e-6	**5.2e-7**	**0.0**	944	56701	**100**	5.74
CSPSO			1.9e-8	1.0e-6	**5.2e-7**	**0.0**	764	68759	**100**	5.78
DE	$f_5(x)$	10	9.8e-7	5.2e-1	1.4e-1	1.3e-1	29892	896811	8	128
PSO			6.8e-7	8.4e-2	3.4e-2	1.7e-2	29514	**885544**	2	**111**
CS			6.5e-7	4.6e-2	**1.9e-2**	1.2e-2	27787	1667298	9	526
CSPSO			7.3e-7	4.7e-2	1.7e-3	**1.1e-2**	**27047**	2434260	**12**	219
DE	$f_6(x)$	10	8.3e-5	3.1e-1	2.5e-2	4.3e-2	30000	900030	0	144.62
PSO			4.4e-5	3.99	7.8e-2	5.6e-1	**898**	**26948**	98	**4.91**
CS			3.9e-7	1.0e-6	**8.0e-7**	**0.0**	901	54145	**100**	18.73
CSPSO			2.5e-7	9.9e-1	1.9e-2	1.4e-1	1106	99580	98	29.66
DE	$f_7(x)$	10	9.8e-7	7.65	4.9e-1	1.18	29968	899099	5	85.19
PSO			6.4e-7	29.92	1.94	5.74	18209	**546283**	47	**70.84**
CS			2.1e-7	4.97	6.5e-1	9.4e-1	18618	1117138	**57**	305
CSPSO			4.2e-7	3.98	**6.3e-1**	**8.5e-1**	**18617**	1675621	55	366
DE	$f_8(x)$	10	7.51	933.81	321.59	188.21	30000	900030	0	62.14
PSO			1.0e-6	90002	10813	29242	29892	896789	1	107
CS			4.1e-7	1.0e-6	8.2e-7	**0.0**	1576	**94638**	**100**	**23.01**
CSPSO			2.5e-7	1.0e-6	**8.0e-7**	**0.0**	**1330**	119761	**100**	31.19
DE	$f_9(x)$	2	8.7e-7	1.0e-6	9.6e-7	0.0	5236	157098	100	6.34
PSO			0.0	1.0e-6	**4.5e-7**	0.0	786	**23595**	100	**0.83**
CS			1.8e-9	1.0e-6	5.8e-7	0.0	1240	74484	100	5.72
CSPSO			7.8e-9	1.0e-6	5.5e-7	0.0	**476**	42949	100	2.67
DE	$f_{10}(x)$	10	9.0e-7	1.0e-6	9.9e-7	0.0	4653	139635	100	8.78
PSO			3.2e-7	1.0e-6	8.7e-7	0.0	3102	93065	100	10.47
CS			2.8e-7	1.0e-6	**8.2e-7**	0.0	704	**42248**	100	10.30
CSPSO			3.7e-7	1.0e-6	**8.2e-7**	0.0	**480**	43329	100	**7.85**

Table.3. Performance evaluation of different EAs with 20-D benchmark functions

Algorithm	**Function**	Min_f	Max_f	$Mean_f$	SD_f	$Iter_{avg}$	FE_{avg}	S_{rate}	$T_{sim}(s)$
DE	$f_1(x)$	9.6e-5	4.01	1.1e-6	2.69	30000	1200040	0	308
PSO		8.4e-7	19.95	7.1e-1	3.49	6062	**242468**	96	**85**
CS		7.7e-7	1.16	**2.3e-2**	**1.6e-1**	3752	300242	**98**	191
CSPSO		7.8e-7	1.16	**2.3e-2**	**1.6e-1**	**3116**	373976	**98**	173
DE	$f_6(x)$	9.6e-7	9.5-1	1.5e-1	2.9e-1	17152	686130	69	**178**
PSO		7.6e-7	22.54	2.7e-1	2.24	28243	1129730	7	309
CS		7.1e-7	1.2e-2	5.9e-4	2.2e-3	5362	**429045**	93	264
CSPSO		6.3e-7	7.3e-3	**2.9e-4**	**1.5e-3**	**4103**	492490	**96**	251
DE	$f_7(x)$	6.6e-4	3.2e-1	6.6e-2	**7.2e-2**	30000	1200040	0	365
PSO		5.8e-7	10.12	1.46	2.55	11629	465160	71	166
CS		5.6e-7	1.98	**2.9e-2**	2.2e-1	**2842**	**227442**	**98**	**158**
CSPSO		7.3e-7	1.98	3.8e-1	5.6e-1	11184	1342172	66	813
DE	$f_8(x)$	7.48	30.39	15.74	4.68	30000	1200040	0	**218**
PSO		2.99	63.82	17.80	16.45	30000	**1200000**	0	313
CS		8.2e-7	9.95	**2.23**	**1.91**	**28772**	2297857	**14**	1246
CSPSO		7.9e-7	11.94	2.88	2.01	29117	3494106	7	1492
DE	$f_9(x)$	344.56	2982.78	1014	483	30000	1200040	0	**190**
PSO		9.9e-7	90008	14861	32811	29592	1183698	3	250
CS		5.7e-7	3.99	4.8e-1	1.30	7859	**628766**	88	293
CSPSO		5.5e-7	3.99	**4.4e-1**	**1.25**	**7260**	871286	**89**	459
DE	$f_{10}(x)$	9.8e-7	1.0e-6	1.0e-6	0.0	9696	387884	100	42
PSO		6.1e-7	1.0e-6	**9.2e-7**	0.0	4445	177802	100	**34**
CS		7.3e-7	1.0e-6	9.3e-7	0.0	1676	134152	100	62
CSPSO		5.7e-7	1.0e-6	**9.2e-7**	0.0	**1434**	**172220**	100	69

Table.4. Performance evaluation of different EAs with 30-D benchmark functions

Algorithm	**Function**	Min_f	Max_f	$Mean_f$	SD_f	$Iter_{avg}$	FE_{avg}	S_{rate}	$T_{sim}(s)$
DE	$f_1(x)$	3.30	6.43	4.30	7.7e-1	30000	1800060	0	720
PSO		7.6e-7	19.96	4.41	7.01	12887	773255	71	434
CS		8.5e-7	1.16	4.9e-2	2.1e-1	6469	776356	95	644
CSPSO		8.4e-7	1.0e-6	**9.7e-7**	**0.0**	**2807**	**505453**	**100**	**355**
DE	$f_5(x)$	9.7e-7	9.9-1	2.3e-1	3.8e-1	22611	1356756	74	547
PSO		6.9e-7	23.48	1.84	6.18	24057	1443456	24	590
CS		7.2e-7	1.0e-6	9.5e-7	**0.0**	3802	**456373**	**100**	**420**
CSPSO		7.3e-7	1.0e-6	**9.4e-7**	**0.0**	**3272**	589036	**100**	450
DE	$f_6(x)$	1.3e-3	1.12	1.2e-1	2.3e-1	30000	1800060	0	811
PSO		8.5e-7	38.35	7.48	7.29	25273	1516390	19	780
CS		6.7e-7	1.0e-6	**9.3e-7**	**0.0**	**3710**	**445273**	**100**	**441**
CSPSO		6.7e-7	1.98	3.9e-1	5.8e-1	11835	2130421	65	1703
DE	$f_7(x)$	10.62	30.71	19.23	4.47	30000	1800060	0	**460**
PSO		9.95	154.57	70.44	31.76	30000	1800000	0	764
CS		8.4e-7	9.95	**3.57**	**2.12**	29989	3598790	**2**	3166
CSPSO		8.8e-7	12.94	3.63	2.67	**29952**	5391439	**2**	4150

DE	$f_8(x)$	1281	7125	3466	1055	30000	1800060	0	**371**
PSO		4.3e-6	90072	20247	37064	30000	1800000	0	554
CS		6.5e-7	3.99	4.8e-1	1.30	11696	**1403602**	88	1011
CSPSO		6.8e-7	3.99	**1.6 e-1**	**7.8e-1**	**9443**	1699873	**96**	1059
DE	$f_{10}(x)$	9.9e-7	1.0e-6	1.0e-6	**0.0**	15516	931017	**100**	134
PSO		7.1e-7	20000	800	3059	6934	416015	93	**99**
CS		7.9e-7	1.0e-6	9.4e-7	**0.0**	2584	**310190**	**100**	211
CSPSO		6.8e-7	1.0e-6	**9.3e-7**	**0.0**	**2221**	399921	**100**	208

Thus, The CS and CSPSO algorithms outperform for most of the tested functions compared to the PSO and DE algorithms. The PSO algorithm requires the least average number of iterations ($Iter_{avg}$), average function evaluations (FE_{avg}), and simulation time (T_{sim}) for Easom function compared to the CS and CSPSO algorithms as listed in Table.2. The PSO algorithm also outperforms the CS algorithm for Schaffer function as listed in Table.2.

The convergence graphs of Beale, Drop-Wave, Easom, and Schaffer functions with 2-D and Ackley, Lévy, Griewank, Rastrigin, Rosenbrock, and Sphere functions with 30-D optimized by the CS, PSO, hybrid CSPSO, and DE algorithms are shown in Fig.9 to Fig.18. The convergence graph of the function shows the average function value performance of all runs with respect to iterations in the respective experiment. The vertical axis is the average of the best function value obtained by the algorithm in each iteration and the horizontal axis is the number of iterations.

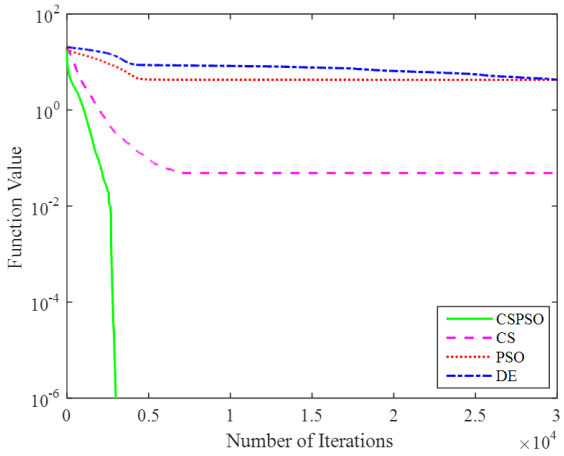

Fig.9. Convergence graph of 30-D Ackley function for the CS, PSO, CSPSO, and DE algorithms

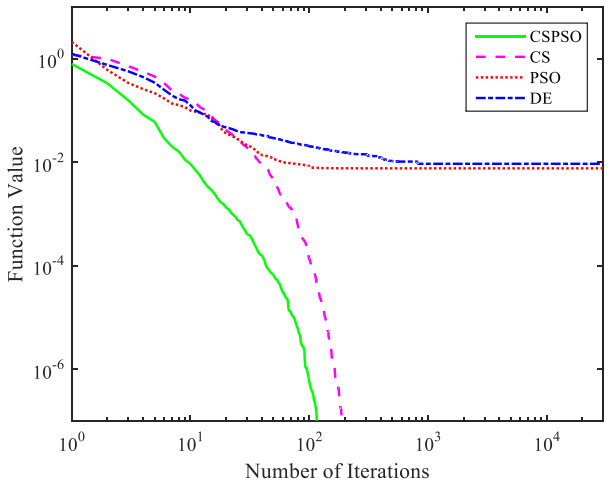

Fig.10. Convergence graph of 2-D Beale function for the CS, PSO, CSPSO, and DE algorithms

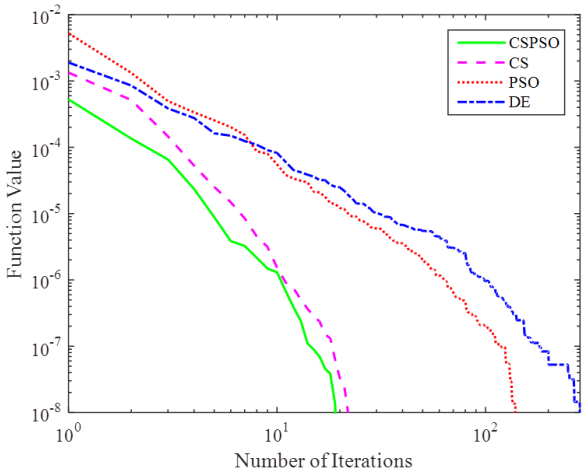

Fig.11. Convergence graph of 2-D Drop-Wave function for the CS, PSO, CSPSO, and DE algorithms

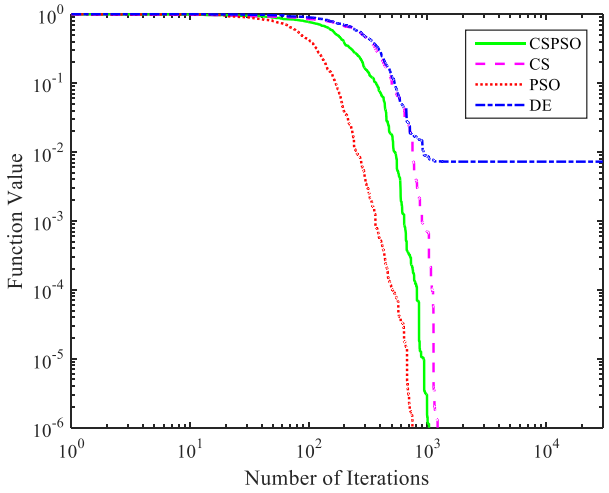

Fig.12. Convergence graph of 2-D Easom function for the CS, PSO, CSPSO, and DE algorithms

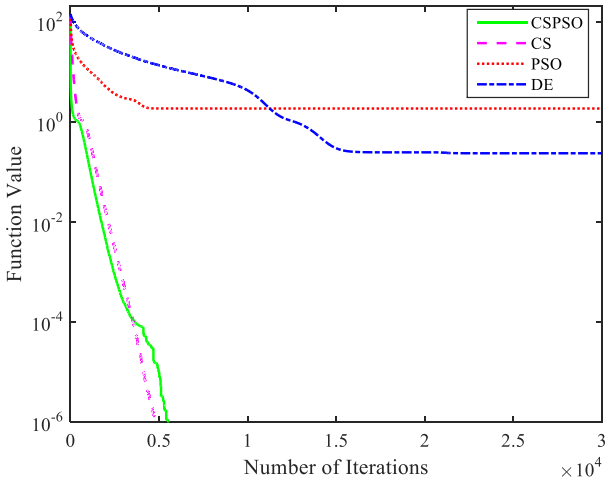

Fig.13. Convergence graph of 30-D Griewank function for the CS, PSO, CSPSO, and DE algorithms

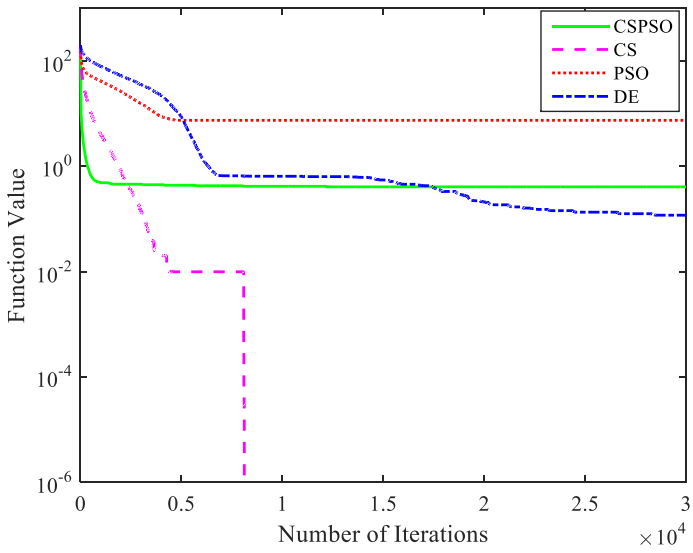

Fig.14. Convergence graph of 30-D Lévy function for the CS, PSO, CSPSO, and DE algorithms

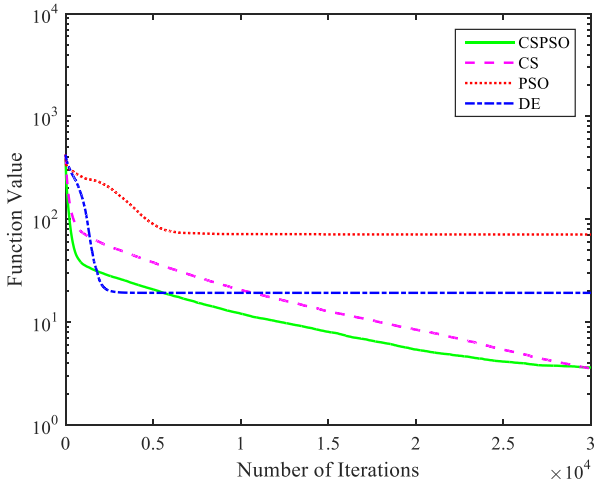

Fig.15. Convergence graph of 30-D Rastrigin function for the CS, PSO, CSPSO, and DE algorithms

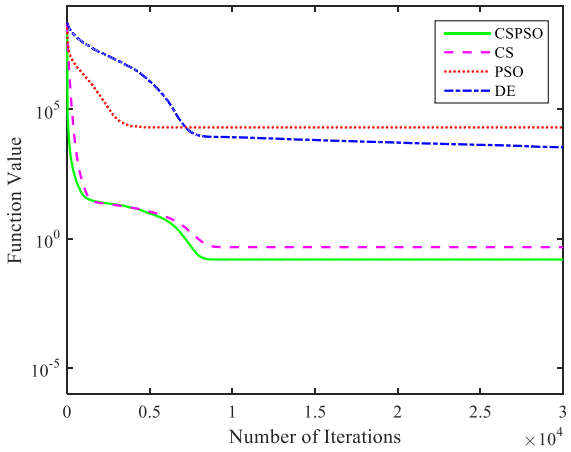

Fig.16. Convergence graph of 30-D Rosenbrock function for the CS, PSO, CSPSO, and DE algorithms

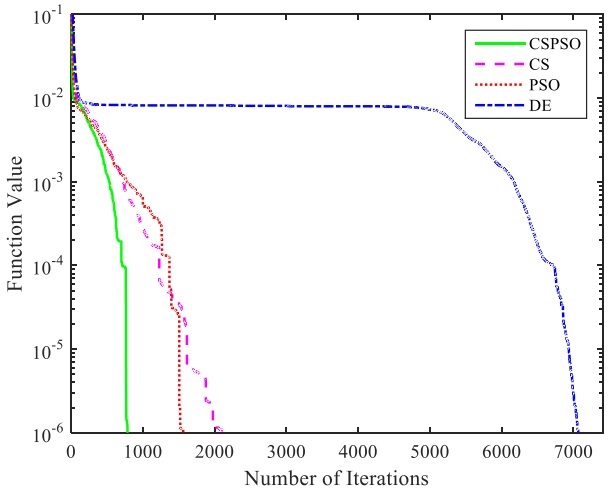

Fig.17. Convergence graph of 2-D Schaffer function for the CS, PSO, CSPSO, and DE algorithms

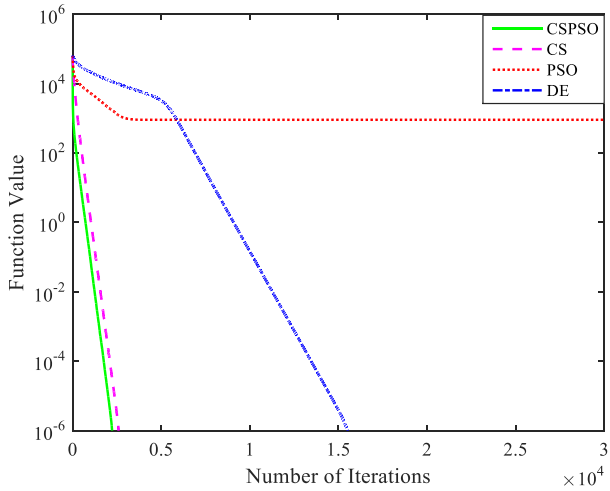

Fig.18.Convergence graph of 30-D Sphere function for the CS, PSO, CSPSO, and DE algorithms

3.5 SUMMARY

After reviewing the previously published literature for the given optimized problem, special attention is paid to the implementation of widely used and promising Evolutionary Algorithms. In this review paper, the focus is set on the performance evaluation of different metaheuristic Evolutionary Algorithms using different unimodal and multimodal benchmark functions. In this chapter, a hybrid algorithm of the CS and PSO algorithms is also compared. The concept of the hybrid algorithm of two or more algorithms plays a very prominent role to improve the searching capabilities of an algorithm and convergence rate. The more function evaluations per iteration which increase the total convergence time limit performance of the hybrid algorithm.

4. CONCLUSIONS

In this paper, the DE, PSO, and CS algorithms are analyzed and implemented using the C programming language. The performance of each algorithm is evaluated using different ten standard unimodal and multimodal benchmark functions. To exploit the benefits of the CS and PSO algorithms, the hybrid CSPSO algorithm is implemented and evaluated. The application and impact of the CS and hybrid CSPSO algorithms are used to optimize the basic building blocks of an analog CMOS IC such as voltage divider, triple cascode current mirror, three-stage current starved VCO, common-source amplifier, cascode amplifier, DA with a current mirror load, two-stage op-amp, FOTA, and Miller OTA. The convergence graphs of the various functions are compared and performance estimation of all the evolutionary algorithms is compared.

REFERENCES

[1] M.F.M. Barros, J.M.C. Guilherme and N.C.G. Horta, "Analog Circuits and Systems Optimization based on Evolutionary Computation Techniques", Springer, 2010.

[2] S.L. Sabat, K.S. Kumar and S.K. Udgata, "Differential Evolution and Swarm Intelligence Techniques for Analog Circuit Synthesis", *Proceedings of World Congress on Nature and Biologically Inspired Computing*, pp. 469-474, 2009.

[3] B. Razavi, "Design of Analog CMOS Integrated Circuits", McGraw-Hill, 2013.

[4] G. Alpaydin, S. Balkir and G. Dundar, "An Evolutionary Approach to Automatic Synthesis of High-Performance Analog Integrated Circuits", *IEEE Transactions on Evolutionary Computing*, Vol. 7, No. 3, pp. 240-252, 2003.

[5] H.Y. Koh, C.H. Sequin and P.R. Gray, "OPASYN: A Compiler for CMOS Operational Amplifiers", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 9, No. 2, pp. 113-125, 1990.

[6] P.M.R. Pereira, "Optimization based Design of LC Voltage Controlled Oscillators", PhD Dissertations, Department of Electronics Engineering, Nova De Lisboa University, pp. 1-198, 2013.

[7] B.D. Gajjar, "Automatic CMOS Analog Circuit Design using Particle Swarm Optimization Algorithm", Master Thesis, Department of Electronics and Communication Engineering, Gujarat University, pp. 1-90, 2011.

[8] R.A. Thakker, C. Sathe, A.B. Sachid, M. Shojaei Baghini, V. Ramgopal Rao and M.B. Patil, "A Novel Table-Based Approach for Design of FinFET Circuits", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 28, No. 7, pp. 1061-1070, 2009.

[9] Y. Massoud and T. Ragheb, "Automated Design Solutions for Fully Integrated Narrow-Band Low Noise Amplifiers", *Proceedings of International Workshop on System on Chip for Real Time Applications Automated*, pp. 109-114, 2006.

[10] H.Y. Koh, C.H. Sequin and P.R. Gray, "OPASYN: A Compiler for CMOS Operational Amplifiers", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 9, No. 2, pp. 113-125, 1990.

[11] A. Savio, L. Colalongo, M. Quarantelli and Z.M. Kovacs Vajna, "Automatic Scaling Procedures for Analog Design Reuse", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 53, No. 12, pp. 2539-2547, 2006.

[12] W. Nye, D.C. Riley, A. Sangiovanni Vincentelli and A.L. Tits, "Delight.Spice: An Optimization-Based System for the Design of Integrated Circuits", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 7, No. 4, pp. 501-519, 2002.

[13] S.S. Sapatnekar, V.B. Rao and P.M. Vaidya, "An Exact Solution to the Transistor Sizing Problem for CMOS Circuits using Convex Optimization", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 12, No. 1, pp. 1621-1634, 1993.

[14] M.M. Hershenson and S.P. Boyd, "Optimal Design of CMOS Op-Amp via Geometric Programming", *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 20, No. 1, pp. 1-21, 2001.

[15] F.A. Potra and S.J. Wright, "Interior-Point Methods", *Journal of Computational and Applied Mathematics*, Vol. 124, No. 1, pp. 281-302, 2000.

[16] R.A. Thakker, M.S. Baghini and M.B. Patil, "Low-Power Low-Voltage Analog Circuit Design using Hierarchical Particle Swarm Optimization", *Proceedings of International Conference on VLSI Design*, pp. 427-432, 2009.

[17] D.H. Wolpert and W.G. Macready, "No Free Lunch Theorems for Search", *IEEE Transactions on Evolutionary Computing*, Vol. 1, No. 1, pp. 67-82, 1997.

[18] A.P. Vaze, "Analog Circuit Design using Genetic Algorithm: Modified", World Academy of Science, Engineering and Technology, International Journal of Electrical, Computer, Energetic, Electronic and Communication Engineering, Vol. 2, No. 2, pp. 301-303, 2008.

[19] H. Gupta and B. Ghosh, "Analog Circuits Design using Ant Colony Optimization", *International Journal of Electronics, Computer and Communications Technologies*, Vol. 2, No. 3, pp. 9-21, 2012.

[20] G. Zhu and S. Kwong, "Gbest-Guided Artificial Bee Colony Algorithm for Numerical Function Optimization", *Applied Mathematics and Computation*, Vol. 217, No. 7, pp. 3166-3173, 2010.

[21] X.S. Yang and S. Deb, "Cuckoo Search via Levy Flights", *Proceedings of World Congress on Nature and Biologically Inspired Computing*, pp. 210–214, 2009.

[22] R.A. Vural and U.E. Ayten, "Optimized Analog Filter Approximation Via Evolutionary Algorithms", *Proceedings of International Conference on Intelligent Systems Design and Applications*, pp. 485-490, 2012.

[23] S.J. Patel and R.A. Thakkar, "Automatic Circuit Design and Optimization using Modified PSO Algorithm", *Journal of Engineering Science and Technology Review*, Vol. 4, No. 1, pp. 192-197, 2016.

[24] P.P. Prajapati and Mihir V. Shah, "Two Stage CMOS Operational Amplifier Design using Particle Swarm Optimization Algorithm",*Proceeding of IEEE UP Section Conference on Electrical, Computer and Electronics*, pp. 1-6, 2015.

[25] H. Soneji and R.C. Sanghvi, "Toward the Improvement of Cuckoo Search Algorithm", *World Congress on Information and Communication Technologies*, pp. 878-883, 2012.

[26] P. Civicioglu and E. Besdok, "A Conceptual Comparison of the Cuckoo-Search, Particle Swarm Optimization, Differential Evolution and Artificial Bee Colony Algorithms", *Artificial Intelligence Review*, Vol. 39, No. 4, pp. 315-346, 2013.

[27] R. Storn and K. Price, "Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces", *Journal of Global Optimization*, Vol. 11, No. 4, pp. 341-359, 1997.

[28] V. Arunachalam, "Water Resources Research Report: Optimization using Differential Evolution", Technical Report, University of Western Ontario, pp. 1-68, 2008.

[29] J. Kennedy and R.C. Eberhart, "*Swarm Intelligence*", Morgan Kaufmann Publishers, 2001.

[30] R.C. Butani, B.D. Gajjar and R.A. Thakker, "Performance Evaluation of Particle Swarm Optimization (PSO) and Artificial Bee Colony (ABC) Algorithm", *Proceedings of International Conference on Advanced Computing, Communication and Networks*, pp. 108-112, 2011.

[31] V. Truong Vu, "A Comparison of Particle Swarm Optimization and Differential Evolution", *Soft Computing*, Vol. 3, No. 3, pp. 13-30, 2012.

[32] E. Elbeltagi, T. Hegazy and D. Grierson, "Comparison among Five Evolutionary-Based Optimization Algorithms", *Advanced Engineering Informatics*, Vol. 19, No. 1, pp. 43-53, 2005.

[33] S. Janson and M. Middendorf, "A Hierarchical Particle Swarm Optimizer", *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, Vol. 35, No. 6, pp. 1272-1280, 2005.

[34] A. Adnan and M.A. Razzaque, "A Comparative Study of Particle Swarm Optimization and Cuckoo Search Techniques through Problem-Specific Distance Function", *Proceedings of International Conference of Information and Communication Technology*, pp. 88-92, 2013.

[35] X.S. Yang, "Harmony Search as a Metaheuristic Algorithm", *Computational Intelligence*, Vol. 191, pp. 1-14, 2009.

[36] E. Valian, S. Mohanna and S. Tavakoli, "Improved Cuckoo search Algorithm for Global Optimization", *Communication and Information Technology*, Vol. 1, No. 1, pp. 31-44, 2011.

[37] Pankaj P. Prajapati, Swati A. Sharma and Mihir V. Shah, "Design of CMOS Operational Amplifier using Differential Evolutionary Algorithm", *Proceedings of International Conference on VLSI Design*, pp. 108-111, 2016.

[38] I. Fister, D. Fister and I. Fister, "A Comprehensive Review of Cuckoo Search: Variants and Hybrids", *International Journal of Mathematical Modelling and Numerical Optimisation*, Vol. 4, No. 4, pp. 387-409, 2013.

[39] S. Roy, "Cuckoo Search Algorithm using Levy Flight : A Review", *International Journal of Modern Education and Computer Science*, Vol. 5, No. 12, pp. 10-15, 2013.

[40] X. Yang, T. O. Ting and M. Karamanoglu, "Random Walks, Levy Flights, Markov Chains and Metaheuristic Optimization", Springer, 2013.

[41] X. Yang and S. Deb, "Cuckoo Search : State-of-the-Art and Opportunities", *Proceedings of International Conference on Soft Computing and Machine Intelligence*, pp. 55-59, 2017.

[42] R.N. Mantegna, "Fast, Accurate Algorithm for Numerical Simulation of Levy Stable Stochastic Processes", *Physical Review*, Vol. 49, No. 5, pp. 4677-4683, 1994.

[43] Pankaj P. Prajapati and Mihir V. Shah, "Optimization of CMOS Current Mirror Load-Based Differential Amplifier using Hybrid Cuckoo Search and Particle Swarm Optimization Algorithm", *Journal of Artificial Intelligence Research and Advances*, Vol. 5, No. 3, pp. 71-78, 2019.

[44] M.A. Mushahhid Majeed and S.R. Patri, "A Hybrid of WOA and mGWO Algorithms for Global Optimization and Analog Circuit Design Automation", *International Journal for Computation and Mathematics in Electrical and Electronic Engineering*, Vol. 38, No. 1, pp. 452-476, 2018.

[45] F.J. Rodriguez, C. Garcia Martinez and M. Lozano, "Hybrid Metaheuristics based on Evolutionary Algorithms and Simulated Annealing : Taxonomy, Comparison, and Synergy Test", *IEEE Transactions on Evolutionary Computing*, Vol. 16, No. 6, pp. 787-800, 2012.

[46] A. Ghodrati and S. Lotfi, "A Hybrid CS/PSO Algorithm for Global Optimization", *Proceedings of International Conference on Intelligent Information and Database Systems*, pp. 89-98, 2012.

[47] X. Yang, "Recent Advances in Swarm Intelligence and Evolutionary Computation", Springer, 2015.

[48] P.P. Prajapati and M.V. Shah, "Performance Estimation of Differential Evolution, Particle Swarm Optimization and

Cuckoo Search Algorithms", *International Journal on Intelligent Systems and Applications*, Vol. 6, pp. 59-67, 2018.

[49] M. Molga and C. Smutnicki, "Test Functions for Optimization Needs", *Proceedings of International Conference on Computer and Information Science*, pp. 1-43, 2005.

[50] M. Jamil and X.S. Yang, "A Literature Survey of Benchmark Functions for Global Optimization Problems", *International Journal of Mathematical Modelling and Numerical Optimisation*, Vol. 4, No. 2, pp. 150-194, 2013.

[51] P.P. Prajapati and M.V. Shah, "Computer Aided CMOS Based Analog Circuit Design", Ph.D. Dissertations, Department of Electrical and Electronics Engineering, Gujarat Technological University, pp. 1-123, 2019.

[52] M.M. Ali and A. Torn, "Population Set-Based Global Optimization Algorithms: Some Modifications and Numerical Studies", *Computers and Operations Research*, Vol. 31, No. 10, pp. 1703-1725, 2004.

[53] Pankaj P. Prajapati and Mihir V. Shah, "Automatic Circuit Design of CMOS Miller OTA using Cuckoo Search Algorithm", *International Journal of Applied Metaheuristic Computing*, Vol. 23, No. 1, pp. 1-13, 2018.

[54] M. Clerc, "The Swarm and the Queen: Towards a Deterministic and Adaptive Particle Swarm Optimization", *Proceedings of Congress on Evolutionary Computation*, Vol. 3, pp. 1951-1957, 1999.

[55] I.C. Trelea, "The Particle Swarm Optimization Algorithm : Convergence Analysis and Parameter Selection", *Information Processing Letters*, Vol. 85, No. 6, pp. 317-325, 2003.

[56] Y. Shi and R.C. Eberhart, "Parameter Selection in Particle Swarm Optimization", Springer, 1998.

[57] C. Eberhart and Y. Shi, "Comparing Inertia Weights and Constriction Factors in Particle Swarm Optimization", *Proceedings of the Congress on Evolutionary Computation*, pp. 84-88, 2000.

[58] Pankaj P. Prajapati and Mihir V. Shah, "Automatic Sizing of CMOS based Analog Circuits using Cuckoo Search Algorithm", *International Journal of Intelligent Systems Technologies and Applications*, Vol. 23, No. 1, pp. 1-14, 2018.

[59] A. Leon-Garcia, "*Probability, Statistics, and Random Processes for Electrical Engineering*", Pearson Prentice Hall, 2008.