An Efficient Artificial Bee Colony Algorithm and Analog Circuit Design Environment

SUBHASH PATEL
Indus University
Ahmedabad
INDIA
subhash.bvm@gmail.com

RAJESH A THAKER
VGEC
Ahmedabad
INDIA

Abstract: The artificial bee colony algorithm (ABC), a population based algorithm, provides solutions with better accuracy compared to other competitive population based algorithms. However, it suffers from slow convergence speed. We suggest modifications in search strategy of ABC to improve overall performance and named this modified algorithm, Efficient ABC algorithm (EABC). The performance of EABC is compared with ABC by conducting experiment on 15 well-known scalable benchmark functions and synthesizing two analog circuits, two stage op-amp and bulk driven OTA, in 130μm CMOS technology. The proposed algorithm is performing significantly better than ABC for 14 benchmark functions and for remaining one the results are comparable. With the two-stage op-amp design problem, the average design error is 0.4% with EABC compared to 2.10% with ABC. Not only that the average design time is only 19.8 minutes with EABC in contrast to 22 minutes with ABC. In case of bulk driven OTA design, the average design error with EABC algorithm is zero compared to 1.26% with ABC. The average design time taken to design bulk driven OTA by EABC is only 4.62 minutes compared to 9.07 minutes with ABC. Apart from this, EABC is also compared with GABC and MABC algorithms, the variants of ABC. This comparison clearly indicates that EABC is performing better than ABC, GABC and MABC.

Key–Words: Artificial bee colony, Optimization, Automatic circuit design, Operational Amplifier, OTA

1 Introduction

Inspired from nature and life-system, the researchers have developed many heuristic algorithms and demonstrated their applications to solve complex problems. The Genetic Algorithm (GA) is based on Darwian law of survival of fittest [1]. The particle swarm optimization (PSO) algorithm simulates the behavior of birds flocking in search of food [2].The ant colony algorithm (ACO) is inspired from the foraging behavior of the ant colonies [3]. The biogeography-based optimization (BBO) is based on speciation and extinction of the species [4]. The ABC algorithm simulates the foraging behavior of the honey bees [5].

The ABC algorithm is tested for solving multi-modal and uni-modal benchmark functions and its performance is compared with other algorithms such as DE, PSO and EA [6, 7]. The comparison shows that, the ABC algorithm provides better performance than the mentioned algorithms and it can be efficiently employed to solve the multi-modal engineering problems with high dimensionality.

In [8] the application of ABC algorithm for global maximum power point tracking in the PV system under conditions of in-homogeneous isolation is demonstrated. The results of [8] show that the ABC algorithm perform better than PSO and EPO (enhanced perturb and observe) techniques. In [9], the ABC algorithm is used to design active analog filters and its performance is compared with GA and PSO algorithms. The results of [9] describe that, ABC outperforms other methods by means of execution time. [10] has demonstrated the application of ABC algorithm for data clustering. For the data clustering problem [10], the performance of the ABC algorithm is compared with PSO algorithm and other nine techniques. The results show that the ABC algorithm can be used efficiently for multivariate data clustering. [11] utilizes ABC for the Synthetic Aperture Radar (SAR) image segmentation. The experiment results of [11] indicate that ABC algorithm based method for SAR image segmentation is superior to the GA based and Artificial Fish Swarm (AFS) optimization based methods in term of segmentation accuracy and segmentation time. In [12], ABC algorithm is applied for the optimization of the truss structures and concluded that for the optimization of the truss structures, the ABC algorithm provides more accurate results compared to HS, SA, PSO and HPSO. However, ABC suffers from slow convergence speed. In [13], the ABC

algorithm and other competitive algorithms are used to design loud speaker. The result of this work indicates that for design of loud speaker, the performance of ABC is better that GA, PSO and DE but suffers from slow convergence speed. Zhang also suggests modified version of ABC algorithm to overcome the problem of slow convergence.

In this work, we present improved and efficient version of the ABC algorithm, namely Efficient Artificial Bee Colony algorithm (EABC). In EABC two different search strategies are presented which are utilized at different phases of the algorithm to improve overall efficiency of the algorithm. The performance of the EABC is tested by performing the experiments on 15 different numerical benchmark functions. The results of these experiments show that the performance of EABC algorithm is better than ABC. The performance of EABC algorithm is also compared with the other variants of the ABC algorithm.

Other than this, we address the problem of the automatic CMOS analog circuit design. With the advancement in CMOS technology, the size of MOSFET device shrinks. The smaller device size brings more non-linearity in the characteristics of the device. This makes deciding optimum device sizes extremely difficult. Under such circumstances, the traditional design approach based on the analytical calculations followed by the simulation fails to provide time efficient ASIC development cycle. On the other hand, with increasing power of the modern CPU, it is possible to use optimization algorithms effectively for circuit design. Researchers have already applied various evolutionary algorithms and swarm optimization techniques to solve the problem of the circuit design. [14] design real life circuits such as RF Low Noise Amplifier, Leapfrog Filter, and Ultra Wideband LNA are designed using NSGA-II algorithm. The modified GA algorithm algorithm based design tool is used to design CMOS operational amplifier by [15]. In [16], design of CMOS buffers and CMOS amplifiers using GA algorithm and PSO based algorithms are discussed. This work demonstrates the application of EABC to design Two stage CMOS amplifier and CMOS bulk-driven OTA. The performance of EABC for CMOS circuit design is compared with ABC and its variants, GABC and MABC.

The paper is organized as follows. Section 2 describes the ABC algorithm. The EABC algorithm is presented in section 3. The performances of comparison of EABC and ABC is discussed in section 4. The framework for the automatic circuit design problem using ABC and EABC along with the design examples is illustrated in section 5. Finally conclusions are drawn in section 6.

2 ABC algorithm

The Artificial Bee Colony algorithm (ABC) simulates the intelligent behavior of the bees and [5] proposed this swarm optimization technique. In ABC algorithm, there are three types of artificial bees: employee bee, onlooker bee and scout bee. The employee bees and onlooker bees are equal in number. Each employee bee tries to find better food source around location in its memory and shares this information with the onlooker bees. The onlooker bees only select food sources where the probabilities for improvement are higher and try to improve selected food sources. The scout bee drops the food sources which cannot be improved after per-determined trials and instead of each dropped food source, new food source is selected randomly. The ABC algorithm follows the iterative process and the major phases are initialization, employee bees phase, onlooker bees phase and scout bee phase.

Let's take swarm population $2 \times N$ for the problem with D dimensions. This leads to N employee bees and N onlooker bees. The algorithm starts with random initialization. First, N numbers of the food sources are initialized randomly. Each food source represents the solution candidate and it can be modeled by an vector $X_i = \{x_{i1}, x_{i2}, ..., x_{iN}\}$. Each food source can be initialized randomly as follows,

$$x_{ij} = x_{maxj} + \phi_{ij}(x_{maxj} - x_{minj}) \qquad (1)$$

where, $i = 1, 2...N$, $j = 1, 2, ..., D$, ϕ_{ij} is uniformly distributed random number between 0 and 1. For j^{th} dimension, x_{maxj} represents upper bound of search space and x_{minj} represents lower bound of search space. After the random initialization, each food source is evaluated for the its fitness.

Each employee bee finds a new food source V_i around assigned food source X_i, by sharing the information with other employee bee. The V_i can be given by following search equation,

$$V_{ij} = x_{ij} + \theta_{ij}(x_{ij} - x_{ik}) \qquad (2)$$

where, $j \in \{1, 2, ..., D\}$ and selected randomly, $k \in \{1, 2, ..., N\}$ and different form i, θ_{ij} is a random number between -1 and 1. V_i is evaluated for its fitness. If fitness of V_i is better than fitness of X_i, then V_i is selected otherwise it is rejected. In other words, greedy selection is applied between food sources X_i and V_i.

After search process for better food site, each Employee bee shares the information about its food source with onlooker bees. Based on the fitness of

Table 1: Benchmark functions used for experiment.

	Function	Search Space	Minimum Value
Sphere	$f_1(X) = \sum_{i=1}^{N} {x_i}^2$	$[-100, 100]^N$	0
Griewank	$f_2(X) = \frac{1}{4000} \sum_{i=1}^{N} {x_i}^2 - \prod_{i=1}^{N} cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$[-600, 600]^N$	0
Rastrigin	$f_3(X) = 10n + \sum_{i=1}^{N}(x_i^2 - 10cos(2\pi x_i))$	$[-5.12, 5.12]^N$	0
Rosenbrock	$f_4(X) = \sum_{i=1}^{N-1}(100(x_i^2 - x_{i+1})^2 + (1 - x_i)^2)$	$[-10, 10]^N$	0
Schwefel	$f_5(X) = \sum_{i=1}^{N}(-x_i sin(\sqrt{\lvert x_i \rvert})) + 418.982887272$	$[-500, 500]^N$	0
Schwefel's 2.21	$f_6(X) = max_i\{\lvert x_i \rvert, 1 \leq i \leq N\}$	$[-100, 100]^N$	0
Alpine	$f_7(x) = \sum_{i=1}^{N} \lvert x_i sin(x_i) + 0.1x_i \rvert$	$[-10, 10]^N$	0
Shifted Sphere	$f_8(X) = \sum_{i=1}^{N} {z_i}^2,\ Z = X - O$	$[-100, 100]^N$	0
Shifted Griewank	$f_9(X) = \frac{1}{4000} \sum_{i=1}^{N} {z_i}^2 - \prod_{i=1}^{N} cos\left(\frac{z_i}{\sqrt{i}}\right) + 1,$ $Z = X - O$	$[-600, 600]^N$	0
Shifted Rastrigin	$f_{10}(X) = 10n + \sum_{i=1}^{N}(z_i^2 - 10cos(2\pi z_i)),\ Z = X - O$	$[-5.12, 5.12]^N$	0
Non Continuous Rastrigin	$f_{11}(X) = 10n + \sum_{i=1}^{N}(y_i^2 - 10cos(2\pi y_i))$ $y_i = \begin{cases} x_i & \text{if} \quad \lvert x_i \rvert < \frac{1}{2} \\ \frac{round(2x_i)}{2} & \text{if} \quad \lvert x_i \rvert \geq \frac{1}{2} \end{cases}$	$[-5.12, 5.12]^N$	0
Dixon-price	$f_{12}(X) = (x_1 - 1)^2 + \sum_{i=2}^{N} i(2x_i^2 - x_{i-1})$	$[-100, 100]^N$	0
Sum Square	$f_{13}(X) = \sum_{i=1}^{N} ix_i^2$	$[-100, 100]^N$	0
Zakhorav	$f_{14}(X) = \left(\sum_{i=1}^{N} 0.5ix_i\right)^4 + \left(\sum_{i=1}^{N} 0.5ix_i\right)^2 + \sum_{i=1}^{N} x_i^2$	$[-5, 10]^N$	0
Ackley	$f_{15}(X) = 20 + e - 20exp(-0.2\sqrt{\frac{1}{N}\sum_{i=1}^{N} x_i^2}) - exp(\frac{1}{N}\sum_{i=1}^{N} cos(2\pi x_i))$	$[-32, 32]^N$	0

each food source, the probability for further improvement associated with each food source is calculated using roulette wheel selection, as follows,

$$p_i = \frac{f_i}{\sum_{i=1}^{i=N} f_i} \tag{3}$$

where f_i is the fitness of the ith food location. The food location with higher fitness value has better chance for the improvement.

The onlooker bees only select certain food sources where probabilities for improvements are higher. The new food source location V_i is found using Equ. 2. Based on the fitness of V_i and X_i, greedy selection process is carried out.

The scout bee drops the food site which cannot be improved after certain per-defined trails T_{max} and finds new food source for each dropped food source by random re-initialization as described by Equ.1

3 Efficient ABC algorithm

The population initialization in evolutionary algorithm is a very crucial task. It affects the convergence speed and quality of the final solution. However if the nature of the problem is unknown, the random initialization is a best choice. In Efficient ABC algorithm (EABC), random population initialization is used. In EABC like ABC, there are three types of bees : employee bees, onlooker bees and scout bees. Half of the bees are employee bees and remaining half become onlooker bees. However, the behavior of the bees is different than ABC.

In order to improve the movement of the employee bee, we follow following search equation during employee bee phase,

$$V_{i,j} = X_{N_1,j} + \Theta_{i,j}(X_{N_2,j} - X_{i,j}) \tag{4}$$

where, N_1, N_2 are mutually exclusive and different from i with $N_1, N_2 \in \{1, 2, 3, ..., SN\}$. $j \in \{1, 2, 3, ..., D\}$ and selected randomly. Θ_{ij} is an uniformly distributed random number between -0.25 to 0.25. Employee bee gets the information about two different food sources i.e. N_1 and N_2 and finds new food source V_i around N_1. The random number $\Theta_{i,j}$ decides the search radius around N_1. The large value of this radius results into slower convergence speed and larger value leads to poor exploration. In ABC, employee bees carries search around its own food source, while in EABC, the search is carried out around other bee's food source. This search strategy improves the exploration. The greedy selection process is applied between food source X_i and new found food source V_i.

In ABC, during onlooker bee phase, onlooker bees try to improve selected food sources. In EABC, the idea of selective improvement is dropped and like employee bee phase, equal chance is given to each food source to improve. However, the different search strategy is used. The new search strategy during the Onlooker phase is described by following equation,

$$V_{ij} = X_{best,j} + \Phi_{i,j}(X_{N_1,j} - X_{N_2,j}) \tag{5}$$

where, N_1, N_2 are mutually exclusive and different from i with $N_1, N_2 \in \{1, 2, 3, ..., SN\}$. $j \in \{1, 2, 3, ..., D\}$ and selected randomly. Φ_{ij} is an uniformly distributed random number between -0.5 to 0.5. This search strategy improves the exploitation. Thus, employee bee phase provides exploration and onlooker bee phase provides exploitation.

The scout bee tries to improve the food sources which cannot be improved after pre-determined trials T_{max} or whose fitness is very less (α times smaller) in comparison with the best food source found so far. To improve such food sources, scout bee follows the mutation and recombination process. The scout bee calculates the new food location V_i using Equ. 6 and then follows recombination process according to Equ. 7.

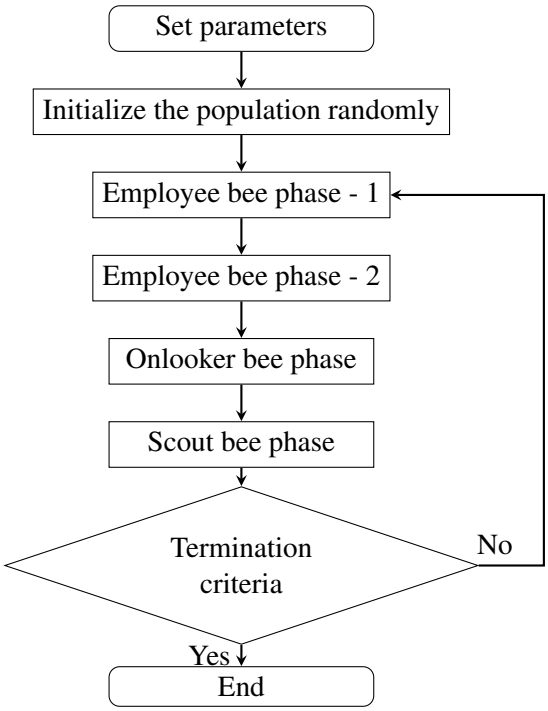
$$U_{i+1} = X_{best} + \Theta_i(X_{N1} - X_{N2}) \tag{6}$$

$$X_{i+1,j} = \begin{cases} X_{i,j} & \text{if } Random[0,1] > p \\ U_{i+1,j} & \text{if } Random[0,1] < p \end{cases} \tag{7}$$

with N_1 and N_2 are mutually exclusive with $N_1, N_2 \in 1, 2, 3, ..., SN$ and selected randomly. Θ_i is a vector containing uniformly distributed random numbers between -0.25 to 0.25. p is a constant number between 0 and 1.

Each algorithmic iteration in EABC contains two employee bee phases, one onlooker bee phase and one scout bee phase. The flow-chart of the algorithm is shown in Fig. 1.

The proper choice of the p and boundary limits of the random numbers Φ and Θ is necessary. Based on the results of experiments conducted on the benchmark functions, values of these constants are suggested. For $p = 0.2$ and $p = 0.3$, satisfactory results are obtained, so these values of p are recommended.



Employee bee phase

$i = 1$
while $i \leq SN$ **do**
Select N_1, N_2 different from i, with $N_1 \neq N_2$;
Generate new food-source V_i using Equ. 4 ;
Apply greedy selection between V_i and X_i ;
end

Onlooker bee phase

$i = 1$
while $i \leq SN$ **do**
Select N_1, N_2 different from i ;
Generate new food-source V_i using Equ. 5 ;
Apply greedy selection between V_i and X_i ;
end

Scout bee phase

$i = 1$
while $i \leq SN$ **do**
if $(Trial_i \geq T_{max})$ *or* $\left(fitness\ of\ X_i \geq \frac{fitness\ of\ X_{best}}{\alpha}\right)$ **then**
Generate mutant vector U_{i+1} using Equ. 6;
Generate new food source X_i using recombination process described by Equ. 7 ;
end
end

Figure 1: Flow chart of EABC algorithm

Table 2: Mean, best, worst and standard deviation values obtained by ABC and EABC through 30 independent runs on function from f_1 to f_7.

Function	D		Mean	Best	Worst	SD
Sphere	30	ABC	1.34e-09	2.45e-10	3.18e-09	7.52e-10
	30	EABC	**1.35e-34**	**5.98e-35**	**3.70e-34**	**7.15e-35**
	60	ABC	1.48e-10	4.19e-10	5.35e-09	1.30e-09
	60	EABC	**4.87e-33**	**2.02e-33**	**1.21e-32**	**2.46e-33**
	100	ABC	1.53e-09	2.28e-10	6.81e-09	1.32e-09
	100	EABC	**3.69e-32**	**1.64e-32**	**7.54e-32**	**1.54e-32**
Griewank	30	ABC	4.07e-08	8.12e-11	3.04e-07	8.48e-08
	30	EABC	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**
	60	ABC	1.49e-09	1.01e-10	1.24e-08	2.43e-09
	60	EABC	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**
	100	ABC	1.18e-09	3.50e-11	7.18e-09	1.63e-09
	100	EABC	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**
Rastrigin	30	ABC	1.04e-05	1.03e-10	2.05e-04	4.59e-05
	30	EABC	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**
	60	ABC	2.98e-01	8.03e-11	1.99e+00	5.83e-01
	60	EABC	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**
	100	ABC	1.66e+00	2.15e-10	4.98e+00	1.34e+00
	100	EABC	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**
Rosenbroke	30	ABC	3.44e-01	2.18e-02	1.90e+00	4.12e-01
	30	EABC	**2.12e-01**	**2.36e-03**	**1.24e+00**	**2.74e-01**
	60	ABC	4.14e-01	3.34e-02	**1.90e+00**	**4.12e-01**
	60	EABC	**4.00e-01**	**3.33e-02**	4.03e+00	7.47e-01
	100	ABC	8.22e-01	**2.09e-02**	5.95e+00	1.07e+00
	100	EABC	**4.06e-01**	3.18e-02	**4.64e+00**	**8.27e-01**
Schwefel	30	ABC	1.22e+02	4.28e-08	2.67e+02	1.00e+02
	30	EABC	**-2.19e-12**	**-2.55e-12**	**-1.70e-12**	**2.34e-13**
	60	ABC	6.38e+02	1.25e+02	1.02e+02	2.25e+02
	60	EABC	**-3.15e-12**	**-4.03e-12**	**-1.93e-12**	**5.43e-13**
	100	ABC	5.93e+02	2.36e+02	8.66e+02	1.56e+02
	100	EABC	**-5.90e-12**	**-7.73e-12**	**-4.60e-12**	**8.15e-13**
Schwefel's 2.21	30	ABC	5.45e+01	4.49e+01	6.19e+01	4.72e+00
	30	EABC	**5.03e+00**	**3.86e+00**	**6.56e+00**	**6.11e-01**
	60	ABC	7.20e+01	6.08e+01	7.78e+01	3.66e+00
	60	EABC	**2.26e+01**	**2.03e+01**	**2.73e+01**	**1.55e+00**
	100	ABC	8.13e+01	7.59e+01	8.45e+01	2.34e+00
	100	EABC	**4.08e+01**	**3.74e+01**	**4.44e+01**	**1.84e+00**
Alpine	30	ABC	1.54e-04	3.32e-05	3.69e-04	8.38e-05
	30	EABC	**7.81e-14**	**2.23e-16**	**9.44e-13**	**1.87e-13**
	60	ABC	2.51e-03	1.89e-04	1.23e-02	2.96e-03
	60	EABC	**2.89e-12**	**2.04e-13**	**1.07e-11**	**2.75e-12**
	100	ABC	1.58e-02	3.77e-04	3.59e-02	8.62e-03
	100	EABC	**1.56e-11**	**5.80e-12**	**3.51e-11**	**6.81e-12**

Table 3: Mean, best, worst and standard deviation values obtained by ABC and EABC through 30 independent runs on function from f_8 to f_{13}.

Function	D		Mean	Best	Worst	SD
Shifted Sphere	30	ABC	1.31e-09	1.18e-10	4.24e-09	1.03e-09
	30	EABC	**1.45e-33**	**3.08e-35**	**4.27e-34**	**8.73e-35**
	60	ABC	1.12e-09	2.47e-10	3.09e-09	7.07e-10
	60	EABC	**4.29e-33**	**1.58e-33**	**1.02e-32**	**1.98e-33**
	100	ABC	1.80e-09	3.56e-10	4.03e-09	1.09e-09
	100	EABC	**5.38e-33**	**1.09e-33**	**1.00e-32**	**2.28e-33**
Shifted Griewank	30	ABC	1.49e-08	1.95e-10	2.80e-07	5.01e-08
	30	EABC	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**
	60	ABC	1.56e-09	1.33e-10	9.94e-09	2.37e-09
	60	EABC	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**
	100	ABC	8.03e-10	4.59e-11	8.90e-09	1.59e-09
	100	EABC	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**
Shifted Rastrigin	30	ABC	1.58e-04	1.66e-10	4.75e-10	8.53e-04
	30	EABC	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**
	60	ABC	2.81e-01	3.09e-11	1.99e+00	5.05e-01
	60	EABC	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**
	100	ABC	9.3e-01	6.18e-10	3.11e+00	1.12e+00
	100	EABC	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**
Non-continuous Rastrigin	30	ABC	2.38-e01	8.41e-08	3.00e+00	6.14e-01
	30	EABC	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**
	60	ABC	2.12e+00	2.26e-05	4.91e+00	1.39e+00
	60	EABC	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**
	100	ABC	5.39e+00	1.23e-07	9.38e+00	2.17e+00
	100	EABC	**0.00e+00**	**0.00e+00**	**0.00e+00**	**0.00e+00**
Dixon-price	30	ABC	1.58e-01	2.38e-02	3.42e-01	8.68e-02
	30	EABC	**1.46e-02**	**3.69e-07**	**2.21e-01**	**4.27e-02**
	60	ABC	1.71e-01	3.80e-02	4.46e-01	9.58e-02
	60	EABC	**2.86e-02**	**4.22e-04**	**3.28e-01**	**6.93e-02**
	100	ABC	2.76e-01	8.25e-02	**1.04e+00**	**2.43e-01**
	100	EABC	**1.02e-01**	**4.85e-04**	1.39e+00	2.59e-01
Sum Square	30	ABC	1.10e-07	7.71e-09	7.21e-07	1.32e-07
	30	EABC	**1.10e-33**	**3.14e-34**	**3.21e-33**	**6.19e-34**
	60	ABC	6.64e-08	1.20e-08	2.35e-07	5.60e-08
	60	EABC	**1.06e-31**	**4.06e-32**	**2.74e-31**	**4.86e-32**
	100	ABC	7.17e-08	1.14e-08	2.21e-07	5.44e-08
	100	EABC	**1.38e-30**	**6.50e-31**	**2.10e-30**	**4.68e-31**

Table 4: Mean, best, worst and standard deviation values obtained by ABC and EABC through 30 independent runs on function from f_{14} to f_{15}.

Function	D		Mean	Best	Worst	SD
Zakhorav	30	ABC	2.45e+02	2.01e+02	3.10e+02	2.60e+01
	30	EABC	**7.50e+01**	**2.71e+01**	**1.24e+02**	**2.10e+01**
	60	ABC	6.54e+02	5.75e+02	7.12e+02	3.72e+01
	60	EABC	**3.55e+02**	**2.41e+02**	**5.38e+02**	**6.38e+01**
	100	ABC	1.25e+03	1.05e+03	1.36e+03	6.92e+01
	100	EABC	**8.25e+02**	**6.43e+02**	**1.07e+03**	**9.35e+01**
Ackley	30	ABC	8.31e-06	2.68e-06	1.55e-05	2.94e-06
	30	EABC	**2.68e-23**	**2.44e-26**	**1.19e-22**	**3.41e-23**
	60	ABC	1.40e-05	7.83e-06	2.25e-05	3.51e-06
	60	EABC	**3.74e-24**	**1.65e-27**	**4.49e-23**	**9.28e-24**
	100	ABC	2.80e-05	1.29e-05	5.89e-05	1.00e-05
	100	EABC	**3.11e-25**	**8.86e-29**	**7.54e-24**	**1.35e-24**

4 Experiments

4.1 Benchmark functions and parameter setting

To test the performance of the EABC on numerical benchmark functions, experiments are conducted to minimize 15 well-known scalable benchmark functions. The Sphere and Sum-Square functions are bowl shape functions. Rosenbrock and Dixon-price functions are multi-modal and inseparable valley shape functions. Griewank, Rastrigin and Schwefel are multi-modal, inseparable functions with multiple local minima. Alpine and Ackley are also multi-modal problems. Zakhorav function is plate shape function. In shifted functions, shift vector O is selected randomly from search space.

To compare the performances of EABC and ABC, the experiments on each benchmark function are conducted for dimensions $D = 30$, $D = 60$ and $D = 100$ with maximum number of the function evaluations 150000, 300000 and 500000 respectively. The population size is set to 150 ($SN = 75$). The value of parameter $limit$ for ABC and EABC is set to $2 \times D$ i.e. 60 for $D = 30$, 120 for $D = 60$ and 200 for $D = 100$. For EABC, value of p is set to 0.2 and value of α is set to 100. All the results discussed are based on 30 independent runs.

4.2 Experimental results

The result of experiments are illustrated in Tables 2, 3 and 4 in terms of mean, best, worst and standard deviation over 30 independent runs. The obtained results indicates that EABC perform significantly better than ABC for all benchmark functions except Rosenbrock function with faster convergence speed. In case of Rosenbrock function the performance of EABC does not improve significantly compared to ABC. The worst result obtained for $D = 60$ with EABC is larger compared to that obtained with ABC. However, the mean error is less for EABC. The graphs shown in Fig. 2 illustrate the convergence performance of ABC and EABC algorithms.

4.3 Comparison with GABC and MABC

The Gbest-guided artificial bee colony algorithm (GABC)[17] and Modified artificial bee colony algorithm (MABC) [18] algorithms are the enhanced versions of the ABC algorithm. In GABC, the search of the better food source is inspired by the PSO algorithm. The MABC is inspired by DE algorithm and uses chaotic iterator and opposition learning based initialization. The comparison of EABC with these two algorithms are based on the number of function evaluations and mean error found over 30 independent runs. The population size is taken 150. The maximum number of function evaluations are 150000, 300000 and 500000 for $D = 30$, $D = 60$ and $D = 100$, respectively. The parameter C is set to 1.5 for GABC algorithm as suggested by [17]. Table 5 shows the comparison between EABC, GABC and MABC algorithms. The result illustrated for MABC is directly taken from its corresponding reference [18].

From the comparison of results, it is very clear the

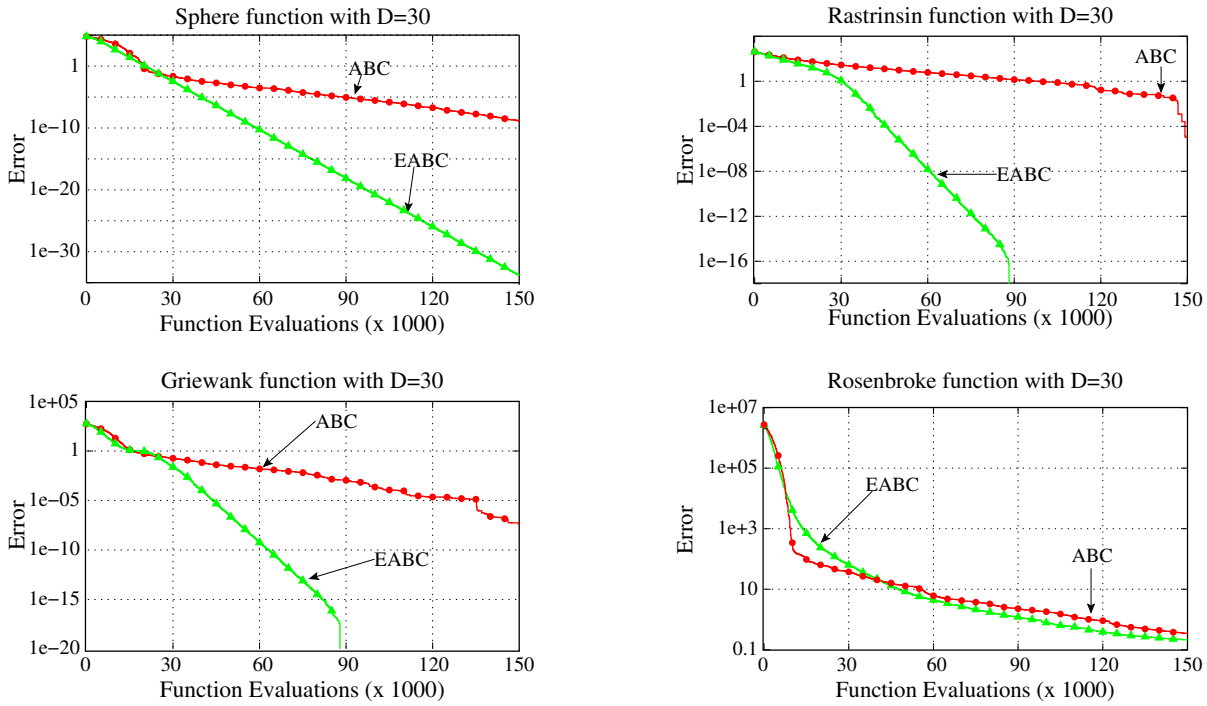


Figure 2: Convergence performances of ABC and EABC algorithms

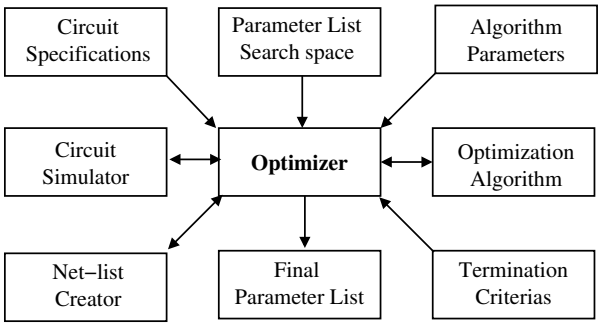


Figure 3: Conceptual block diagram of Optimizer

performance of the EABC is better than GABC. The performance of EABC is better or comparable with MABC for the all taken cases except the Alpine function. For Alpine function MABC is found better.

5 Automatic analog circuit design

The task of the automatic circuit design is carried out by optimizer, a design framework. The optimization algorithm is a heart of the optimizer. The optimizer generates the circuit based on parameters provided by algorithm, initiate circuit simulator, analyze the simulator output and provides necessary feedback to optimization algorithm. Based on this feedback, optimization algorithm generates new parameters. The conceptual block diagram of the optimizer is illustrated in Fig. 3. For the CMOS circuits, design parameters include size of various transistors. Initially, upper and lower values of the various circuit parameter are decided broadly without involving any circuit related calculations. This creates search space for the circuit design problem. With this information, the optimization algorithm generates the set of circuit parameter. The optimizer generates a circuit for simulator according to these parameters and circuit is simulated against pre-defined test cases. The output of simulator is utilized by cost function to calculate the design error. The optimization algorithm uses this error to generate new parameter set. The aim of the optimization process is reduce design error to null.

5.1 Formulation of cost function

For the formulation of the cost function, we have used normalized root mean square error method. Let's consider a circuit design problem with M specifications. The set of the desired specifications is represented by vector DS. During the optimization process, after each circuit simulation, obtained specifications are collected in vector OS. The cost function can be described by following equation,

$$f_e(\%) = \sqrt{\frac{1}{M}\sum_{i=1}^{M} E_i} \times 100 \qquad (8)$$

Table 5: Comparison of EABC and GABC algorithms : Mean value of error over 30 independent runs

Function	D	GABC	MABC	EABC
	30	6.74e-20	9.43e-32	**1.35e-34**
Sphere	60	4.40e-19	6.03e-29	**4.87e-33**
	100	1.15e-18	1.43e-27	**3.69e-32**
	30	4.55e-09	**0.00e+00**	**0.00e+00**
Griewank	60	1.81e-13	**0.00e+00**	**0.00e+00**
	100	1.44e-15	**0.00e+00**	**0.00e+00**
	30	2.36e-16	**0.00e+00**	**0.00e+00**
Rastrigin	60	5.98e-15	**0.00e+00**	**0.00e+00**
	100	3.86e-14	**0.00e+00**	**0.00e+00**
	30	1.76e+00	6.11e-01	**2.12e-01**
Rosenbroke	60	8.19e+00	1.51e+00	**4.00e-01**
	100	2.57e+01	1.98e+00	**4.06e-01**
	30	9.99e-06	**1.58e-16**	7.81e-14
Alpine	60	8.46e-05	**8.20e-16**	2.89e-12
	100	2.66e-04	**5.83e-15**	1.56e-11
	30	6.36e-04	**-1.21e-13**	-2.19e-12
Schwefel	60	2.59e-02	3.56e-11	**-3.15e-12**
	100	4.26e+01	1.19e-10	**-5.90e-12**

$$E_i = \begin{cases} 0 & \text{if } i^{th} \text{ is satisfied} \\ \left(\frac{OS_i - DS_i}{DS_i}\right)^2 & \text{Otherwise} \end{cases} \quad (9)$$

5.2 Setting of algorithm parameters for circuit design

To compare the performance of EABC with ABC and its variants, we have designed two stage CMOS operational amplifier and bulk driven operational transconductance amplifier (OTA) in 130nm technology. The population size is set to 30 ($N = 15$), maximum number of the circuit evaluations is set to 5000 and value of T_{max} is set to twice the problem dimension for all the algorithms i.e. $T_{max} = 2D$. For EABC algorithm value of p and α are set to 0.2 and 10, respectively. The C parameter for GABC is set to 1.5. Selective probability p is set to 0.7 for MABC algorithm as suggested by [18]. The optimizer and optimization algorithms are implemented using C++ and experiments are conducted on computer with 4GB of RAM and FX-8350 processor. Each circuit is designed 25 times independently using EABC, ABC, GABC and MABC algorithms.

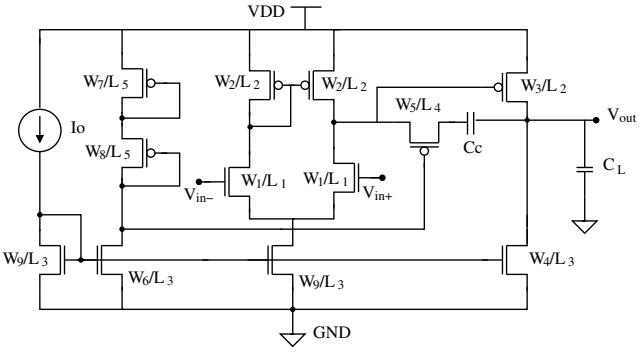


Figure 4: Two-stage operational amplifier: circuit diagram.

Table 6: Two stage operational amplifier: Search space for design variables.

Parameter	Search space
W_1 to W_9	0.2 μm to 10 μm
L_1 to L_5	0.2 μm to 1 μm
I_0	0.5 μA to 10 μA
C_C	0.001 pF to 1 pF

5.3 Two-Stage CMOS operational amplifier

The two stage operational amplifier is very versatile analog circuit and found in many circuits such as mixer, amplifiers, DAC and ADC as a building block. The circuit of op-amp is illustrated in Fig 4[19]. The major specifications of two stage op-amp considered for the design are gain, unity gain bandwidth (UGB), phase margin (PM), power consumption (PC), rise slew rate (RSR), fall slew rate (FSR), common mode rejection ration (CMRR) and power supply rejection ration (PSRR). The desired specifications are described in Table 7. The design parameters are width and length of the transistors, value of the current source I_0 and capacitor C_C. The circuit is designed in $130nm$ technology to drive load of $0.05pF$ with $1.2V$ supply voltage. The search space i.e. upper and lower bounds on width and length of transistor, value of I_0 and value on C_C, is illustrated in Table 6. The mean of design error in % over 25 independent design runs, % error for Worst and Best design calculated using Eq. 8 after 5000 circuit evaluations are illustrated in Table 8. The worst, best and mean specifications found during 25 independent design runs are indicated in Table 9. The number of times the op-amp is design successfully i.e. with all the specifications are satisfied and hence zero error, using EABC,

Table 7: Two stage operational amplifier: Search space for design variables.

Specification	Desired value
Gain (dB)	≥ 80
UGB (MHz)	≥ 100
PM (degree)	≥ 62
PC (μW)	≤ 20
RSR (V/μS)	≥ 60
FSR (V/μS)	≥ 60
|PSRR| (dB)	≥ 80
CMRR (dB)	≥ 75

Table 8: Two stage operational amplifier: Mean design error in % over 25 independent design runs, % error for best and worst design cases.

Algorithm	Mean design error	Error for best design	Error for worst design
EABC	0.40	0	1.58
ABC	2.10	0	6.33
GABC	0.74	0	6.18
MABC	1.45	0	2.55

ABC, GABC and MABC algorithms is illustrated in Fig. 5. Fig 6 describes the variations in cost function value with the circuit evaluations and thus compares the convergence speed of EABC with ABC, GABC and MABC algorithm.

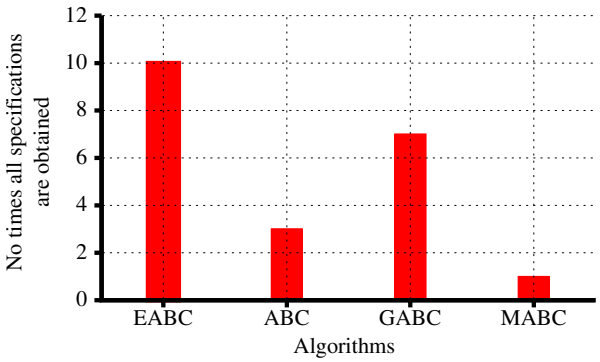


Figure 5: Two stage operational amplifier: Number of times all specifications are obtained during 25 design runs.

From obtained results following observations can be made.

- The mean of cost function value over 25 independent two stage op-amp design runs is only 0.4% with EABC, while that is 2.1% for ABC, 0.74% for GABC and 1.45% for MABC. The average design time is 19.88 minutes with EABC, 22 minutes with ABC, 21.21 minutes with GABC and 22.3 minutes with MABC. This clearly indicates that for two-stage op-amp design problem, the performance of EABC is better than ABC, MABC and GABC algorithms.

- In case of numerical benchmark functions experiment, MABC algorithm outperforms the GABC algorithm, however for two stage op-amp design problem, GABC algorithm performs better than MABC.

- With EABC algorithm, from 25 independent design runs, op-amp is designed 10 times with no error i.e. all design constrains are satisfied. While for ABC algorithm this is 3 times, for GABC algoirhtm 7 times and for MABC algorithm it is only once. Moreover, the worst design obtained during design runs has error of 1.58% with EABC. It is 6.33% for ABC, 6.18% for GABC and 2.55% for MABC.

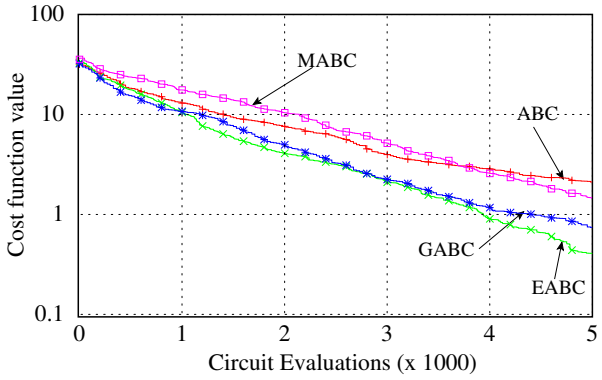


Figure 6: Two stage operational amplifier: variations in mean error with circuit evaluations.

5.4 Bulk-driven OTA

In the bulk-driven circuit technique, the voltage signal is applied at the bulk terminal of the MOSFET. In low voltage application, i.e. supply voltage is less than 1V, this technique enhances the performance of the circuit by overcoming the limitations imposed by the threshold voltage. Another advantage of bulk-driven technique for low voltage application is that, it does not require any modification in the structure of MOSFET [20, 21, 22].

The operational transconductance amplifier (OTA) is used widely to drive large capacitive load. The circuit diagram of the bulk driven OTA is shown in Fig.7. This circuit is proposed by [23]. In this work, we design the same OTA circuit with $130nm$ technology to drive the load of $15pF$. The desired specifications are shown in in Table 10. The design

Table 9: Two stage operational amplifier: Mean of obtained specifications (MOS), Specifications of best obtained design (BOS), Specifications of worst obtained design (WOS) over 25 independent design runs

Specifications	EABC			ABC			GABC			MABC		
	MOS	BOS	WOS	MOS	BOS	WOS	MOS	BOS	WOS	MOS	BOS	WOS
Gain (dB)	79.7	82.5	77.2	78.9	82.8	79.6	79.5	80.2	80.8	79.7	80.0	83.1
UGB (MHz)	101.4	101.1	97.9	98.9	102.0	86.7	102.3	107.0	100.3	100.5	107.6	110.9
PM (degree)	62.8	64.8	61.1	61.8	62.6	58.9	63.0	65.2	57.8	63.3	63.8	57.7
PC (μW)	19.1	19.7	19.3	19.5	19.7	22.2	19.0	17.8	15.2	19.6	19.7	23.2
RSR (V/μS)	71.3	65.5	82.9	72.4	62.1	67.4	72.7	86.2	80.1	69.8	71.9	66.7
FSR (V/μS)	65.3	61.0	77.4	66.4	63.7	67.7	67.5	60.2	60.2	65.5	74.5	73.7
|PSRR| (dB)	92.1	80.1	94.0	85.1	81.0	84.6	87.6	80.3	83.0	85.0	92.2	88.6
CMRR (dB)	76.0	79.9	74.1	76.0	75.3	79.6	76.9	77.2	73.1	76.8	75.6	75.7
Design Time (min)	19.88	7.02	22.57	22.00	16.52	22.57	21.21	13.41	22.57	22.3	15.78	22.57

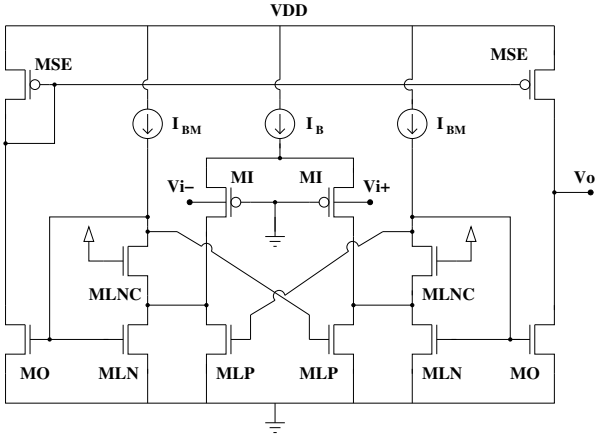


Figure 7: Bulk-driven OTA: circuit diagram.

parameters with their upper and lower bounds are illustrated in Table 11.

Table 10: Bulk driven OTA: Desired specifications.

Specifications	Desired value
Gain (dB)	≥ 42
UGB (MHz)	≥ 10
PM (degree)	≥ 60
Power Consumption (μW)	≤ 100
Rise slew rate ($V/\mu S$)	≥ 10
Fall slew rate ($V/\mu S$)	≥ 10

The mean design error over 25 independent bulk driven OTA design runs, the best design and worst design errors are illustrated in Table 12. The worst, best and mean specifications found during 25 independent design runs are indicated in Table 13. Fig. 9 shows

Table 11: Bulk-driven OTA: search space for design variables.

Parameter	Search space
Width of all transistors (μm)	1 to 100
Length of all transistors (μm)	0.2 to 5
I_B (μA)	3 to 50
I_{BM} (μA)	3 to 50
VDD (V)	0.6

Table 12: Bulk-driven OTA: Mean design error in % over 25 independent design runs, % error for best and worst design cases.

Algorithm	Mean design error	Error for best design	Error for worst design
EABC	0	0	0
ABC	1.26	0	9.6
GABC	1.41	0	8.0
MABC	0.14	0	2.0

how the cost function value reduces with the circuit evaluations.From the obtained results following observations can be made.

- From 25 design runs, EABC algorithm has successfully designed OTA all 25 times. ABC algorithm is successful for 13 times, GABC and MABC algorithms are successful for 11 and 22 times respectively. Thus EABC algorithm provides better accuracy of result.
- The average bulk driven OTA design time is 4.62 minutes with EABC, 9.07 minutes with ABC,

Table 13: Bulk-driven OTA: Mean of obtained specifications (MOS), Specifications of best obtained design (BOS), Specifications of worst obtained design (WOS) over 25 independent design runs

Specifications	EABC			ABC			GABC			MABC		
	MOS	BOS	WOS	MOS	BOS	WOS	MOS	BOS	WOS	MOS	BOS	WOS
Gain (dB)	43.4	46.7	42.1	41.9	42.7	37.9	42.0	43.0	38.6	42.8	43.0	42.5
UGB (MHz)	11.0	13.6	10.1	11.0	10.1	10.2	11.4	10.7	10.5	11.0	12.3	10.6
PM (degree)	62.6	61.7	60.4	62.3	63.9	60.4	62.8	62.5	60.3	64.0	61.3	58.8
PC (μW)	90.1	71.1	99.6	92.2	94.1	90.3	92.6	99.6	70.5	89.5	83.6	82.9
RSR (V/μS)	15.5	14.9	11.8	14.5	20.2	11.1	14.9	10.6	19.3	14.5	20.6	15.4
FSR (V/μS)	23.5	23.7	12.8	18.0	20.5	12.1	16.8	13.4	15.8	27.8	71.1	21.3
Design Time (min)	4.62	2.63	7.61	9.07	1.88	11.65	9.59	4.96	11.65	8.25	4.17	11.65

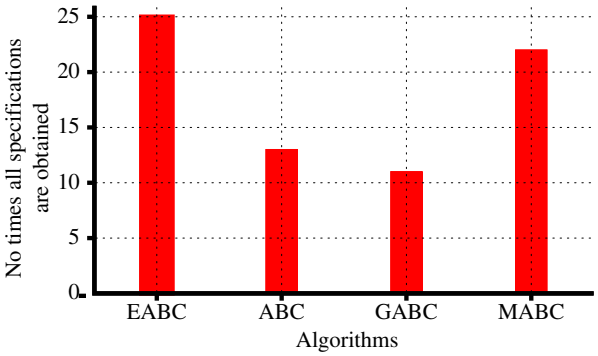


Figure 8: Bulk driven OTA: number of times all specifications are obtained during 25 design runs.

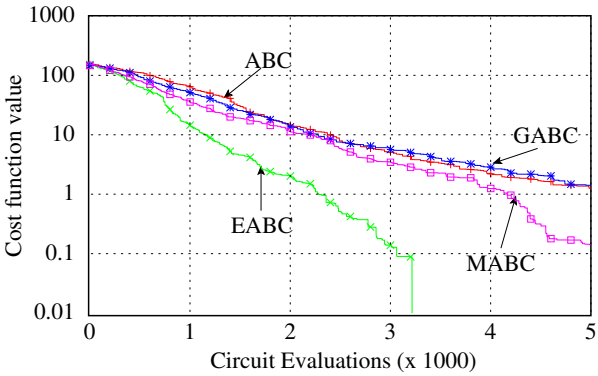


Figure 9: Bulk driven OTA: variations in mean error with circuit evaluations.

9.59 minutes with GABC and 8.25 minutes with MABC. It is clear that for OTA design problem, the convergence speed of EABC is better than ABC, GABC and MABC algorithm. The worst design OTA by ABC has error 9.69% while that for GABC and MABC is 8.07% and 2.00%, respectively.

- EABC algorithm can design blulk driven OTA faster and with better accuracy compared to ABC, GABC and MABC algorithms.

6 Conclusion

In this paper, we suggest modifications in ABC algorithm and based on that present new algorithm named EABC algorithm. EABC algorithm has proper balance between exploitation and exploration. The experiment conducted on 15 benchmark functions shows that EABC outperforms the ABC algorithm. EABC algorithm is also compared with other variants of the ABC algorithms such as GABC and MABC. The comparison shows that performance of EABC is better than GABC and MABC. We also present application of ABC based optimization algorithms for optimum design of CMOS circuits. With this design technique, even less experienced circuit designer can design the optimum circuit. We designed two circuits namely two-stage CMOS op-amp and bulk driven OTA using EABC, ABC, GABC and MABC algorithms. For the circuit design application, the EABC exhibits faster convergence speed with better accuracy of results compared to ABC, GABC and MABC algorithms. For two-stage op-amp, average design error is only 0.4% with EABC in contrast with 2.1% for ABC, 0.74% for GABC and 1.45% for MABC. The average design time of Two stage op-amp found for EABC is 19.88 minutes, while that is for 22 minutes for ABC, 21.21 minutes for GABC and 22.3 minutes for MABC. In case of bulk driven OTA, average design error is zero with EABC in contrast with 1.26% for ABC, 1.41% for GABC and 0.14% for MABC. The average time for designing bulk driven OTA using EABC is 4.62 minutes in comparison with 9.07 minutes for ABC, 9.59 minutes for GABC and 8.25

minutes for MABC. Thus, EABC algorithm is more promising algorithm compared to ABC algorithm.

References:

[1] S. Sivanandam and S. Deepa, *Introduction to genetic algorithms*. Springer Science & Business Media, 2007.

[2] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Neural Networks, 1995. Proceedings., IEEE International Conference on*, vol. 4, pp. 1942–1948 vol.4, Nov 1995.

[3] M. Dorigo and C. Blum, "Ant colony optimization theory: A survey," *Theoretical computer science*, vol. 344, no. 2, pp. 243–278, 2005.

[4] D. Simon, "Biogeography-based optimization," *Evolutionary Computation, IEEE Transactions on*, vol. 12, no. 6, pp. 702–713, 2008.

[5] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm," *Journal of global optimization*, vol. 39, no. 3, pp. 459–471, 2007.

[6] D. Karaboga and B. Basturk, "On the performance of artificial bee colony (abc) algorithm," *Applied soft computing*, vol. 8, no. 1, pp. 687–697, 2008.

[7] D. Karaboga and B. Akay, "A comparative study of artificial bee colony algorithm," *Applied Mathematics and Computation*, vol. 214, no. 1, pp. 108–132, 2009.

[8] K. Sundareswaran, P. Sankar, P. Nayak, S. P. Simon, and S. Palani, "Enhanced energy output from a pv system under partial shaded conditions through artificial bee colony," *Sustainable Energy, IEEE Transactions on*, vol. 6, no. 1, pp. 198–209, 2015.

[9] R. A. Vural, T. Yildirim, T. Kadioglu, and A. Basargan, "Performance evaluation of evolutionary algorithms for optimal filter design," *Evolutionary Computation, IEEE Transactions on*, vol. 16, no. 1, pp. 135–147, 2012.

[10] D. Karaboga and C. Ozturk, "A novel clustering approach: Artificial bee colony (abc) algorithm," *Applied soft computing*, vol. 11, no. 1, pp. 652–657, 2011.

[11] M. Ma, J. Liang, M. Guo, Y. Fan, and Y. Yin, "Sar image segmentation based on artificial bee colony algorithm," *Applied Soft Computing*, vol. 11, no. 8, pp. 5205–5214, 2011.

[12] M. Sonmez, "Artificial bee colony algorithm for optimization of truss structures," *Applied Soft Computing*, vol. 11, no. 2, pp. 2406–2418, 2011.

[13] X. Zhang, X. Zhang, S. Ho, and W. Fu, "A modification of artificial bee colony algorithm applied to loudspeaker design problem," *Magnetics, IEEE Transactions on*, vol. 50, pp. 737–740, Feb 2014.

[14] G. Nicosia, S. Rinaudo, and E. Sciacca, "An evolutionary algorithm-based approach to robust analog circuit design using constrained multi-objective optimization," *Knowledge-Based Systems*, vol. 21, no. 3, pp. 175–183, 2008.

[15] M. Barros, J. Guilherme, and N. Horta, "Analog circuits optimization based on evolutionary computation techniques," *INTEGRATION, the VLSI journal*, vol. 43, no. 1, pp. 136–155, 2010.

[16] R. A. Thakker, M. S. Baghini, and M. B. Patil, "Automatic design of low-power low-voltage analog circuits using particle swarm optimization with re-initialization," *Journal of Low Power Electronics*, vol. 5, no. 3, pp. 291–302, 2009.

[17] 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.

[18] W. f. Gao and S. y. Liu, "A modified artificial bee colony algorithm," *Computers & Operations Research*, vol. 39, no. 3, pp. 687–697, 2012.

[19] P. Allen and D. Holberg, *CMOS Analog Circuit Design*. OUP USA, 2012.

[20] B. Blalock and P. Allen, "A low-voltage, bulk-driven mosfet current mirror for cmos technology," in *Circuits and Systems, 1995. ISCAS '95., 1995 IEEE International Symposium on*, vol. 3, pp. 1972–1975 vol.3, Apr 1995.

[21] B. Blalock, P. Allen, and G. Rincon-Mora, "Designing 1-v op amps using standard digital cmos technology," *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on*, vol. 45, pp. 769–780, Jul 1998.

[22] R. He and L. Zhang, "Evaluation of modern mosfet models for bulk-driven applications," in *Circuits and Systems, 2008. MWSCAS 2008. 51st Midwest Symposium on*, pp. 105–108, Aug 2008.

[23] J. Carrillo, G. Torelli, R. Perez-Aloe, and J. Duque-Carrillo, "1-v rail-to-rail bulk-driven cmos ota with enhanced gain and gain-bandwidth product," in *Circuit Theory and Design, 2005. Proceedings of the 2005 European Conference on*, vol. 1, pp. I/261–I/264 vol. 1, Aug 2005.