ABC_DE_FP: A Novel Hybrid Algorithm for Complex Continuous Optimization Problems

Article *in* International Journal of Bio-Inspired Computation · January 2018

DOI: 10.1504/IJBIC.2018.10014476

2 authors:

Parul Agarwal
Jaypee Institute of Information Technology
24 PUBLICATIONS **287** CITATIONS
SEE PROFILE

Shikha Mehta
Jaypee Institute of Information Technology
47 PUBLICATIONS **561** CITATIONS
SEE PROFILE

ABC_DE_FP: a novel hybrid algorithm for complex continuous optimisation problems

Parul Agarwal* and Shikha Mehta

Department of Computer Science and Engineering,
Jaypee Institute of Information and Technology,
Noida, India
Email: parul.agarwal@jiit.ac.in
Email: shikha.mehta@jiit.ac.in
*Corresponding author

Abstract: Artificial bee colony (ABC) algorithm evolved as one of the efficient swarm intelligence-based algorithm in solving various global optimisation problems. Though numerous variants of ABC are available, algorithm depicts poor convergence rate in many situations. Therefore, maintaining balance between intensification and diversification of an algorithm still needs attention. In this context, a novel hybrid ABC algorithm (ABC_DE_FP) has been developed by integrating FPA and DE in original ABC algorithm. To assess the efficacy of proposed hybrid algorithm, it is primarily compared with contemporary ABC variants such as GABC, IABC and AABC over simple benchmark problems. Thereafter, it is evaluated with respect to original ABC, FPA, hybrid ABC_FP, ABC_DE and ABC_SN over CEC2014 optimisation problems for up to 100 dimensions. Results reveal that proposed algorithm considerably outperforms its counterparts in terms of minimum error value attained and convergence speed for majority of global numerical optimisation functions.

Keywords: artificial bee colony algorithm; CEC 2014 benchmark functions; nature inspired algorithms; flower pollination algorithm; FPA; differential evolution; convergence speed; complex continuous optimisation problems; minimum error value; intensification and diversification.

Reference to this paper should be made as follows: Agarwal, P. and Mehta, S. (2019) ‘ABC_DE_FP: a novel hybrid algorithm for complex continuous optimisation problems’, *Int. J. Bio-Inspired Computation*, Vol. 14, No. 1, pp.46–61.

Biographical notes: Parul Agarwal received her BTech and MTech in Computer Science and Engineering. Presently, she is pursuing her PhD from the JIIT under the supervision of Dr. Shikha Mehta. She is working on nature inspired algorithms, data mining and machine learning.

Shikha Mehta is an Associate Professor in department of computer science engineering at JIIT, Noida. She received her Doctorate from the University of Delhi, New Delhi, in 2013. She has experience of over 16 years in academics. She has participated in many international and national conferences in India and abroad and is actively involved as reviewer for many international journals and conference proceedings. She has guided approximately 100 graduate and post-graduate project thesis. She is currently guiding four PhD scholars. Her research interests include nature inspired algorithms, social network analysis, streaming data analytics, parallel and distributed databases, etc.

1 Introduction

Nature inspired algorithms have been successfully applied to optimise wide variety of numerical optimisation problems in the last few years. Algorithms state-of-art prospects and applications are defined in Agarwal and Mehta (2014). New progress in swarm intelligence-based algorithms along with current issues were presented in Duan and Luo (2015).

Artificial bee colony (ABC) algorithm, developed by Karaboga and Basturk (2008), is one of the swarm intelligence-based algorithm that imitates foraging behaviour of honey bees. ABC is an efficient nature inspired algorithm as it outperforms genetic algorithm (GA), differential evolution (DE), particle swarm optimisation (PSO) and evolution strategies (ES) (Karaboga and Akay, 2009). Karaboga and Basturk (2008) also proved the efficacy of ABC algorithm on various numerical optimisation functions against PSO, DE and evolutionary algorithm (EA). Empirical analysis of five nature inspired algorithms on CEC 2014 benchmark functions is illustrated in Agarwal and Mehta (2017). It was observed that ABC gives better solution as compared to other algorithms. Scientist and researchers were attracted towards ABC due to its excellent performance and simpler implementation.

In spite of having good efficiency and easier implementation ABC suffers from few drawbacks. As ABC algorithm is metaheuristic and stochastic in nature, it takes long time to get best results. Hence, convergence speed of ABC needs improvement. The algorithm should be prohibited from getting trapped in local optimal solutions. Regarding exploration and exploitation, ABC is inferior at exploitation due to its update equation for searching new solution (Zhu and Kwong, 2010). A good optimisation algorithm maintains balance between exploration and exploitation. This encouraged the researchers to develop modified versions of ABC algorithm. The modified versions were obtained either by tuning or adding some parameters in algorithm or hybridising some other algorithm or concepts in different phases of existing ABC algorithm. The aim for all algorithms is same, i.e., to improve solution quality or to accelerate execution or both. Zhu and Kwong (2010) introduced a new version of ABC algorithm called guided ABC (GABC) algorithm by modifying the update equation of solution. The modified equation for finding new solution was formed by integrating global best solution in existing equation. Akay and Karaboga (2012) presented a modification in typical ABC algorithm and tested it against real parameter optimisation problems. The changes in update equation of original ABC algorithm were made by adding new parameters.

A novel approach of introducing simulated annealing (SA) in ABC algorithm (ABC_SN) was given by Chen et al. (2012). Yurtkuran and Emel (2015) presented an adaptive version of ABC algorithm (AABC) by introducing various search strategies in different phases of algorithm for enhancing exploration and exploitation capabilities. An ABC algorithm with multiple search approaches was introduced by Gao et al. (2015).

Flower pollination algorithm (FPA) recently been used in some applications and few variants have been proposed by researchers. It was also used in optimising various clustering datasets when combined with K-means clustering algorithm (Agarwal and Mehta, 2015). Agarwal and Mehta (2016) introduced an enhanced version of FPA and tested it with datasets of different domains. A hybrid FPA was developed to handle ill conditioned systems of linear and non linear equations (Abdel-Baset and Hezam, 2016).

Other studies on benchmark problems include a new form of memetic algorithm (MA) for large-scale global optimisation (Mehta, 2016). Banati and Mehta (2012) introduced an automated tool (SEVO) for EAs, i.e., GA, MA, PSO and shuffled frog leaping algorithm (SFLA) and tested these algorithms over sixteen benchmark functions. It has been used in context aware filtering (Mehta and Banati, 2014).

Although various techniques and number of variants have been proposed, there exists no algorithm that finds exact solution to all kinds of optimisation problems. Rather there is no such algorithm that maintains perfect synchronisation between exploration and exploitation.

In view of obtaining better stability, solution quality and convergence speed, a new hybrid approach of ABC algorithm is introduced in this paper. The hybrid algorithm modifies standard ABC algorithm by incorporating FPA in onlooker bee phase and DE in employed and onlooker bee phases. The algorithm is tested on simple and complex CEC 2014 (Liang et al., 2013) benchmark functions.

Subsequent sections are organised as: Section 2 discusses standard optimisation algorithms. Section 3 elaborates proposed algorithm. Experimental study along with analysis of results is presented in Section 4. Section 5 is dedicated to conclusion along with direction for future work.

2 Overview of standard algorithms

2.1 *Artificial bee colony algorithm*

ABC algorithm is successful in optimising various objective functions. Algorithm uses three phases of bees: employed bees, onlooker bees and scout bees. Employed bees are associated with some food source (FS) and determine the new one in neighbourhood of existing FS. Hence the new solution is computed using equation (1).

$$\mathrm{POS}_{ij} = X_{ij} + \Phi_{ij}\left(X_{ij} - X_{kj}\right) \tag{1}$$

where j is a random integer in the range [1, D] and k ∈ (1, 2, …, FS) is a randomly chosen index that has to be different from i. FS is the number of FSs (population size) and D is the dimension of each FS. Φ_{ij} is a uniformly distributed real random number in the range [–1, 1]. After computing the new FS position, the nectar amount, i.e., fitness of solution (FS) is computed by applying equation (2).

$$\text{Fitness} = \begin{cases} \dfrac{1}{1+f_i}, & \text{if } f_i \geq 0 \\ 1+\text{abs}(f_i), & \text{if } f_i < 0 \end{cases} \tag{2}$$

where 'f_i' denotes objective function value. If fitness of computed solution improves (i.e., nectar amount is high) then associated FS is replaced by new one otherwise old one remains in population and a counter for entering into scout bee increases by 1. The information of neighbouring FSs are shared by onlooker bees. They explore new FSs using equation (1) depending upon the probability obtained from equation (3).

$$P_i = \frac{\text{Fitness}_i}{\sum_{i=1}^{FS} \text{Fitness}_i} \tag{3}$$

Onlooker bees compute the fitness of explored FSs using equation (2) and memorise the FS with highest nectar amount. If nectar amount does not improve throughout the life cycle and counter reaches 'limit' value then employed bee becomes scout bee. 'Limit' is a threshold value which has to be exceeded for entering into scout bee phase. The new random FS is explored by scout bee using equation (4).

$$X_{ij} = lb_j + \text{rand}(0,1)\left(ub_j - lb_j\right) \tag{4}$$

where i = 1, …, FS and j = 1, …, D, where FS is the number of FSs and D is dimension parameter of optimisation problem. ub and lb are upper and lower bound already defined for a given problem.

2.2 Differential evolution algorithm

DE is a strong EA developed by Storn and Price. There are number of schemes developed for executing algorithm, but DE/rand/1 is the most widely applied scheme and same has been used in this paper. DE algorithm uses three phases in sequence: mutation, crossover and selection.

2.2.1 Mutation

In this phase, mutant vector v_i is computed using each target vector x_i (candidate solution) as shown in equation (5).

$$v_i = x_a + F \cdot (x_b - x_c) \tag{5}$$

where i = 1, 2, …, FS, and a, b and c are randomly chosen (integer) index of individuals in the population which are different from each other. F is scaling factor between [0, 2].

2.2.2 Crossover

A new vector 'u' named trial vector is generated from mutant vector and target vector as given in equation (6).

$$u_{ij} = \begin{cases} v_{ij}, & \text{if rand}[0,1] \le Cr \text{ or } j == j_{rand} \\ x_{ij}, & \text{otherwise} \end{cases} \tag{6}$$

where j = 1, 2, …, D and j_{rand} is index selected randomly from 1 to D in order to ensure that trial vector get at least one parameter from mutant vector. Cr is the crossover probability to regulate the parameter of mutant vector in trial vector.

2.2.3 Selection

For optimisation problem, target vector for next generation is selected by applying greedy selection between trial vector and target vector on basis of fitness value of objective function.

2.3 Flower pollination algorithm

FPA, developed by Yang (2012) is inspired from pollination behaviour of flowering plants. FPA is based on following rules:

1 Flower constancy is the probability of reproduction which is proportional to the flowers similarity.

2 Global pollination is a term for biotic and cross pollination procedures.

Rule 1 and 2 can be combined to give following equation

$$x_i^{t+1} = x_i^t + L(x_i^t - g_*) \tag{7}$$

where x_i^t is ith solution of tth generation, g_* is the current best solution at present generation and L represents levy flight distribution (Yang, 2012) portraying potency of pollination.

3 Local pollination is carried out by abiotic and self pollination techniques.

Rule 1 and 3 can be combined to give following equation

$$x_i^{t+1} = x_i^t + \in \left(x_j^t - x_k^t\right) \tag{8}$$

where x_j^t and x_k^t are flowers of same species and $\in$ is the uniform distribution in range [0, 1].

4 A switch probability p is used to control the local and global pollination and it ranges from [0, 1].

3 Proposed hybrid ABC (ABC_DE_FP) algorithm

In ABC algorithm, the objective of bees is to find the FS with maximum nectar amount; and in FPA the objective is to find the flower with maximum pollination capability. Combining the two techniques employs that onlooker bee act as pollinators in flower pollination and the pollination is global in nature. FS in ABC algorithm is flower. Onlooker bee determines the flower with maximum nectar amount. Hence objective of hybrid algorithm is same as that of ABC and FPA. In literature it is established that these algorithms give considerable performance individually. Therefore, it is expected that combining two techniques with DE may enhance solution quality and convergence speed for achieving best solution as DE helps in escalating the exploitation of ABC algorithm.

Algorithm 1 Flower pollination algorithm step in hybrid ABC (ABC_DE_FP)

```
1. if rand > p
2. L is the step size determined by Levy distribution
3. Perform global pollination (global search) using equation (7)
4. else
5. Perform exploitation (local search) with mutation strategy of
   DE using equation (5)
6. end if
```

In hybrid ABC algorithm, FSs in the population are initialised randomly within the bounds already defined in objective function. In the employed bee phase, original ABC uses equation (1) to search for local solution in the neighbourhood of existing candidate solution. In literature this step of finding new solution cannot assure of achieving better results (Chen et al., 2012). Hence, it slows down the convergence speed. Moreover, if a better solution is found then algorithm might be caught in local optima. Thus, to overcome the drawbacks of poor solution quality and slow execution time, mutation strategy of DE [given in equation (5)] is inculcated in place of local search strategy [equation (1)] of employed and onlooker bee phase. After finding mutant solution, trial solution is generated using crossover strategy of DE [defined in equation (6)]. The probability of attaining better FS increases by applying greedy selection criteria over fitness values of trial and previous solution. This might improve the exploitation process (local search) of ABC algorithm as using mutant solution and crossover rate may helps to improve execution time of algorithm. Additionally, to maintain synchronisation between exploration and exploitation, FPA technique is incorporated into onlooker bee phase. Onlooker bee phase plays an important role in exploration as well as exploitation in original ABC algorithm. Thus, pseudo code of FPA, which is added into hybrid ABC, is shown in Algorithm 1.

Algorithm 2 Proposed algorithm (ABC_DE_FP)

```
1.  Initialise the input parameters FES, D, FS, limit, p and Cr
2.  Determine the initial solution randomly within search
    space defined by lower (lb) and upper bound (ub)
3.  Compute fitness from initialised food source using
    equation (2).
4.  For N_iter = 1: FES
        Employed bee phase
5.  For each food source(FS)
6.  Compute the mutant solution using mutant strategy of DE
    [equation (5)]
7.  Compute the trial solution using crossover strategy of DE
    [equation (6)]
8.  Determine the fitness of trial solution using equation (2).
9.  Apply greedy approach to find best solution using fitness
    of trial and previous solution
10. If solution improves, update the population with better
    food source else increment counter of scout by 1.
11. End for loop
12. Determine the probability 'pr' using fitness value of
    corresponding FS [equation (3)]
        Onlooker bee phase
13. For each 'i' food source (FS)
14. if (rand < pr(i))
15. Apply FPA as shown in Algorithm 1
16. Compute the trial solution using crossover strategy of DE
    [equation (6)]
17. Determine the fitness of trial solution using equation (2).
18. Apply greedy approach to find best solution using fitness
    of trial and previous solution
19. If solution improves, update the population with better
    food source else increment counter of scout by 1.
20. End if
21. End for loop
22. Memorise the global minima from updated population
23. If scout counter exceeds limit, enter into scout bee phase.
        Scout bee phase
24. Determine the new solution, i.e., explore new food source
    using equation (4)
25. Compute the fitness of explored solution
26. The algorithm again enters employed bee phase until
    termination condition is achieved.
27. End while
```

Pseudo code of proposed hybrid ABC algorithm is given in Algorithm 2. In onlooker bee phase, the balance between exploitation (local pollination) and exploration (global pollination) is maintained using a switch probability 'p'. Global pollination, i.e., exploring global optima in FPA is performed by using Levy distribution [equation (7)]. This helps in modelling the direction of search. The solution obtained from exploration process is termed as mutant solution in proposed algorithm. Local pollination, i.e., performing local search in neighbourhood of existing solution is performed by using mutation strategy of DE. Due to switch probability 'p', either local or global mutant solution is obtained. Thereafter, same steps of crossover and greedy selection defined in employed bee phase are applied. For each flower, if no better solution is obtained from employed and onlooker bee phase, a counter is incremented for each phase. If the counter reaches a limit, algorithm enters into scout bee phase. The cycle of three phases continues unless predefined numbers of fitness function evaluations are achieved. Input to the algorithm is maximum fitness function evaluations (FES), FS, dimension of FS (D), Limit, switch probability (p), crossover rate (Cr) and scale factor (F).

4 Experimental study and analysis of results

4.1 Experimental setup

The efficacy of proposed hybrid algorithm is initially tested on simple benchmark functions (Yurtkuran and Emel, 2015) and then on complex functions defined by IEEE congress on evolutionary computation 2014 (Liang et al., 2013). All algorithm are implemented in MATLAB R2013a version with 64 bit Windows7 operating system, Intel Core i5 processor with 16.0GB RAM. Algorithms employ certain parameters that are used to regulate their behaviours. List of parameter along with the values used in implementation are shown in Table 1. In ABC algorithm and its variants, the value of limit (Karaboga and Akay, 2009) is defined on basis of FS and dimension (D) given below in equation (9).

$$\text{Limit} = \text{FS} * \text{D} \tag{9}$$

Population size is taken as recommend in Shan et al. (2015) and Gao et al. (2015). The value of switch probability 'p' is suggested in Chakraborty et al. (2014). Moreover, value of Cr (Chakraborty et al., 2014) is analysed over 0.2 to 0.9 and the best value was found to be 0.7.

Table 1 Parameter setting

Algorithms	*Parameters*	*Values*
Common parameter	Population size (n)	40
FPA and ABC_FP, ABC_DE_FP	p	0.7
ABC_DE, ABC_DE_FP	Cr	0.7
	Mutation ratio	[0.2, 0.8]
ABC_SN	Beta	0.9
	Alpha	5E-7
	T	100

Scaling factor is taken as mutation ratio and is generated randomly. It is uniformly distributed array of number between 0.2 and 0.8. The limits are recommended in Hati et al. (2013). The best values of alpha, beta and T of SA are defined in Chen et al. (2012).

Meticulous experiments were performed in order to evaluate the performance of proposed algorithm. The efficacy of proposed algorithm (ABC_DE_FP) is evaluated with respect to contemporary ABC variants under following experiments:

- Experiment1: performance comparison of proposed algorithm over existing ABC variants on simple test functions.
- Experiment2: performance evaluation of proposed algorithm on CEC2014 benchmark functions.
  - Wilcoxon rank sum test analysis
  - run length distribution.

4.2 Experiment1: performance comparison of proposed algorithm over existing ABC variants on simple test functions

In this experiment proposed hybrid algorithm is compared with other ABC variants, i.e., GABC (Zhu and Kwong, 2010), improved ABC (IABC) (Gao and Liu, 2011) and adaptive ABC (AABC) (Yurtkuran and Emel, 2015). The formulation of test functions used in this experiment given in Yurtkuran and Emel (2015).

Hybrid ABC_DE_FP is executed on test functions and compared with various ABC variants on same input parameter values given in Yurtkuran and Emel (2015). Table 2 shows the best, mean, median, worst and standard deviation of 30 runs on ten test functions.

Highlighted values represent the minimum (best) value attained by respective algorithm. It can be observed from Table 2 that ABC_DE_FP performs better than ABC, GABC, IABC and AABC algorithms for majority of the test functions. Since AABC perform better than ABC, GABC and IABC, so comparison analysis of proposed algorithm is made against AABC. For F1 (Rosenbrock) function, best value of ABC_DE_FP improves by 63% though mean value lags by 34% approximately. The ratio of improvement is relatively high for ABC_DE_FP. For F2 and F3 functions, i.e., Ackley and Rastrigin functions, improvement in proposed algorithm is by 18.6% and 82% respectively at best value. While at mean value enhancement is by 10% and 16.3% with respect to AABC. At F5 function, i.e., Weierstarss function, ABC_DE_FP improves by 65% on best value although it lags by 2.3% only on mean value. For F6 (Schwefel 2.26), F7 (Shifted Sphere) and F10 (Shifted Rastrigin), ABC_DE_FP depicts comparable performance to AABC. For F8 (Shifted Schwefel 1.2), although mean value depreciates by 5% but its best value promotes by 15%. However, on function F9 (Shifted Rosenbrock), the best value is provided by GABC while ABC_DE_FP and AABC exhibit similar performance. On F4 (Griewank) function, ABC_DE_FP does not give better performance.

Overall, proposed algorithm ABC_DE_FP depicts either better or comparable performance with respect to best existing variant of ABC in most of the benchmark test problems. Since these optimisation functions were simple; next experiment evaluates the efficiency of ABC_DE_FP on complex problems.

4.3 Experiment 2: performance evaluation of proposed algorithm on CEC2014 benchmark functions

In order to assess the efficacy of proposed hybrid algorithm over complex optimisation problems, it is compared with original ABC algorithm (ABC), FPA, hybrid of ABC and FPA (ABC_FP), ABC and DE (ABC_DE) (Xiang et al., 2014) and ABC and simulated annealing (ABC_SN) (Chen et al., 2012). ABC_FP algorithm has been developed by author of this paper. The proposed hybrid ABC algorithm and its variants are evaluated on the 30 complex benchmark functions obtained from problem definition of CEC 2014 (Liang et al., 2013).

The main focus of our work is to develop an efficient algorithm for single objective optimisation problems. These benchmark functions are best defined in problem suit of CEC 2014 (Liang et al., 2013). Single objective benchmark functions of CEC 2014 can be converted into dynamic, niching composition and many other class of problems. Thus, developing an algorithm for single objective complex optimisation problems of CEC2014 can be transformed to various class of problem.

CEC 2014 test functions are composed of several new features of basic function for the purpose of optimisation. These functions are evaluated on 10, 30, 50 and 100 dimension. Thirty functions are subdivided into four subclasses named as: unimodal functions (F1–F3), simple multimodal functions (F4–F16), hybrid functions (F17–F22) and composition functions (F23–F30). Search range is defined from –100 to +100 for all test functions. Bias (Fi*) for all test functions ranges from 100 for function F1 with increment of 100 for every next consecutive function upto 30,000 for function F30. Maximum function evaluations (MaxFes) are treated as terminating criteria and taken as 10,000 * D, where D is dimension of problem. Each algorithm is executed 30 times independently with 10,000*D fitness function evaluations in each run. Results are recorded in the form of minimum fitness value and afterwards error is computed as below:

$$\text{Error} = \text{F}\min - \text{Fi}^* \tag{10}$$

Table 2 Best, median, worst, mean and standard deviation fitness values of 30 runs at 100 dimensions

	Algorithm	*Best*	*Median*	*Worst*	*Mean*	*S.D.*
F1	ABC	5.51E+01	1.91E+02	1.35E+02	1.26E+02	3.55E+01
	GABC	6.17E+01	1.97E+02	1.44E+02	1.35E+02	2.96E+01
	IABC	8.06E+01	2.05E+02	1.32E+02	1.32E+02	2.81E+01
	AABC	6.22E+01	1.88E+02	1.10E+02	1.15E+02	3.14E+01
	ABC_DE_FP	*2.33E+01*	1.56E+02	2.10E+02	1.55E+02	4.53E+01
F2	ABC	2.769E-07	1.558E-06	6.007E-07	6.968E-07	3.246E-07
	GABC	2.779E-12	7.685E-12	4.268E-12	4.557E-12	1.182E-12
	IABC	4.414E-13	9.921E-13	6.173E-13	6.206E-13	1.028E-13
	AABC	1.714E-13	2.105E-13	1.892E-13	1.895E-13	9.253E-15
	ABC_DE_FP	*1.39E-13*	1.71E-13	2.21E-13	1.71E-13	1.82E-14
F3	ABC	1.119E-10	2.30E+00	8.559E-03	3.971E-01	5.807E-01
	GABC	4.547E-13	3.201E-10	4.150E-12	2.372E-11	5.972E-11
	IABC	4.547E-13	3.411E-12	1.023E-12	1.114E-12	5.861E-13
	AABC	3.411E-13	7.958E-13	5.684E-13	5.343E-13	1.162E-13
	ABC_DE_FP	*6.23E-14*	4.55E-13	1.11E-12	4.47E-13	2.49E-13
F4	ABC	1.721E-14	4.139E-11	2.275E-13	3.540E-12	9.267E-12
	GABC	1.443E-15	2.136E-08	2.998E-15	7.734E-10	3.898E-09
	IABC	1.887E-15	4.451E-07	3.442E-15	1.484E-08	8.127E-08
	AABC	9.992E-16	1.991E-10	2.276E-15	6.674E-12	3.635E-11
	ABC_DE_FP	*1.89E-15*	6.66E-10	2.65E-06	2.16E-07	6.17E-07
F5	ABC	1.033E-04	2.421E-04	1.677E-04	1.638E-04	4.978E-05
	GABC	1.137E-13	2.274E-13	1.847E-13	1.791E-13	5.022E-14
	IABC	8.527E-14	2.274E-13	1.421E-13	1.478E-13	4.403E-14
	AABC	5.684E-14	1.137E-13	1.137E-13	9.095E-14	2.935E-14
	ABC_DE_FP	*1.99E-14*	1.14E-13	1.71E-13	9.31E-14	3.72E-14
F6	ABC	−4.118E+04	−4.005E+04	−4.059E+04	−4.059E+04	2.61E+05
	GABC	−4.190E+04	−4.115E+04	−4.166E+04	−4.163E+04	1.53E+05
	IABC	−4.190E+04	−4.178E+04	−4.190E+04	−4.188E+04	4.39E+04
	AABC	−4.190E+04	−4.190E+04	−4.190E+04	−4.190E+04	6.021E-06
	ABC_DE_FP	*-4.19E+04*	-4.19E+04	-4.17E+04	-4.19E+04	6.86E+01
F7	ABC	−4.500E+02	−4.500E+02	−4.500E+02	−4.500E+02	1.812E-13
	GABC	−4.500E+02	−4.500E+02	−4.500E+02	−4.500E+02	8.527E-14
	IABC	−4.500E+02	−4.500E+02	−4.500E+02	−4.500E+02	4.903E-14
	AABC	−4.500E+02	−4.500E+02	−4.500E+02	−4.500E+02	2.820E-02
	ABC_DE_FP	*-4.50E+02*	-4.50E+02	-4.50E+02	-4.50E+02	2.31E-13
F8	ABC	1.24E+05	1.69E+05	1.84E+05	1.61E+05	2.43E+04
	GABC	1.47E+05	1.54E+05	1.78E+05	1.60E+05	1.15E+04
	IABC	1.88E+05	1.90E+05	2.04E+05	1.95E+05	7.92E+03
	AABC	1.33E+05	1.46E+05	1.78E+05	1.52E+05	1.88E+04
	ABC_DE_FP	*1.13E+05*	1.47E+05	1.64E+05	1.44E+05	1.38E+04
F9	ABC	3.94E+02	3.95E+02	3.97E+02	3.95E+02	1.39E+00
	GABC	*2.92E+02*	3.97E+02	4.87E+02	4.14E+02	4.12E+01
	IABC	3.93E+02	3.98E+02	4.61E+02	4.10E+02	2.86E+01
	AABC	3.90E+02	3.92E+02	4.00E+02	3.03E+02	3.88E+00
	ABC_DE_FP	3.92E+02	4.38E+02	5.31E+02	4.38E+02	3.35E+01
F10	ABC	−3.290E+02	−3.280E+02	−3.270E+02	−3.279E+02	7.213E-01
	GABC	−3.300E+02	−3.300E+02	−3.300E+02	−3.300E+02	7.204E-07
	IABC	−3.300E+02	−3.300E+02	−3.300E+02	−3.300E+02	5.684E-14
	AABC	−3.300E+02	−3.300E+02	−3.300E+02	−3.300E+02	0.00E+00
	ABC_DE_FP	*–3.30E+02*	–3.30E+02	–3.30E+02	–3.30E+02	0.00E+00

Table 3 Mean error values of algorithms over 30 runs for 10 D

Func.	*D = 10*					
	ABC_DE_FP	*ABC*	*FPA*	*ABC_FP*	*ABC_DE*	*ABC_SN*
1	2.04E+05	2.97E+05	*0.00E+00*	2.88E+05	4.21E+05	1.03E+06
2	1.13E+02	1.35E+02	*0.00E+00*	3.32E+01	3.86E+02	3.99E+05
3	2.01E+02	3.81E+02	*0.00E+00*	2.47E+02	5.30E+02	2.21E+03
4	1.22E+00	2.50E-01	1.18E+01	*8.00E-02*	4.28E+00	7.95E+00
5	*1.62E+01*	1.88E+01	2.02E+01	1.66E+01	1.93E+01	2.02E+01
6	2.10E+00	2.15E+00	2.90E+00	2.32E+00	2.38E+00	*1.28E+00*
7	*0.00E+00*	*0.00E+00*	7.00E-02	1.00E-02	2.00E-02	7.00E-02
8	*0.00E+00*	*0.00E+00*	9.16E+00	*0.00E+00*	5.00E-02	3.10E-01
9	*4.19E+00*	8.21E+00	1.31E+01	7.62E+00	7.00E+00	1.29E+01
10	*8.00E-02*	1.10E-01	2.38E+02	1.00E-01	1.55E+00	1.97E+01
11	*1.58E+02*	2.54E+02	6.68E+02	2.40E+02	4.27E+02	5.47E+02
12	*1.40E-01*	2.60E-01	3.90E-01	*1.40E-01*	4.60E-01	5.80E-01
13	*1.20E-01*	1.30E-01	2.50E-01	*1.20E-01*	1.60E-01	1.90E-01
14	2.10E-01	*1.50E-01*	1.80E-01	*1.50E-01*	2.30E-01	1.60E-01
15	*7.20E-01*	9.40E-01	1.22E+00	8.30E-01	1.25E+00	2.34E+00
16	*2.12E+00*	2.23E+00	2.90E+00	2.14E+00	2.54E+00	3.17E+00
17	1.18E+05	2.38E+05	*7.02E+01*	2.06E+05	1.62E+05	7.66E+04
18	9.90E+02	8.41E+02	*5.35E+00*	1.81E+02	4.70E+03	9.58E+02
19	*3.10E-01*	4.50E-01	1.90E+00	5.30E-01	4.20E-01	1.28E+00
20	2.03E+02	4.06E+02	*6.95E+00*	7.82E+01	1.40E+03	2.76E+02
21	4.65E+03	1.84E+04	*1.10E+01*	8.32E+03	2.00E+04	1.33E+04
22	*5.20E-01*	2.12E+00	2.41E+01	1.54E+00	1.42E+00	7.07E+00
23	3.29E+02	2.58E+02	3.29E+02	*2.52E+02*	3.29E+02	3.19E+02
24	*1.14E+02*	1.20E+02	1.19E+02	1.21E+02	1.17E+02	1.23E+02
25	*1.29E+02*	1.40E+02	1.30E+02	1.44E+02	1.45E+02	1.70E+02
26	1.00E+02	*1.00E+02*	*1.00E+02*	1.00E+02	1.00E+02	1.00E+02
27	1.85E+02	*3.64E+01*	8.12E+01	8.31E+01	6.32E+01	2.90E+02
28	*3.67E+02*	*3.67E+02*	4.15E+02	3.77E+02	3.70E+02	3.75E+02
29	3.25E+02	3.24E+02	*2.47E+02*	2.56E+02	6.88E+02	3.49E+02
30	5.99E+02	6.57E+02	*5.27E+02*	5.99E+02	7.31E+02	6.59E+02

Table 4 Mean error values of algorithms over 30 runs for 30 D

Func.	*D = 30*					
	ABC_DE_FP	*ABC*	*FPA*	*ABC_FP*	*ABC_DE*	*ABC_SN*
1	9.25E+06	1.13E+07	*2.95E+05*	2.91E+06	1.83E+07	3.23E+07
2	1.63E+03	4.31E+02	2.49E+05	1.11E+02	3.56E+04	5.11E+07
3	6.89E+02	9.29E+02	*5.18E+02*	*6.33E+02*	2.66E+03	4.60E+03
4	4.86E+01	*2.37E+01*	9.06E+01	3.46E+01	6.39E+01	1.41E+02
5	*2.00E+01*	2.04E+01	2.08E+01	2.00E+01	2.04E+01	2.06E+01
6	*1.46E+01*	1.53E+01	2.11E+01	1.53E+01	1.46E+01	1.47E+01
7	*0.00E+00*	*0.00E+00*	1.90E-01	*0.00E+00*	2.00E-02	6.90E-01
8	*0.00E+00*	*0.00E+00*	6.93E+01	*0.00E+00*	1.30E-01	2.30E+00
9	*4.58E+01*	8.55E+01	1.13E+02	9.48E+01	4.67E+01	9.87E+01
10	*4.50E-01*	1.73E+00	2.29E+03	4.70E-01	3.65E+00	4.16E+01
11	*1.83E+03*	2.22E+03	3.60E+03	2.02E+03	2.47E+03	3.93E+03
12	*1.50E-01*	4.10E-01	1.08E+00	*1.50E-01*	5.10E-01	1.01E+00
13	*2.20E-01*	2.40E-01	5.30E-01	2.30E-01	2.80E-01	4.00E-01
14	2.60E-01	2.00E-01	2.70E-01	*1.90E-01*	2.70E-01	*1.90E-01*
15	*5.12E+00*	1.08E+01	3.30E+01	7.71E+00	7.62E+00	1.98E+01
16	*9.54E+00*	1.06E+01	1.22E+01	9.77E+00	1.02E+01	1.19E+01

Table 4 Mean error values of algorithms over 30 runs for 30 D (continued)

Func.	*D = 30*					
	ABC_DE_FP	*ABC*	*FPA*	*ABC_FP*	*ABC_DE*	*ABC_SN*
17	3.87E+06	3.26E+06	*1.30E+03*	1.61E+06	4.40E+06	3.45E+06
18	1.66E+03	2.29E+03	*2.93E+02*	3.17E+02	3.23E+03	1.17E+05
19	*7.03E+00*	7.13E+00	1.13E+01	7.04E+00	8.00E+00	1.12E+01
20	7.58E+03	5.73E+03	*2.13E+02*	8.57E+03	7.87E+03	1.27E+04
21	3.59E+05	4.52E+05	*6.74E+02*	2.03E+05	7.72E+05	1.01E+06
22	2.64E+02	*1.06E+02*	2.39E+02	2.99E+02	4.02E+02	3.12E+02
23	*3.15E+02*	3.15E+02	3.15E+02	3.15E+02	3.16E+02	3.17E+02
24	*2.23E+02*	2.27E+02	2.34E+02	2.28E+02	2.27E+02	2.26E+02
25	*2.04E+02*	2.08E+02	2.05E+02	2.08E+02	2.10E+02	2.08E+02
26	1.04E+02	*1.00E+02*	1.04E+02	1.00E+02	1.01E+02	1.00E+02
27	4.12E+02	4.10E+02	4.96E+02	*4.06E+02*	4.24E+02	5.96E+02
28	8.56E+02	9.67E+02	1.33E+03	1.02E+03	*8.40E+02*	9.30E+02
29	1.41E+03	1.11E+03	7.77E+06	*9.30E+02*	1.80E+03	2.28E+03
30	3.40E+03	4.08E+03	3.98E+03	*2.10E+03*	5.90E+03	1.88E+04

Table 5 Mean error values of algorithms over 30 runs for 50 D

Func.	*D = 50*					
	ABC_DE_FP	*ABC*	*FPA*	*ABC_FP*	*ABC_DE*	*ABC_SN*
1	1.27E+07	1.94E+07	*2.66E+06*	6.44E+06	2.53E+07	4.84E+07
2	*2.61E+03*	1.35E+03	6.48E+06	6.09E+02	8.09E+03	2.69E+08
3	*7.90E+03*	8.50E+03	9.48E+03	8.60E+03	1.00E+04	2.18E+04
4	*3.83E+01*	4.84E+01	1.78E+02	6.97E+01	8.21E+01	1.94E+02
5	*2.00E+01*	2.05E+01	2.11E+01	2.00E+01	2.05E+01	2.08E+01
6	*2.53E+01*	3.34E+01	4.47E+01	3.18E+01	3.01E+01	2.68E+01
7	*0.00E+00*	*0.00E+00*	9.40E-01	*0.00E+00*	7.00E-02	1.53E+00
8	*0.00E+00*	*0.00E+00*	1.52E+02	*0.00E+00*	6.60E-01	5.37E+00
9	*1.02E+02*	2.02E+02	2.72E+02	2.10E+02	1.03E+02	2.30E+02
10	*8.90E-01*	4.92E+00	4.53E+03	1.25E+00	2.31E+01	1.11E+02
11	*4.02E+03*	4.92E+03	6.67E+03	4.54E+03	4.66E+03	7.95E+03
12	*1.20E-01*	5.00E-01	1.52E+00	1.40E-01	5.20E-01	1.21E+00
13	*2.90E-01*	3.40E-01	6.50E-01	3.20E-01	3.00E-01	4.90E-01
14	*1.80E-01*	2.50E-01	3.30E-01	2.40E-01	2.90E-01	2.00E-01
15	*1.22E+01*	2.71E+01	1.02E+02	1.86E+01	1.56E+01	5.03E+01
16	*1.75E+01*	1.94E+01	2.16E+01	1.82E+01	1.83E+01	2.12E+01
17	4.61E+06	8.67E+06	3.36E+04	*2.35E+06*	1.09E+07	8.67E+06
18	*1.43E+03*	4.35E+03	1.93E+03	1.44E+03	2.03E+03	9.54E+05
19	2.29E+01	*1.75E+01*	4.73E+01	1.77E+01	2.66E+01	4.11E+01
20	3.40E+04	2.12E+04	*1.52E+03*	3.38E+04	2.74E+04	4.13E+04
21	2.47E+06	3.90E+06	*9.68E+03*	1.66E+06	5.29E+06	5.92E+06
22	7.45E+02	*1.06E+02*	8.68E+02	7.80E+02	1.01E+03	9.21E+02
23	*3.44E+02*	3.44E+02	3.44E+02	3.44E+02	3.49E+02	3.64E+02
24	*2.55E+02*	2.58E+02	2.91E+02	2.58E+02	2.67E+02	2.59E+02
25	*2.11E+02*	2.16E+02	2.12E+02	2.14E+02	2.18E+02	2.19E+02
26	*1.00E+02*	*1.00E+02*	1.20E+02	1.00E+02	1.01E+02	1.01E+02
27	1.12E+03	*5.18E+02*	1.40E+03	1.06E+03	1.07E+03	1.36E+03
28	*1.20E+03*	1.75E+03	2.53E+03	1.97E+03	1.29E+03	1.37E+03
29	*1.50E+03*	1.68E+03	9.33E+07	1.57E+03	1.24E+04	1.75E+04
30	1.13E+04	1.12E+04	2.09E+04	*1.01E+04*	1.22E+04	1.32E+04

Table 6 Mean error values of algorithms over 30 runs for 100 D

Func.	*D = 100*					
	ABC_DE_FP	*ABC*	*FPA*	*ABC_FP*	*ABC_DE*	*ABC_SN*
1	*3.03E+07*	1.73E+08	4.83E+07	3.04E+07	1.33E+08	7.84E+08
2	*1.94E+04*	5.19E+04	3.99E+08	5.44E+04	1.81E+06	1.65E+10
3	*2.41E+04*	3.59E+04	2.43E+04	3.54E+04	3.31E+04	3.19E+05
4	*1.63E+02*	2.45E+02	4.87E+02	1.64E+02	1.99E+02	1.41E+03
5	*2.00E+01*	2.10E+01	2.13E+01	2.00E+01	2.08E+01	2.13E+01
6	*8.37E+01*	1.06E+02	1.08E+02	9.09E+01	8.43E+01	7.61E+01
7	*0.00E+00*	1.11E+00	3.85E+00	0.00E+00	4.00E-02	6.26E+01
8	*1.58E-1*	8.37E+01	4.42E+02	4.90E-01	4.61E+00	7.65E+01
9	*4.04E+02*	9.94E+02	6.82E+02	8.06E+02	4.07E+02	8.60E+02
10	*2.49E+00*	1.63E+03	1.07E+04	3.00E+00	8.40E+01	2.01E+03
11	*1.13E+04*	1.66E+04	1.45E+04	1.31E+04	1.29E+04	2.67E+04
12	*2.90E-01*	1.37E+00	2.35E+00	3.00E-01	8.90E-01	3.46E+00
13	*5.10E-01*	1.26E+00	6.40E-01	5.20E-01	*5.10E-01*	1.05E+00
14	*3.50E-01*	1.33E+01	*3.50E-01*	1.64E+00	4.00E-01	6.10E-01
15	*4.36E+01*	6.35E+02	4.16E+02	6.35E+01	5.45E+01	1.79E+05
16	*3.49E+01*	4.47E+01	4.52E+01	3.52E+01	4.21E+01	4.72E+01
17	*1.27E+05*	8.05E+07	4.46E+05	6.52E+06	5.33E+07	1.65E+08
18	*2.30E+03*	3.62E+06	3.35E+03	2.36E+03	3.20E+03	1.80E+09
19	*7.53E+01*	1.14E+02	1.39E+02	7.57E+01	1.01E+02	1.69E+02
20	*1.50E+04*	1.69E+05	2.25E+04	1.50E+05	1.46E+05	3.52E+06
21	*2.66E+05*	5.05E+07	2.95E+05	4.30E+06	2.47E+07	1.26E+08
22	*2.00E+03*	3.85E+03	2.02E+03	2.49E+03	3.31E+03	5.08E+03
23	*3.48E+02*	3.60E+02	3.51E+02	3.49E+02	3.60E+02	6.43E+02
24	*3.54E+02*	3.56E+02	4.45E+02	3.60E+02	3.72E+02	3.58E+02
25	*2.41E+02*	2.71E+02	2.42E+02	2.47E+02	2.59E+02	3.27E+02
26	*2.02E+02*	2.05E+02	1.89E+02	1.97E+02	2.02E+02	1.10E+02
27	*2.36E+03*	2.59E+03	3.28E+03	2.16E+03	2.37E+03	3.18E+03
28	*3.08E+03*	8.66E+03	6.77E+03	7.26E+03	2.78E+03	3.23E+03
29	*4.35E+03*	1.08E+04	4.20E+08	4.41E+03	6.39E+04	3.64E+06
30	*1.41E+04*	1.84E+05	2.30E+04	1.56E+04	1.16E+05	7.14E+05

Performance is assessed on basis of best, worst, mean, median and standard deviation of 30 error values (runs) on each function. To statistically determine the difference in algorithms, Wilcoxon rank sum test analysis is performed. Computational complexity of algorithms is analysed through run length distribution graphs. The function error values at each run are recorded after (0.01, 0.02, 0.03, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0) * MaxFes fitness function evaluations. At each iteration point, mean of 30 run is calculated for every function.

The mean of error values for ABC_DE_FP, ABC, FPA, ABC_FP, ABC_DE and ABC_SN algorithms on 30 independent runs at 10 and 30 dimensions are shown in Table 3 and 4. Table 5 and 6 represents mean error values of algorithms at 50 and 100 dimensions respectively. With respect to nature inspired algorithms on benchmark functions, order of magnitude signifies the accuracy of algorithm. Data of tables are shown with two decimal places. Acceptable tolerance of known optimal value of zero is .01. Also, best mean error value is highlighted for each function. Mean value of algorithms are statistically compared with other algorithms through Wilcoxon rank sum test analysis which is presented in sub section.

4.3.1 Wilcoxon rank sum test analysis

Wilcoxon rank sum test represent the statistical test analysis for determining the significant difference in error values of proposed versus contemporary algorithms. For each function, rank sum ‘p’ is computed for two compared algorithms over 30 runs. Significantly, $p < 0.05$ and maximum number evaluations are defined by D * 10,000. Mean of 30 error values is calculated for compared algorithm (mean_error_value1) and ABC_DE_FP (mean_error_value2). Following steps are executed for Wilcoxon rank sum test analysis:

```
Input: p value, mean_error_value1, mean_error_value2
if $p < 0.05$ and mean_error_value1 < mean_error_value2
return significantly better (+)
else if $p < 0.05$ and mean_error_value1 > mean_error_value2
return significantly worse (−)
else
return no significant difference (≈)
```

Wilcoxon rank sum test shown in Table 7 recapitulate the experimental results of every algorithm on 10, 30, 50 and 100 dimensions. For 100 dimensions, out of total 30 (CEC2014) benchmark functions, ABC_DE_FP performs better than ABC for 17 functions while it perform worse for 9 functions only. For rest of the four functions, there is no significant difference. Similarly ABC_DE_FP performs better for 20 functions than FPA and worse for seven functions only. Correspondingly all the results of Table 7 can be analysed. It can be noticed that for the majority of functions, proposed algorithm ABC_DE_FP performs better than all its defined variants. ABC_DE_FP wins over our own variant ABC_FP at higher dimension (100D) though it is inferior at 10, 30 and 50 dimensions. Hence ABC_DE_FP overshadow ABC_FP (own variant) as well as all the other contemporary algorithms on majority of benchmark functions.

Table 7 Wilcoxon rank sum test analysis

Vs. ABC_DE_FP		*100 D*	*50 D*	*30 D*	*10 D*
ABC	Better(+)	9	9	7	4
	Worse(−)	17	16	14	14
	No sig diff(≈)	4	5	9	12
FPA	Better(+)	7	5	7	13
	Worse(−)	20	20	20	15
	No sig diff(≈)	3	5	3	2
ABC_FP	Better(+)	10	17	18	9
	Worse(−)	16	9	7	8
	No sig diff(≈)	4	4	5	13
ABC_DE	Better(+)	6	2	1	1
	Worse(−)	16	16	16	22
	No sig diff(≈)	8	12	13	7
ABC_SN	Better(+)	5	3	5	4
	Worse(−)	24	24	25	23
	No sig diff(≈)	1	3	0	3

4.3.2 Computational complexity analysis (run length distribution)

Performance of proposed hybrid algorithm (ABC_DE_FP) with respect to competitive algorithms is exhibited in the form of convergence graphs for 100 dimensions. Computational complexity of algorithms over the mean error value attained on fixed number of iterations is analysed in subsections.

Figure 1 RLD for F1 (see online version for colours)

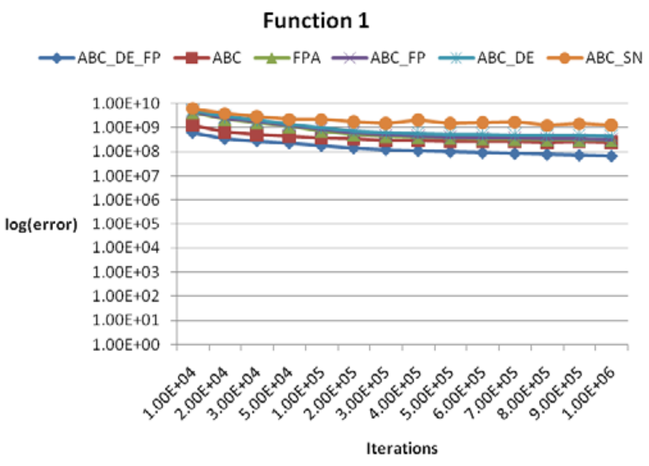


Figure 2 RLD for F2 (see online version for colours)

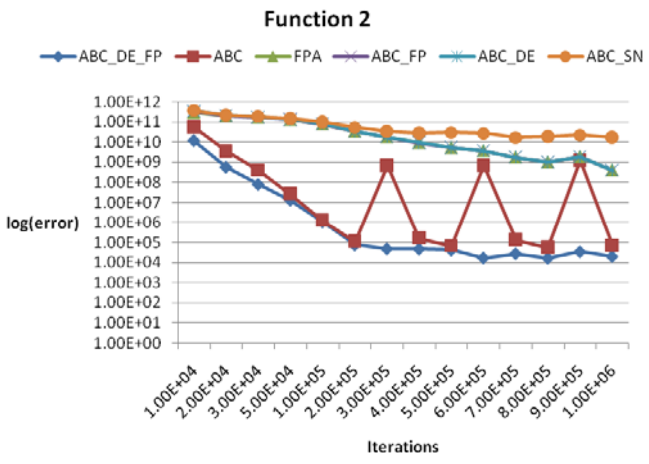
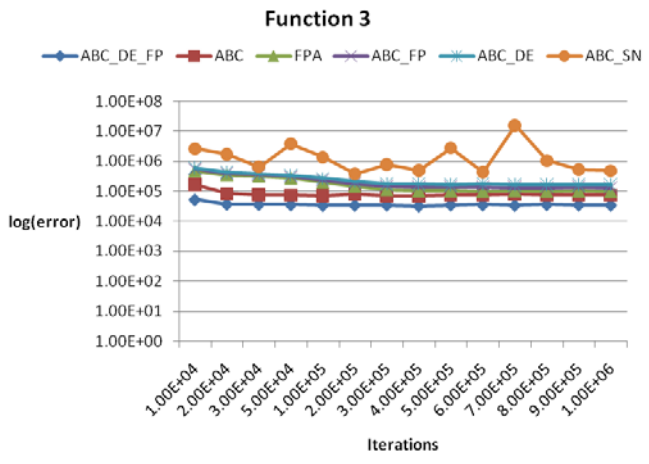


Figure 3 RLD for F3 (see online version for colours)



4.3.2.1 Unimodal functions

Run length distribution representing the behaviour of algorithms on unimodal functions is shown in Figures 1–3. It has been observed that minimum error is achieved by ABC_DE_FP as compared to other algorithms. For F1 function, rate of convergence for all competitor algorithms is nearly same. However, proposed algorithm reaches its minima at 2.00E+05 iteration. In F2 function, ABC_DE_FP exhibit rapid pace of convergence and obtain its global minima at 6.00E+05 iteration. ABC portrays non-uniform convergence speed and comparable performance to ABC_DE_FP at 2.00E+05 iteration. This is due to the lack of proper balance between exploration and exploitation techniques. In hybrid ABC algorithm, this drawback is overcome by integrating FPA and DE. Thus, ABC_DE_FP depicts uniform convergence speed while attaining global minima. In F3 function, though algorithm depicts nearly same convergence rate yet ABC_DE_FP achieve minimum error value.

Figure 4 RLD for F4 (see online version for colours)

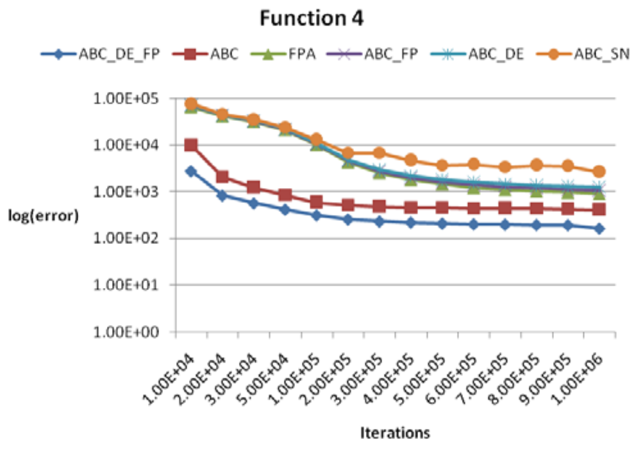
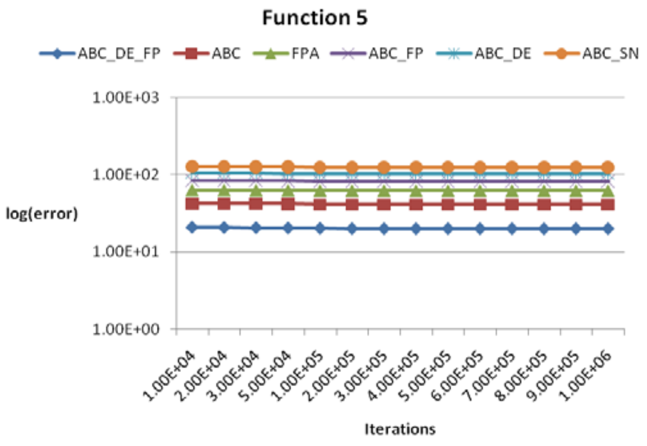


Figure 5 RLD for F5 (see online version for colours)



4.3.2.2 *Simple multimodal functions*

Convergence graphs of proposed ABC_DE_FP with its competitive algorithms for simple multimodal functions are shown in Figures 4–16. Algorithms do not depict any rate of change in the convergence speed for functions F5, F6, F9, F11, F13 and F16. However, minimum error value is shown by ABC_DE_FP. This is due to the fact that once a minimum value is achieved; algorithm is not able to exploit more promising regions of search space. For functions F4, F7, F12 and F15, ABC_DE_FP depicts uniform convergence speed, i.e., shape of curve is almost rectangular hyperbola. Over the iterations algorithm performance gets stagnate, however global minimum is first achieved by ABC_DE_FP. On F14 function, ABC_DE_FP reaches its minima at 2.00E+04 iteration and shows no change in rate of convergence afterwards. For the functions F8 and F10, ABC and ABC_DE_FP depicts irregular and zig zag behaviour over the iterations. This behaviour shows that algorithm’s convergence rate is dependent on function’s property. Overall analysis concludes that ABC_DE_FP is the first one to achieve minimum error value on simple multimodal functions.

Figure 6 RLD for F6 (see online version for colours)

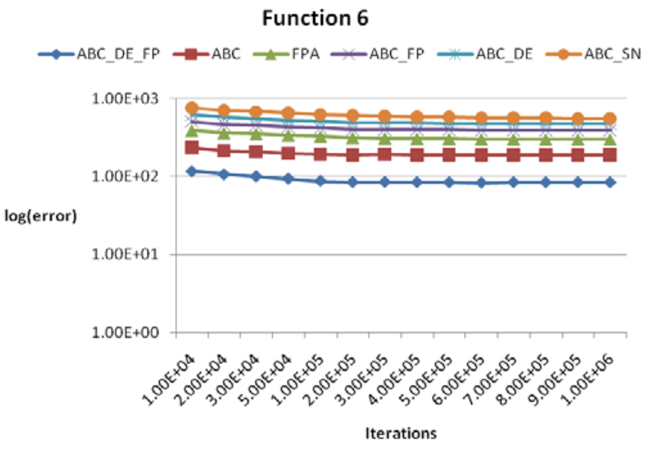


Figure 7 RLD for F7 (see online version for colours)

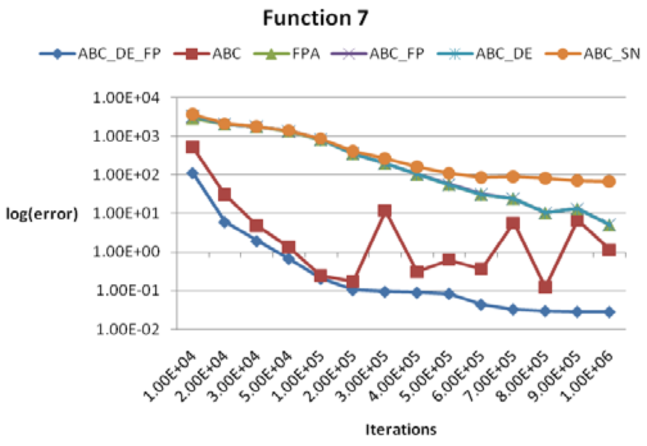


Figure 8 RLD for F8 (see online version for colours)

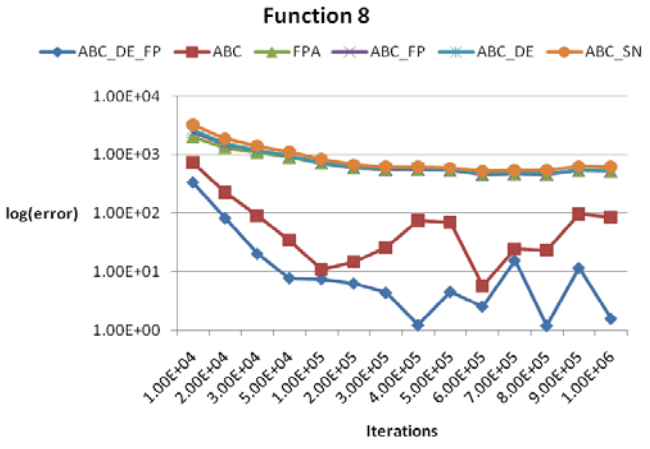


Figure 9 RLD for F9 (see online version for colours)

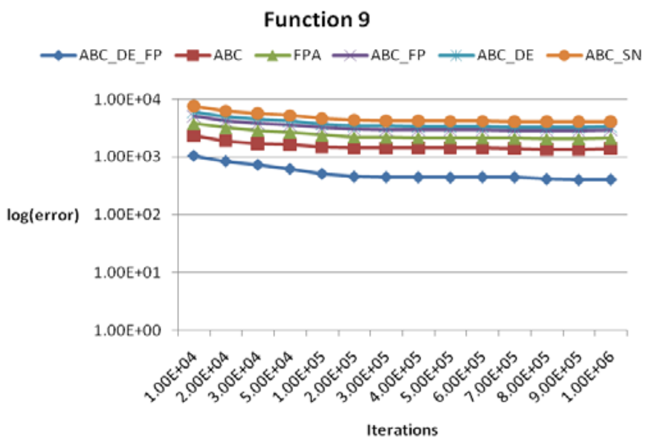


Figure 10 RLD for F10 (see online version for colours)

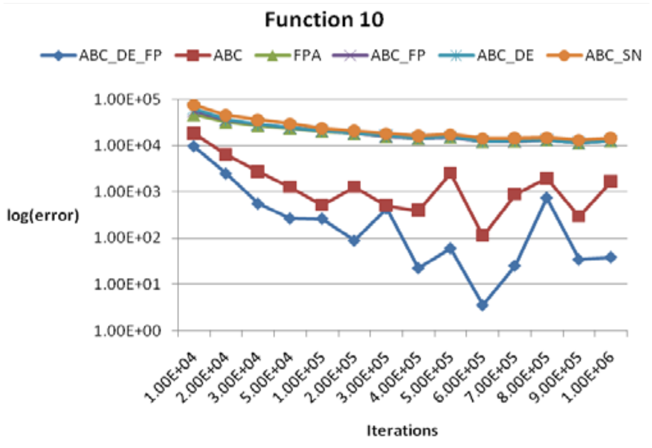
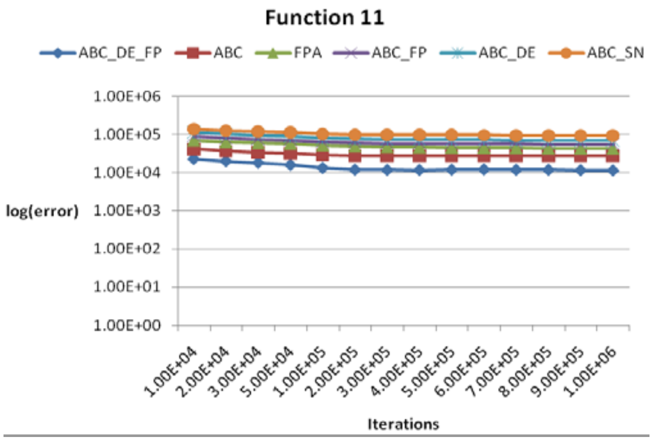


Figure 11 RLD for F11 (see online version for colours)



4.3.2.3 *Hybrid functions*

Hybrid functions are composed of various subcomponents, each consisting of different basic functions including unimodal or multimodal functions. Run length distribution graphs depicting convergence rate of various algorithms along with proposed algorithm is shown in Figures 17–22. On functions F17 and F21, all algorithms show similar performance at initial stage. After 3.00E+04 iterations, ABC_DE_FP reduces the error value over the iterations while other algorithms remain stagnates and show no improvement in performance. Hence ABC_DE_FP depicts better convergence speed. This is due to the improvement in exploitation process of proposed algorithm. For F18 and F19 functions, ABC_DE_FP attains minimum error value and shows good rate of convergence. This improvement is because of integrating FPA into ABC algorithm. For this reason ABC_FP (our variant) coincides with ABC_DE_FP and hence exhibit approximately similar performance. In case of F20 and F22 functions, ABC_DE_FP depicts minimum error value though algorithms show approximately same convergence rate over the iterations.

Figure 12 RLD for F12 (see online version for colours)

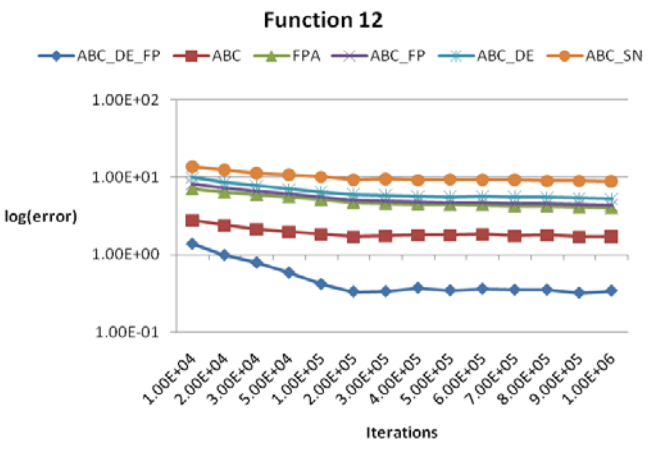


Figure 13 RLD for F13 (see online version for colours)

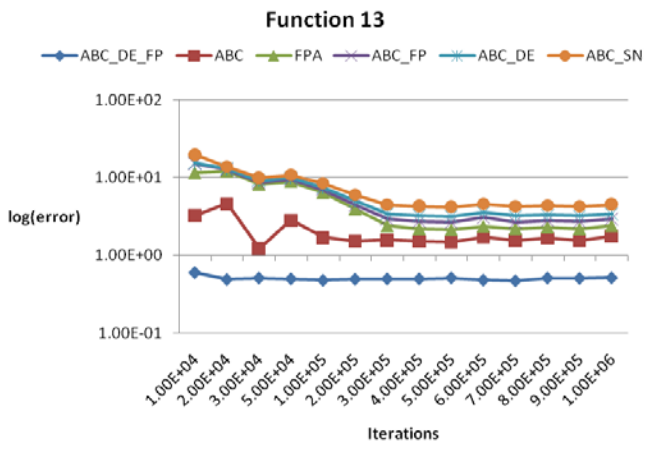


Figure 14 RLD for F14 (see online version for colours)

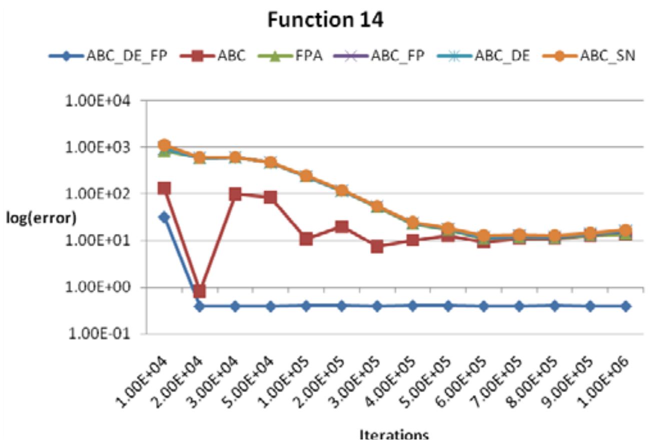


Figure 15 RLD for F15 (see online version for colours)

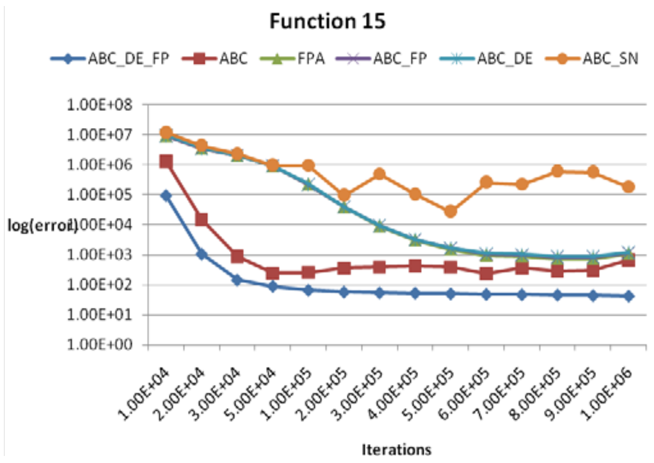


Figure 16 RLD for F16 (see online version for colours)

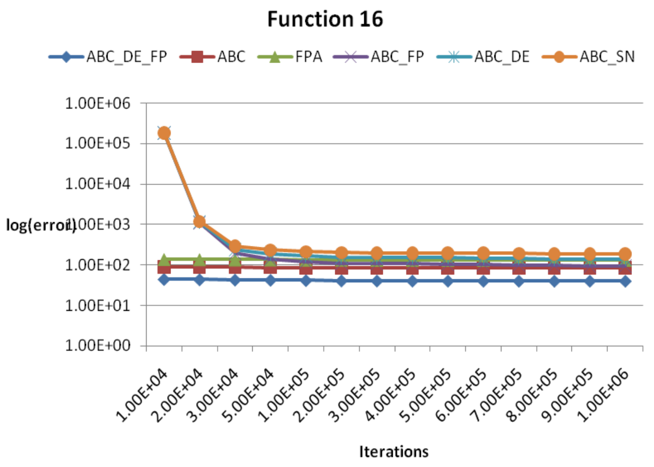
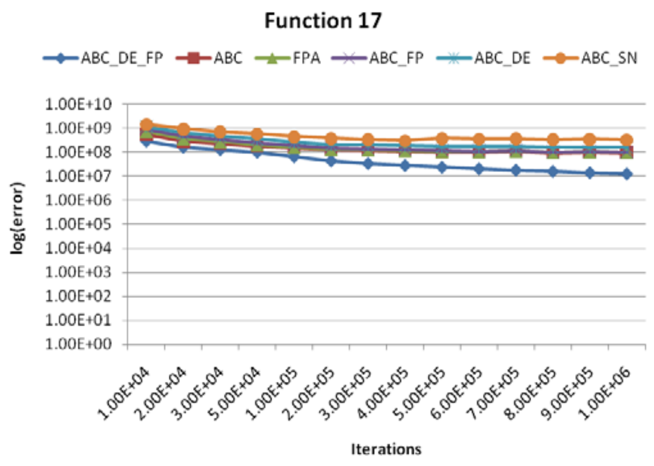


Figure 17 RLD for F17 (see online version for colours)



4.3.2.4 *Composition functions*

The composition functions defined in problem definition of CEC 2014 (Liang et al., 2013) is composed of various sub components. ABC_DE_FP shows minimum error value for functions F23, F24, F25, F26, F27 and F28. The algorithm's convergence rate becomes constant after achieving global minima in approximately 1.00E+05 iterations. The functions are so complex that algorithm's ability of exploring the complete search space deteriorates. On functions F29 and F30, ABC and ABC_DE_FP improves its global minima over the increasing iterations. The performance curve is near to rectangular hyperbola. However, the efficacy of ABC_DE_FP is better as compared to other algorithms.

Figure 18 RLD for F18 (see online version for colours)

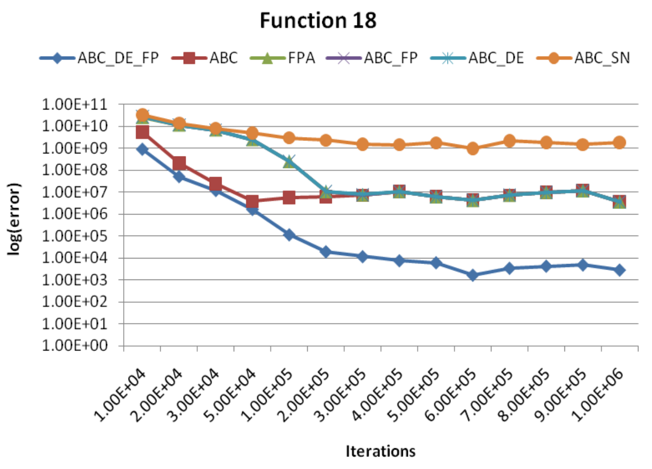


Figure 19 RLD for F19 (see online version for colours)

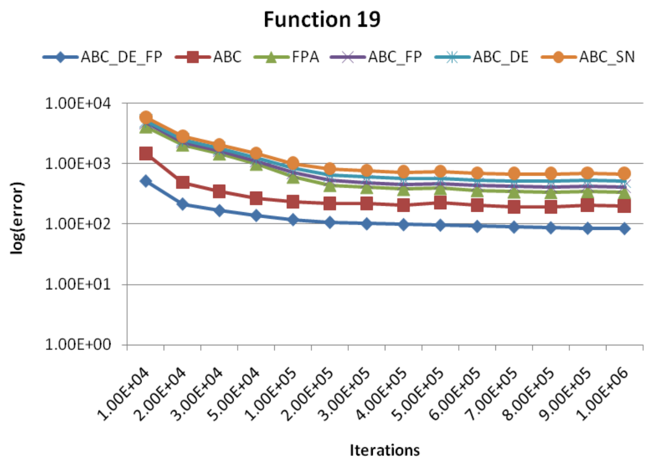
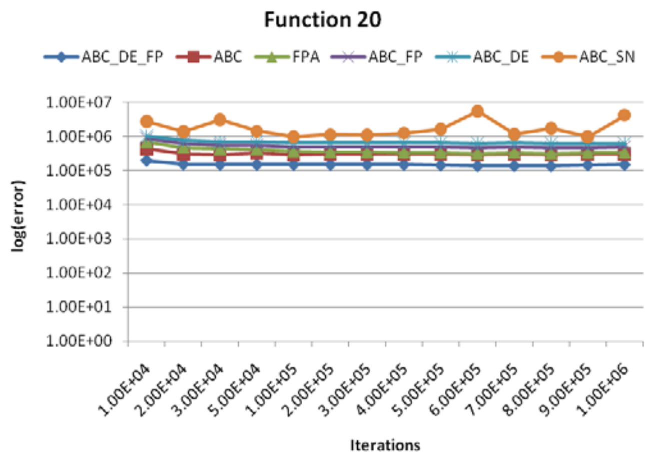


Figure 20 RLD for F20 (see online version for colours)



4.4 Discussion

The studies were performed to assess the efficacy of ABC_DE_FP for simple as well as complex optimisation problems. Results substantiate that for simple problems, proposed algorithm either performs better or similar to the best existing ABC variant. On an average though mean value of ABC_DE_FP lags by 1% but its best value improves by 49%. Analysis over complex problems of CEC2014 establishes that ABC_DE_FP performs significantly better than ABC, FPA, ABC_FPA, ABC_DE and ABC_SN. The statistical significance of the results was established through Wilcoxon rank sum test. Run length distribution graphs portrays that the convergence speed of proposed algorithm in attaining global optima is faster than existing algorithms. Thus, from these studies it can be inferred that, proposed algorithm is more suitable to composite and hybrid (according to CEC2014) type of problems where global minimum value is difficult to achieve.

Figure 21 RLD for F21 (see online version for colours)

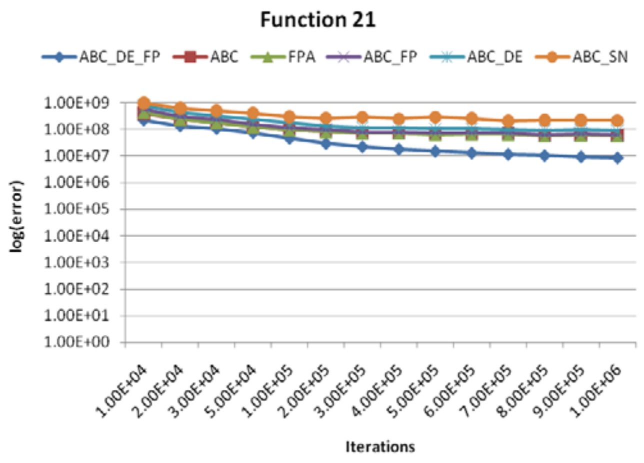


Figure 22 RLD for F22 (see online version for colours)

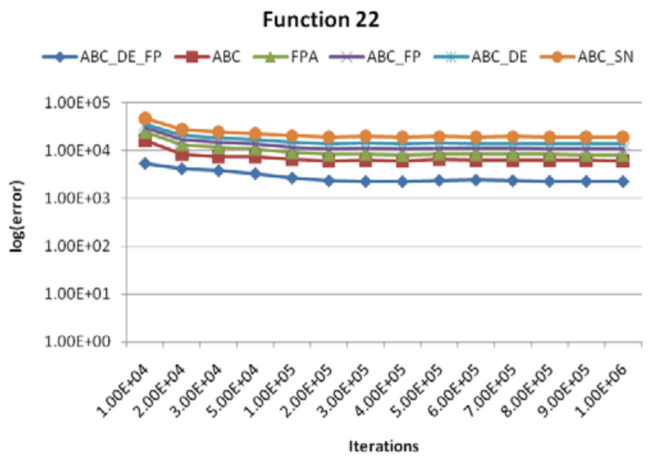


Figure 23 RLD for F23 (see online version for colours)

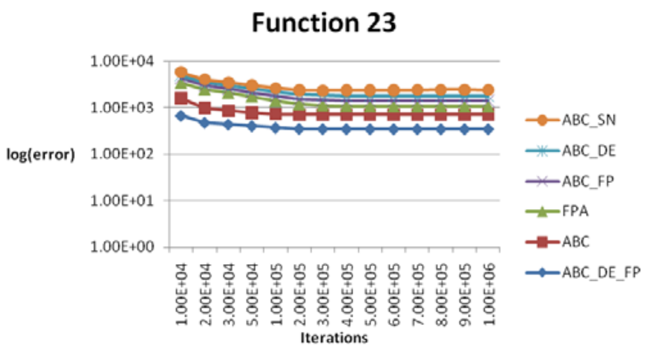


Figure 24 RLD for F24 (see online version for colours)

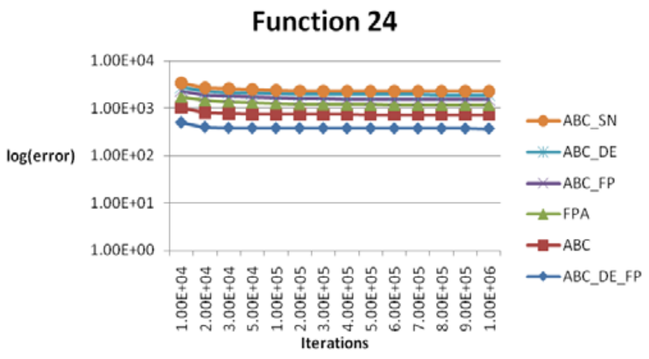


Figure 25 RLD for F25 (see online version for colours)

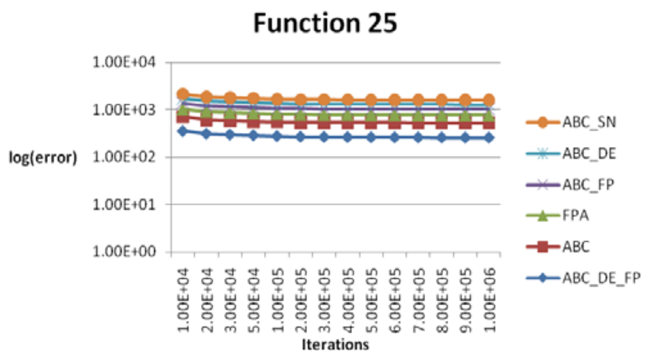


Figure 26 RLD for F26 (see online version for colours)

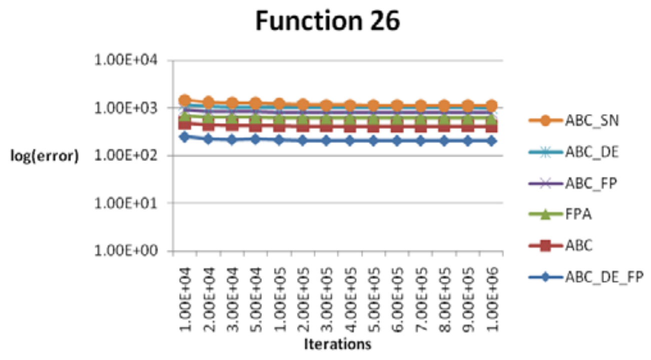


Figure 27 RLD for F27 (see online version for colours)

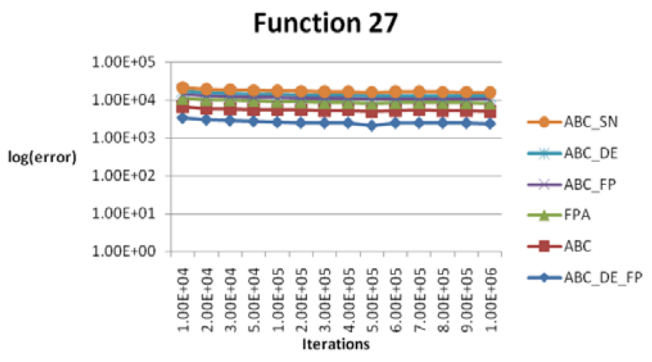


Figure 28 RLD for F28 (see online version for colours)

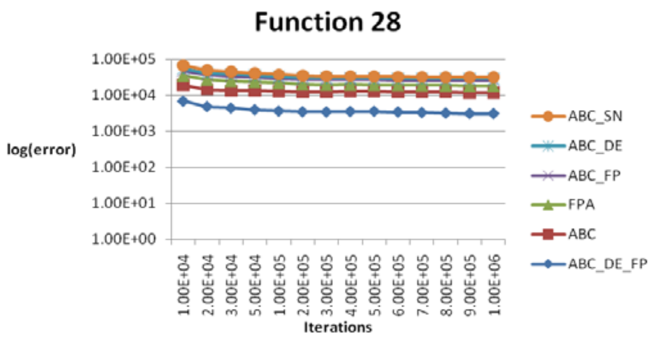


Figure 29 RLD for F29 (see online version for colours)

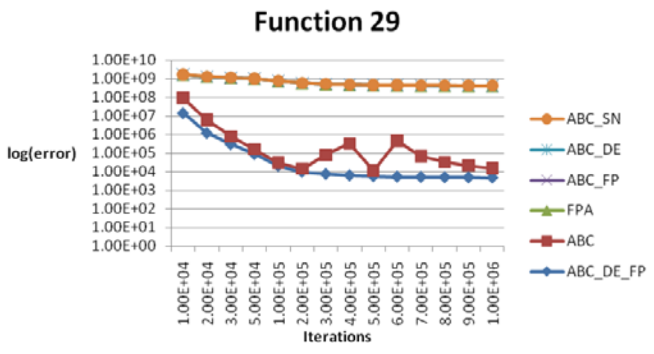
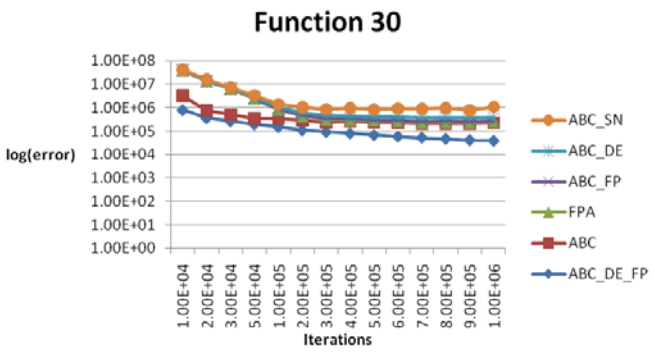


Figure 30 RLD for F30 (see online version for colours)



5 Conclusions

The paper presented an improved version of artificial bee colony algorithm (ABC_DE_FP) in terms of exploitation and convergence speed. The proposed algorithm was combination of ABC algorithm with FPA and DE. In order to exhibit the efficacy of proposed algorithm, it was tested on simple benchmark function against contemporary ABC variants. The algorithm provides improvement in terms of best value attained. Another experiment was conducted on four sets of benchmark function obtained from CEC2014 problem definition at 10, 30, 50 and 100 dimensions. The proposed algorithm was tested against ABC, FPA and various ABC variants. Wilcoxon rank sum test established the statistical significance of the results. Convergence rate of algorithms were analysed via run length distribution graphs. The results illustrated that performance of ABC_DE_FP is effectively and competitively better though its convergence behaviour is dependent on function's property. Overall, ABC_DE_FP algorithm was more suitable for complex optimisation problems.

The work motivates researchers to apply real world applications on proposed algorithm. In order to improve solution quality on composition functions, exploration step of hybrid algorithm can be enhanced further. Problem definitions of CEC 2016 were mostly based on multi objective optimisation. In future our next objective is to test the efficacy of proposed algorithm on CEC 2016 problem suit.

References

Abdel-Baset, M. and Hezam, I.M. (2016) 'A hybrid flower pollination algorithm for solving ill-conditioned set of equations', *International Journal of Bio-Inspired Computation*, Vol. 8, No. 4, pp.215–220.

Agarwal, P. and Mehta, S. (2014) 'Nature-inspired algorithms: state-of-art, problems and prospects', *International Journal of Computer Applications*, Vol. 100, No. 14, pp.14–21.

Agarwal, P. and Mehta, S. (2015) 'Comparative analysis of nature inspired algorithms on data clustering', *IEEE International Conference on Research in Computational Intelligence and Communication Networks* (*ICRCICN*), pp.119–124.

Agarwal, P. and Mehta, S. (2016) 'Enhanced flower pollination algorithm on data clustering', *International Journals of Computers and Applications*, August, Vol. 7074, Nos. 2–3, pp.144–155, Taylor and Francis.

Agarwal, P. and Mehta, S. (2017) 'Empirical analysis of five nature-inspired algorithms on real parameter optimization problems', *Artificial Intelligence Review*, pp.1–57.

Akay, B. and Karaboga, D. (2012) 'A modified artificial bee colony algorithm for real-parameter optimization', *Information Sciences*, Vol. 192, pp.120–142.

Banati, H. and Mehta, S. (2012) 'SEVO: bio-inspired analytical tool for uni-modal and multimodal optimization', *Advances in Intelligent and Soft Computing*, *130 AISC*, Vol. 1, pp.557–566.

Chakraborty, D., Saha, S. and Dutta, O. (2014) 'DE-FP A: a hybrid differential evolution-flower pollination algorithm for function minimization', *International Conference on High Performance Computing and Applications (ICHPCA)*, IEEE, pp.1–6.

Chen, S-M., Sarosh, A. and Dong, Y-F. (2012) 'Simulated annealing based artificial bee colony algorithm for global numerical optimization', *Applied Mathematics and Computation*, Vol. 219, No. 8, pp.3575–3589.

Duan, H. and Luo, Q. (2015) 'New progresses in swarm intelligence-based computation', *International Journal of Bio-Inspired Computation*, Vol. 7, No. 1, pp.26–35.

Gao, W. and Liu, S. (2011) 'Improved artificial bee colony algorithm for global optimization', *Information Processing Letters*, Vol. 111, No. 17, pp.871–882.

Gao, W. et al. (2015) 'Artificial bee colony algorithm with multiple search strategies', *Applied Mathematics and Computation*, Vol. 271, pp.269–287.

Hati, A.N. et al. (2013) 'Modified artificial bee colony algorithm using differential evolution and polynomial mutation for real-parameter optimization', in *2013 International Conference on Advances in Computing*, *Communications and Informatics* (*ICACCI*), pp.534–539.

Karaboga, D. and Akay, B. (2009) 'A comparative study of artificial bee colony algorithm', *Applied Mathematics and Computation*, Vol. 214, No. 1, pp.108–132.

Karaboga, D. and Basturk, B. (2008) 'On the performance of artificial bee colony (ABC) algorithm', *Applied Soft Computing Journal*, Vol. 8, No. 1, pp.687–697.

Liang, J.J., Qu, B.Y. and Suganthan, P.N. (2013) *Problem Definitions and Evaluation Criteria for the CEC 2014 Special Session and Competition on Single Objective Real-Parameter Numerical Optimization*, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore.

Mehta, S. and Banati, H. (2014) 'Context aware filtering using social behavior of frogs', *Swarm and Evolutionary Computation*, Vol. 17, pp.25–36.

Mehta, S. (2016) 'Memetic algorithm with constrained local search for large-scale global optimization', *Journal of Intelligent Systems*, Vol. 26, No. 2, pp.287–300

Shan, H., Yasuda, T. and Ohkura, K. (2015) 'A self adaptive hybrid enhanced artificial bee colony algorithm for continuous optimization problems', *Bio Systems*, Vol. 132, pp.43–53.

Xiang, W., Ma, S. and An, M. (2014) 'HABCDE: a hybrid evolutionary algorithm based on artificial bee colony algorithm and differential evolution', *Applied Mathematics and Computation*, Vol. 238, pp.370–386.

Yang, X-S.S. (2012) 'Flower pollination algorithm for global optimization', *Unconventional Computation and Natural Computation*, Vol. 7445, pp.240–249.

Yurtkuran, A. and Emel, E. (2015) 'An adaptive artificial bee colony algorithm for global optimization', *Applied Mathematics and Computation*, Vol. 271, pp.1004–1023.

Zhu, G. and Kwong, S. (2010) 'Gbest-guided artificial bee colony algorithm for numerical function optimization', *Applied Mathematics and Computation*, Vol. 217, No. 7, pp.3166–3173.