CHAPTER 6

A NOVEL HYBRID NATURE INSPIRED ALGORITHM FOR COMPLEX OPTIMIZATION PROBLEMS

It was observed from previous chapter that ABC and FPA is the best performer on majority of complex optimization function. However, it suffers from slow convergence rate and exploration capability. In order to maintain synchronization between exploration and exploitation mechanism for achieving global optima, this chapter presents a novel hybrid nature inspired algorithm. This algorithm is named ABC_DE_FP which combines evolutionary mechanisms of artificial bee colony algorithm, differential evolution and flower pollination algorithm. The proposed algorithm is assessed on thirty single objective benchmark functions of CEC2014. ABC_DE_FP is validated against state of art ABC variants and experimental results were analyzed on the basis of Wilcoxon rank sum test and convergence graphs. Analysis revealed that ABC_DE_FP outperform other contemporary existing algorithms on majority of the functions and maintains balance between exploration and exploitation.

6.1. INTRODUCTION

The in-depth fair comparative analysis of five existing contemporary nature inspired algorithms made in the previous chapter revealed that artificial bee colony algorithm (ABC) is the best algorithm on high dimension complex benchmark functions. Flower pollination algorithm attains next position after ABC algorithm. ABC is developed by Karaboga [145] and is an efficient nature inspired algorithm. It outperform genetic algorithm (GA), differential evolution (DE), particle swarm optimization (PSO) and evolution strategies (ES) on benchmark functions [146]. Karaboga and Basturk [131] also proved the efficacy of ABC algorithm on various numerical optimization functions against PSO, DE and Evolutionary Algorithm (EA). A survey on advances in ABC with its application is presented in [147]. Empirical analysis of five nature inspired algorithms on CEC 2014 benchmark functions is illustrated in [148]. It has been observed that ABC gives better solution as compared to other algorithms. Scientist and researchers are attracted towards ABC due to its excellent performance and simpler implementation. It has been applied for optimizing constraint based problems [149], training neural network [150], clustering problems [84], leaf-constrained minimum spanning tree [151]. ABC algorithm is also employed in enhancement of image contrast [152] and multilevel threshold on iris image[153]. Discrete ABC is applied in lot-streaming flow shop scheduling problem [39], solving travelling salesman problem [154] and many more.

In spite of having good efficiency and easier implementation, ABC suffers from few drawbacks. As ABC algorithm is meta-heuristic and stochastic in nature, it takes long time to get best results. Hence, convergence speed of ABC needs improvement. The algorithm should be restricted 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 [86]. Exploration, also termed as diversification, is a technique to discover promising solution on wide search space to find global optima. Exploitation, also termed as intensification, is a technique to find promising solution in neighborhood of existing solution where global optima may occur. These two techniques contradict each other when put into practice [88]. Hence, a good optimization algorithm maintains balance between these two techniques. This encouraged the researchers to develop modified versions of artificial bee colony algorithm. The modified versions are developed by researchers by either tuning or adding some parameters in algorithm or by hybridizing 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 [86] introduced a new version of ABC algorithm called guided ABC algorithm (GABC) 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 [81] presented a modification in typical ABC algorithm and tested it against real parameter optimization problems. The changes in update equation of original

ABC algorithm are made by adding new parameters. The new parameters include modification rate (for regulating the frequency of parameter change) and scaling factor (for discovering value of parameter change in update equation). Hati et al. [79] proposed a new version of ABC algorithm by applying mutation strategy of differential evolution (DE) in employed and onlooker bee phase and polynomial mutation in scout bee phase. The new version of ABC algorithm is examined against real parameter optimization problems. Inspired by GABC, Xiang [77] developed a hybrid approach of modified ABC and modified DE algorithm in order to get better solution and convergence speed. Additionally, a new population varying scheme has been introduced to sustain balance among exploration and exploitation. The hybrid algorithm is compared against original ABC, DE and few DE variants on 20 benchmark functions.

A novel approach of introducing simulated annealing (SA) in ABC algorithm (ABC_SN) is given by Chen et al. [82]. In order to improve exploitation technique of ABC algorithm, a local search algorithm called simulated annealing is incorporated into employed bee phase. The new algorithm is evaluated against GABC and ABC on six numerical optimization functions. Yurtkuran and Emel [75] presented an adaptive version of ABC algorithm (AABC) by introducing various search strategies in different phases of algorithm for enhancing exploration and exploitation capabilities. Probabilistic selection is applied for selecting a search rule (technique), and further the rule is updated by applying roulette wheel technique based on previous solution. AABC is scrutinized on few well known real parameter numerical optimization functions against various ABC variants. An ABC algorithm with multiple search approaches is introduced by Gao et al. [76]. Adaptive selection method is used for choosing a particular search strategy. In order to boost exploitation ability of an algorithm, Gaussian distribution is applied. The proposed algorithm depicted better performance against various ABC variants and selected state-of-art algorithms over 22 numerical functions of optimization. Shan et al. [155] offered a new hybrid approach of ABC algorithm composed of Lévy flight distribution in standard ABC algorithm. To prove the efficacy of new algorithm, experiments were carried out on IEEE Congress on Evolutionary Computation 2013 benchmark functions and compared against original ABC, PSO and DE algorithms.

Flower pollination algorithm (FPA) is a global optimization population based algorithm developed by Xin-She Yang [47] in 2012. FPA has recently been used in some applications and few variants have been proposed by researchers. FPA is used to solve multi-objective optimization problems [113]. It has been also successfully applied in optimizing disc brake design problem [113]. Nigdeli et al. [156] also used FPA for designing structural engineering problems. A review on FPA along with its application in linear as well as non linear applications is presented by Chiroma et al. [105]. It has been also used in optimizing various clustering datasets when combined with K-means clustering algorithm [115] and optimal reactive power dispatch problem [157]. Agarwal and Mehta [158] introduced an enhanced version of flower pollination algorithm and tested over clustering datasets of different domains. The algorithm outperformed in terms of solution quality and convergence speed. A hybrid flower pollination algorithm is developed to handle ill conditioned systems of linear and non linear equations [112]. Hezam et al. [73] introduced a hybrid algorithm of FPA with tabu search for solving unconstrained optimization problems. The analysis of results revealed that proposed hybrid algorithm gives better solutions and stability on ten benchmark functions. For globally optimizing constrained optimization problems, Raouf et al. [110] came up with a hybrid approach of FPA and PSO. The algorithm has been examined on seven test problems.

Although various techniques and number of variants have been proposed, there exists no algorithm that finds exact solution to all kinds of optimization problems. Rather there is no such algorithm that maintains perfect synchronization 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 chapter. The hybrid algorithm modifies standard ABC algorithm by incorporating flower pollination algorithm in onlooker bee phase and differential evolution in employed and onlooker bee phases. The algorithm is tested on simple and complex CEC 2014 [137] benchmark functions.

Subsequent sections are organized as follows: section 6.2 discusses standard optimization algorithms. Section 6.3 elaborates proposed algorithm. Experimental setup and results is shown in section 6.4. Section 6.5 is dedicated to the analysis of results and section 6.6 concludes the chapter.

6.2. OVERVIEW OF ABC, FPA AND DE ALGORITHMS

Description of ABC and FPA is defined in previous chapter of section 5.2.1 and 5.2.5 respectively.

Differential Evolution (DE) is a strong evolutionary algorithm developed by Storn and Price [46], [159] in 1997. Yang and Deb [160] improved the search efficiency of DE by combining it with two stage eagle strategy. It has been successfully applied in various application domains. Coletta et. al. used DE to optimize the combination of classifier and cluster ensembles[161]. DE has excellent convergence speed and has also been integrated with various algorithms to obtain better results [162], [163]. There are number of schemes developed for executing algorithm [46], but DE/rand/1 is the most widely applied scheme and same has been used in this work. DE algorithm uses three phases in sequence: mutation, crossover and selection. Like other evolutionary algorithm DE also initializes the candidate solution in population randomly and then enters into following three phases:

a. Mutation

In this phase, mutant vector v_i is computed using each target vector x_i (candidate solution) as shown in Equation 6.1.

$$v_i = x_a + F.(x_b - x_c) \tag{6.1}$$

Where $i=1,2,\ldots FS$ (FS is the number of food sources), 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].

b. Crossover

A new vector u_{ij} named trial vector is generated from mutant vector v_{ij} and target vector x_{ij} as given in Equation 6.2.

$$u_{ij} = \begin{cases} v_{ij}, & if\ rand[0,1] \leq Cr\ or\ j == j_{rand} \\ x_{ij}, & otherwise \end{cases} \tag{6.2}$$

Where $j=1,\ 2\ldots.D$ 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.

c. Selection

For optimization problem, target vector for next generation is selected by applying greedy selection between trial vector and target vector on the basis of fitness value of objective function. In case of minimization type of problem, the vector is selected according to the lower fitness value of trial vector or target vector of corresponding population index.

6.3. HYBRID ABC_DE_FP ALGORITHM

In order to maintain balance between exploration and exploitation in ABC algorithm, a new hybrid nature inspired algorithm named ABC_DE_FP is developed. The algorithm amalgamates FPA and DE evolutionary operators in ABC algorithm. In ABC_DE_FP, ABC algorithm is the base algorithm. Initially scout bees search for random food sources. In employed bee phase, food sources are exploited by evolutionary strategies of DE algorithm. While in onlooker bee phase, exploration is performed through global pollination of FPA while exploitation is done through evolutionary strategies of DE algorithm. The onlooker bees act as pollinators in flower pollination and pollination is global in nature as shown in figure 6.1. Food source in artificial bee colony algorithm is flower. Onlooker bee determines the flower with maximum nectar amount. Hence objective of hybrid algorithm is same as that of ABC. 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 process of an algorithm.

As shown in Figure 6.1, in hybrid ABC algorithm, food sources in the population are initialized randomly within the bounds already defined in objective function. In the employed bee phase, original ABC uses Equation 6.3 to search for local solution in the neighborhood of existing candidate solution.

$$X_{ij}(\text{new}) = X_{ij} + \Phi_{ij}(X_{ij} - X_{kj}) \tag{6.3}$$

Where X_{ij} is the i^{th} solution/food source of population at j^{th} index, k is randomly chosen solution from present population other than i. Φ_{ij} is a uniformly distributed random real number in the range [-1, 1].

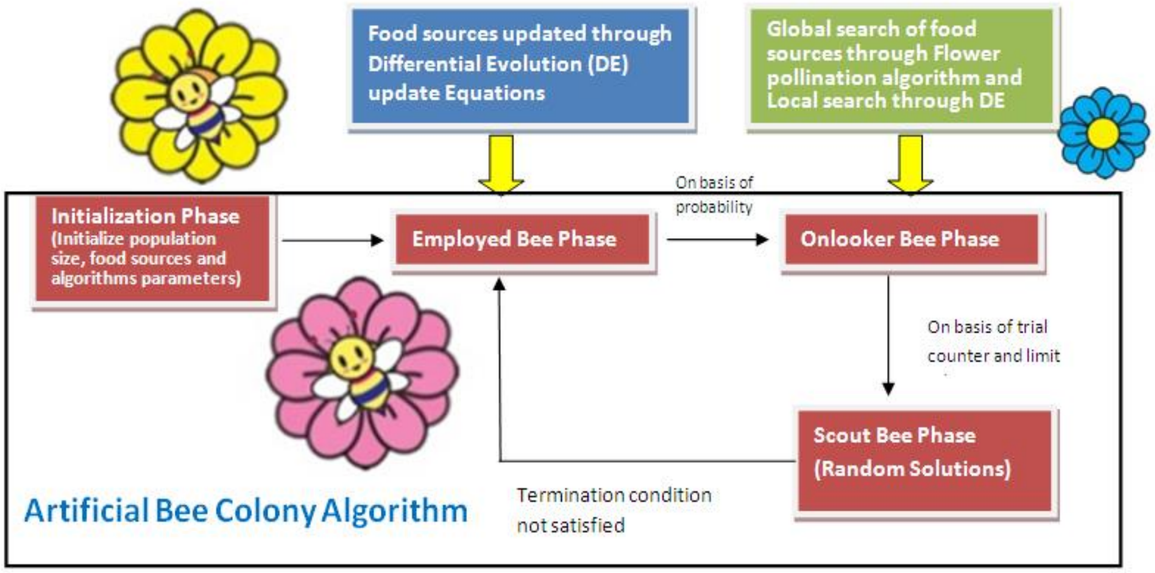

Food sources updated through Differential Evolution (DE) update Equations
Global search of food sources through Flower pollination algorithm and Local search through DE
Initialization Phase (Initialize population size, food sources and algorithms parameters)
Employed Bee Phase
On basis of probability
Onlooker Bee Phase
On basis of trial counter and limit
Scout Bee Phase (Random Solutions)
Termination condition not satisfied
Artificial Bee Colony Algorithm

Figure 6.1: Pictorial representation of ABC_DE_FP algorithm

In literature this step of finding new solution cannot assure of achieving better results[82]. Hence, it slows down the convergence speed. Moreover, even 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 6.1) is inculcated in place of local search strategy (Equation 6.3) of employed and onlooker bee phase as shown in Figure 6.1. After finding mutant solution, trial solution is generated using crossover strategy of DE (defined in Equation 6.2). The probability of attaining better food source 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 synchronization between exploration and exploitation, global pollination mechanism of FPA is incorporated into onlooker bee phase (shown in Equation 6.4).

$$x_i^{t+1} = x_i^t + \mathrm{L}\,(x_i^t - g_*) \tag{6.4}$$

Where x_i^t is *i*th solution of *t*th generation, g_* is the current best solution at present generation and *L* represents Lévy flight distribution [47] [138] portraying potency of pollination. Onlooker bee phase plays an important role in exploration as well as exploitation in original ABC algorithm. The steps of FPA integrated into onlooker bee phase of ABC algorithm is shown in Algorithm 6.1.

Algorithm 6.1: Steps of FPA in ABC_DE_FP

```
1.  if rand>p
2.          L is the step size determined by Lévy distribution
3.          Perform global pollination (global search) using Equation 6.4
4.      else
5.          Perform exploitation (local search) with mutation strategy of DE using Equation 6.1
6.  end if
```

ABC_DE_FP algorithm is depicted through flowchart shown in Figure 6.2 and Algorithm 6.2. 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 Lévy distribution (Equation 6.4). This helps in modeling the direction of search. Due to switch probability *p*, either local or global mutant solution is obtained. Thereafter, same steps of crossover and greedy selection are applied as defined in employed bee phase. 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. Scout bee finds the new flowers (solution) randomly using Equation 6.5.

$$X_{ij} = lb_j + rand(0,1)\left(UB_j - LB_j\right) \tag{6.5}$$

Where *i=1… FS* and *j=1….D*, where *FS* is the number of food sources and *D* is dimension parameter of optimization problem. *UB* and *LB* are upper and lower bound already defined for a given problem. The cycle of three phases continues unless predefined numbers of fitness function evaluations are achieved. Input parameters of algorithm are: maximum fitness function evaluations (*MaxFES*), food source (*FS*), dimension of FS (*D*), Limit, switch probability (*p*), crossover rate (*Cr*) and scale factor (*F*).

Algorithm 6.2: ABC_DE_FP

```
1.  Initialize the given parameters MaxFes, D, FS, p, F and Cr.
2.  Determine the initial target solution randomly within LB and UB using Equation 6.5.
3.  Compute fitness from initialized food source using Equation 6.5.
4.  For N_iter =1: MaxFES
5.      For each food source(FS)
6.          Compute the mutant solution using mutant strategy of DE as given in Equation 6.1.
7.          Compute the trial solution using crossover strategy of DE using Equation 6.2.
8.          Determine the fitness of trial solution.
9.          Apply greedy approach to find best solution using fitness of trial and previous target 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.
13.     For each ‘i’ food source(FS)
14.         if (rand<pr(i))
15.             if rand<p
16.                 Apply global pollination using Equation 6.4
17.             else
18.                 Apply mutant strategy of DE using Equation 6.1.
19.             Compute the trail solution using crossover strategy of DE using Equation 6.2
20.         Determine the fitness of trail solution.
21.         Apply greedy approach to find best solution using fitness of trail and previous target solution
22.         if solution improves,
23.             Update the population with better food source else increment counter of scout by 1.
24.         End if
25.     End for loop
26.     Memorize the global minima from updated population
27.     If scout counter exceeds limit
28.         Determine the new solution i.e. explore new food source using Equation 6.5
29.         Compute the fitness of explored solution
30.     End if
31.     The algorithm again enters employed bee phase until termination condition is achieved.
32.     End while
```

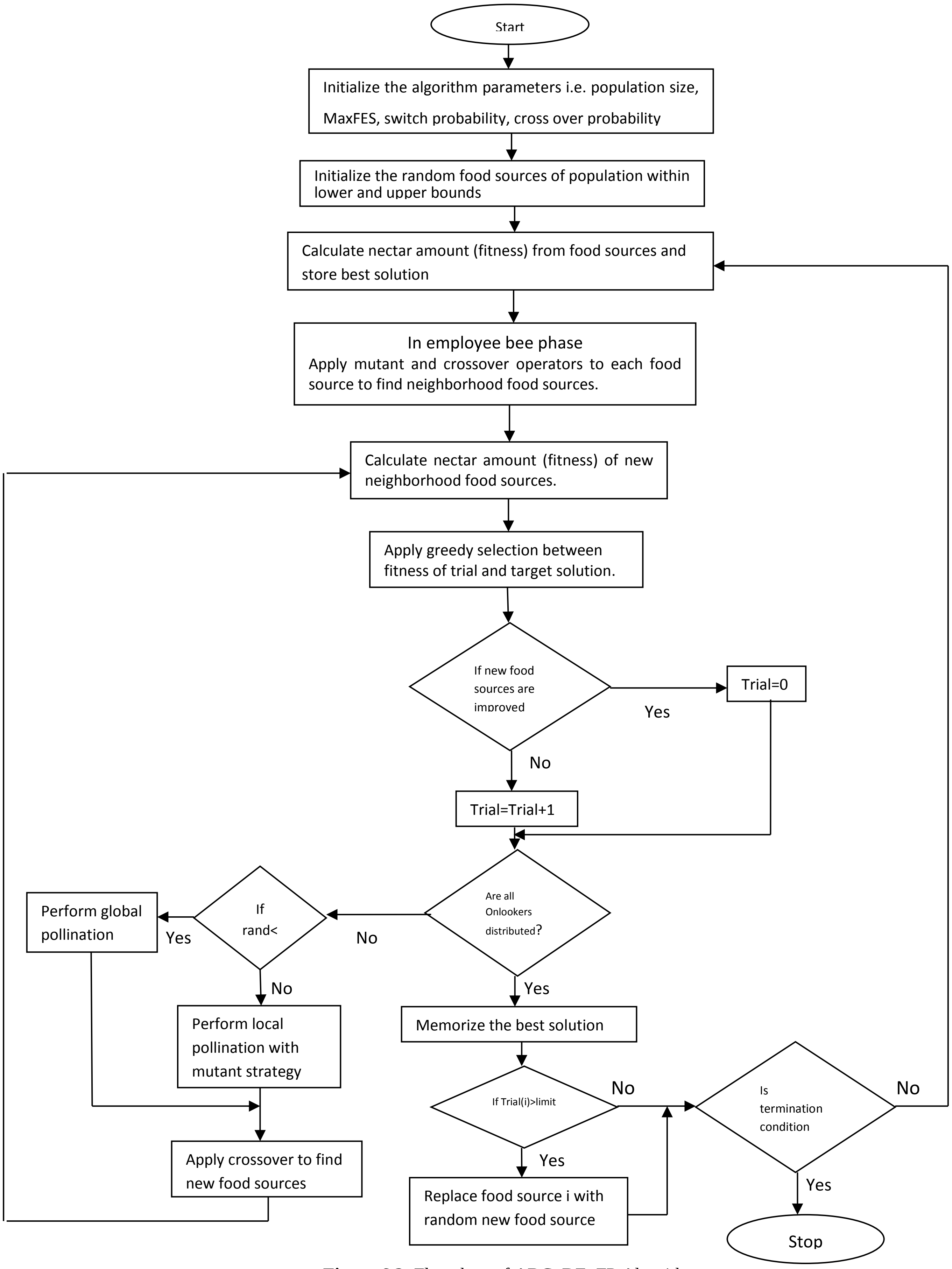

Start
Initialize the algorithm parameters i.e. population size, MaxFES, switch probability, cross over probability
Initialize the random food sources of population within lower and upper bounds
Calculate nectar amount (fitness) from food sources and store best solution
In employee bee phase
Apply mutant and crossover operators to each food source to find neighborhood food sources.
Calculate nectar amount (fitness) of new neighborhood food sources.
Apply greedy selection between fitness of trial and target solution.
If new food sources are improved
Yes
Trial=0
No
Trial=Trial+1
Are all Onlookers distributed?
No
If rand<
Yes
Perform global pollination
No
Perform local pollination with mutant strategy
Apply crossover to find new food sources
Yes
Memorize the best solution
If Trial(i)>limit
No
Yes
Replace food source i with random new food source
Is termination condition
No
Yes
Stop

Figure 6.2: Flowchart of ABC_DE_FP Algorithm

6.4. EXPERIMENTAL STUDY AND RESULTS

6.4.1. Experimental Setup

The efficacy of proposed hybrid algorithm is initially tested on simple benchmark functions [75] and then on complex and hybrid benchmark functions described by CEC2014 [137]. 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 behaviors. List of parameter along with the values used in implementation are shown in Table 6.1. In ABC algorithm and its variants, the value of *limit* [146] is defined on basis of food source (FS) and dimension (D) given below in Equation 6.6.

$$Limit = FS * D \tag{6.6}$$

Population size is taken as recommend in [76], [164]. The value of switch probability p is suggested in [78]. Moreover, value of Cr [78] is analyzed over 0.2 to 0.9 and the best value is found to be 0.7. Scaling factor F is taken as mutation ratio and is generated randomly. It is uniformly distributed array of number between 0.2 and 0.8. ABC algorithm parameter *limit* are recommended in [79]. The best values of *alpha, beta* and *T* of simulated annealing are defined in [82].

Table 6.1: 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

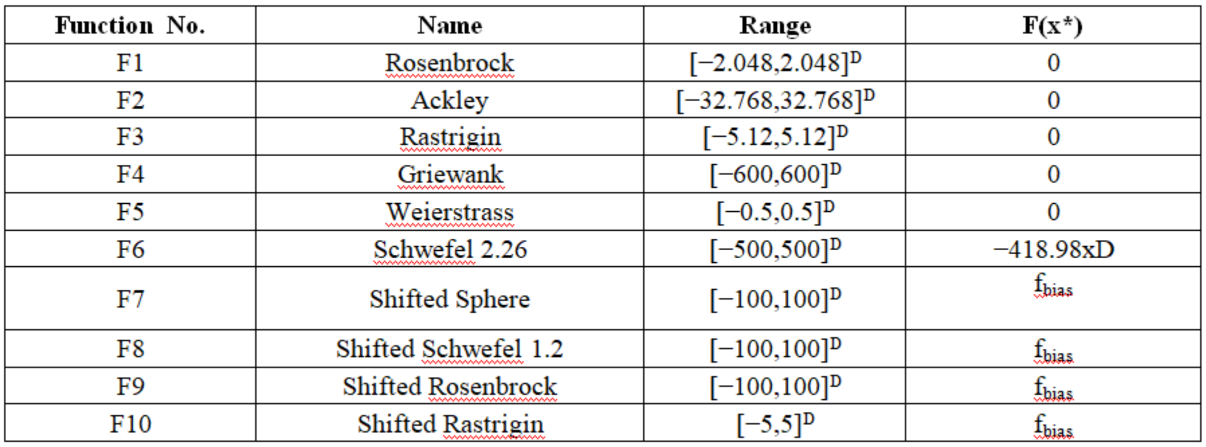
Meticulous experiments are performed to evaluate the performance of presented algorithm. The efficacy of presented ABC_DE_FP is evaluated with respect to contemporary ABC variants on following functions:

- Simple benchmark test functions
- CEC2014 benchmark functions. Further performance is analyzed through:
    - Wilcoxon Rank Sum Test Analysis
    - Convergence Graphs

6.4.2. Comparison of ABC_DE_FP Algorithm with respect to ABC Variants on Simple Test Functions

In this experiment presented hybrid algorithm is compared with other ABC variants i.e. guided ABC (GABC) [86], improved ABC (IABC) [165] and adaptive ABC (AABC) [75]. The formulation of test functions used in this experiment are shown in Table 6.2 [75].

Table 6.2: Simple Test Functions[75]



Function No.	Name	Range	F(x*)
F1	Rosenbrock	$[-2.048,2.048]^D$	0
F2	Ackley	$[-32.768,32.768]^D$	0
F3	Rastrigin	$[-5.12,5.12]^D$	0
F4	Griewank	$[-600,600]^D$	0
F5	Weierstrass	$[-0.5,0.5]^D$	0
F6	Schwefel 2.26	$[-500,500]^D$	−418.98xD
F7	Shifted Sphere	$[-100,100]^D$	f_{bias}
F8	Shifted Schwefel 1.2	$[-100,100]^D$	f_{bias}
F9	Shifted Rosenbrock	$[-100,100]^D$	f_{bias}
F10	Shifted Rastrigin	$[-5,5]^D$	f_{bias}

Hybrid ABC_DE_FP is executed on test functions (defined in Table 6.2) and compared with various ABC variants on same input parameter values given in [75]. Table 6.3 shows the best, mean, median, worst and standard deviation of 30 runs on 10 benchmark test functions.

Table 6.3: 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

Bold values represent the minimum (best) value attained by respective algorithm. It can be observed from Table 6.3 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 comparative analysis of presented algorithm is made against AABC. For F1 (Rosenbrock) function, best value of ABC_DE_FP improves by 63% though mean value lags by 35% 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 presented 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. On F4 (Griewank) function ABC_DE_FP does not give considerable results.

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), mean value improves by 5% and 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 F10 function, algorithms give same fitness value.

Overall, presented 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 optimization functions were simple; next experiment evaluates the efficiency of ABC_DE_FP on complex problems.

6.4.3. Performance Evaluation of ABC_DE_FP Algorithm on CEC2014 Benchmark Functions

In order to assess the efficacy of presented hybrid algorithm over complex optimization problems, it is compared with original artificial bee colony algorithm (ABC), flower pollination algorithm (FPA), hybrid of ABC and FPA (ABC_FP), ABC and DE (ABC_DE) [77] and ABC and simulated annealing (ABC_SN) [82]. ABC_FP algorithm has been developed by us to make comparison. The proposed hybrid ABC algorithm and its variants are evaluated on the 30 single objective complex optimization functions obtained from problem definition of CEC 2014 [137].

Maximum function evaluations (MaxFes) are treated as terminating criteria and taken as 10000*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 thereafter error is computed from bias of each function [137]. Performance is assessed on the 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 analyzed through convergence 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. 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, 30, 50 and 100 dimensions are shown in Table 6.4, Table 6.5, Table 6.6 and Table 6.7 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 optimal value of zero is 1.00E-02. Also, best mean error values are highlighted for each function.

Table 6.4: Mean error values of algorithms over 30 runs for 10D

	D=10					
Function No.	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 6.5: Mean error values of algorithms over 30 runs for 30D

	D=30					
Function No.	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
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 6.6: Mean error values of algorithms over 30 runs for 50D

	D=50					
Func.	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 6.7: Mean error values of algorithms over 30 runs for 100D

	D=100					
Func.	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

It can be observed from Table 6.4 and Table 6.5 that ABC_DE_FP give comparable performance to other algorithms on small dimension functions (10 and 30 dimensions). However at high dimensions such as 50 dimensions (Table 6.6), ABC_DE_FP gives better performance on majority of the functions. At 100 dimensions given in Table 6.7, presented algorithm outperforms all other algorithms. Hence, ABC_DE_FP efficacy enhances and is scalable to dimensions. These results are established through statistical analysis and convergence behavior of algorithms described in subsequent section.

6.5. ANALYSIS OF RESULTS

6.5.1. Wilcoxon Rank Sum Test Analysis

Mean error value of ABC_DE_FP algorithm is statistically compared with other algorithms through Wilcoxon rank sum test analysis. This statistical test analysis determines the significant difference in error values of presented versus contemporary algorithms. Wilcoxon rank sum test shown in Table 6.8 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 ABC performs better for 9 functions. For rest of the 4 functions, there is no significant difference.

Table 6.8: Wilcoxon Rank Sum Test Analysis

Vs. ABC_DE_FP		**100D**	**50D**	**30D**	**10D**
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

Similarly ABC_DE_FP performs better for 20 functions than FPA and worse for 7 functions only. Correspondingly all the results of Table 6.8 can be analyzed. It can be noticed that for the majority of functions, presented 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 the majority of benchmark functions.

6.5.2. Computational Complexity Analysis (Convergence Behavior)

Performance of presented 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 analyzed in subsections.

6.5.2.1. Unimodal Functions

Convergence graph representing the behavior of algorithms on unimodal functions is shown in Figure 6.3 to 6.5. It can be 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, presented algorithm reaches its minima at 2.00E+05 iterations. In F2 function, ABC_DE_FP exhibit rapid pace of convergence and obtain its global minima at 6.00E+05 iterations. ABC portrays non uniform convergence speed and comparable performance to ABC_DE_FP at 2.00E+05 iterations. This is due to the lack of suitable balance among exploration and exploitation techniques. In presented 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.

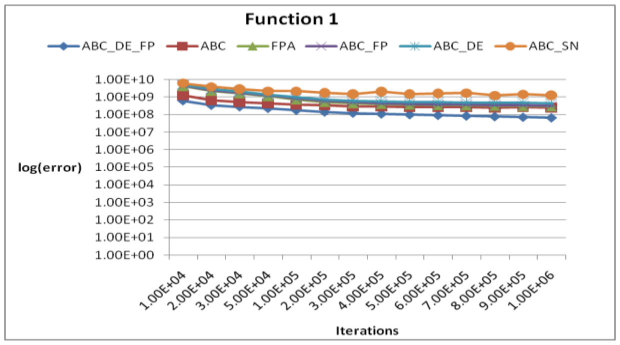


Figure 6.3: Convergence Graph of algorithms for F1

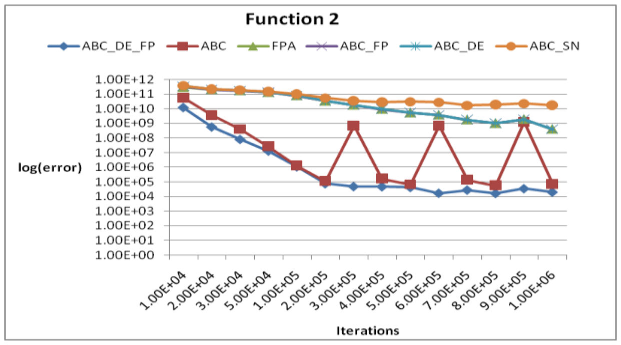


Figure 6.4: Convergence Graph of algorithms for F2

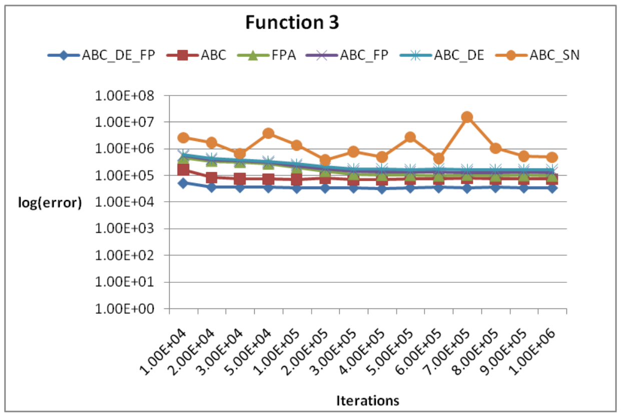


Figure 6.5: Convergence Graph of algorithms for F3

6.5.2.2. Simple Multimodal Functions

Convergence graphs of presented ABC_DE_FP with its competitive algorithms for simple multimodal functions are shown in Figure 6.6 to 6.18. 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 iterations 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 behavior over the iterations. This behavior 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.

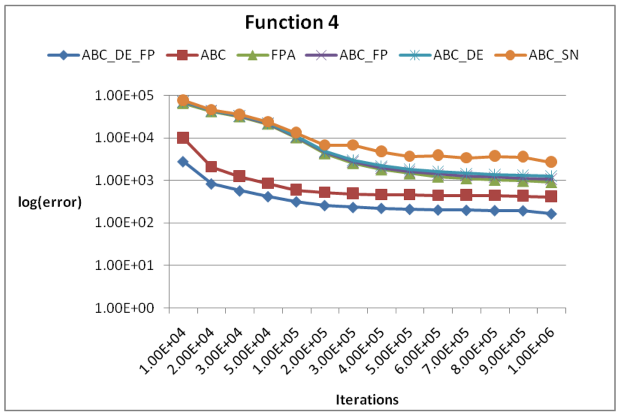


Figure 6.6: Convergence Graph of algorithms for F4

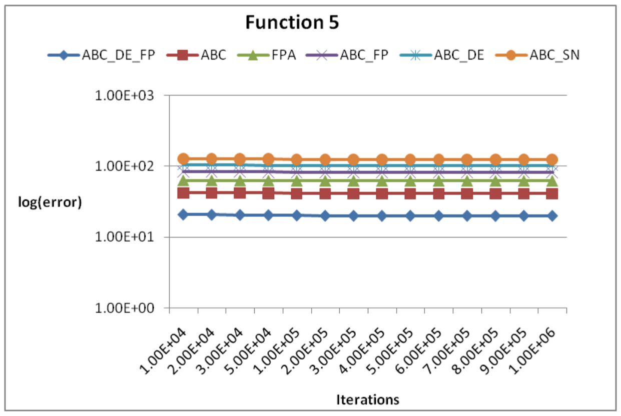


Figure 6.7: Convergence Graph of algorithms for F5

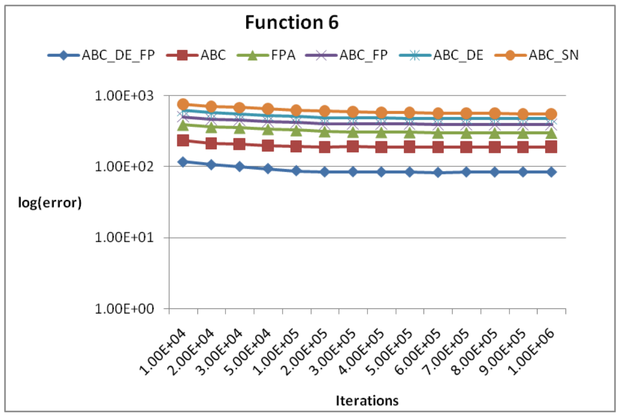


Figure 6.8: Convergence Graph of algorithms for F6

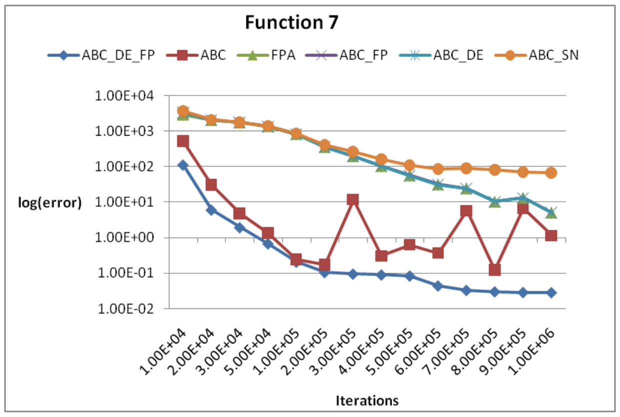


Figure 6.9: Convergence Graph of algorithms for F7

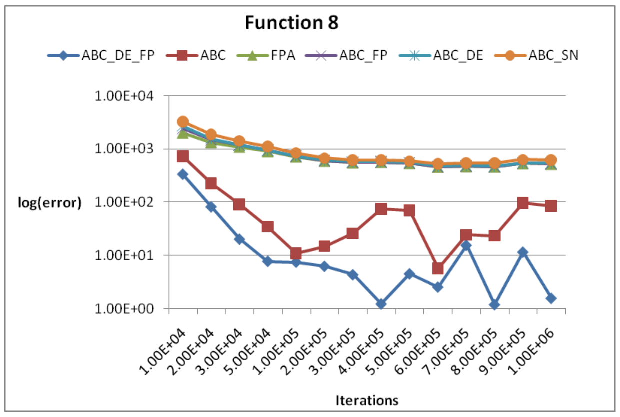


Figure 6.10: Convergence Graph of algorithms for F8

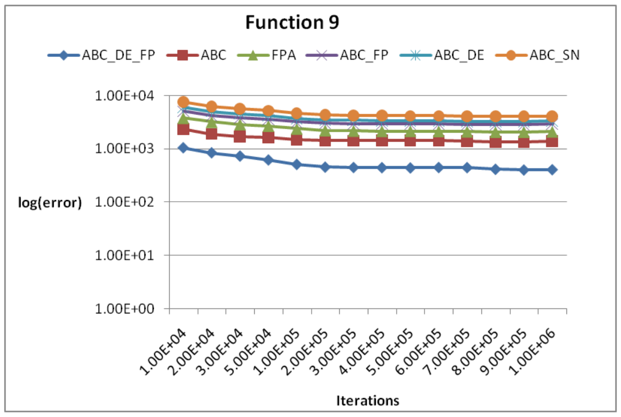


Figure 6.11: Convergence Graph of algorithms for F9

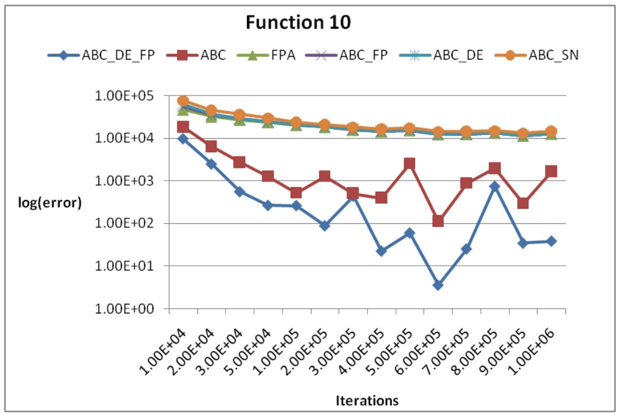


Figure 6.12: Convergence Graph of algorithms for F10

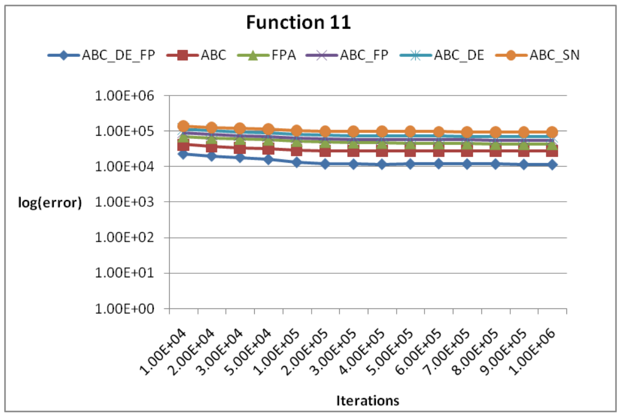


Figure 6.13: Convergence Graph of algorithms for F11

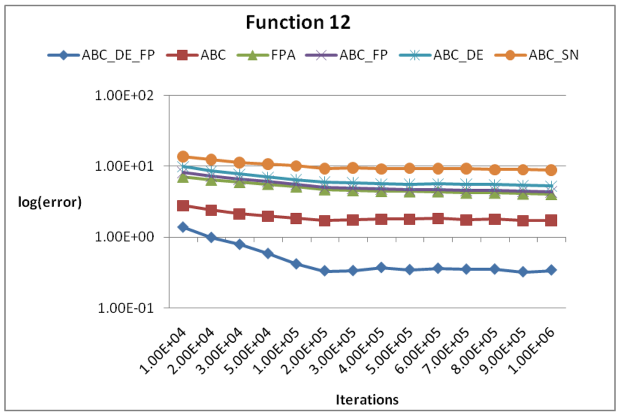


Figure 6.14: Convergence Graph of algorithms for F12

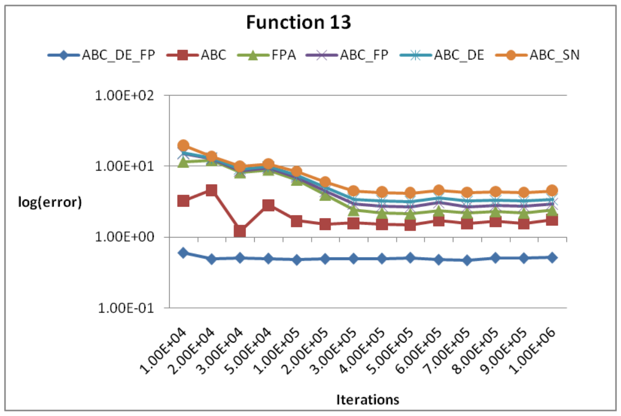


Figure 6.15: Convergence Graph of algorithms for F13

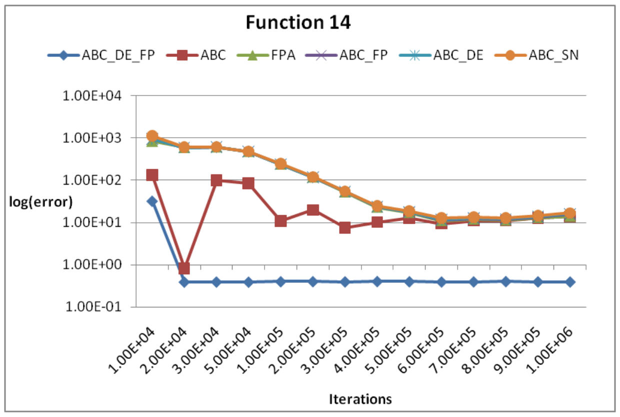


Figure 6.16: Convergence Graph of algorithms for F14

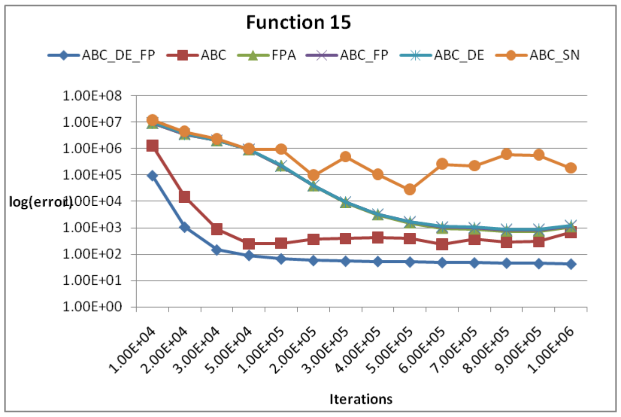


Figure 6.17: Convergence Graph of algorithms for F15

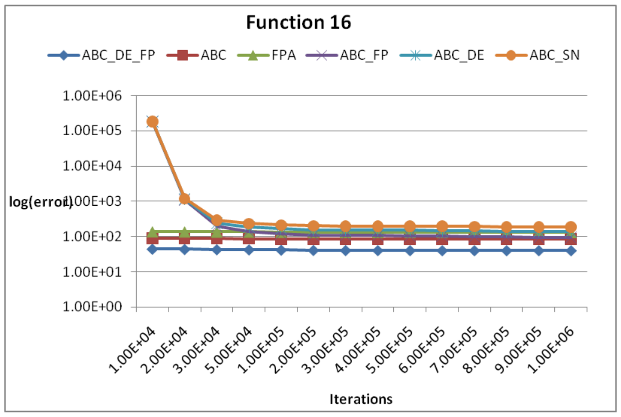


Figure 6.18: Convergence Graph of algorithms for F16

6.5.2.3. Hybrid Functions

Hybrid functions are composed of various subcomponents, each consisting of different basic functions including unimodal or multimodal functions. Convergence rate of various algorithms along with presented algorithm is shown in Figure 6.19 to 6.24. 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 stagnate and show no improvement in performance. Hence ABC_DE_FP depicts better convergence speed. This is due to the improvement in exploitation process of presented 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.

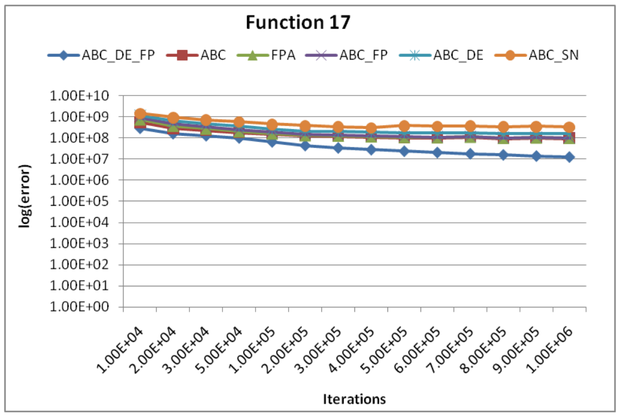


Figure 6.19: Convergence Graph of algorithms for F17

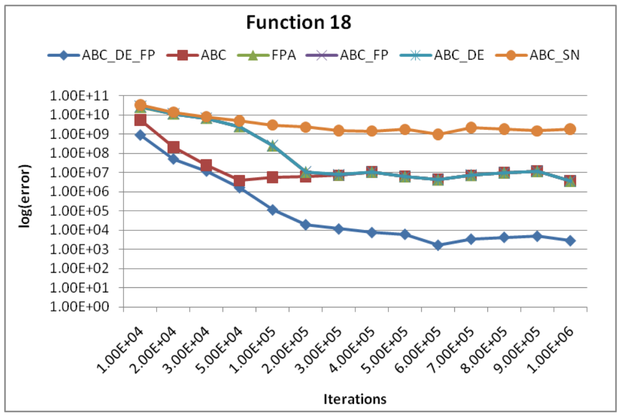


Figure 6.20: Convergence Graph of algorithms for F18

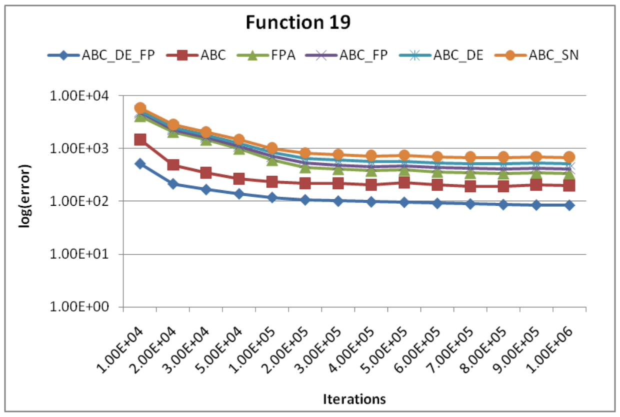


Figure 6.21: Convergence Graph of algorithms for F19

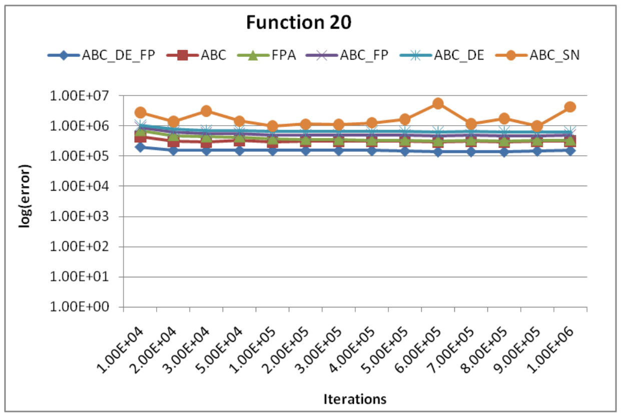


Figure 6.22: Convergence Graph of algorithms for F20

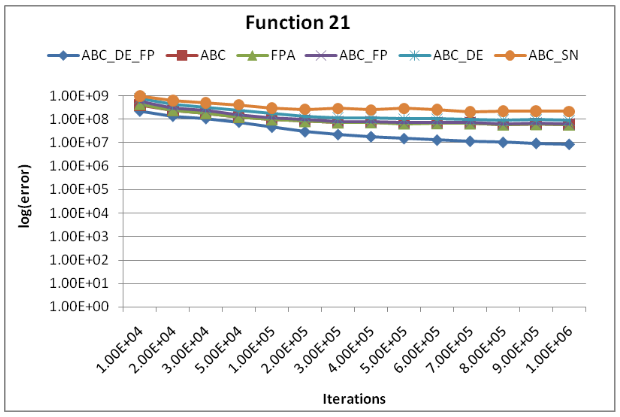


Figure 6.23: Convergence Graph of algorithms for F21

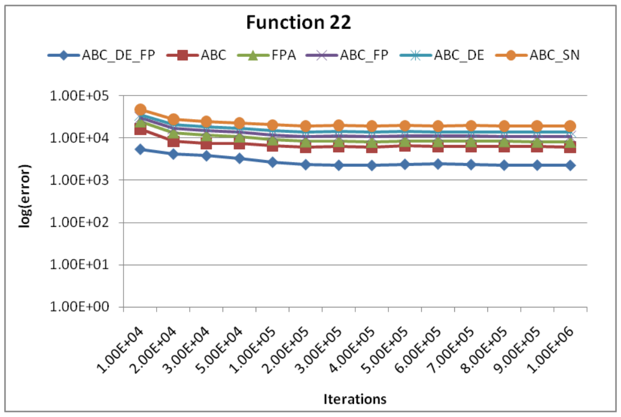


Figure 6.24: Convergence Graph of algorithms for F22

6.5.2.4. Composition Functions

The composition functions defined in problem definition of CEC 2014 [137] is composed of various sub components. Each component uses different basic functions that can be unimodal or multimodal or hybrid. Hence these functions are more complicated than hybrid functions. ABC_DE_FP shows minimum error value for functions F23, F24, F25, F26, F27 and F28 as given in Figure 6.25 to 6.32. 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 iterations. The performance curve is near to rectangular hyperbola. However, the efficacy of ABC_DE_FP is better as compared to other algorithms.

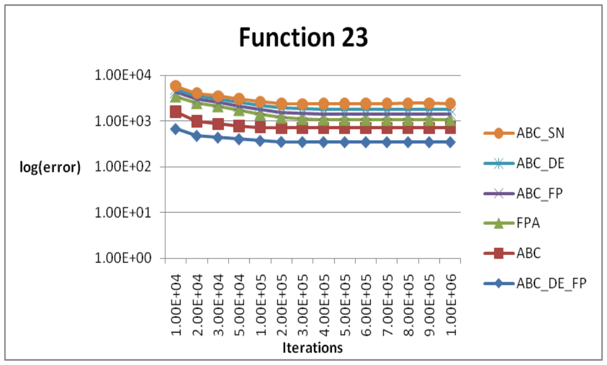


Figure 6.25: Convergence Graph of algorithms for F23

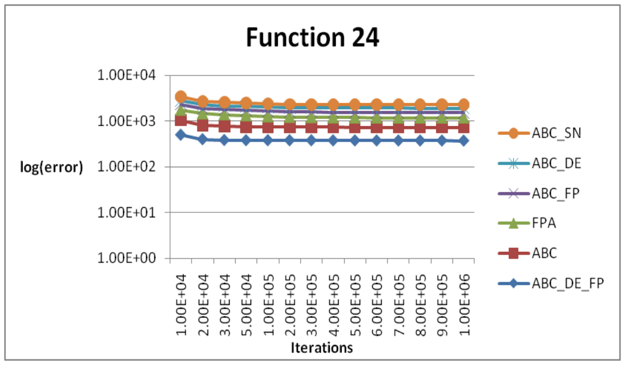


Figure 6.26: Convergence Graph of algorithms for F24

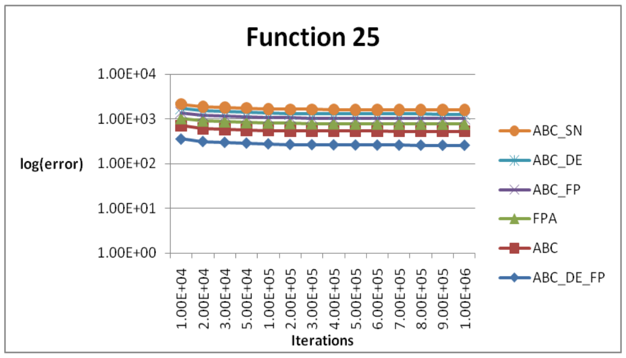


Figure 6.27: Convergence Graph of algorithms for F25

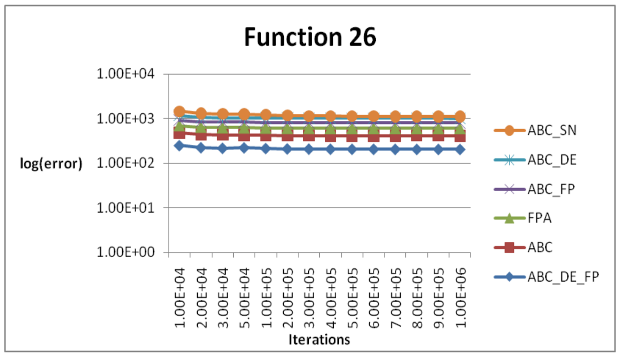


Figure 6.28: Convergence Graph of algorithms for F26

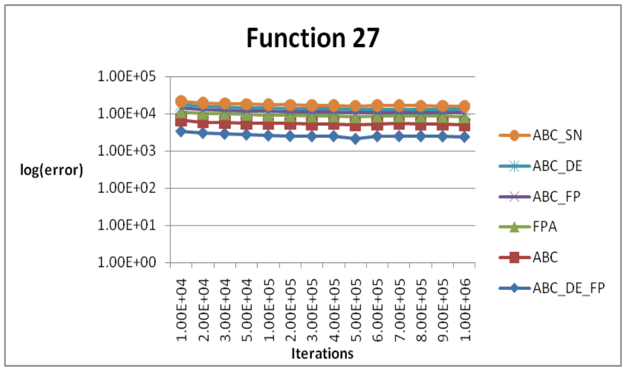


Figure 6.29: Convergence Graph of algorithms for F27

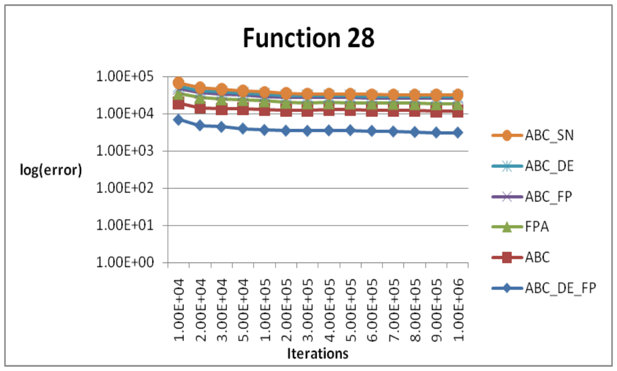


Figure 6.30: Convergence Graph of algorithms for F28

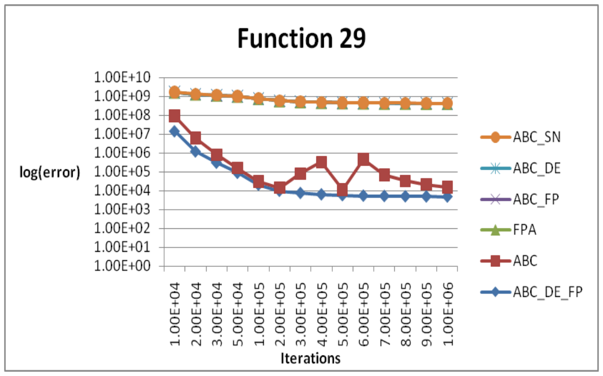


Figure 6.31: Convergence Graph of algorithms for F29

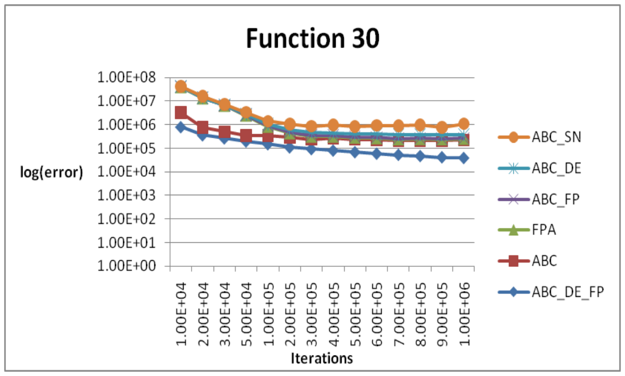


Figure 6.32: Convergence Graph of algorithms for F30

Discussion

The experiments are performed to assess the efficacy of ABC_DE_FP for simple as well as complex optimization problems. Results substantiate that for simple problems, presented 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 is established through Wilcoxon Rank Sum test. Convergence graphs portrays that the convergence speed of presented algorithm in attaining global optima is faster than existing algorithms. Thus, from these studies it can be inferred that, presented algorithm is more suitable to composite and hybrid (according to CEC2014) type of problems where global minimum value is difficult to achieve.

6.6. CONCLUSION

This chapter presents an improved hybrid version of artificial bee colony algorithm (ABC_DE_FP) in terms of exploitation and convergence speed. The presented algorithm is the combination of artificial bee colony algorithm (ABC), flower pollination algorithm and differential evolution. In order to exhibit the efficacy of presented algorithm, it is tested on simple benchmark function against contemporary ABC variants. The algorithm provides improvement in terms of best value attained. Another study is conducted on four sets of benchmark function obtained from CEC2014 problem definition at 10, 30, 50 and 100 dimensions. The presented algorithm is tested against ABC, FPA and various ABC variants. It has been observed that ABC_DE_FP performance improves with increase in dimensions. Wilcoxon rank sum test established the statistical significance of obtained results. Convergence rate of algorithms are analyzed via convergence graphs. The results illustrated that performance of ABC_DE_FP is effectively and competitively better though its convergence behavior is dependent on function’s property. Overall, ABC_DE_FP algorithm is more suitable for complex optimization problems.

Since ABC_DE_FP is scalable with dimensions hence in the next chapter binary version ABC_DE_FP is applied for clustering high dimensional data. High dimensional clustering is performed through subspace clustering. Thus, a hybrid algorithm of subspace clustering and binary ABC_DE_FP is developed named S_FAD algorithm and assessed on high dimensional real datasets.