International Journal of Recent Research and Review, Vol. VII, Issue 3, September 2014
ISSN 2277 – 8322

Fast-Convergent Artificial Bee Colony with an Adaptive Local Search

Aishwarya Varshney[1], R. L. Yadav[2], Vijay Kumar[3]
[1] *Research Scholar, Kautilya Institute of Technology, Jaipur, Rajasthan, India; Email: aishvarshney@gmail.com*
[2] *Assistant Professor, Kautilya Institute of Technology Jaipur, Rajasthan, India; Email: ram.bitspilani@gmail.com*
[3] *Assistant Professor, Kautilya Institute of Technology Jaipur, Rajasthan, India; Email: vijay_matwa@yahoo.com*

***Abstract* - Fast Convergent Artificial Bee Colony (ABC) algorithm has been emerged as one of the latest swarm intelligence based algorithm. Though, there are some areas where ABC outperforms the other optimization techniques but, the drawbacks like preference on exploration at the cost of exploitation and skipping the true solution due to large step sizes, are also associated with it. In this paper, to improve the exploitation capability of the ABC algorithm, a new local search strategy is developed and incorporated with ABC. The proposed local search strategy is based on linearly decreasing inertia weight which has been applied in particle swarm optimization (PSO) algorithm. The proposed local search strategy greedily reduces the interval of step size for the best solution in the colony. The modified ABC is named as fast convergent ABC with an adaptive local search (FABCLS). To validate the performance of the proposed algorithm FABCLS, 15 benchmark optimization functions of different complexities are considered and results comparison section shows the clear superiority of the proposed modification over the basic ABC.**

***Keywords* - Artificial Bee Colony, Local Search Algorithm, Exploration-Exploitation, Swarm Intelligence.**

I. INTRODUCTION

Swarm Intelligence is one of the recent outcomes of the research in the field of Nature inspired algorithms. Collaborative trial and error method is the main concept behind the Swarm Intelligence which enables the algorithmic procedure to find the solution. Researchers are analyzing such collaboration among the social insects while searching food for them and creating the intelligent structures known as Swarm Intelligence. Ant colony optimization (ACO) [1], particle swarm optimization (PSO) [2] & bacterial foraging optimization (BFO) [3] are some examples of swarm intelligence based techniques. The work presented in the articles [4-5] proved its efficiency and potential to deal with non linear, non convex and discrete optimization problems. D.Karaboga [6] contributed the recent addition to this category known as artificial bee colony (ABC) optimization algorithm. The ABC algorithm mimics the foraging behavior of honey bees while searching food for them. ABC is a simple and population based optimization algorithm. Here the population consists of possible solutions in terms of food sources for honey bees whose fitness is regulated in terms of nectar amount which the food source contains. The swarm updating in ABC is due to two processes namely, the variation process and the selection process which are responsible for explo-ration and exploitation, respectively. However the ABC achieves a good solution at a significantly faster rate but, like the other optimization algorithms, it is also weak in refining the already explored search space. On the other part, it is also required to tune the ABC control parameters based on problem. Also literature says that basic ABC itself has some drawbacks like stop proceeding toward the global optimum even though the population has not converged to a local optimum [7] and it is observed that the position update equation of ABC algorithm is good at exploration but poor at exploitation [8] i.e, has not a proper balance between exploration and exploitation. Therefore these drawbacks require a modification in position update

equation and/or a local search approach to be implemented in ABC. These drawbacks have also addressed in earlier research. To enhance the exploitation, Wei-feng Gao *et.al* [9] improved position update equation of ABC such that the bee searches only in neighborhood of the previous iteration's best solution. Anan Banharnsakun et al. [10] proposed the best-so-far selection in ABC algorithm and incorporated three major changes: The best-so-far method, an adjustable search radius, and an objective-value-based comparison in ABC. To solve constrained optimization problems, D. Karaboga and B. Akay [11] used Debs rules consisting of three simple heuristic rules and a probabilistic selection scheme in ABC algorithm. Dervis Karaboga [9] examined and suggested that the limit should be taken as SN ×D, where, SN is the population size and D is the dimension of the problem and coefficient φ_{ij} in position update equation should be adopted in the range of [-1, 1]. Further, Kang *et al.* [12] introduced exploitation phase in ABC using Rosenbrock's rotational direction method and named modified ABC as Rosenbrock ABC (RABC). In this paper, the effective modification in ABC that we made is the incorporation of a local search strategy in ABC after all its three (emplyoed, onlooker and scout bee) phases, based on linearly decreasing inertia weight (LDIW) [13]. This local strategy exploits only the global best position of the current swarm after each iteration.

Rest of the paper is organized as follows: In section 2, proposed modified ABC (FABCLS) is explained. In Section 3, performance of the proposed strategy is analyzed via numerical experiment. Finally, in section 4, paper is concluded.

II . FAST-CONVERGENT ARTIFICIAL BEE COLONY WITH AN ADAPTIVE LOCAL SEARCH

This section explains the proposed modified ABC algorithm. In this paper a new self adaptive local search strategy is incorporated with ABC. The motivation for the proposed modifications is described as follows:

As mentioned earlier, if φij and difference between randomly selected solution and current solution is high in position update equation of ABC then there will be sufficient chance to skip the global optima. In this situation, some local search strategy can help the search procedure. During the iterations, local search algorithm illustrates very strong exploitation capability [14]. Therefore, the exploitation capability of ABC algorithm may be enhanced by incorporating a local search strategy with ABC algorithm.

In this way, the exploration and exploitation capability of ABC algorithm could be balanced as the global search capability of the ABC algorithm explores the search space or tries to identify the most promising search space regions, while the local search strategy will exploit the identified search space.

Therefore, in this paper a self adaptive local search strategy is proposed and incorporated with the ABC. The proposed local search strategy is inspired from linearly decreasing inertia weight (LDIW) [13].

In the proposed local search strategy, the required step size ($i.e., \varphi(x - x_{random})$) to update an individual is reduced self adaptively to exploit the search area in the proximity of the best candidate solution. Thus, the proposed strategy is named as self adaptive local search (SALS). In SALS, the step size is reduced as a logarithmic function of iteration counter as shown in equations (2) and (3). In SALS, the random component (φ_{ij}) of basic ABC algorithm is optimized to direct the best solution to update its position. This process can be seen as an optimization problem solver which minimizes the unimodal continuous objective function $f(\varphi_{ij})$ in the direction provided by the best solution x_{best} over the variables *w1,w2* in the interval [−1, 1] or simply it optimizes the following mathematical optimization problem:

$$min\ f(\varphi)\ in\ [-1, 1]; \qquad (1)$$

SALS process starts with initial range *(w1 = −1, w2 = 1)* and generates two points in this interval by diminishing the edges of the range through a greedy way using the equations (2) and (3). At a time, either of these equations

is executed depends which *f(w1)* or *f(w2)* has better fitness. If *w1* provides better fitness then edge *w2* of the range shrinks towards w1 otherwise *w1* shifts itself near to *w2*. The detailed implementation of SALS can be seen in Algorithm 1.

$$w1 = w1 + (w2 - w1) \times t / maxiter \quad (2)$$
$$w2 = w2 - (w2 - w1) \times t / maxiter \quad (3)$$

where, *t* and *maxiter* are the current iteration counter and maximum allowable iterations in the local search algorithm respectively.

Algorithm 1 terminates when either iteration counter exceeds the maximum iteration allowable in local search or absolute difference between *w1* and *w2* falls below a user defined parameter ε.

In algorithm 2, D is the dimension of the problem, U(0, 1) is a uniformly distributed random number in the range (0, 1), p_r is a perturbation rate and is used to control the amount of disturbance in the best solution x_{best} and x_k is a randomly selected solution in the population.

As the modifications are proposed to improve the convergence speed of ABC, while maintaining the diversity in the swarm. Therefore, the proposed strategy is named as "Fast-convergent Artificial Bee Colony with an adaptive Local Search" (FABCLS).

The FABCLS is composed of four phases: employed bee phase, onlooker bee phase, scout bee phase. The last phase, namely self adaptive local search phase is executed after the completion of scout bee phase. The pseudo-code of the proposed FABCLS algorithm is shown in Algo 3.

Algorithm 1: Self Adaptive Local Search (SALS) Strategy:

Input: Optimization function M inf(x), the best solution x_{best}*;*

Initialize termination parameters ε*, maximum number of iteration counter maxiter and variables w1 = −1, w2 =1 and itercount = 1;* ***while*** *(*$|w1 - w2| > \varepsilon$ *and itercount* $\leq$ *maxiter)* ***do***

Generate two new solutions x_{new1} *and* x_{new2} *from* x_{best} *by using w1 and w2, respectively using Algorithm 2;*

Calculate $f(x_{new1})$ *and* $f(x_{new2})$*;*

if $f(x_{new1}) < f(x_{new2})$ ***then***

$w2 = w2 - (w2 - w1) \times itercoun / maxiter$;

if $f(x_{new1}) < f(x_{best})$ ***then***

$x_{best} = x_{new1}$;

end if

else

$w1 = w1 + (w2 - w1) \times itercount / maxiter$;

if $f(x_{new2}) < f(x_{best})$ ***then***

$x_{best} = x_{new2}$;

end if

end if

Set itercount = itercount + 1;

end while

Algorithm 2: New solution generation:

Input: w and best solution x_{best} *;*

for *j = 1 to D* ***do***

if $U(0, 1) < p_r$ ***then***

$x_{newj} = x_{bestj} + w(x_{bestj} - x_{kj})$;

else

$x_{newj} = x_{bestj}$;

end if

end for

Return x_{new}

Algorithm 3: Accelerating ABC using adaptive local search (FABCLS):

Initialize the population and control parameters;

while *Termination criteria is not satisfied* ***do***

Step 1: Employed bee phase.

Step 2: Onlooker bees phase to update the food sources based on their profitability.

Step 3: Scout bee phase to determine the new food sources for exhausted food sources.

Step 4: Apply self adaptive local search strategy phase to exploit the best solution found so far using Algorithm 1

end while

Return the best solution.

III . EXPERIMENTAL RESULTS AND DISCUSSION

A. Test problems under consideration

To validate the effectiveness of the proposed algorithm FABCLS, 15 mathematical optimization problems (f_1 to f_{15}) of different characteristics and complexities are taken into consideration (listed in Table 1).These all problems are continuous in nature. Test problems $f_1 - f15$ are taken from [15].

B. Experimental setting

The results obtained from the proposed FABCLS are stored in the form of success rate, average number of function evaluations, standard deviation of the fitness and mean error. Results for these test problems (Table I) are also obtained from the basic ABC and recent variant of ABC named Modified AB (MABC) [16] for the comparison purpose. The following parameter setting is adopted while implementing our proposed and other considered algorithms to solve the problems:

- The number of simulations/run =100,
- Colony size NP = 50 [17-18] and Number of food
- sources SN = NP/2, φ_{ij} = rand[−1, 1] and limit = Dimension×Number of food sources=D × SN[11, 16],
- C = 1.5 [8],
- The terminating criteria: Either acceptable error (Table I) meets or maximum number of function evaluations (which is set to be 200000)is reached,
- The proposed local search in ABC runs either 10 times (based on empirical experiment) for each iteration or ε = 0.001 whichever comes earlier in algorithm.
- Parameter settings for the other considered algorithms ABC and MABC are adopted from their original articles.

C. Results Analysis of Experiments Results Analysis of Experiments

Table II presents the numerical results for benchmark problems of Table 1 with the experimental settings shown in section *B*. Table II shows the results of the proposed and other considered algorithms in terms of average number of function evaluations (AFE), standard deviation (SD), mean error (ME) and success rate (SR). It can be observed from Table II that FABCLS outperforms the considered algorithms most of the time in terms of accuracy, reliability and efficiency. Some other statistical tests like acceleration rate (AR) [19], and boxplots have also been done in order to analyze the algorithms output more intensively.

D. Statistical Analysis

Algorithms ABC, MABC and FABCLS are compared based on SR, AFE, and ME. First SR of all these algorithms is compared and if it is not possible to distinguish the performance of algorithms based on SR then comparison is made on the basis of AFE. ME is used for comparison if the comparison is not possible on the basis of SR and AFE both. It is clear from Table II, when the results of all functions are evaluated together, the FABCLS algorithm is the cost effective algorithm for most of the functions.

Since boxplot [20] can efficiently represent the empirical distribution of results, the boxplots for average number of function evaluations for all algorithms FABCLS, ABC, and MABC have been represented in Figure 1.Figure 1 shows that FABCLS is cost effective in terms of function evaluations as interquartile range and median of average number of function evaluations are very low for FABCLS.

Further, we compare the convergence speed of the considered algorithms by measuring the AFEs. A smaller AFEs means higher convergence speed. In order to minimize the effect of the stochastic nature of the algorithms, the reported function evaluations for each test problem is averaged over 100 runs.

In order to compare convergence speeds, we use the acceleration rate (AR) which is defined as follows, based on the AFEs for the two algorithms

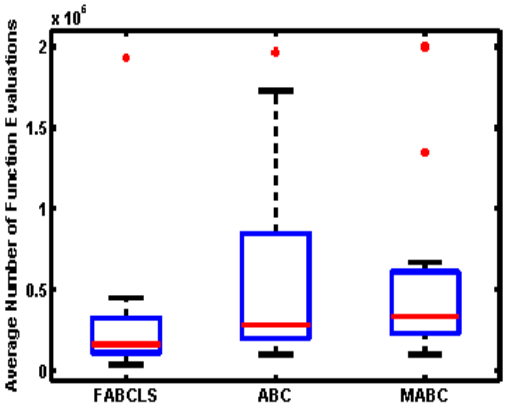

Fig 1: Boxplots graph for Average number of function evaluation

ALGO and FABCLS: the AFEs for the two algorithms ALGO and FABCLS:

$$AR = AFE_{ALGO} / AFE_{FABCLS} \quad (4)$$

where, ALGO2 {ABC, and MABC} and AR > 1 means FABCLS is faster. In order to investigate the AR of the proposed algorithm as compare to the considered algorithms, results of Table II are analyzed and the value of AR is calculated using equation (4).

Table III shows a comparison between FABCLS - ABC, and FABCLS - MABC in terms of AR. It is clear from the Table III that convergence speed of FABCLS is better than considered algorithms for most of the functions.

IV . Conclusion

ABC is a simple algorithm with very less parameters with drawbacks like premature convergence and poor in exploitation. This article proposed a self adaptive local search strategy and incorporated it with ABC to enhance the exploitation capability. The proposed algorithm has been extensively compared with other recent variants of ABC namely, MABC. Through the extensive experiments, it can be stated that the proposed algorithm is a competitive algorithm to solve the continuous optimization problems.

V. References

[1] B. Akay and D. Karaboga. A modified artificial bee colony algorithm for real-parameter optimization. Information Sciences, doi:10.1016/j.ins.2010.07.015, 2010.

[2] J. Kennedy and R. Eberhart. Particle swarm optimization. In Neural Networks, 1995. Proceedings.,IEEE International Conference on, volume 4, pages 1942–1948. IEEE, 1995.

[3] K.M. Passino. Biomimicry of bacterial foraging for distributed optimization and control. Control Systems Magazine, IEEE, 22(3):52–67, 2002.

[4] K.V. Price, R.M. Storn, and J.A. Lampinen. Differential evolution: a practical approach to global optimization. Springer Verlag, 2005.

[5] J. Vesterstrom and R. Thomsen. A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems. In Evolutionary Computation, 2004. CEC2004. Congress on, volume 2, pages 1980–1987. IEEE, 2004.

[6] D. Karaboga. An idea based on honey bee swarm for numerical optimization. Techn. Rep. TR06, Erciyes Univ. Press, Erciyes, 2005.

[7] D. Karaboga and B. Akay. A comparative study of artificial bee colony algorithm. Applied Mathematics and Computation, 214(1):108–132, 2009.

[8] G. Zhu and S. Kwong. Gbest-guided artificial bee colony algorithm for numerical function optimization. Applied Mathematics and Computation, 217(7):3166–3173, 2010.

[9] W. Gao and S. Liu. A modified artificial bee colony algorithm. Computers & Operations Research, 2011.

[10] A. Banharnsakun, T. Achalakul, and B. Sirinaovakul. The best-so-far selection in artificial bee colony algorithm. Applied Soft Computing, 11(2):2888–2901, 2011.

[11] Dervis Karaboga and Bahriye Akay. A modified artificial bee colony (abc) algorithm for constrained optimization problems. Applied Soft Computing, 11(3):3021–3031, 2011.

[12] F. Kang, J. Li, and Z. Ma. Rosenbrock artificial bee colony algorithm for accurate global optimization of numerical functions. Information Sciences, 181(16):3508–3531, 2011.

[13] Yuhui Shi and Russell C Eberhart. Empirical study of particle swarm optimization. In Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on, volume 3. IEEE, 1999.

[14] H.Wang, D.Wang, and S. Yang. A memetic algorithmwith adaptive hill climbing strategy for dynamic optimization problems. Soft Computing-A Fusion of Foundations, Methodologies and Applications,13(8):763–780, 2009.

[15] M.M. Ali, C. Khompatraporn, and Z.B. Zabinsky. A numerical evaluation of several stochastic algorithms on selected continuous global optimization test problems. Journal of Global Optimization,31(4):635–672, 2005.

[16] M. Dorigo and G. Di Caro. Ant colony optimization: a new meta-heuristic. In Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on, volume 2. IEEE, 1999.

[17] K. Diwold, A. Aderhold, A. Scheidler, and M. Middendorf. Performance evaluation of artificial bee colony optimization and new selection schemes. Memetic Computing, pages 1–14, 2011.

[18] M. El-Abd. Performance assessment of foraging algorithms vs. evolutionary algorithms. Information Sciences, 182(1):243–263, 2011.

[19] S. Rahnamayan, H.R. Tizhoosh, and M.M.A. Salama. Opposition-based differential evolution. Evolutionary Computation, IEEE Transactions on, 12(1):64–79, 2008.

[20] D.F. Williamson, R.A. Parker, and J.S. Kendrick. The box plot: a simple visual method to interpret data. Annals of internal medicine, 110(11):916, 1989.

Table I
Test Problems

Test Problem	Objective Function	Search Range	Optimum Value	D	Acceptable Error
Parabola (Sphere)	$f_1(x) = \sum_{i=1}^{D} x_i^2$	[-5.12 , 5.12]	f(0) = 0	30	1.0E – 05
De Jong's f4	$f_2(x) = \sum_{i=1}^{D} i(x_i)^4$	[-5.12 , 5.12]	f(0) = 0	30	1.0E – 05
Griewank	$f_3(x) = 1 + \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right)$	[-600 , 600]	f(0) = 0	30	1.0E – 05
Rosenbrock**	$f_4(x) = \sum_{i=1}^{D-1}(100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2)$	[-30 , 30]	f(0) = 0	30	1.0E – 02
Ackley	$f_5(x) = 20 + e + exp(-\frac{0.2}{D}\sqrt{\sum_{i=1}^{D} x_i^2}$ $-exp\left(\frac{1}{D}\sum_{i=1}^{D} cos(2\pi x_i)\right) x_i)$	[-1 , 1]	f(0) = 0	30	1.0E – 05
Alpine	$f_6(x) = \sum_{i=1}^{D} \lvert x_i \sin x_i + 0.1\, x_i \rvert$	[-10 , 10]	f(0) = 0	30	1.0E – 05
Michalewicz function	$f_7(x) = -\sum_{i=1}^{D} \sin x_i \left(\sin \left(\frac{i x_i^2}{\pi}\right) 20 \right)$	[0 , π]	f_{min} = -9.66015	10	1.0E – 05
Salomon Problem (SAL) f(0,0,0,0,0,0)=0	$f_8(x) = 1 - \cos(2\pi \sqrt{\sum_{i=1}^{D} x_i^2} + 0.1 \left(\sqrt{\sum_{i=1}^{D} x_i^2}\right)$	[-100, 100]	f(0) = 0	30	1.00E-01
Axis parallel hyperellipsoid	$f_9(x) = \sum_{i=1}^{D} i x_i^2$	[-5.12 , 5.12]	f(0) = 0	30	1.0E – 05
Sum of different powers	$f_{10}(x) = \sum_{i=1}^{D} \lvert x_i \rvert^{i+1}$	[-1 , 1]	f(0) = 0	30	1.0E – 05
Step function[100, 100]f(-0.5<=x <=0.5)=0	$f_{11}(x) = \sum_{i=1}^{D} \lfloor x_i + 0.5 \rfloor^2$	[-100, 100]	f(-0.5<=x <=0.5)=0	30	1.0E – 05
Inverted cosine wave function (Masters) [-5, 5] f(000..0)=-D+1	$f_{12}(x) = -\sum_{i=1}^{D-1}(exp\left(\frac{-(x_i^2 + x_{i+1}^2 + 0.5 x_i x_{i+1})}{8}\right) x I)$ where $I = \cos(4 \sqrt{x_i^2 + x_{i+1}^2 + 0.5 x_i x_{i+1}})$	[-5,5]	f(0) = -D+1	10	1.0E – 05
Neumaier 3 Problem (NF3) (Neumaier, 2003b)	$f_{13}(x) = \sum_{i=1}^{D} (x_i - 1)^2 - \sum_{i=2}^{D} x_i x_{i-1}$	$[-D^2, D^2]$	$f_{min} = -\frac{D(D+4)(D-1)}{6}$	10	1.00E-01
Rotated hyper-ellipsoid function	$f_{14}(x) = \sum_{i=1}^{D} \sum_{j=1}^{i} x_j^2$	[-65.536, 65.536]	f(0) = 0	30	1.0E – 05
Levi montalvo 1	$f15(x) = \frac{\pi}{D}(10\sin2(\pi y_i) + \sum_{i=1}^{D-1}((y_i - 1)2 \text{ x } (1 + 10\sin2(\pi y_{i+1})) + (y_D - 1)2)$ where $y_i = 1 + \frac{1}{4}(x_i + 1)$	[-10,10]	f(0) = 0	30	1.0E – 05

Table II
Comparison of the results of test problems

TP	Algorithm	SD	ME	AFE	SR
f_1	FABCLS	9.16E-06	9.16E-06	11789.2	100
	ABC	8.17E-06	2.02E-06	20409	100
	MABC	8.95E-06	9.48E-07	22359	100
f_2	FABCLS	8.43E-06	8.43E-06	6483.4	100
	ABC	4.90E-06	3.11E-06	9578.5	100
	MABC	8.63E-06	1.27E-06	22584	100
f_3	FABCLS	8.12E-05	8.12E-05	30623.27	99
	ABC	2.28E-04	1.26E-03	45942.82	97
	MABC	9.24E-06	5.42E-07	44038.5	100
f_4	FABCLS	2.86E+01	2.86E+01	193962.93	7
	ABC	1.60E+00	3.08E+00	172861	28
	MABC	3.60E+01	2.41E+01	200000	0
f_5	FABCLS	9.50E-06	9.50E-06	30774.74	100
	ABC	8.28E-06	1.56E-06	49107	100
	MABC	9.51E-06	4.31E-07	43333	100
f_6	FABCLS	8.97E-06	8.97E-06	45730.86	100
	ABC	8.21E-06	2.27E-06	77527	100
	MABC	1.01E-03	7.61E-04	199704.63	2
f_7	FABCLS	4.25E-06	-9.66E+00	19095.88	100
	ABC	3.66E-06	3.74E-06	27213.73	100
	MABC	6.70E-06	3.28E-06	36367.7	100
f_8	FABCLS	9.20E-01	9.20E-01	23958.86	100
	ABC	9.77E-01	6.61E-02	158845.77	60
	MABC	9.33E-01	3.82E-02	27650	100
f_9	FABCLS	8.97E-06	8.97E-06	13379.12	100
	ABC	7.90E-06	1.99E-06	22765	100
	MABC	9.22E-06	6.91E-07	25772	100
f_{10}	FABCLS	5.73E-06	5.73E-06	4315.28	100
	ABC	5.49E-06	2.90E-06	16095.5	100
	MABC	7.57E-06	1.90E-06	9489	100
f_{11}	FABCLS	0.00E+00	0.00E+00	8218.3	100
	ABC	0.00E+00	0.00E+00	11501.02	100
	MABC	0.00E+00	0.00E+00	15844	100
f_{12}	FABCLS	7.96E-06	-9.00E+00	31889.81	100
	ABC	2.39E-02	1.34E-01	87198.49	89
	MABC	8.08E-06	1.64E-06	66422.19	100
f_{13}	FABCLS	9.03E-02	-2.10E+02	20770.93	100
	ABC	9.15E-01	6.63E-01	196658.24	5
	MABC	1.01E-01	2.08E-02	134805.99	96
f_{14}	FABCLS	9.17E-06	9.17E-06	16816.12	100
	ABC	7.75E-06	2.20E-06	27957.5	100
	MABC	9.23E-06	6.90E-07	33065	100
f_{15}	FABCLS	9.15E-06	9.15E-06	11165.68	100
	ABC	6.98E-06	2.24E-06	19607.5	100
	MABC	9.06E-06	7.90E-07	22738	100

Table III

Acceleration Rate (AR) of FABCLS as compared to the ABC and MABC, TP: Test Problems

TP	ABC	MABC
f_1	1.789789021	1.96079635
f_2	1.630366328	3.844045848
f_3	1.427561747	1.36838962
f_4	0.893905381	1.034286084
f_5	1.538661587	1.35774579
f_6	1.732237123	4.462132855
f_7	1.692745642	2.262139944
f_8	6.837846389	1.190251731
f_9	1.715578036	1.942186565
f_{10}	4.593910368	2.708310738
f_{11}	1.404979294	1.935523278
f_{12}	2.563734002	1.952887338
f_{13}	15.03419457	10.30569318
f_{14}	1.645067827	1.945601992
f_{15}	1.831326494	2.123712958