Modified Artificial Bee Colony Algorithm Using Differential Evolution and Polynomial Mutation for Real-Parameter Optimization

Aditya Narayan Hati[1] Rajkumar Darbar[2] Nanda Dulal Jana[1] Jaya Sil[3]

[1]Dept. of Information Technology, NIT Durgapur, India.
[2]School of Information Technology, IIT Kharagpur, India.
[3]Dept. of Computer Science Engineering, BESU, Howrah, WB, India

smartyadi88@gmail.com, rajdarbar.r@gmail.com, nanda.jana@gmail.com, js@cs.becs.ac.in

Abstract—Artificial Bee Colony (ABC) is a swarm based stochastic search algorithm inspired by the foraging behavior of honeybees. Due to the simplicity of implementation and promising optimization capability, ABC is successfully applied to solve wide class of scientific and engineering optimization problems. But, it has problems of premature convergence and trapping in local optima. In this paper, to enhance the performance of ABC, we have proposed a modified version of ABC algorithm using Differential Evolution (DE) and Polynomial Mutation (PM) called DE-PM-ABC. The comparison with ABC by Karaboga [1], MABC [27] by Liu et al. using some benchmark functions of CEC 2005 demonstrates that our approach achieves a good trade-off between exploration and exploitation and thus obtains better global optimization result and faster convergence speed.

Keywords — Artificial Bee Colony (ABC), Differential Evolution (DE), Polynomial Mutation (PM), Real parameter optimization.

I. INTRODUCTION

In recent years, swarm intelligence (SI) has become a research interest to many research scientists. Many SI algorithms such as particle swarm optimization (PSO) [2], ant colony optimization (ACO) [3], and bee colony optimization (BCO) [4] etc. have been proposed in order to solve a wide range of optimization problems. These algorithms are well-known due to their ability to produce low cost, fast and reasonably accurate solutions for the numerical optimization problems.

The artificial bee colony algorithm [1][5] also belongs to swarm intelligence family. It is simple in concept, easy to implement and there are fewer control parameters setting [6][7]. It is proven to be a better than other algorithms for global optimization [8-11]. But it has problems of premature convergence and trapping in local optimal [12, 13]. For this reasons, to improve the performance of basic version of ABC algorithm, some modification in parameters and also proper exploration and exploitation equations are necessary.

Liu X et al. [19] proposed artificial bee colony programming (ABCP) employing randomized distribution, bit hyper-mutation and a novel crossover operator (layer noise crossover (LNX)) to significantly improve the performance of the original ABC algorithm. Li Bao et al. [23] compared and analyzed several selection strategies of ABC such as disruptive selection strategy, tournament selection strategy and rank selection strategy. In paper [24], Nadezda Stanarevic proposed a new approach for extending ABC algorithm based on five mutation strategies “borrowed” from differential evolution (DE) algorithm in order to improve the exploitation process. Andrej Aderhold et al. [25] studied the influence of the population size on the optimization behavior of ABC. In paper [26], Wei-Ping Lee et al. proposed diversity strategy based artificial bee colony algorithm (DABC) due to overcome the problem of premature convergence and trapping in local optimum of ABC.

In this paper, we have proposed a modified artificial bee colony algorithm (DE-PM-ABC) using Differential Evolution (DE) and Polynomial Mutation (PM) for real parameter optimization. The DE/rand/1 mutation strategy is used in employed bee and onlooker bee phase. The polynomial mutation is used in scout bee phase. The DE-PM-ABC has well-balanced between exploitation and exploration. We have compared the performance of DE-PM-ABC approach against the basic ABC algorithm and MABC algorithm [27]. The result shows that DE-PM-ABC method has better convergence rate and global searching capability and it outperforms the basic ABC and MABC.

II. OVERVIEW OF ABC

The Artificial Bee Colony (ABC) is a swarm based a stochastic search algorithm which imitates the foraging behavior of honeybees. In ABC algorithm [1, 9, 14], the colony of artificial bees consists of three groups of bees: Employed bees, Onlookers and Scouts. In ABC, artificial bees fly around in a multidimensional search space and the employed bees choose food sources depending on the experience of themselves and the onlooker bees choose food sources based on their nest mates experience and adjust their positions. Scout bees fly and choose the food sources randomly without using experience. Each food source chosen represents a possible solution to the problem under consideration. The nectar amount of the food source represents the quality or fitness of the solution. The number of employed bees or the onlooker bees is equal to the number of food sources or possible solutions in the population. In other words, the number of employed bees is equal to the number of food sources around the hive. The employed bee whose food source has been abandoned becomes a scout. In ABC algorithm the position of food source determines the solution and the amount of nectar represents the fitness of the respective solution. The step-by-step procedure of ABC is as follows:

PROCEDURE ABC

```
1. Initialize Population
2. Repeat
3.     Place the employed bees on their food
       sources.
4.     Place the onlooker bees on the food
       sources depending on their nectar
       amounts.
5.     Send the scouts to the search area for
       discovering new food sources.
6.     Memorize the best food source found so
       far.
7. Until requirements are met
```

Here, SN denotes the number of food source and D denotes the dimension of a food source. SN is selected as the half of the number of bees. To initialize the population equation (1) is used.

$$X_{ij}=X_{ij}^{LB}+\left(X_{ij}^{UB}-X_{ij}^{LB}\right)\times U[0,1] \quad (1)$$

Here X is the food source where i and j denote the food source number and the component of a food respectively. U[0, 1] is a uniform random number generator which generate a number between 0 and 1. In order to produce a candidate food position from the old one in memory, the ABC uses the following expression

$$V_{ij}=X_{ij}+\left(X_{ij}-X_{kj}\right)\times\Phi_{ij} \quad (2)$$

In equ (2), V and X are the new and old food sources respectively. Here, only one component is chosen to modify from the whole food source. The value of Φ is chosen between -1 to 1. This equation is used to modify the food source for both employed bee and onlooker bee phases.

After generation of the new food source from the previous food source in employed bee phase, the new food source is selected through greedy selection process depending on the probability value associated with that food source, p_i, calculated by the following expression

$$P_i = \frac{\text{fitness}_i}{\sum \text{fitness}_i} \quad (3)$$

Here, fitness_i is the fitness value of the food source i. The fitness value is selected through the following equation

$$\text{fitness}_i = \begin{cases} \dfrac{1}{1+\text{objval}_i} & ;\ \text{if objval}_i \geq 0 \\ 1+\text{abs}(\text{objval}_i);\ \text{Otherwise} \end{cases} \quad (4)$$

Where objval_i is the objective function value for the food source i. In scout bee phase, the food sources which are abandoned by employed and onlooker bees are updated by the equation (1).

To do this, a counter, named trial, is used. If the value of trial crosses a predefined value called limit, then the food source is reinitialized with the above equation. The value of limit is determined by **Limit=SN×D.**

III. DE-PM-ABC

In this section we are describing our proposed modified version of artificial bee colony algorithm i.e. DE-PM-ABC.

A. Differential Evolution (DE)

Differential evolution (DE) **[16]** is a population based stochastic heuristic for global optimization. There are several variants, proposed by Storn. DE can be represented by DE/x/y/z format, where x represents the choice of the target vectors from the population pool for mutation, y represents the number of difference vectors used in mutation and z represents the crossover strategy. Also there are only three parameters to control, they are, population size (np), scale factor (f) and crossover probability (cr). The DE algorithm can be described in the following way.

1) Initialization

In DE, each component (eg. ith component) of a vector has an upper bound x_{ij}^{UB} and a lower bound x_{ij}^{LB}. Here an initialized vector is called target vector (X_{ij}) of D dimension where i represents the component number, j represents the target vector number in the current population matrix and a set of target vectors is called population matrix pop of dimension $D\times np$. The population matrix is initialized as follows:

$$X_i=X_{ij}^{LB}+\left(X_{ij}^{UB}-X_{ij}^{LB}\right)\times U[0,1] \quad (5)$$

Where the U[0, 1] is a uniform random number generator which generates a number between 0 and 1 with uniform random distribution.

2) Mutation

Mutation is a kind of exploration technique that can explore the search space rapidly. It creates donor vectors (V_{ij}) of dimension $D\times np$. The mutation is done by the following equation

$$V_{ij}=X_{r_1,j}+\left(X_{r_2,j}-X_{r_3,j}\right)\times f\ ;\ r_1\neq r_2\neq r_3\neq i \quad (6)$$

Here f is a control parameter called scale factor. It controls the speed of convergence towards optimal solutions depending on its value. Range of f is given as [0, 1].

3) Crossover

Crossover generates trial vector (U_{ij}) of dimension $D\times np$. There are two types of crossover strategies: (i) Binomial crossover and (ii) exponential crossover. Binomial crossover is very similar to uniform crossover in evolutionary algorithms. It is described as follows:

$$U_{ij}=\begin{cases} V_{ij} \text{ if`jrand} < cr \vee j=k \\ X_{ij} \text{ otherwise} \end{cases} \quad (7)$$

jrand is a random value generated for each j with uniform distribution. This crossover suggests that in trial vector, there must be at least one component from the donor vector. The cr is called crossover probability. It is defined as $cr\in[0,1]$.

4) Selection

Selection is a greedy method. Selection is also termed as mother-child competition. As its name indicates, mother X_i and child U_i compete with each other for survival chance in the next generation. The mother-child competition is done as follows:

$$X_i=\begin{cases} U_i \text{ iff}\left(U_i\right)\leq f\left(X_i\right) \\ X_i \text{ otherwise} \end{cases} \quad (8)$$

Thus, each individual of the temporary (trial) population is compared with its counterpart in the current population. The one with the lower objective function value will survive from the tournament selection to the population of the next generation. As a result, all the individuals of the next generation are as good as or better than their counterparts in the current generation.

B. Polynomial Mutation

If X_i is the value of ith parameter selected for mutation with a mutation probability P_m, then polynomial mutation (PM) can be represented by equation (9). This polynomial mutation is based on polynomial probability distribution which is written in equation (10).

$$X_i(t+1)=X_i(t)+\left(X_i^U-X_i^L\right)\times\delta_i \quad (9)$$

$$P(\delta)=0.5\times(\eta_m+1)\times(1-(\delta))^{\eta_m} \quad (10)$$

$$P_m=\frac{1}{D}+\left(1-\frac{1}{D}\right)\frac{t}{t_{max}} \quad (11)$$

Here, X_i^U is the Upper Bound of X_i ; X_i^L is the Lower Bound of X_i; η_m is polynomial distribution index that can take any non-negative value. The distribution is controlled by this parameter. By varying its value, the perturbation can be varied in the mutated solution. If the value of η_m is large, a small perturbation in the value of a variable is achieved.

D is dimension of the problem; t is current iteration number; t_{max} is maximum iteration number. The parameter δ_i is calculated as follows:

$$\delta_i = \begin{cases} (2r_i)^{\frac{1}{\eta_m}+1} & ; \text{if } r_i < 0.5 \\ 1 - \left(2(1-r_i)^{\frac{1}{\eta_m}+1}\right) & ; \text{if } r_i \geq 0.5 \end{cases} \tag{12}$$

Here, r_i is a random number in [0,1]. The algorithm for polynomial mutation (PM) is described below.

```
01. i←1
02. repeat
03.    $r_i$ ← U[0,1]
04.    if $r_i$< 0.5 then
05.            $\delta_i = (2r_i)^{1/\eta_m+1}$
06.    else
07.            $\delta_i = 1-\left(2(1-r_i)^{1/\eta_m+1}\right)$
08.    end if
09.    q ← U[0,1]
10.    if q ≤$P_m$ then
11.            $X_i(t+1) = X_i(t) + \left(X_i^{UB} - X_i^{LB}\right) \times \delta_i$
12.    end if
13.  i = i + 1
14. until   i = np
```

C. Proposed method

It is well known that both exploration and exploitation are necessary for the population-based meta-heuristic optimization algorithms, such as GA, PSO, DE and so on. In these optimization algorithms, the exploration refers to the ability to investigate the various unknown regions in the solution space to discover the global optimum. While, the exploitation refers to the ability to apply the knowledge of the previous good solutions to find better solutions [18]. In practice, the exploration and exploitation contradict with each other, and in order to achieve good optimization performance, the two abilities should be well balanced.

In our present study, the performance of artificial bee colony (ABC) greatly depends on exploration and exploitation which is done by the scout and employed bees respectively. But in real situation, ABC algorithm often faces some problems like weak global search capability, local optimum and so on due to imbalance between exploration and exploitation. Details of these problems have briefly explained in different research articles. For example, in paper [19], Liu X et al. mentioned that ABC algorithm focuses on the neighborhood search, while its global search capability is weak. This mainly manifests the following two aspects. Onlookers select food source completely depend on the fitness of the corresponding employed bees. Under the strategy, those individuals with better fitness have large chances to get more onlookers. After the mutation operator, the modified onlooker bees carry out the neighborhood search. But those employed bees with worse fitness only obtain few onlookers during the greedy selection strategy, so their neighborhoods are not explored enough and discarded finally. What's more, the diversity information of population is lost, and the evolutionary process could fall into local optimum. The scouts are up to global search task through distributing food sources randomly, but the amount of scouts is about 10% to the population size, so the global search capability of ABC algorithm is poor. On the other hand, the strategy of global search is only randomly distribution, which is too simple to work for practical problem.

In another work, Zhu G et al. [20] noted that according to the solution search equation of ABC algorithm described by equation (2), the new candidate solution is generated by moving the old solution towards (or away from) another solution selected randomly from the population. However, the probability that the randomly selected solution is a good solution is the same as that the randomly selected solution is a bad one, so the new candidate solution is not promising to be a solution better than the previous one. On the other hand, in equation (2), the coefficient Φ_{ij} is a uniform random number in [-1, 1] and x_{kj} is a random individual in the population, therefore, the solution search dominated by equation (2) is random enough for exploration. To sum up, the solution search equation described by equation (2) is good at exploration but poor at exploitation. So, to improve the performance of ABC algorithm at a certain level, modification is very necessary.

Here, we proposed DE-PM-ABC algorithm using Differential Evolution and Polynomial Mutation. In our proposed algorithm, we modified exploitation strategy of employed bee (EB) and onlooker bee (OB) using equation (6) i.e. DE/rand/1. Similarly, we have also improved the exploration strategy of scout bee (SB) polynomial mutation procedure. Basically, in this paper, we have introduced some further modification in 'Improved ABC' method proposed by X. T. Li et al. [21] by incorporating the concept of polynomial mutation in scout bee phase. The algorithm is as follows:

PROCEDURE DE-PM-ABC

```
1. Initialize the population randomly in the search space by the following equation:
   $X_i = X_i^L + \left(X_i^U - X_i^L\right) \times U[0,1]$
2. Calculate the fitness of the initialized population.
3. While termination condition not satisfied
3.1. if U[0,1] ≤ MR
        $V_{ij} = X_{r_1,j} + X_{r_2,j} - X_{r_3,j}$ ; $r_1 \neq r_2 \neq r_3$
     else
        $V_{ij} = X_{i,j}$
3.2. Calculate the fitness and make a greedy selection  between V and X.
3.3. Update the trial counter for those whose food source is not updated.
3.4. Calculate the probability $P_i = \frac{fitness_i}{\sum fitness_i}$
3.5. Do the same from 3.1 to 3.3 for the Onlooker Bees.
3.6. if  $trial_i$ > limit
Update the scout bee population using polynomial mutation.
4. end.
```

IV. EXPERIMENTAL RESULTS & ANALYSIS

To test the algorithm, we have considered 10 benchmark functions from CEC 2005 problem-set [22]. Then, we have compared DE-PM-ABC with basic ABC strategy proposed by Karaboga [1, 5] and modified ABC (MABC), proposed by Liu et al [27]. The 10 benchmark functions are described in Table 1. The function 1-4 is unimodal and rests are multi modal functions. The f9 and f10 is a expanded multi modal function.

The algorithms are tested on Intel® Pentium(R) CPU B960 @ 2.20GHz × 2 processor with 4 GB DDR3 RAM. The operating system platform is Ubuntu 12.04, 32 bit. The programming language is MATLAB R2012a.

For the basic ABC algorithm, the number of bees (i.e. number of population) is `50` and the number of food source (i.e. number of employed bee or onlooker bee) is `25`. The parameter D is the dimension of the food source whose value is `10`. The number of employed bees (n_e) is same as the number of onlooker bees (n_o) which is `50%` of the total population. Max cycle is set to `D*1e+04`. The limit is set to n_e`*D*0.5`. The value of Φ is chosen between `[-1, 1]` with uniform distribution. For MABC **[27]**, mutation ratio (MR) is equal to `0.2`. In DE-PM-ABC, the polynomial distribution index (η_m) is `100` for polynomial distribution. The parameter MR is `0.80`. These three algorithms are run for `25` times for each of these functions. The results of these three algorithms on the function set of table are shown in Table 2. We have considered error `1*e-06` for function 1 – 3, and `1*e-02` for other functions as negligible. In `Table 2`, the best value, mean error and standard deviation of the results are shown for all three algorithms. Best value denotes the best result obtained in 25 runs for each function. The mean error demonstrates the average error obtained in 25 runs for each function. The standard deviation shows the stability of the algorithm for a function as it denotes the deviation of the results in obtained in 25 runs for each function. From the results of `Table 2`, it is clear that the DE-PM-ABC is very competitive with respect to the remaining two algorithms for most of these benchmark functions in all four criteria. The following `Figure 1 - 5` depicts the convergence rate of the three algorithms which is shown in '`log scale`'. However, these figures also show that the convergence rate of DE-PM-ABC is better than the rest two algorithms.

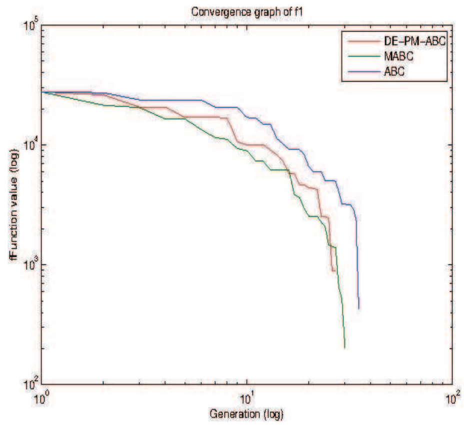

Figure 1: Convergence graph of function `f`$_1$

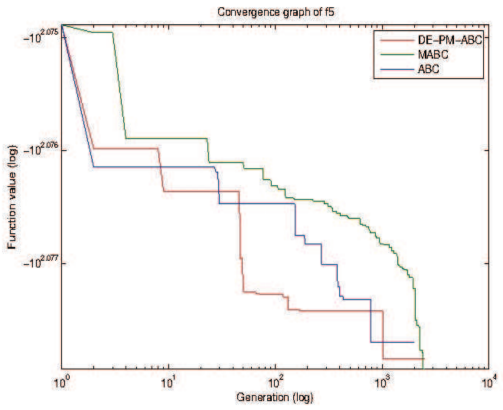

Figure 2: Convergence graph of function `f`$_5$

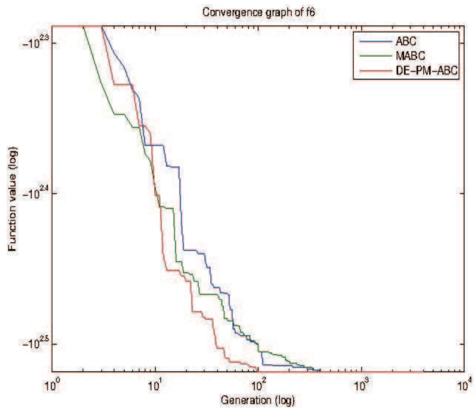

Figure 3: Convergence graph of function `f`$_6$

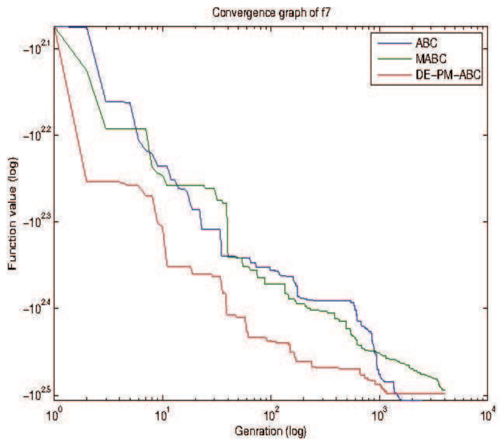

Figure 4: Convergence graph of function `f`$_7$

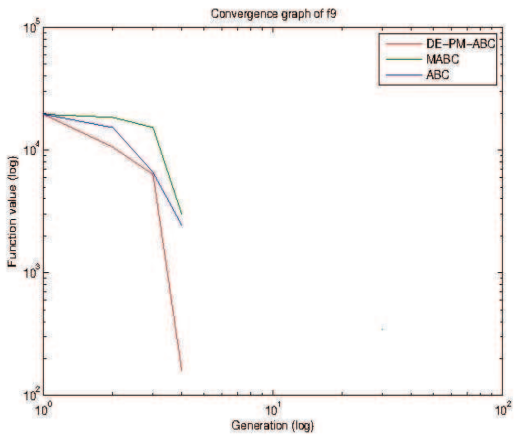

Figure 5: Convergence graph of function `f`$_9$

Table 1: Benchmark functions selected from CEC 2005 function sets

Function Definition	Dimension	Range	Optimum
$f_1(X)=\sum_{i=1}^{D} X_i^2+f_{bias}$; $X=X-o$;	10	[-100,100]	- 450
$f_2(X)=\sum_{i=1}^{D}\left(\sum_{j=1}^{i} X_j\right)^2+f_{bias}$; $X=X-o$	10	[-100,100]	- 450
$f_3(X)=(A_i X-B_i)+f_{bias}$; $\det(A)\neq 0, a_{ij}\in[-500,500], B_i=A_i*o, o_i\in[-100,100]$	10	[-100,100]	- 310
$f_4(X)=\sum_{i=1}^{D}\left(100\left(X_i^2-X_{i+1}\right)^2+\left(X_i-1\right)^2\right)+f_{bias}$; $X=(X-o+1)$	10	[-100,100]	390
$f_5(X)=-20e^{\left(0.2\sqrt{\left(\sum_{i=1}^{D} X_i^2/D\right)}\right)}-e^{\left(\sum_{i=1}^{D}\cos(2\pi X_i)/D\right)}+20+e+f_{bias}$; $X=(X-o)*M$	10	[-32,32]	-140
$f_6(X)=\sum_{i=1}^{D}\left(X_i^2-10\cos(2\pi X_i)+10\right)+f_{bias}$; $X=(X-o)$	10	[-5,5]	-330
$f_7(X)=\sum_{i=1}^{D}\left(X_i^2-10\cos(2\pi X_i)+10\right)+f_{bias}$; $X=(X-o)*M$	10	[-5,5]	-330
$f_8(X)=\sum_{i=1}^{D}\left(\sum_{k=o}^{k_{max}}\left[a_k\cos\left(2\pi b_k\left(X_i+0.5\right)\right)\right]\right)-D\sum\left[a_k\cos\left(2\pi b_k\times 0.5\right)\right]+f_{bias}$; $a=0.5$, $b=3$, $k_{max}=20$; $X=(X-o)*M$	10	[-0.5,0.5]	90
$f_9(X)=\sum_{i=1}^{D-1}F_G\left(F_R\left(X_i,X_{i+1}\right)\right)+F_G\left(F_R\left(X_D,X_1\right)\right)+f_{bias}$; $F_G=\sum_{i=1}^{D}X_i^2/4000-\prod_{i=1}^{D-1}\cos\left(X_i/\sqrt{i}\right)+1$, $F_G=\sum_{i=1}^{D}\left(100\left(X_i^2-X_{i+1}\right)^2+\left(X_i-1\right)^2\right)$;$X=X-o+1$	10	$[-\pi, \pi]$	-460
$f_{10}(X)=\sum_{i=1}^{D-1}G\left(X_i,X_{i+1}\right)+G\left(X_D,X_1\right)+f_{bias}$;$G(X,Y)=0.5+\left(\sin^2\sqrt{\left(X^2+Y^2\right)}-0.5\right)/\left(1+0.001\left(X^2+Y^2\right)\right)^2$;$X=(X-o)*M$	10	[-100,100]	-300

Table 2: The Best Value, Mean Error and Standard Deviation of ABC, MABC and DE-PM-ABC

Function	Algorithms	Best value	Mean Error	Std deviation
f_1	ABC	**-4.500000e+02**	3.424535e-07	2.541939e-07
	MABC	**-4.500000e+02**	1.679737e-07	1.634574e-07
	DE-PM-ABC	**-4.500000e+02**	**0.746875e-07**	**1.004682e-07**
f_2	ABC	-4.460094e+02	1.485044e+02	5.494593e+01
	MABC	-4.509998e+02	1.539517e+01	4.007401e+00
	DE-PM-ABC	**-4.500000e+02**	**1.522740e+00**	**1.858761e+00**
f_3	ABC	-3.032140e+02	7.487511e+01	5.430259e+01
	MABC	-3.058662e+02	1.476858e+01	8.856474e+00
	DE-PM-ABC	**-3.107268e+02**	**1.074121e+01**	**7.467809e+00**
f_4	ABC	3.908337e+02	2.420477e+00	2.188215e+00
	MABC	3.901095e+02	1.107611e+00	**1.512197e+00**
	DE-PM-ABC	**3.900310e+02**	**1.025185e+00**	1.554276e+00
f_5	ABC	-1.197246e+02	2.032683e+01	8.446907e-02
	MABC	-1.197711e+02	2.032729e+01	7.814739e-02
	DE-PM-ABC	**-1.328411e+02**	**2.031046e+01**	**4.094885e-02**
f_6	ABC	-3.299971e+02	5.859473e-03	3.869127e-03
	MABC	-3.309994e+02	5.604473e-03	3.134349e-03
	DE-PM-ABC	**-3.300099e+02**	**3.960360e-03**	**2.641545e-03**
f_7	ABC	-3.158970e+02	2.337617e+01	8.970598e+00
	MABC	-3.207792e+02	1.775862e+01	3.433692e+00
	DE-PM-ABC	**-3.300034e+02**	**1.762412e+01**	**2.354493e+00**
f_8	ABC	9.522356e+01	5.691881e+00	8.044037e-01
	MABC	9.489143e+01	5.657784e+00	7.175481e-01
	DE-PM-ABC	**9.418693e+01**	**5.193835e+00**	**4.796464e-01**
f_9	ABC	-1.299876e+02	2.305121e-01	8.592573e-02
	MABC	-1.298804e+02	2.312772e-02	2.847127e-02
	DE-PM-ABC	**-1.299992e+02**	**2.573974e-03**	**1.000220e-02**
f_{10}	ABC	-2.970434e+02	3.482768e+00	1.857258e-01
	MABC	-2.966374e+02	3.391965e+00	1.314412e-01
	DE-PM-ABC	**-2.966385e+02**	**3.354515e+00**	**4.120291e-02**