A NOVEL PLANT PROPAGATION ALGORITHM: MODIFICATIONS AND IMPLEMENTATION

Muhammad Sulaiman[1], **Abdel Salhi**[2], **Eric S Fraga**[3], **Wali Khan Mashwani**[4], **Muhammad M Rashidi**[5]

[1]Department of Mathematics, Abdul Wali Khan University, Mardan, KPK, Pakistan
[2]Department of Mathematical Sciences, University of Essex, Colchester, UK
[3]Centre for Process Systems Engineering, Department of Chemical Engineering, University College London (UCL), UK
[4] Department of Mathematics, Kohat University of Science & Technology (KUST), Pakistan.
[5]Shanghai Key Lab of Vehicle Aerodynamics and Vehicle Thermal Management Systems, Tongji University, 4800 Cao An Rd., Jiading, Shanghai 201804, China
E-mail: sulaiman513@yahoo.co.uk, mashwanigr8@gmail.com

ABSTRACT*: The Plant Propagation Algorithm, epitomised by the Strawberry Algorithm, has been previously successfully tested on low dimensional continuous Optimization problems. It is a neighbourhood search algorithm. In this paper, we introduce a robust and efficient version of the algorithm and explain how it can be implemented to compete with one of the best available alternatives, namely the Artificial Bee Colony algorithm and we present an improved and more effective variant on standard continuous optimization test problem instances in high dimensions. Computational and comparative results are included.*

Keywords: Optimization; Heuristics; Nature-Inspired Algorithms; Artificial Bee Colony Algorithm; Strawberry/Plant Propagation Algorithm

1. INTRODUCTION

Although there are already many good algorithms and heuristics for Optimization and search problems [23], the growing complexity of these problems in practice and the frequency with which they occur mean that new and more effective algorithms have to be developed. Note that frequently occurring problems may justify introducing new algorithms with only slight improvements [17, 15, 14, 16]. But, designing new algorithms which are easier to implement and require fewer arbitrarily set of parameters, for instance, is a worthwhile quest in itself. So, there are many reasons for trying to invent new algorithms.

As it happens, a lot of attempts at creating new algorithms look to Nature for inspiration. It seems that natural phenomena such as the survival of living entities and the success of some species in a given environment, rely on Optimization and search to overcome the constraints that these environments impose on them. The survival of species often depends on their ability to adapt, to find food quickly, avoid predation and give their off-spring the best chance to survive and thrive. These are typically Optimization/search problems.

The Plant Propagation Algorithm (PPA) is a Nature-inspired algorithm, [24, 3, 21, 18], for Optimization and search. It emulates the way plants, in particular the strawberry plant, propagate. Basic PPA has been described and tested on single objective Optimization problems [18, 21]. Although the problems considered were of low dimensions, it was established that the algorithm has merits and deserves further investigation and testing on higher dimension instances. The attraction of the algorithm is its simplicity and the relatively small number of parameters requiring arbitrary setting.

This paper addresses the issue of testing the Modified Plant Propagation Algorithm (MPPA) on larger, higher dimension problems, and compares the algorithm to its basic version

PPA, and two other Nature-inspired ones specifically the Artificial Bee Colony (ABC) algorithm [10] and the Modified Artificial Bee Colony (MABC) algorithm [2, 5]. The PPA is presented both in its original and modified forms. Extensive comparative results on high-dimensional test instances are reported and discussed. The paper ends with a conclusion and further issues for consideration.

2. THE ARTIFICIAL BEE COLONY ALGORITHM

The Artificial Bee Colony algorithm proposed in [10], simulates the foraging behaviour of bees living in a colony. Three groups of bees participate in the foraging process: worker bees, onlooker bees and scout bees. The majority of the population is composed of worker bees and onlooker bees. The scouts are recruited from worker bees. Algorithm 1 below describes the ABC algorithm.

2.1 THE MODIFIED ARTIFICIAL BEE COLONY ALGORITHM

The MABC algorithm has been suggested in [5]. The new method improves upon the exploitation aspect. MABC uses Differential Evolution [13] in step 4 of the ABC algorithm, and removes step 5 (or the Scouts phase). MABC has a good performance on continuous unconstrained Optimization, it is a hybrid algorithm of ABC. It is, at the moment, the algorithm to beat among a large selection of benchmark functions

[5].

3. THE STRAWBERRY ALGORITHM

PPA as the Strawberry algorithm [18, 20, 21], is a neighbour-hood search algorithm and a population-based metaheuristic. However, it can be seen as a multi-path following algorithm unlike Simulated Annealing (SA) [1, 19, 21], for instance, which is a single path following algorithm.

Exploration and exploitation are the two main properties global Optimization algorithms ought to have [24, 23, 3, 22]. Exploration refers to the property of covering the whole search space, while exploitation refers to the property of searching for local optima, near good solutions. Effective global Optimization methods exhibit both properties.

Consider what a strawberry plant, and possibly any plant which propagates through runners, will do to maximise its chances of survival. If it is in a good spot in the ground, with enough water, nutrients and light, then it is reasonable to assume that there is no pressure on it to leave that spot to guarantee its survival. So, it will send many short runners that will give new strawberry plants and occupy the neighbourhood as best they can. If, on the other hand, the mother plant is in a spot that is poor in water, nutrients, light or any element necessary for a plant to survive, then it will try to find a better spot for its off-spring. Therefore, it will send a few runners further afield to explore distant neighbourhoods. One can also assume that it will send only a few, since sending a long runner is a big investment for a plant which is in a poor spot. We may further assume that the quality of the spot (abundance of nutrients, water and light) is reflected in the growth of the plant. With this in mind, and the following notation, PPA can be described as follows.

A plant p_i is in spot X_i in dimension n. This means $X_i =$ $[x_{i,j}]$, for $j = 1, ..., n$. In PPA, exploitation is implemented through sending of many short runners by plants in good spots. Exploration is implemented by sending few long runners by plants in poor spots; the long runners allow distant neighbourhoods to be explored.

The parameters used in PPA are the population size NP which is the number of strawberry plants to start with, the maximum number o generations g_{max}, and the maximum number of possible runners n_{max} per plant. g_{max} is effectively the stopping criterion in this initial version of PPA. The algorithm uses the objective function value at different plant positions X_i , $i = 1, ..., NP$ to rank them as would a fitness function in genetic algorithms, [8].

Let $N_i \in (0, 1)$ be the normalised objective function value for X_i. The number of plant runners, n_i for this solution is given by

$$n_r^i = \lceil n_{max} N_i \alpha_i \rceil, \qquad (1)$$

where $\alpha_i \in (0, 1)$ is a randomly generated number. Every solution generates at least one runner. Each runner generated has a distance, $dx_i \in [-1, 1]^n$, calculated by

$$dx_j^i = 2(1-N_i)(r-0.5), \text{ for } j = 1,\ldots,n, \qquad (2)$$

where $r \in [0, 1]$ is also randomly generated. We note that the number of runners is proportional to the fitness, i.e. the value of the objective function, whereas the distance is inversely proportional to it.

Having calculated the n-dimensional vector dx_i , the new point to explore, $Y_i = [y_{i,j}]$, for $j = 1, ..., n$, is given by

$$y_{i,j} = x_{i,j} + (b_j - a_j)dx_j^i, \text{ for } j = 1,\ldots,n, \qquad (3)$$

where a_j and b_j are the lower and upper bounds of the search domain, respectively. Note that, if the bounds of the search domain are violated, the point is adjusted to be within the domain.After all individuals/plants in the population have sent out their allocated runners, new plants are evaluated and the whole increased population is sorted. To keep the population size constant, individuals with lower fitness are eliminated.

3 MODIFIED PLANT PROPAGATION ALGORITHM (MPPA)

The modifications with respect to the previous version of PPA as in Algorithm 2, [18, 21], concern the strategy for generating runners, whether they should be short or long, and whether the new points resulting from these runners are retained or not. Also, while in Algorithm 2 the number of runners is varying between 1 and n_{max}, in the modified version it is assumed to be a fixed number, although all generated runners may not lead to points that will make it into the new population. This is because, after sorting, their rank may be above NP, the population size.

3.1 AN ALTERNATIVE IMPLEMENTATION OF THE PROPAGATION PHASE

The population is initialized randomly by

$$x_{i,j} = a_j + (b_j - a_j)\alpha_j, \ j = 1,\ldots,n, \qquad (4)$$

Algorithm 1 : Artificial Bee Colony (ABC) algorithm, [10]

```
1: Generate random food sources for all worker bees;
2: while Stopping criterion is not satisfied do
3:     Each worker bee goes to its assigned food source,
       finds a neighbouring source, evaluates it and
       displays the information to onlooker bees in the hive,
       through a dance;
4:     Each onlooker bee, after watching worker bees
       dancing, selects one of the sources and goes to it.
       It then finds a neighbouring source and evaluates it.
5:     Food sources that are not good are abandoned and
       replaced by new ones discovered by worker bees that
       have become scouts.
6:     Record the best food source found so far.
7: end while
8: return Best solution
```

Algorithm 2 : Plant Propagation Algorithm (PPA), [18]

```
1: Generate a random population $P = \{X_i| , i = 1,\ldots,NP\}$;
2: while Stopping criterion is not satisfied do
3:     Compute $N_i = Nomalized(f(X_i)), \forall X_i \in P, N_i \in (0,1)$;
4:     Sort $P$ in ascending order of fitness values $N_i$ (for mini-
       mization);
5:     Create new population $\Phi$;
6:     for each $X_i, i = 1,\ldots,NP$ do
7:         $r_i \leftarrow$ Generate set of runners where both the size of the
           set and the distance for each runner (individually) are
           proportional to the fitness values $N_i$;
8:         $\Phi \leftarrow \Phi \cup r_i$ {append the set of runners to population;}
9:     end for
10:    $P \leftarrow \Phi$ {new population};
11:    Rank the population $P$ and omit the individuals having
       higher rank than $NP$
12: end while
13: return Best solution
```

where $\alpha_j \in (0, 1)$ is a randomly generated real number for each j. After the population is initialized, MPPA proceeds to generate for every member in the population a number nr of runners; nr is assumed to be a fixed constant. These runners lead to new solutions as per the equations (5-7), on the off chance that the limits of the search area are maltreated, the coordinates are conformed respectively to be inside the search space [9].

$$y_{i,j} = x_{i,j} + \beta_j x_{i,j}, \ j = 1,\ldots,n, \qquad (5)$$

where $\beta_j \in [-1, 1]$ is a randomly generated number for each j. The term $\beta_j \cdot x_{i,j}$ is the length with respect to the j^{th} coordinate of the runner, and $y_{i,j} \in [a_j , b_j]$. If the bounds of the search domain are violated, the point is adjusted to be within the domain. The generated individual Y is evaluated according to the objective function and is stored in Φ, equation (5) helps in exploring the neighbourhood of $x_{i,j}$. As the search becomes refined, that is the algorithm is in exploitation mode then the coordinates produced by equation

(5) are smaller and smaller [9]. This is represented in Figure (2), where the horizontal axis shows the total number of perturbations produced during 30 independent experiments. Note that in beginning of each experiment the step size is larger and as the search is refined the step size decreases gradually, in the latter case the algorithm is in exploitation mode. In Algorithm 3, If this newly created solution, by equation (5), is not improving the objective function, then another individual is created with a runner based on equation (6), [9]. The number of runners of a certain length generated by equation (6), when solving f_2 , is shown in Figure (1), this shows the frequency of exploration for the optimum solution.

$$y_{i,j} = x_{i,j} + \beta_j b_j,\ j = 1,\dots,n, \qquad (6)$$

where b_j is the j^{th} upper bound and here again $y_{i,j} \in [a_j, b_j]$. This can be considered as a solution at the end of a long runner. Again, if the generated individual does not improve the

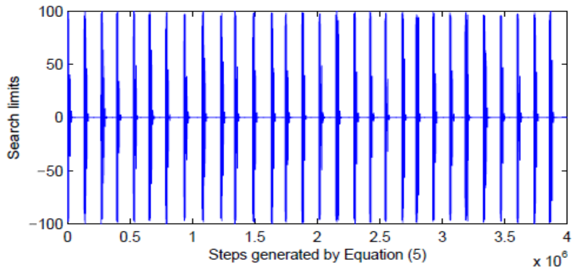


Figure 1. Performance of equation (5) in solving f_2

objective value, another runner is created by equation (7),

$$y_{i,j} = x_{i,j} + \beta_j a_j,\ j = 1,\dots,n, \qquad (7)$$

where a_j is the j^{th} lower bound and $y_{i,j} \in [a_j, b_j]$. This can be considered as a solution at the end of a long runner. The number of runners of a certain length generated by equation (7), when solving f_2 , is shown in Figure (3), this shows the frequency of exploration for the optimum solution.

Equations (5-7), are implemented by Algorithm 3 turn by turn if any of the search equation fails in improving the current solution [9]. MPPA improves the balance between exploration and exploitation of the search space as depicted in Figures (1-3). Note that the above equations may lead to infeasibility. In these situations, the offending entry is set by default to the boundary, lower or upper as per the concerned equation. To keep the size of the population constant, the plants with ranks > NP after sorting, are eliminated.

4. EXPERIMENTAL STUDIES

MPPA has been applied to a set of 18 benchmark functions of dimensions D = 30, 60 and 100, as shown in Tables 1-3. The set of experiments are carried out using n * 5000 function evaluations, where n is the dimension of the given test problem. The population size is NP = 75, as used in [5]. We try to match the number of evaluations carried out in tests in [5]. Note that although MPPA generates more points per iteration than MABC, it runs for less generations. In this way the number of function evaluation is kept the same for both algorithms. One, therefore, can talk of a fair comparison although some readers may find the concept of "fair comparison" hard to achieve in empirical studies on algorithms. The solution quality is listed in terms of best, worst, median, mean and standard deviation of the objective values found by each algorithm over 30 independent runs, as shown in Tables 5-7.

For functions 1, 2, 3, 5, 7, through 13, 16, and 17, MPPA found the optimum solutions while MABC did not. For functions 4 both algorithms found the optimum. For functions 6, 15, and 18 MPPA generated better solutions in terms of quality than MABC, although these solutions are suboptimal. Only for function 14 did MPPA generate a solution of lower quality than that found by MABC.

It is also important to note that MPPA outperformed ABC on all functions except Function 14, in dimension 30. Note that, at least for the time being, it is not necessary to compare with other algorithms such as the Genetic Algorithm (GA) [8, 7], Particle Swarm Optimization (PSO) [4], Differential Evolution (DE) [13], Harmony Search algorithm (HS) [6] and others, since all of these have been outperformed by ABC as reported in [11, 12]. MPPA is compared to its basic version PPA, as shown in Table 4, the approximate error in solutions, points to the superiority of MPPA.

5 CONCLUSION

Optimization problems are becoming more and more complex and unavoidable in most human activities. Although a variety of algorithms and heuristics to deal with them have been developed, new approaches are needed as the size and complexity of these problems increase. In recent years, heuristics and in particular those inspired by Nature, are becoming more efficient and robust. We have designed a Nature- inspired algorithm based on the way plants and in particular the strawberry plant, propagate. The original PPA algorithm has been only summarily tested on low dimension problems to establish its credentials. Surprisingly, despite being very simple and requiring few parameters, it managed to solve those problems rather well, [18]. In this paper, we have presented a modified version of PPA, which is referred to as MPPA. The improvements concern the way new solutions at the end of runners are calculated, i.e. Equations (5-7). The resulting algorithm has been tested on a more extensive test bench with a large number of functions having interesting characteristics such as multimodality and non separability in high dimensions, up to a 100. The results show that MPPA outperforms ABC and its more robust modification MABC on most of the test functions. In conclusion, MPPA provides us with a robust, easy to implement method for nonlinear, non-convex high dimensional optimization problems.

ACKNOWLEDGMENTS

This work is supported by Abdul Wali Khan University, Mardan, Pakistan, Grant No. F.16-5/ P& D/ AWKUM /238.

Algorithm 3 : Modified Plant Propagation Algorithm (MPPA)

```
1:  Create a random population of plants
    $pop = \{X_i \mid i = 1,2,...,NP\}$, $f \leftarrow$ Objective function ,
2:  Evaluate the population,
3:  Assume a fixed number of runners as $n_r = 3$,
4:  while The stopping criteria is not reached do
5:      Create $\Phi$;
6:      for all plants $i = 1$ to $NP$ do
7:          $\Phi_i = X_i$;
8:          for $k = 1$ to $n_r$ do
9:              Generate a new solution $Y$ according to
                equation (5);
10:             Evaluate it and store it in $\Phi$;
11:             Calculate $diff = |f(Y)| - |f(X_i)|$;
12:             if diff$\geq 0$ then
13:                 Generate a new solution Y according
                    to equation (6);
14:                 Evaluate it and store it in $\Phi$;
15:                 Compute $diff = |f(Y)| - |f(X_i)|$;
16:                 if diff$\geq$0 then
17:                     Generate a new runner using
                        equation (7);
18:                     Evaluate it and store it in $\Phi$;
19:                 end if
20:             end if
21:         end for
22:     end for
23:     Evaluate and append $\Phi$ to current population $pop$;
24:     Sort the population in ascending order of the objective
        values and omit the plants with rank$> NP$;
25:     Update current best;
26: end while
27: return Best soulution
```

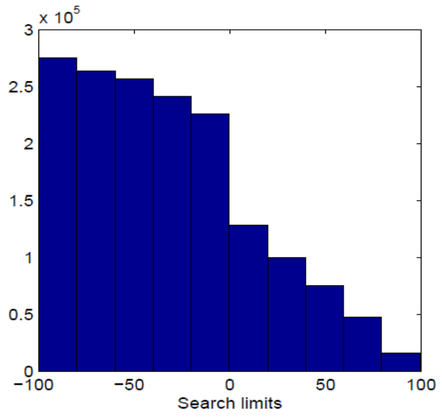


Figure 3. Performance of equation (7) in solving f_2

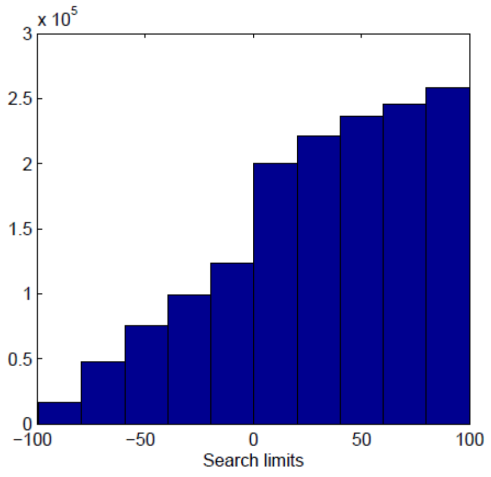


Figure 2. Performance of equation (6) in solving f_2

Table 1. Unimodal (U) and Separable (S) Test functions

Ftn No.	Range	Dim	Function	Formulation	Min
1	[-100, 100]	[30,60,100]	Sphere	$f(x) = \sum_{i=1}^{n} x_i^2$	0
2	[-100, 100]	[30,60,100]	Elliptic	$f(x) = \sum_{i=1}^{n} (10^6)^{\frac{(i-1)}{(n-1)}} x_i^2$	0
3	[-10, 10]	[30,60,100]	Axis Parallel Hyperellipsoid	$f(x) = \sum_{i=1}^{n} \lvert x_i \rvert^{(i+1)}$	0
4	[-100, 100]	[30,60,100]	Step	$f(x) = \sum_{i=1}^{n} (\lfloor x_i + 0.5 \rfloor)^2$	0
5	[-1.28, 1.28]	[30,60,100]	De Jong's 4 (no noise)	$f(x) = \sum_{i=1}^{n} i x_i^4$	0
6	[-1.28, 1.28]	[30,60,100]	Quartic (noise)	$f(x) = \sum_{i=1}^{n} i x_i^4 + random[0,1)$	0

Table 2. Unimodal (U) and Non-separable (N) Test functions

Ftn No.	Range	Dim	Function	Formulation	Min
7	[-10, 10]	[30,60,100]	Sum of Different Powers	$f(x)=\sum_{i=1}^{n} ix_i^2$	0
8	[-10, 10]	[30,60,100]	Schwefel's Problem 2.22	$f(x)=\sum_{i=1}^{n}\lvert x_i\rvert+\prod_{i=1}^{n}\lvert x_i\rvert$	0
9	[-100, 100]	[30,60,100]	Schwefel's Problem 2.21	$f(x)=max_i\{\lvert x_i\rvert, 1\le i\le n\}$	0
10	[-10, 10]	[30,60,100]	Rosenbrock	$f(x)=\sum_{i=1}^{n-1}[100(x_{i+1}-x_i^2)^2+(x_i-1)^2]$	0

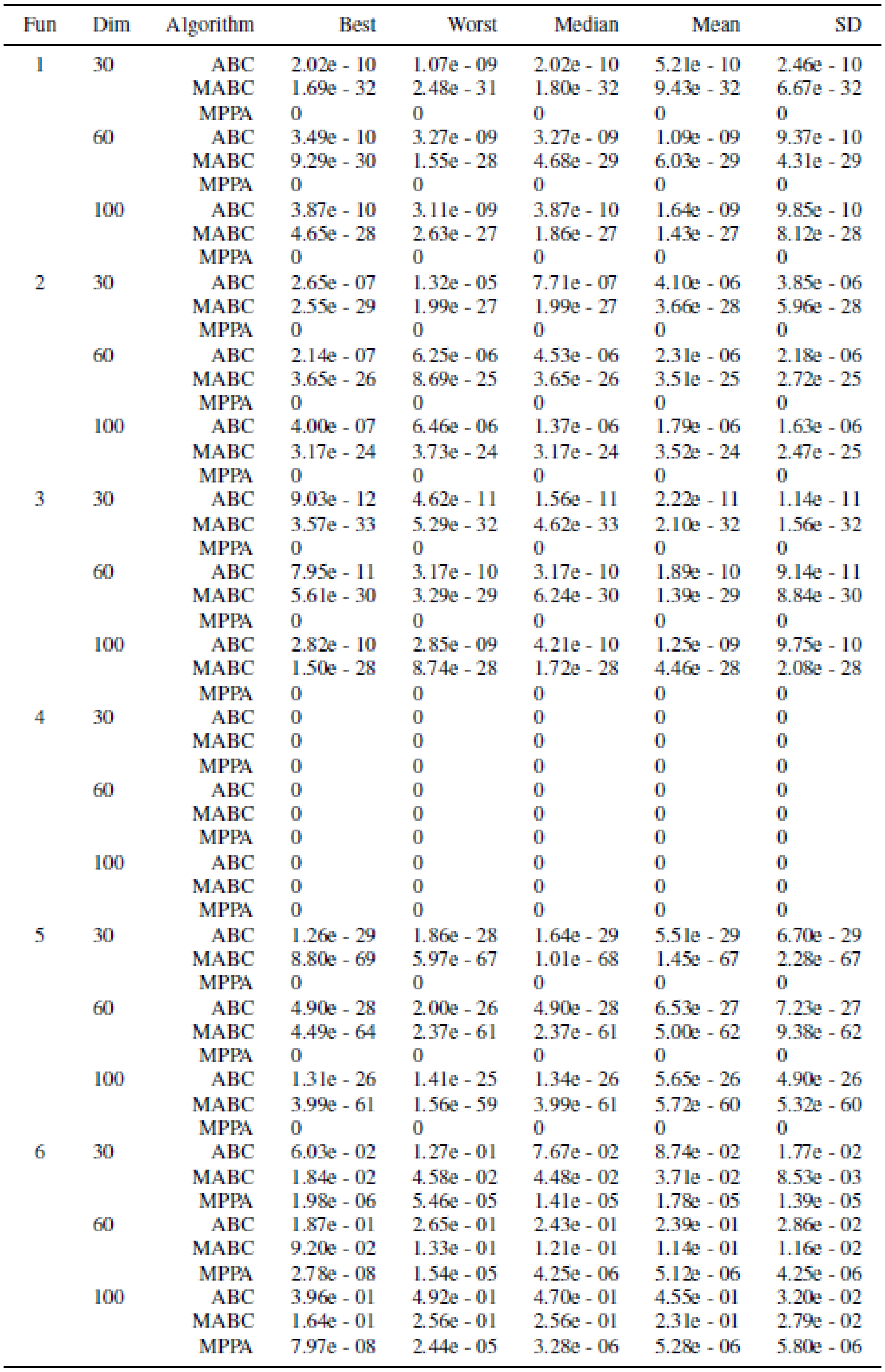
Table 3. Multimodal (M), Separable (S)/Non-separable (N) Test functions ;

Ftn No.	Range	D	Type	Function	Formulation	Min
11	[-5.12, 5.12]	[30,60,100]	MS	Rastrigin	$f(x)=[x_i^2-10\cos(2\pi x_i)+10]$	0
12	[-5.12, 5.12]	[30,60,100]	M	Non-Continuous	$f(x)=[y_i^2-10\cos(2\pi y_i)+10]$	0
				Rastrigin	$y_i=\begin{cases} x_i & \lvert x_i\rvert<\frac{1}{2} \\ \frac{round(2x_i)}{2}; & \lvert x_i\rvert\ge\frac{1}{2}\end{cases}$	
13	[-600, 600]	[30,60,100]	MN	Griewank	$f(x)=\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}})+1$	0
14	[-500, 500]	[30,60,100]	MS	Schwefel	$f(x)=418.98288727243369*n-\sum_{i=1}^{n}x_i\sin(\sqrt{\lvert x_i\rvert})$	0
15	[-32, 32]	[30,60,100]	MN	Ackeley's Path	$f(x)=-20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2})-\exp(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i))+20+e$	0
16	[-10, 10]	[30,60,100]	M	Alpine	$f(x)=\sum_{i=1}^{n}\mid x_i\cdot\sin(x_i)+0.1\cdot x_i\mid$	0
17	[-0.5, 0.5]	[30,60,100]	M	Weierstrass	$f(x)=\sum_{i=1}^{n}(\sum_{k=0}^{kmax}[a^k\cos(2\pi b^k(x_i+0.5))])$	
					$-D\sum_{k=0}^{kmax}[a^k\cos(2\pi b^k(0.5)], a=0.5, b=3, k_{max}=20.$	0
18	[-100, 100]	[30,60,100]	MN	Schaffer	$f(x)=0.5+\frac{\sin^2(\sqrt{\sum_{i=1}^{n}x_i^2})-0.5}{(1+0.001(\sum_{i=1}^{n}x_i^2))^2}$	0

Table 4. Summary of Results Obtained with the PPA [18] and MPPA Methods. Experiments were Repeated 10 Times for Each Problem and Solutions Obtained were Compared with PPA [18]. a and b are the Lower and Upper Bounds on X, the Decision variables.

Function	Range	Dim	PPA	MPPA	True value	$Error_{PPA}$	$Error_{MPPA}$
Six hump camel back	[-3, 2]	2	-1.0316	-1.0316	-1.0316	0	0
Branin	[-5, 15]	2	0.3980	0.3980	0.397887	1.13e-04	1.13e-04
Easom	[-100, 100]	2	-0.9997	-0.9997	-1	3.0e-04	3.0e-04
Goldstein Price	[-2, 2]	2	3.00536	3	3	5.36e-03	0
Martin Gaddy	[-20, 20]	2	4.4749e-08	0	0	4.4749e-08	0
Rastrigin	[-10, 10]	2	9.2e-03	0	0	9.2e-03	0
Rosenbrock	[-5, 10]	2	2.0e-04	0	0	2.0e-04	0
Schwefel	[-500, 500]	2	-833.203	-837.9603	-837.9658	4.76	5.5e-03

Table 5. Results Obtained by MPPA, ABC and MABC on test functions of Table 1.



Fun	Dim	Algorithm	Best	Worst	Median	Mean	SD
1	30	ABC	2.02e - 10	1.07e - 09	2.02e - 10	5.21e - 10	2.46e - 10
		MABC	1.69e - 32	2.48e - 31	1.80e - 32	9.43e - 32	6.67e - 32
		MPPA	0	0	0	0	0
	60	ABC	3.49e - 10	3.27e - 09	3.27e - 09	1.09e - 09	9.37e - 10
		MABC	9.29e - 30	1.55e - 28	4.68e - 29	6.03e - 29	4.31e - 29
		MPPA	0	0	0	0	0
	100	ABC	3.87e - 10	3.11e - 09	3.87e - 10	1.64e - 09	9.85e - 10
		MABC	4.65e - 28	2.63e - 27	1.86e - 27	1.43e - 27	8.12e - 28
		MPPA	0	0	0	0	0
2	30	ABC	2.65e - 07	1.32e - 05	7.71e - 07	4.10e - 06	3.85e - 06
		MABC	2.55e - 29	1.99e - 27	1.99e - 27	3.66e - 28	5.96e - 28
		MPPA	0	0	0	0	0
	60	ABC	2.14e - 07	6.25e - 06	4.53e - 06	2.31e - 06	2.18e - 06
		MABC	3.65e - 26	8.69e - 25	3.65e - 26	3.51e - 25	2.72e - 25
		MPPA	0	0	0	0	0
	100	ABC	4.00e - 07	6.46e - 06	1.37e - 06	1.79e - 06	1.63e - 06
		MABC	3.17e - 24	3.73e - 24	3.17e - 24	3.52e - 24	2.47e - 25
		MPPA	0	0	0	0	0
3	30	ABC	9.03e - 12	4.62e - 11	1.56e - 11	2.22e - 11	1.14e - 11
		MABC	3.57e - 33	5.29e - 32	4.62e - 33	2.10e - 32	1.56e - 32
		MPPA	0	0	0	0	0
	60	ABC	7.95e - 11	3.17e - 10	3.17e - 10	1.89e - 10	9.14e - 11
		MABC	5.61e - 30	3.29e - 29	6.24e - 30	1.39e - 29	8.84e - 30
		MPPA	0	0	0	0	0
	100	ABC	2.82e - 10	2.85e - 09	4.21e - 10	1.25e - 09	9.75e - 10
		MABC	1.50e - 28	8.74e - 28	1.72e - 28	4.46e - 28	2.08e - 28
		MPPA	0	0	0	0	0
4	30	ABC	0	0	0	0	0
		MABC	0	0	0	0	0
		MPPA	0	0	0	0	0
	60	ABC	0	0	0	0	0
		MABC	0	0	0	0	0
		MPPA	0	0	0	0	0
	100	ABC	0	0	0	0	0
		MABC	0	0	0	0	0
		MPPA	0	0	0	0	0
5	30	ABC	1.26e - 29	1.86e - 28	1.64e - 29	5.51e - 29	6.70e - 29
		MABC	8.80e - 69	5.97e - 67	1.01e - 68	1.45e - 67	2.28e - 67
		MPPA	0	0	0	0	0
	60	ABC	4.90e - 28	2.00e - 26	4.90e - 28	6.53e - 27	7.23e - 27
		MABC	4.49e - 64	2.37e - 61	2.37e - 61	5.00e - 62	9.38e - 62
		MPPA	0	0	0	0	0
	100	ABC	1.31e - 26	1.41e - 25	1.34e - 26	5.65e - 26	4.90e - 26
		MABC	3.99e - 61	1.56e - 59	3.99e - 61	5.72e - 60	5.32e - 60
		MPPA	0	0	0	0	0
6	30	ABC	6.03e - 02	1.27e - 01	7.67e - 02	8.74e - 02	1.77e - 02
		MABC	1.84e - 02	4.58e - 02	4.48e - 02	3.71e - 02	8.53e - 03
		MPPA	1.98e - 06	5.46e - 05	1.41e - 05	1.78e - 05	1.39e - 05
	60	ABC	1.87e - 01	2.65e - 01	2.43e - 01	2.39e - 01	2.86e - 02
		MABC	9.20e - 02	1.33e - 01	1.21e - 01	1.14e - 01	1.16e - 02
		MPPA	2.78e - 08	1.54e - 05	4.25e - 06	5.12e - 06	4.25e - 06
	100	ABC	3.96e - 01	4.92e - 01	4.70e - 01	4.55e - 01	3.20e - 02
		MABC	1.64e - 01	2.56e - 01	2.56e - 01	2.31e - 01	2.79e - 02
		MPPA	7.97e - 08	2.44e - 05	3.28e - 06	5.28e - 06	5.80e - 06

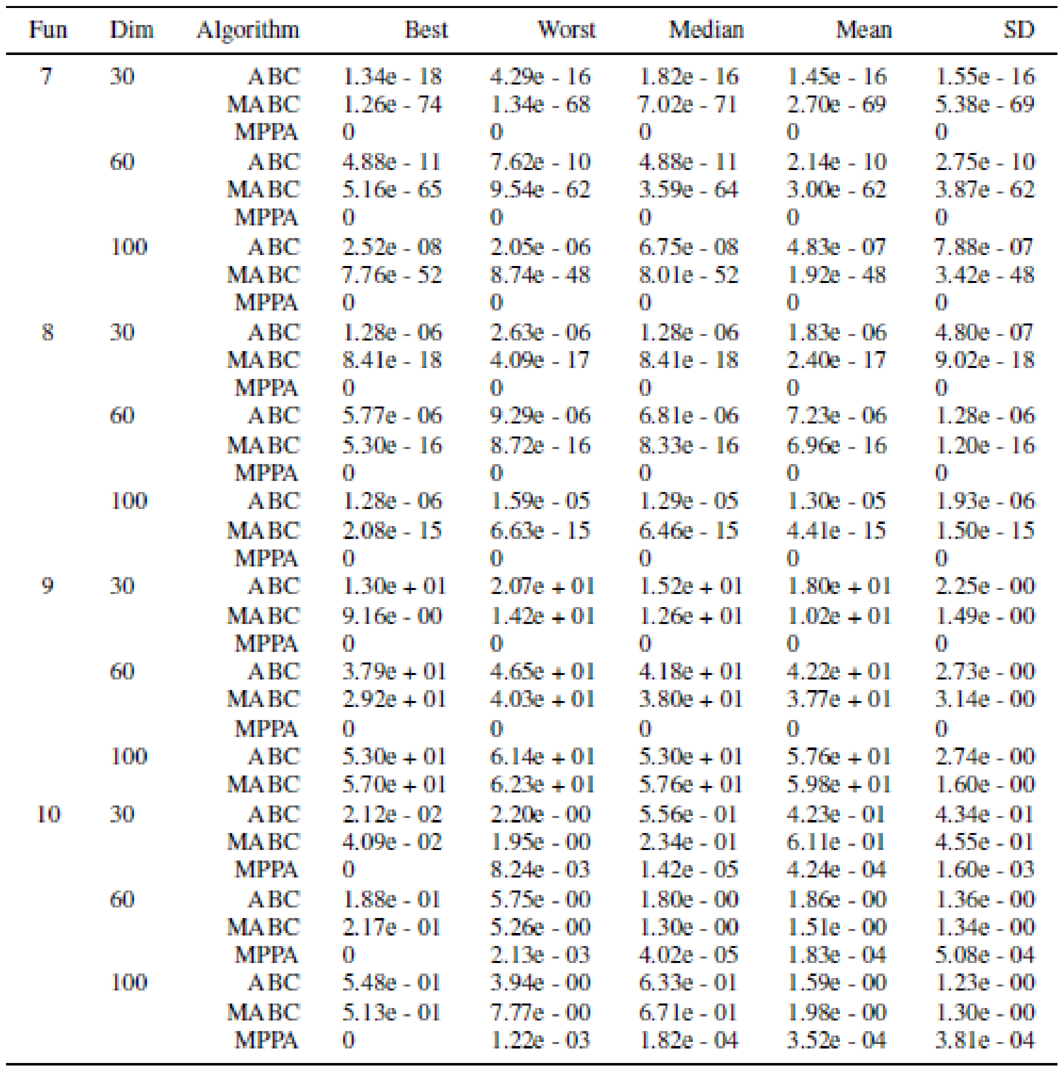


Table 6. Results Obtained by MPPA, ABC and MABC on test functions of Table 2.

Fun	Dim	Algorithm	Best	Worst	Median	Mean	SD
7	30	ABC	1.34e - 18	4.29e - 16	1.82e - 16	1.45e - 16	1.55e - 16
		MABC	1.26e - 74	1.34e - 68	7.02e - 71	2.70e - 69	5.38e - 69
		MPPA	0	0	0	0	0
	60	ABC	4.88e - 11	7.62e - 10	4.88e - 11	2.14e - 10	2.75e - 10
		MABC	5.16e - 65	9.54e - 62	3.59e - 64	3.00e - 62	3.87e - 62
		MPPA	0	0	0	0	0
	100	ABC	2.52e - 08	2.05e - 06	6.75e - 08	4.83e - 07	7.88e - 07
		MABC	7.76e - 52	8.74e - 48	8.01e - 52	1.92e - 48	3.42e - 48
		MPPA	0	0	0	0	0
8	30	ABC	1.28e - 06	2.63e - 06	1.28e - 06	1.83e - 06	4.80e - 07
		MABC	8.41e - 18	4.09e - 17	8.41e - 18	2.40e - 17	9.02e - 18
		MPPA	0	0	0	0	0
	60	ABC	5.77e - 06	9.29e - 06	6.81e - 06	7.23e - 06	1.28e - 06
		MABC	5.30e - 16	8.72e - 16	8.33e - 16	6.96e - 16	1.20e - 16
		MPPA	0	0	0	0	0
	100	ABC	1.28e - 06	1.59e - 05	1.29e - 05	1.30e - 05	1.93e - 06
		MABC	2.08e - 15	6.63e - 15	6.46e - 15	4.41e - 15	1.50e - 15
		MPPA	0	0	0	0	0
9	30	ABC	1.30e + 01	2.07e + 01	1.52e + 01	1.80e + 01	2.25e - 00
		MABC	9.16e - 00	1.42e + 01	1.26e + 01	1.02e + 01	1.49e - 00
		MPPA	0	0	0	0	0
	60	ABC	3.79e + 01	4.65e + 01	4.18e + 01	4.22e + 01	2.73e - 00
		MABC	2.92e + 01	4.03e + 01	3.80e + 01	3.77e + 01	3.14e - 00
		MPPA	0	0	0	0	0
	100	ABC	5.30e + 01	6.14e + 01	5.30e + 01	5.76e + 01	2.74e - 00
		MABC	5.70e + 01	6.23e + 01	5.76e + 01	5.98e + 01	1.60e - 00
10	30	ABC	2.12e - 02	2.20e - 00	5.56e - 01	4.23e - 01	4.34e - 01
		MABC	4.09e - 02	1.95e - 00	2.34e - 01	6.11e - 01	4.55e - 01
		MPPA	0	8.24e - 03	1.42e - 05	4.24e - 04	1.60e - 03
	60	ABC	1.88e - 01	5.75e - 00	1.80e - 00	1.86e - 00	1.36e - 00
		MABC	2.17e - 01	5.26e - 00	1.30e - 00	1.51e - 00	1.34e - 00
		MPPA	0	2.13e - 03	4.02e - 05	1.83e - 04	5.08e - 04
	100	ABC	5.48e - 01	3.94e - 00	6.33e - 01	1.59e - 00	1.23e - 00
		MABC	5.13e - 01	7.77e - 00	6.71e - 01	1.98e - 00	1.30e - 00
		MPPA	0	1.22e - 03	1.82e - 04	3.52e - 04	3.81e - 04

Table 7. Results Obtained by MPPA, ABC and MABC on test functions of Table 3.

Fun	Dim	Algorithm	Best	Worst	Median	Mean	SD
11	30	ABC	3.58e - 10	1.46e - 01	5.50e - 10	4.81e - 03	2.57e - 02
		MABC	0	0	0	0	0
		MPPA	0	0	0	0	0
	60	ABC	3.21e - 10	1.99e - 00	1.28e - 05	3.71e - 01	5.97e - 01
		MABC	0	0	0	0	0
		MPPA	0	0	0	0	0
	100	ABC	5.40e - 09	1.99e - 00	1.99e - 00	1.10e - 00	8.21e - 01
		MABC	0	0	0	0	0
		MPPA	0	0	0	0	0
12	30	ABC	8.96e - 10	1.00e - 00	1.77e - 08	1.12e - 01	2.97e - 01
		MABC	0	0	0	0	0
		MPPA	0	0	0	0	0
	60	ABC	2.17e - 07	3.025e - 00	1.09e - 00	1.47e - 00	9.47e - 01
		MABC	0	0	0	0	0
		MPPA	0	0	0	0	0
	100	ABC	2.00e - 00	7.21e - 00	5.02e - 00	4.74e - 00	2.01e - 00
		MABC	0	0	0	0	0
		MPPA	0	0	0	0	0
13	30	ABC	4.74e - 12	1.35e - 07	1.77e - 08	1.61e - 08	3.99e - 08
		MABC	0	0	0	0	0
		MPPA	0	0	0	0	0
	60	ABC	7.99e - 12	1.05e - 09	1.45e - 11	1.39e - 10	3.10e - 10
		MABC	0	0	0	0	0
		MPPA	0	0	0	0	0
	100	ABC	6.29e - 13	3.05e - 08	1.01e - 10	2.01e - 09	1.32e - 09
		MABC	0	0	0	0	0
		MPPA	0	0	0	0	0
14	30	ABC	1.54e - 06	2.37e + 02	3.76e - 01	8.86e + 01	8.62e + 01
		MABC	- 1.81e - 12	0	0	- 1.21e - 13	4.53e - 13
		MPPA	3.82e - 04	3.55e + 03	5.36e - 04	3.55e + 02	1.08e + 03
	60	ABC	3.55e + 02	7.69e + 02	7.69e + 02	5.40e + 02	1.41e + 02
		MABC	2.91e - 11	3.63e - 11	2.91e - 11	3.56e - 11	2.18e - 12
		MPPA	7.63e - 04	7.10e +03	7.79e - 04	7.10e - 02	2.16e +03
	100	ABC	7.81e + 02	1.55e + 03	1.51e + 03	1.29e + 03	2.23e + 02
		MABC	1.09e - 10	1.23e - 10	1.16e - 10	1.19e - 10	4.06e - 12
		MPPA	1.27e - 03	1.18e +04	1.30e -03	7.89e +02	3.00e +03
15	30	ABC	2.26e - 06	8.32e - 06	7.17e - 06	4.83e - 06	2.12e - 06
		MABC	3.64e - 14	4.35e - 14	3.99e - 14	4.13e - 14	2.17e - 15
		MPPA	8.88e - 16	8.88e - 16	8.88e - 16	8.88e - 16	0
	60	ABC	2.44e - 06	1.57e - 05	2.44e - 06	7.79e - 06	3.63e - 06
		MABC	1.14e - 13	1.57e - 13	1.32e - 13	1.37e - 13	1.24e - 14
		MPPA	8.88e - 16	8.88e - 16	8.88e - 16	8.88e - 16	0
	100	ABC	5.12e - 06	1.38e - 05	5.94e - 06	1.02e - 05	2.92e - 06
		MABC	3.27e - 13	3.98e - 13	3.27e - 13	3.56e - 13	2.29e - 14
		MPPA	8.88e - 16	8.88e - 16	8.88e - 16	8.88e - 16	0
16	30	ABC	2.99e - 05	1.05e - 04	1.05e - 04	7.66e - 05	2.76e - 05
		MABC	3.74e - 18	6.54e - 16	1.48e - 17	1.58e - 16	2.48e - 16
		MPPA	0	0	0	0	0
	60	ABC	2.18e - 04	1.24e - 03	3.25e - 04	5.78e - 04	3.51e - 04
		MABC	2.55e - 16	1.90e - 15	4.45e - 16	8.20e - 16	4.69e - 16
		MPPA	0	0	0	0	0
	100	ABC	7.13e - 04	1.27e - 02	7.13e - 04	7.62e - 03	5.10e - 03
		MABC	2.38e - 15	9.68e - 15	7.76e - 15	5.83e - 15	1.97e - 15
		MPPA	0	0	0	0	0
17	30	ABC	1.38e - 01	1.62e - 01	1.39e - 01	1.46e - 01	1.09e - 02
		MABC	0	0	0	0	0
		MPPA	0	0	0	0	0
	60	ABC	1.87e - 01	3.51e - 01	2.93e - 01	2.77e - 01	6.77e - 02
		MABC	0	1.42e - 14	7.10e - 15	9.94e - 15	5.68e - 15
		MPPA	0	0	0	0	0
	100	ABC	6.66e - 00	7.48e - 00	7.26e - 00	7.07e - 00	4.08e - 00
		MABC	4.26e - 14	5.68e - 14	4.26e - 14	5.21e - 14	6.69e - 15
		MPPA	0	0	0	0	0
18	30	ABC	4.147e - 01	4.598e - 01	4.524e - 01	4.413e - 01	1.81e - 02
		MABC	2.277e - 01	3.455e - 01	3.121e - 01	2.952e - 01	3.17e - 02
		MPPA	3.14e - 03	9.57e - 02	3.36e - 02	3.81e - 02	2.65e - 02
	60	ABC	4.960e - 01	4.976e - 01	4.974e - 01	4.971e - 01	5.90e - 04
		MABC	4.796e - 01	4.903e - 01	4.850e - 01	4.840e - 01	3.62e - 03
		MPPA	9.29e - 03	1.98e - 01	5.13e - 02	6.84e - 02	4.71e - 02
	100	ABC	4.996e - 01	4.998e - 01	4.998e - 01	4.997e - 01	4.51e - 05
		MABC	4.988e - 01	4.992e - 01	4.991e - 01	4.990e - 01	1.75e - 04
		MPPA	0	1.77e - 01	7.85e - 02	8.79e - 02	5.33e - 02

REFERENCES

[1] E.H.L. Aarts, J.H.M. Korst, and P.J.M. Van Laarhoven. Simulated annealing. *Local search in combinatorial optimization*, pages 91–120, 1997.

[2] B. Akay and D. Karaboga. A modified artificial bee colony algorithm for real-parameter optimization. *Information Sciences*, 2010.

[3] J. Brownlee. *Clever algorithms: nature-inspired programming recipes*. Jason, B., 2011.

[4] M. Dorigo, L.M. Gambardella, M. Birattari, A. Martinoli, R. Poli, and T. Stützle. *Ant Colony Optimization and Swarm Intelligence: 5th International Workshop, ANTS 2006, Brussels, Belgium, September 4-7, 2006, Proceedings*, volume 4150. Springer, 2006.

[5] Wei-feng Gao and San-yang Liu. A modified artificial bee colony algorithm. *Computers & Operations Research*, 39(3):687–697, 2012.

[6] Z.W. Geem, J.H. Kim, and GV Loganathan. A new heuristic optimization algorithm: harmony search. *Simulation*, 76(2):60–68, 2001.

[7] D.E. Goldberg. *Genetic algorithms in search, optimization, and machine learning*. Addison-Wesley Professional, 1989.

[8] John H Holland. Adaption in natural and artificial systems. 1975.

[9] R. Hooke and FA Jeeves. Direct search solution of numerical and statistical problems. *Journal of the Association for Computer Machines*, 8, 1961.

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

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

[12] D. Karaboga and B. Basturk. On the performance of artificial bee colony (abc) algorithm. *Applied Soft Computing*, 8(1):687–697, 2008.

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

[14] M. A. Z. Raja, R. Samar, and M. M. Rashidi. Application of three unsupervised neural network models to singular nonlinear bvp of transformed 2d bratu equation. *Neural Computing and Applications*, 25(7-8):1585–1601, 2014.

[15] MM Rashidi, M Ali, N Freidoonimehr, and F Nazari. Parametric analysis and optimization of entropy generation in unsteady mhd flow over a stretching rotating disk using artificial neural network and particle swarm optimization algorithm. *Energy*, 55:497–510, 2013.

[16] MM Rashidi, O Anwar Bég, A Basiri Parsa, and F Nazari. Analysis and optimization of a transcritical power cycle with regenerator using artificial neural networks and genetic algorithms. *Proceedings of the Institution of Mechanical Engineers, Part A: Journal of Power and Energy*, page 0957650911407700, 2011.

[17] MM Rashidi, N Galanis, F Nazari, A Basiri Parsa, and L Shamekhi. Parametric analysis and optimization of regenerative clausius and organic rankine cycles with two feedwater heaters using artificial bees colony and artificial neural network. *Energy*, 36(9):5728–5740, 2011.

[18] A. Salhi and E.S. Fraga. Nature-inspired optimisation approaches and the new plant propagation algorithm. *in Proceedings of the The International Conference on Numerical Analysis and Optimization (ICeMATH '11), Yogyakarta, Indonesia*, pages K2–1–K2–8, 2011.

[19] A. Salhi, LG. Proll, D. R. Insua, and JI. Martin. Experiences with stochastic algorithms for a class of constrained global optimisation problems. *Recherche operationnelle, Paris*, 34(2):183–198, 2000.

[20] Muhammad Sulaiman and Abdellah Salhi. A seed-based plant propagation algorithm: the feeding station model. *The Scientific World Journal*, 2015, 2015.

[21] Muhammad Sulaiman, Abdellah Salhi, Birsen Irem Selamoglu, and Omar Bahaaldin Kirikchi. A plant propagation algorithm for constrained engineering optimisation problems. *Mathematical Problems in Engineering*, 2014, Article ID 627416, 10 pages, 2014, doi:10.1155/2014/627416.

[22] Jianyong Sun, Jonathan M Garibaldi, Natalio Krasnogor, and Q Zhang. An intelligent multi-restart memetic algorithm for box constrained global optimisation. *Evolutionary Computation*, 21(1):107–147, 2013.

[23] El-G. Talbi. *Metaheuristics: from design to implementation*, volume 74. John W & Sons, 2009.

[24] X-S Yang. *Nature-inspired metaheuristic algorithms*. Luniver Press, 2011.