A New Modified Firefly Algorithm

http://dx.doi.org/10.3991/ijes.v4i2.5879

Divya Gupta[1], Medha Gupta[2]
[1] Wize Commerce Private Limited, Gurgaon, India
[2] Ambedkar Institute of Advanced Communication Technologies and Research (AIACTR), GGSIPU, New Delhi, India

***Abstract*—Nature inspired meta-heuristic algorithms studies the emergent collective intelligence of groups of simple agents. Firefly Algorithm is one of the new such swarm-based metaheuristic algorithm inspired by the flashing behavior of fireflies. The algorithm was first proposed in 2008 and since then has been successfully used for solving various optimization problems. In this work, we intend to propose a new modified version of Firefly algorithm (MoFA) and later its performance is compared with the standard firefly algorithm along with various other meta-heuristic algorithms. Numerical studies and results demonstrate that the proposed algorithm is superior to existing algorithms.**

***Index Terms*—Benchmark Functions, Firefly Algorithm, Nature-Inspired Algorithms, Unconstrained Optimization**

I. INTRODUCTION

Swarm Intelligence is a collective behavior of decentralized, self-organized natural or artificial systems. The concept was introduced by Gerardo Beni and Jing Wang in 1989 [1]. It is the study of computational systems inspired by the 'collective intelligence'. The collective intelligence emerges through the cooperation of large numbers of homogeneous agents in the environment. The collective behavior of the organisms like ants [2], bees [3] [4], fish [5] etc. inspires artificial intelligence to simulate and solve real world problems. In such systems, there are a number of simple agents which follow simple and fixed rules that determine their possible behavior during interaction among themselves and the surroundings. Although the individual particles are ignorant of it, their collective behavior leads to an intelligent global behavior. In nature such systems are commonly used to solve problems such as effective foraging for food, prey evading, or colony relocation. The information is typically stored throughout the participating homogeneous agents, or is stored or communicated in the environment itself such as through the use of pheromones in ants, dancing in bees, and proximity in fish and birds. Like evolutionary computation, swarm intelligence 'algorithms' or 'strategies' are considered adaptive strategies and are typically applied to search and optimization domains.

Optimization is the process of selecting the best element (with regard to some criteria) from some set of available alternatives or to find minimum or maximum output for an experiment. The input consists of variables; the process or function is known as the cost function, the objective function or the fitness function; and the output is the cost or fitness [6]. An optimization problem [7] is a real world problem where the objective function is not differentiable and its values can be acquired by simulation. These problems are either single-objective or multi-objective. In single objective optimization problems only one optimum solution with a single solution space exists [8]. In these problems if a new solution has a better objective function value than the old solution, the new solution is accepted. An algorithm works in this space by accepting or rejecting the solutions based on their respective function values.

Optimization problems based on swarm intelligence are known as meta-heuristic algorithms [9] which have features like self-organization, no central control, derivative free and easy to implement. These features lead to an emergent behavior that overcome the main limitations of conventional methods and can be conveniently applied to various optimization problems. These metaheuristic algorithms have two major components namely exploitation and exploration. The exploration ability ensures that the algorithm can search the whole space and escape from local optima while the exploitation ability guarantees the algorithm can search carefully and converge to the optimal point [10].

Swarm Intelligence principles have been successfully applied in a variety of problem domains including function optimization problems, finding optimal routes, scheduling, structural optimization, and image and data analysis [11] [12]. Computational modelling of swarms has been further applied to a wide-range of diverse domains, including machine learning [13], bioinformatics and medical informatics [14], dynamical systems and operations research [15]; they have been even applied in finance and business [16].

The firefly algorithm [17] is a swarm-based metaheuristic algorithm which imitates the social behavior of fireflies. Each firefly flashes its light with some brightness which attracts other fireflies within the neighborhood. The closer the two fireflies are, the more attractive they will seem. When the attractiveness is proportional to the objective function the search space is explored by moving the fireflies towards more attractive neighbors. The search strategy consists of controlled randomization, efficient local search and selection of the best solutions. Though, the algorithm has advantages of being easy to implement and understand and has been successfully applied to various engineering optimization problems since its emergence in 2008, it consists of disadvantages such as getting trapped in local optima or no memorizing capability.

The main aim of this paper is to eliminate all the existing limitations of Firefly algorithm by proposing a new modified version of the same. Later a comprehensive comparative study on the performance of the proposed algorithm (MoFA) with the standard Firefly algorithm (SFA) and other metaheuristic algorithms for optimizing a very large set of numerical functions is presented.

II. FIREFLY ALGORITHM

A. Standard Firefly Algorithm (SFA)

Firefly algorithm was proposed by Dr. Xin She Yang in 2008 and is inspired by the mating behavior of fireflies [17]. Fireflies belong to the family of insects that are capable to produce natural light used to attract a mate or a prey. There are about two thousand firefly species which produce short and rhythmic flashes. These flashes often appear to be in a unique pattern and produce an amazing sight in the tropical areas during summer. If a firefly is hungry or looks for a mate its light glows brighter in order to make the attraction of insects or mates more effective. The brightness of the bioluminescent light depends on the available quantity of a pigment called luciferin, and more pigment means more light. Based on the bioluminescent communication phenomenon of fireflies, the SFA uses three main basic rules:

1. A firefly will be attracted by other fireflies regardless of their sex.
2. Attractiveness is proportional to their brightness and decreases as the distance among them increases.
3. The landscape of the objective function determines the brightness of a firefly [18].

Based on above three rules the pseudo code of SFA is prepared which is used by all the modified algorithms too.

Pseudo code for SFA:

Inputs: objective function *f(x)*; variable boundary; population size *n*; maximum attractiveness β_0; absorption coefficient γ; randomization parameter α
Outputs: best solution

1. Initialization;
2. For variable $i = 1{:}n$
3. Randomly produce x_i within the variable ranges;
4. End for i;
5. Evaluate the function values of the firefly population;
6. Do while (Termination Criterion Are Not Met)
7. For $i = 1{:}n$
8. For variable $j = 1{:}n$
9. If $f(x_j) < f(x_i)$
10. If ($I_j > I_i$), move firefly i towards j;
11. End if
12. Evaluate new solutions and update light intensity;
13. End for j;
14. End for i;
15. Evaluate the new firefly populations;
16. Record the best solution achieved so far;
17. End while;

The process of Firefly algorithm starts with the initialization of the population of fireflies. Each firefly in a population represents a candidate solution and the population size determines the size of the search space. In the implementation of the algorithm, the flashing light or the brightness of a firefly is formulated in such a way that it gets associated with the objective function to be optimized.

In the Firefly algorithm, there are two important issues: the variation of the light intensity and the formulation of the attractiveness. As we know the intensity *(I)* of flashes decreases as the distance (r) increases, with the medium given, it can be determined by equation $I(r) = I_0 e^{-\gamma r^2}$. Attractiveness ($\beta$) function is defined using an absorption coefficient (γ) and distance (r) by the equation $\beta = \beta_0 e^{-\gamma r^2}$.

The Cartesian distance between any two i[th] and j[th] fireflies at x_i and x_j respectively, the Cartesian distance is determined by equation $r_{ij} = \sqrt{\sum_{k=1}^{d}(x_{i,k} - x_{j,k})^2}$ where $x_{i,k}$ is the k^{th} component of the spatial coordinate x_i of the i^{th} firefly and d is the number of dimensions.

The movement of a firefly occurs when ith firefly gets attracted to another more attractive (brighter) jth firefly which is determined by equation $x_i = x_i + \beta_0 e^{-\gamma r_{ij}^2}(x_j - x_i) + \alpha\epsilon$. Here the second term is due to the attraction while the third term is randomization with α being the randomization parameter and ε being the vector of random numbers drawn from a Gaussian distribution. The randomization parameter enhances the searching ability of the algorithm whereas the attraction term ensures that the attraction of a firefly to another is maximum when they are at the same place (i.e. r=0) and minimum when they are not in close proximity (i.e. r=∞).

For any large number of fireflies (N), if N>>m, where m is the number of local optima of an optimization problem, the convergence of the algorithm can be achieved. Here, the initial location of N fireflies is distributed uniformly in the entire search space, and as the iterations of the algorithm continue fireflies converge into all the local optimum. By comparing the best solutions among all these optima, the global optima are achieved [18]. The parameter γ characterizes the contrast of the attractiveness and its value varies from 0.1 to 10 determining the convergence speed of the Firefly algorithm.

Advantages of this algorithm were:

- Easy implementation
- Easy to understand

Disadvantages of this algorithm were:

- Gets trapped into several local optima
- No Memorizing Capability

B. Proposed Firefly Algorithm (MoFA)

This algorithm eliminates the weaknesses of the firefly algorithm by enhancing its exploitation and exploration ability by reducing the randomness of the algorithm and improving the collective movement of the fireflies. In the SFA, the fireflies move regardless of the global optima which decreases the ability of the firefly algorithm to find global best. Therefore in the MoFA, while comparing the brightness of two fireflies the global optimum affects the movement of fireflies. The firefly with either the maximum or minimum value is allowed to influence other fireflies leading to better results iteratively. Also this algorithm enhances the exploitation quality of the algorithm by gradually reducing the randomness and by adding a

social dimension to each firefly (the global best). This increases the global exploration chances of fireflies.

In the MoFA implementation, randomization parameter α was not kept fixed and was linearly decreased from α_0 to α_∞ with iterations, where α_0 was the initial value and α_∞ was the final value. This strategy could keep balance between the exploration and exploitation abilities of the proposed algorithm. In the early stage, larger α provided better global searching ability and in the later stage, smaller α offered better convergence. The distance function (r_i) was determined by the equation

$r_{i,best} = \sqrt{(x_i - x_{gbest})^2 + (y_j - y_{best})^2}$.

The movement of i^{th} firefly is determined by equation $x_i = x_i + \left(\beta_0 e^{-\gamma r_{i,j}{}^2}(x_j - x_i) + \beta_0 e^{-\gamma r_{i,best}{}^2}(x_{gbest} - x_i)\right) + \alpha\varepsilon + \lambda\varepsilon(x_i - g_{best})$ where $\varepsilon = \left(rand - {}^1/_2\right)$.

Here, the i^{th} firefly got attracted to the best solution if no local best solution existed in the neighbourhood. Also, the current global best (*gbest*) is explicitly used redefine the distance and movement of fireflies along with decoding the final best solutions. Thus, the proposed algorithm reduces the randomness of the algorithm to obtain convergence quickly and influences the movement of fireflies towards global optima for reducing the probability of the algorithm in getting trapped into several local optima.

III. OTHER META-HEURISTICS ALGORITHMS USED

In order to test the performance of the MoFA, we compare its results with the results obtained using other commonly applied heuristic algorithms. Following algorithms are used for comparison in this work.

A. Random Search (RS)

It is the simplest optimization algorithm which randomly generates a large number of possible solutions and chooses the one which results in lowest cost.

B. Genetic Algorithm (GA)

This algorithm [19], [20] is modeled on the principles of evolution via natural selection, employing a population of individuals that undergo selection in the presence of variation-inducing operators such as mutation and crossover. A fitness function is used to evaluate individuals, and the reproductive success varies with the fitness function.

C. Bat Algorithm (BA)

This algorithm [21], [22] is based on the echolocation behavior of bats. It uses a frequency-tuning and automatic balance of exploration and exploitation by controlling loudness and pulse emission rates. Initially, each bat is randomly assigned a frequency and then once a solution is selected among the current best solutions, a new solution for each bat is generated locally using random walk.

IV. BENCHMARK FUNCTIONS

To test the performance of the proposed algorithm, 15 benchmark problems have been used. This set includes many different kinds of problems such as unimodal, multimodal, separable, non-separable, low-dimensional, high-dimensional, linear, non-linear, noisy and noiseless problems. Initial range, formulation, characteristics and the dimension of these problems are listed in Table I.

Multimodal functions have more than one local optimum and are used to test the ability of algorithms for getting rid of local optimum whereas Unimodal functions have only one global optimum. Another set of test problems consists of Separable/Non-Separable functions. A p-variable separable function can be expressed as the sum of p functions of one variable. Non-separable functions cannot be written in this form and hence are more difficult than Separable functions. The high-dimensional functions ($D \geq 25$) are more difficult to optimize than the low-dimensional ($D < 25$) problems and hence the dimensionality of the search-space plays a vital role in these functions. Also expected fitness depends on the random noise generator (Gaussian or uniform) which makes sure that the algorithm never gets the same value on the same point [6]. In this work, 6 unimodal, 9 multimodal, 4 Separable and 11 Non-Separable functions, 10 low-dimensional and 5 high-dimensional functions have been used.

V. EXPERIMENTAL ANALYSIS

To confirm the efficiency of the proposed method, the algorithms were tested on 15 benchmark functions which are discussed in previous section. Also, in order to ensure the validity of the results, each experiment was repeated ten times and the best value of each run along with the statistical tests was tabulated. The experimental environment was implemented in MATLAB programs and executed on a DELL INSPIRON Computer with the configuration of Intel Core I3 CPU M330 at 2.13 GHz and 3GB RAM. Unless specified otherwise, in all cases the values of the common parameters such as population size, N and total evaluation number was chosen to be same. The population size was kept 10 and the maximum evaluation number was 1000 for all functions. Also to make comparison clear, the values below e-12 were assumed to be 0.

Table II gives the objective function values obtained by the algorithms while applied on the benchmark functions. The best, mean, standard deviation (stdev) and standard error of mean (SEM) values have been listed. It can be clearly seen that the proposed algorithm MoFA emerges as a clear winner (values in bold) with respect to precision. The best and mean values are closest to the respective objective function values and the standard deviation values are lowest for the proposed algorithm. Although the proposed algorithm converges slowly than the Genetic and Bat algorithms with respect to CPU time, the precise results by the proposed algorithm makes us ignore the slower speed which is due to the computational complexity of the algorithm i.e. $O(n^2)$, where n is the population size.

It can also be observed that the proposed algorithm converges to the optima much faster than the existing firefly algorithm. The proposed algorithm does not cause fireflies to move regardless of its previous better situation and is therefore able to memorize any history of better situation for each firefly leading to its memorization capability. The probability for the proposed algorithm to get trapped into several local optima reduces due to high precision of the objective function values obtained. Also the algorithm emerges out to be more robust and flexible than the existing algorithm on the basis of dimension and other control parameters such as γ and α [paper sent for publication].

VI. CONCLUSION

In this work, a new modified Firefly algorithm was proposed and its performance was compared on various parameters by using 15 benchmark functions with other nature-inspired meta-heuristic algorithms. The extensive simulation results given in previous section certified that MoFA in contrast to SFA have a minor chance of premature convergence. Also, MoFA outperforms other metaheuristic algorithms with greater accuracy. Thus, the proposed method can be applied to many real-time applications such as Travelling Salesman Problem, Vertex Cover Problem and Data Clustering.

REFERENCES

[1] Beni G., Wang J., Swarm Intelligence in Cellular Robotic Systems, Proceed, NATO Advanced Workshop on Robots and Biological Systems, Tuscany, Italy, June 26–30 (1989).

[2] A. Colorni, M. Dorigo et V. Maniezzo, Distributed Optimization by Ant Colonies, actes de la première conférence européenne sur la vie artificielle, Paris, France, Elsevier Publishing, 134-142, 1991.

[3] D. Karaboga and B. Akay, "A Comparative Study of artificial bee colony (ABC) algorithm", Applied Mathematics and Computation 214(2009), pp. 109-131, 2009.

[4] Abbass, H. (2001) 'MBO: marriage in honey bees optimization – ahaplometrosis polygynous swarming approach', Proceedings of the 2001 Congress on Evolutionary Computation CEC2001, IEEE Press, pp.207–214. http://dx.doi.org/10.1109/CEC.2001.934391

[5] X. L. Li, "A new optimization algorithm: artificial fish school algorithm", Hangzhou, Zhejing University, 2003.

[6] Randy L.Haupt, Sue Ellen Haupt, Practical Genetic Algorithms, Second Edition, John Wiley &Sons, 2004.

[7] Ausiello, Giorgio; et al. (2003), Complexity and Approximation (Corrected ed.), Springer, ISBN 978-3-540-65431-5.

[8] MA Sasa, Xue Jia, Fang Xingqiao, Liu Dongqing, "Research on Continuous Function Optimization Algorithm Based on Swarm Intelligence", 5th International Conference on Computation, 2009, pg no. 61-65.

[9] Yang, X. S. (2008), Nature-Inspired Metaheuristic Algorithms,Frome: Luniver Press.

[10] Xin-She Yang:Nature-Inspired Metaheuristic Algorithms, Second Edition, Luniver press,2011, p. 160.

[11] A. P. Engelbrecht (ed.), Computational Intelligence: An Introduction. John Wiley & Sons, England, 2002.

[12] C. P. Lim, L. C. Jain, and S. Dehuri, Innovations in Swarm Intelligence: Studies in Computational Intelligence, Vol. 248, Springer, 2009. http://dx.doi.org/10.1007/978-3-642-04225-6

[13] S. Das, B. K. Panigrahi, and S. S. Pattnaik, Nature-Inspired Algorithms for Multi-objective Optimization, Handbook of Research on Machine Learning Applications and Trends: Algorithms Methods and Techniques, Hershey, New York, Vol. 1, pp. 95–108, 2009.

[14] S. Das, A. Abraham, and A. Konar, Swarm Intelligence Algorithms in Bioinformatics, Studies in Computational Intelligence. Vol. 94, pp. 113–147, 2008. http://dx.doi.org/10.1007/978-3-540-76803-6_4

[15] K. E. Parsopoulos and M N. Vrahatis, Particle Swarm Optimization and Intelligence: Advances and Applications, Information Science Reference, Hershey, Pennsylvania, 2010. http://dx.doi.org/10.4018/978-1-61520-666-7

[16] E. Bonabeau, C. Meyer, Swarm Intelligence: A Whole New Way to Think About Business, Harvard Business Review, Vol.79, No.5, pp. 106-114, 2001.

[17] X. S. Yang, "Firefly algorithm for multimodal optimization." In: Stochastic Algorithms: foundations and applications, SAGA, lecture notes in computer sciences, pp. 169-178, 2009. http://dx.doi.org/10.1007/978-3-642-04944-6_14

[18] Adil Hashmi, Nishant Goel, Shruti Goel, Divya Gupta, "Firefly Algorithm for Unconstrained Optimization", IOSR Journal of Computer Engineering, Vol. 11, Issue 1,2013.

[19] Bäck, Thomas (1995). Evolutionary algorithms in theory and practice : evolution strategies, evolutionary programming, genetic algorithms. Oxford: Oxford University Press. p. 328.

[20] X. S. Yang, "A New Metaheuristic Bat-Inspired Algorithm" Department of Engineering, University of Cambridge, April 2010.

[21] Xin-She Yang and Amir H. Gandomi, "Bat Algorithm: A Novel Approach for Global Engineering Optimization", Engineering Computations, Vol. 29, Issue 5, 2012, pp.464- 483. http://dx.doi.org/10.1108/02644401211235834

[22] X. S. Yang, "A New Metaheuristic Bat-Inspired Algorithm" Department of Engineering, University of Cambridge, April 2010.

AUTHORS

Divya Gupta is with Wize Commerce Private Limited Gurgaon, India (gupta.divya3091@gmail.com),

Medha Gupta is with Ambedkar Institute of Advanced Communication Technologies and Research (AIACTR), GGSIPU, New Delhi, India (medhaguptacse@gmail.com)

Submitted 26 April 2016. Published as resubmitted by the authors 27 May 2016.

TABLE I
BENCHMARK FUNCTIONS USED (U-UNIMODAL, M-MULTIMODAL, N-NON-SEPARABLE, S-SEPARABLE)

S.no	Function	C	Range	D	Formulation
1.	Sphere	US	[-100,100]	30	$f(x) = \sum_{i=1}^{n} x_i^2$
2.	Powell	UN	[-4,5]	24	$f(x)= \sum_{i=1}^{n/k} (x_{4i-3} +10\, x_{4i-2})^2 +5(x_{4i-1} - x_{4i})^2 +(x_{4i-2} - x_{4i-1})^4 + 10(x_{4i-3} - x_{4i})^4$
3.	Matyas	UN	[-10, 10]	2	$f(x)= 0.26(x_1^2 + x_2)^2 - 0.48\, x_1 x_2$
4.	Easom	UN	[-100,100]	2	$f(x)= -\cos(x_1)\cos(x_2)\exp(-(x_1-\prod)^2-(x_2-\prod)^2)$
5.	Zakharov	UN	[-5, 10]	10	$\mathbf{f}(x) = \sum_{i=1}^{n} x_i^2 + (\sum_{i=1}^{n} 0.5 i x_i)^2 + (\sum_{i=1}^{n} 0.5 i x_i)^4$
6.	Dixon-Price	UN	[-10, 10]	30	$f(x)= (x_1-1)^2 + \sum_{i=2}^{n} i(2x_i^2 - x_{i-1})^2$
7.	Rastringin	MS	[-5.12,5.12]	30	$f(x) = \sum_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10]$
8.	Michalewicsz 2	MS	[0,π]	2	$f(x)= -\sum_{i=1}^{n} sin(x_i)(\sin(i x_i^2/\prod))^{2m}$; m=10

9.	Bohachevsky 1	MS	[-100,100]	2	$f(x)= x_1^2 +2 x_2^2 -0.3 \cos(3\prod x_1) - 0.4 \cos(4\prod x_2)+ 0.7$
10.	Goldstein-Price	MN	[-2, 2]	2	$\boldsymbol{f}(x) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)][30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$
11.	Ackley	MN	[-15, 30]	30	$\boldsymbol{f}(x) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$
12.	Griewank	MN	[-600,600]	30	$\boldsymbol{f}(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \Pi_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$
13.	Bohachevsky 2	MN	[-100,100]	2	$f(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1)(4\pi x_2) + 0.3$
14.	Bohachevsky 3	MN	[-100,100]	2	$f(x)= x_1^2 +2 x_2^2 -0.3 \cos(3\prod x_1 + 4\prod x_2)+ 0.3$
15.	Hump Camel-back	MN	[-5,5]	2	$f(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x1x2 - 4x_2^2 + 4x_2^4$

TABLE II
OBJECTIVE FUNCTION VALUES OBTAINED BY VARIOUS ALGORITHMS

Function	F(x)		RS	GA	BAT	SFA	MoFA
		Best	1.68e+3	1.78e+3	1.21e-1	1.71e-8	**4.16e-9**
		Mean	1.73e+3	1.78e+3	1.71	5.93e-3	**7.79e-8**
Griewank	0	StdDev	3.09e+1	**0**	2.25	4.23e-3	**1.14e-7**
		SEM	9.78	**0**	0.71	1.34e-3	**3.6e-8**
		CPU time (sec)	0.031	0.055	0.391	0.922	**0.719**
		Best	5.17e+2	5.22e+2	7.28e-9	4.48e-11	**0**
		Mean	5.20e+2	5.23e+2	4.48	0.895	**2.65e-9**
Rastringin	0	StdDev	1.83	0.52	3.82	1.28	**3.14e-9**
		SEM	0.58	0.16	1.21	0.404	**9.91e-10**
		CPU time (sec)	0.031	0.054	0.281	0.891	**0.703**
		Best	3.55e-57	0	-5.46e-1	-1	**-1**
		Mean	3.55e-58	0	-5.46e-2	-3.33e-1	**-3.33e-1**
Easom	-1	StdDev	1.12e-57	0	1.73e-1	0.5	0.48
		SEM	3.55e-58	0	5.46e-2	1.58e-1	1.52e-1
		CPU time (sec)	0.094	0.19	0.297	0.375	0.359
		Best	4.37e+6	4.94e+6	1.01e-10	0	**0**
		Mean	4.56e+6	4.94e+6	1.14e-9	**3.81e-11**	**2.97e-11**
Zakharov	0	StdDev	1.18e+5	0	1.18e-9	3.09e-11	**2.29e-11**
		SEM	3.74e+4	0	3.74e-10	0	**0**
		CPU time (sec)	0.015	0.070	0.235	0.422	**0.609**
		Best	1.50e+7	2.06e+7	5.22e+2	2.58e-1	**6.60e-3**
		Mean	1.72e+7	2.89e+7	2.04e+3	6.68e-1	**9.19e-2**
Powell	0	StdDev	1.34e+6	5.13e+6	9.54e+2	4.69e-1	**9.28e-2**
		SEM	4.25e+5	1.62e+6	3.01e+2	1.48e-1	**2.93e-2**
		CPU time (sec)	0.015	0.080	0.484	0.984	**0.844**
		Best	1.56e+6	1.83e+6	2.58e+4	1.24e-4	**5.40e-3**
		Mean	1.64e+6	1.96e+6	3.88e+4	1.32e-4	**7.96e-3**
Sphere	0	StdDev	6.40e+4	9.17e+4	1.05e+4	3.87e-5	**1.43e-3**
		SEM	2.02e+4	2.90e+4	3.32e+3	1.22e-5	**4.52e-4**
		CPU time (sec)	0.016	0.056	0.25	1.078	**0.703**
		Best	4.89e+8	7.63e+8	5.23e+4	0.73	**0.71**
		Mean	5.89e+8	8.32e+8	2.01e+5	1.52	**0.76**
Dixon-Price	0	StdDev	4.42e+7	4.28e+7	1.14e+5	1.55	**5.78e-2**
		SEM	1.40e+7	1.35e+7	3.62e+4	0.49	**1.83e-2**
		CPU time (sec)	0.016	0.090	0.25	0.812	**0.688**
		Best	22.4	22.34	5.68e-5	3.23e-5	**2.37e-5**
		Mean	22.4	22.35	5.09	5.14e-5	**5.10e-5**
Ackley	0	StdDev	0	3.16e-3	3.82	1.12e-5	**1.79e-5**
		SEM	0	9.99e-4	1.21	3.5e-6	**5.66e-6**
		CPU time (sec)	0.016	0.070	0.422	0.688	**0.719**
		Best	9.45	9.87e+2	3.41e-11	6.01e-11	**0**
		Mean	1.72e+3	9.87e+2	4.98e-2	9.00e-11	**0**
Matyas	0	StdDev	2.76e+3	0	8.25e-2	7.76e-11	**0**
		SEM	8.73e+2	0	2.61e-2	2.45e-11	**0**
		CPU time (sec)	0.125	0.239	0.234	0.39	**0.56**
		Best	1.80e+9	1.91e+9	3	3	**3**
		Mean	1.80e+9	1.91e+9	13.8	3	**3**
Goldstein-Price	3	StdDev	0	0	13.94	0	**0**
		SEM	0	0	4.41	0	**0**
		CPU time (sec)	0.094	0.10	0.297	0.578	**0.515**
		Best	2.93e+5	2.96e+5	0.883	1.61e-7	**1.11e-7**
Bohachevsky 1	0	Mean	2.93e+5	2.96e+5	2.92e+2	3.11e-7	**2.95e-7**
		StdDev	0	0	3.64e+2	1.97e-7	**2.24e-7**

		SEM	0	0	1.15e+2	6.22e-8	**7.08e-8**
		CPU time (sec)	0.094	0	0.484	0.532	0.562
		Best	2.93e+5	2.96e+5	7.77	6.07e-8	**1.52e-8**
		Mean	2.93e+5	2.96e+5	2.05e+2	2.23e-7	**2.09e-7**
Bohachevsky 2	0	StdDev	0	0	1.86e+2	1.85e-7	**1.77e-7**
		SEM	0	0	5.86e+1	5.85e-8	**5.60e-8**
		CPU time (sec)	0.094	0.129	0.328	0.485	0.526
		Best	2.82e+5	2.96	2.03	5.35e-8	**1.52e-8**
		Mean	2.92e+5	2.96	2.96e+2	1.79e-7	**1.40e-7**
Bohachevsky 3	0	StdDev	3.46e+3	0	4.05e+2	1.49e-7	**2.06e-7**
		SEM	1.09e+3	0	1.28e+2	4.71e-8	**6.51e-8**
		CPU time (sec)	0.094	0.112	0.781	0.468	**0.562**
		Best	4.94e+6	5.12e+6	4.80e-8	4.65e-8	**4.32e-8**
		Mean	5.04e+6	5.12e+6	8.16e-2	4.68e-8	**4.65e-8**
Hump	0	StdDev	3.47e+4	0	2.58e-1	1.51e-10	**1.97e-9**
		SEM	1.10e+4	0	8.16e-2	4.76e-11	**6.22e-10**
		CPU time (sec)	0.11	0.229	0.266	0.547	**0.542**
		Best	1.52	1.97	-1.8013	-1.8013	**-1.8013**
		Mean	1.694	1.97	-1.749	-1.8013	**-1.8013**
Michalewics	-1.8013	StdDev	0.127	0	0.270	0	**0**
		SEM	4.01e-2	0	8.53e-2	0	**0**
		CPU time (sec)	0.094	0.119	0.328	0.469	**0.461**