SOLUTION OF CONTINUOUS, DIFFERENTIABLE, NON-SEPARABLE, NON-SCALABLE AND UNI-MODEL TEST FUNCTIONS USING META-HEURISTICS

Seemab. Z.[1], A. Sana.[1,2], M. F. Tabassum[2,3]
[1] Department of Mathematics, Lahore Garrison University, Lahore, Pakistan.
[2] Department of Mathematics, University of Management and Technology, Lahore, Pakistan.
[3] Department of Mathematics, Lahore Leads University, Kamahan Campus, Lahore, Pakistan.
* Corresponding Author: Sana Akram sanaakram03@gmail.com, +92-321-4736571

ABSTRACT: *Test functions are important to validate new optimization algorithms and to compare the performance of various algorithms. There are many test functions in the literature, but there is no standard list or set of test functions one has to follow. New optimization algorithms should be tested using at least a subset of functions with diverse properties so as to make sure whether or not the tested algorithm can solve certain type of optimization efficiently. Here we provide a selected list of test problems for unconstrained optimization. In this paper we use three meta-heuristic optimization algorithms that are Particle Swarm Optimization (PSO), Differential Evolution (DE) and Artificial Bee Colony (ABC). To check the performance of these methods we use special suit of test functions. These test functions are continuous, differentiable, non-separable, non-scalable and uni-model in two dimensional. Test functions are important to check the accuracy of optimization algorithms and compare performance of them. The optimum results of these test functions are obtained by using MATLAB programming environment which demonstrated the effectiveness and applicability of test functions. Particle Swarm Optimization, Differential Evolution and Artificial Bee Colony are ranked with respect to function evaluation and then compare their results.*

Keywords: Continuous, Differentiable, Non-separable, Non-scalable and Uni-model test functions, Particle Swarm Optimization, Differential Evolution, Artificial Bee Colony

INTRODUCTION

Always theorists working in nonlinear programming area, as well as practical optimizers need to evaluate nonlinear optimization algorithms. Due to the hypothesis introduced in order to prove the convergence and the complexity of algorithms, the theory is not enough to establish the efficiency and the reliability of a method. As a consequence the only way to see the "power" of an algorithm remains its implementation in computer codes and its testing on large classes of test problems of different structures and characteristics. George B. Dantzig said "*the final test of a theory is its capacity to solve the problems which originated it"*. This is the main reason we assembled here this collection of large-scale unconstrained optimization problems to test the theoretical developments in mathematical programming.

Nonlinear programming algorithms need to be tested at least in two different senses. Firstly, testing is always profitable into the process of development of an algorithm in order to evaluate the ideas and the corresponding algebraic procedures. Clearly, well designed test problems are very powerful in clarifying the algorithmic ideas and mechanisms. Secondly, a reasonably large set of test problems must be used in order to get an idea about the hypothesis used in proving the quality of the algorithm (local and global convergence, complexity) and to compare algorithms at an experimental level.

Generally, two types of (unconstrained) nonlinear programming problems can be identified: "*artificial problems*" and "*real-life problems*". The artificial nonlinear programming problems are used to see the behavior of the algorithms in different difficult situations like long narrow valleys, functions with significant null-space effects, essentially uni-modal functions, functions with a huge number of significant local optima, etc. Figures 1-6 present some types of artificial nonlinear function in unconstrained optimization. All of them are of 2 variables, thus having the possibility for their graphical representation.

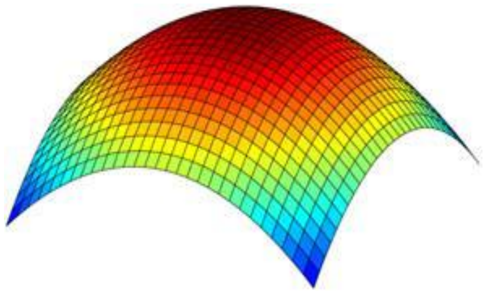
Fig.1. Unimodal function.

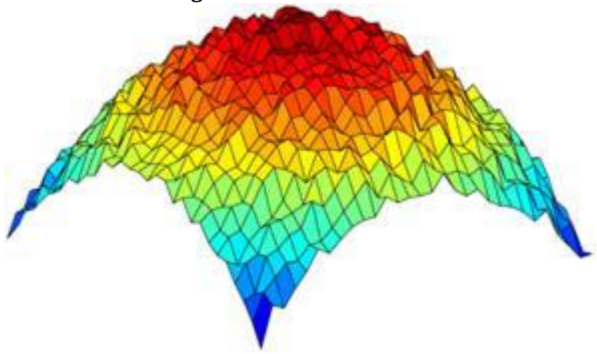
Fig.2. Essentially unimodal functions.

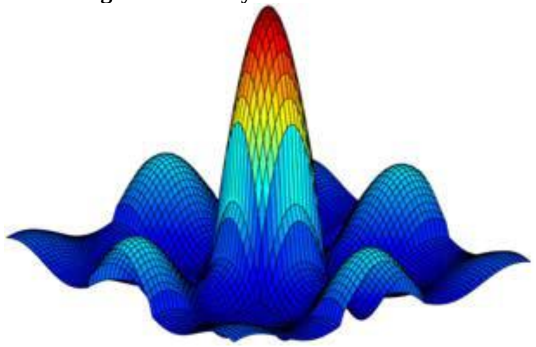
Fig.3. Functions with a small number of significant local optima.

The main characteristic of artificial nonlinear programming problems is that they are relatively easy to manipulate and to use into the process of algorithmic invention. Besides, the algorithmist may rapidly modify the problem in order to place the algorithm in different difficult conditions.

Real-life problems, on the other hand, are coming from different sources of applied optimization problems like physics, chemistry, engineering, biology, economy, oceanography, astronomy, meteorology, etc. Unlike artificial (unconstrained) nonlinear programming problems, real-life problems are not easily available and are difficult to manipulate. They may have complicated algebraic (or differential) expressions, may depend on a huge amount of data, and possible are dependent on some parameters which must be estimated in a specific way. A very nice collection of real-life unconstrained optimization problems is that given by [1, 2].

In this collection we consider only artificial unconstrained optimization test problems. All of them are presented in extended or generalized form. The main difference between these forms is that while the problems in generalized form have the Hessian matrix as a block diagonal matrix, the extended forms have the Hessian as a multi-diagonal matrix. Many individuals have contributed, each of them in important ways, to the preparation of this collection. We do not mention them here. An important source of problems was the CUTE collection established by [3]. Some other problems are from [4, 5] or are extracted from some other papers or technical reports. Generally, the problems in extended forms are slightly more difficult to be solved. The special suit of derivative free optimization is also discussed in [30-35]

In this script a nonlinear mathematical test functions are selected as a test case for the capabilities of DE, PSO and ABC methods. The remaining of this paper is organized as follows: Section 2 presents the characteristics of test functions and the concepts of the proposed three methods in details. In Section 3, the performances of methods are tested on different unconstrained optimization test problems and the results are compared with each other in terms of number of function evaluations (computational cost). Finally, conclusions are given in Section 4.

METHODS AND MATERIALS

Characteristics of Test Functions

The goal of any global optimization (GO) is to find the best possible solutions x* from a set X according to a set of criteria $F = \{f_1, f_2, \ldots, f_n\}$. These criteria are called objective functions expressed in the form of mathematical functions. An objective function is a mathematical function $f : D \subset \Re^n \rightarrow \Re$ subject to additional constraints. The set D is referred to as the set of feasible points in a search space. In the case of optimizing a single criterion f, an optimum is either its maximum or minimum. The global optimization problems are often defined as minimization problems, however, these problems can be easily converted to maximization problems by negating f. A general global optimum problem can be defined as follows:

$$\underbrace{\min imize}_{x} \quad f(x)$$

The true optimal solution of an optimization problem may be a set of $x^* \in D$ of all optimal points in D, rather than a single minimum or maximum value in some cases. There could be multiple, even an infinite number of optimal solutions, depending on the domain of the search space. The tasks of any good global optimization algorithm are to find globally optimal or at least suboptimal solutions. The objective functions could be characterized as continuous, discontinuous, linear, non-linear, convex, non-convex, uni-modal, multimodal, separable and non-separable.

According to [6], it is important to ask the following two questions before start solving an optimization problem; (i) what aspects of the function landscape make the optimization process difficult? (ii) What type of a priori knowledge is most effective for searching particular types of function landscape? In order to answer these questions, benchmark functions can be classified in terms of features like modality, basins, valleys, separability and dimensionality [7].

Modality: The number of ambiguous peaks in the function landscape corresponds to the modality of a function. If algorithms encounter these peaks during a search process, there is a tendency that the algorithm may be trapped in one of such peaks. This will have a negative impact on the search process, as this can direct the search away from the true optimal solutions.

Basins: A relatively steep decline surrounding a large area is called a basin. Optimization algorithms can be easily attracted to such regions. Once in these regions, the search process of an algorithm is severely hampered. This is due to lack of information to direct the search process towards the minimum. According to [6], a basin corresponds to the plateau for a maximization problem, and a problem can have multiple plateaus.

Valleys: A valley occurs when a narrow area of little change is surrounded by regions of steep descent [6]. As with the basins, minimizers are initially attracted to this region. The progress of a search process of an algorithm may be slowed down considerably on the floor of the valley.

Separability: The separability is a measure of difficulty of different benchmark functions. In general, separable functions are relatively easy to solve, when compared with their inseparable counterpart, because each variable of a function is independent of the other variables. If all the parameters or variables are independent, then a sequence of n independent optimization processes can be performed. As a result, each design variable or parameter can be optimized independently.

Dimensionality: The difficulty of a problem generally increases with its dimensionality. According to [7, 8], as the number of parameters or dimension increases, the search space also increases exponentially. For highly nonlinear problems, this dimensionality may be a significant barrier for almost all optimization algorithms.

Artificial Bee Colony Algorithm

In this method, two types of foragers, called employed and unemployed foragers are considered in the artificial hive of ABC. In the initialization of the algorithm, a new food source was produced for each employed forager. Employed foragers moved to nectar foraged from food source by conveying the position information of food sources to the hive. Onlooker bees, unemployed for-agers, were intended to try to improve food source positions of employed foragers by considering information sent by employed foragers to the hive. In this algorithm, food sources signified feasible solutions for the optimization problem. Scout bee was considered as the other type of unemployed foragers. If the employed or onlooker bees could not improve a food source by in a certain time, the status of employed bee allotted to that food source was turned to a scout bee. ABC algorithm comprises four phases' i.e. initialization, employed bee, onlooker bee and scout bee, realized sequentially. It is worth mentioning that ABC algorithm was an iterative algorithm and half of the population in the hive was employed bee and the other half was onlooker bee and only one scout bee could occur at the each iteration. Phases of the algorithm were articulated as follows [9].

Initialization phase: This phase was utilized only once. Firstly, the population size was determined. Half of the population comprised of employed bees and the remaining half was onlooker bees [10]. A new food source was produced for each employed bee by using Eq. (1).

$$x_{i,j} = x_j^{\min} + \lambda\left(x_j^{\max} - x_j^{\min}\right) \quad i=1,2,...,N, j=1,2,...,D \qquad (1)$$

where $x_{i,j}$ was j^{th} dimension of i^{th} employed bee, $x_j^{\max}$ and $x_j^{\min}$ were lower and upper bounds of j^{th} parameters, respectively, λ was a random number in range of *[0,1], N* represented the number of employed bees and D was the dimensionality of the optimization problem. Moreover, in this phase, the abandonment counter (AC) of each employed bee was reset. Thereafter, the fitness values of the food sources of the employed bees were calculated by using following:

$$\text{fit}_i = \begin{cases} \dfrac{1}{1+f_i} & \text{if } (f_i \geq 0) \\ 1+\text{abs}(f_i) & \text{otherwise} \end{cases} \qquad (2)$$

where fit_i denoted the fitness value of food source of i^{th} employed bee, f_i was the value of objective function specific for the optimization problem of food source of i^{th} employed bee. Further, in this phase, limit value was set and a counter was created and reset for each food source [10].

Employed bee phase: In this phase, each employed bee aimed to find a new food source to improve self-solution by using

$$v_{i,j} = x_{i,j} + \phi\left(x_{i,j} - x_{k,j}\right)$$

$$i,k \in 1,2,...,N, \quad j \in 1,2,...,D \quad \text{and} \quad i \neq k \qquad (3)$$

where $v_{i,j}$ was the j^{th} dimension of i^{th} candidate solution, $x_{i,j}$ was j^{th} dimension of i^{th} employed bee, $x_{k,j}$ was j^{th} dimension of k^{th} employed bee, ϕ acted as a random number in range of *[−1,+1], N* was the number of employed bee and *D* was the dimensionality of the optimization problem. Moreover, in this phase, the neighbor of candidate solution (k) and dimension of the problem (j) were randomly selected among the employed bee population and between dimensionality of the problem, respectively.

The fitness value of the new food source was calculated by using Eq. (2) and in case of better than old one, the new food source position was learnt by the employed bee and the AC of the food source was reset and AC was increased by 1.

Onlooker bee phase: In this phase, the employed bee shared the position information about the self-food source by dancing in the dance area of the hive. The onlooker bees watched out the dance and designated an employed bee by using fitness values of food sources of the employed bees and roulette wheel. The probability for selection was calculated as under:

$$p_i = \frac{\text{fit}_i}{\sum_{j=1}^{N} \text{fit}_j} \qquad (4)$$

where p_i was the probability of the selected i^{th} employed bee. After the selection, the onlooker bees aimed to improve the solutions of employed bees by using Eq. (3). If new solution gained by the onlooker bee was better than the solution of employed bee, the employed bee got the solution of the onlooker bee memorized and the AC was reset. Otherwise, AC was incremented by 1 [10].

Scout bee phase: In this phase, the abandonment counter with maximum content was fixed and matched with pre-determined limit value. If value of the AC having maximum content was greater than the limit value, the employed bee of food source of AC with maximum content becomes a scout bee. A new solution was generated for this new scout bee by using Eq. (1). AC of the new food source was reset. After generating new solution for itself, the scout bee returned to instatement i.e. employed bee.

Table-1: Parameter table for ABC

Description	Values
Maximum Number of Iterations	MaxIt=2000
Population Size (Colony Size)	nPop=10
Number of Onlooker Bees	nOnlooker=nPop
Abandonment Limit Parameter	L=200
Acceleration Coefficient Upper Bound	a=1

Particle Swarm Optimization

Natural creatures such as birds or fish behave as swarm. Many people research on them to know about how these creatures' behave as swarm. Reynold research on boid and develop the rules of swarm behavior. Boyd and Richerson research on decision process of people and develop that people use two things when they make decision, ist they use their own experience and second they use the experience of other's people. After this in 1990's Dorigo gave the idea of ant colony optimization (ACO) that was based on behavior of insects such as ants. After these researches Kennedy and Eberhert work on the swarm behavior of birds and develop Particle Swarm Optimization [11].

The purpose of PSO was too treated with nonlinear optimization problems but now it is also use for solving combinatorial optimization problems. PSO also treat with both discrete and continuous problems.

During research on boid reynold use the following three rules.

i. Every agent keeps away from nearest agent.
ii. Try to go towards centre of swarm
iii. Try to go towards destination

And the research results then use in Particle Swarm optimization.

During research on the decision process of human beings, Boyd and Recherson develop the idea of individual learning and culture transmission. Boyd and Recherson develop that people use two type of information when they make decisions. Firstly they use their own experience i.e. they try different choices and know which one is better. Secondly they use experience of other's people i.e. they know which choice their neighbors have best so far [12-14].

Eberhert and Kennedy research on swarm behavior of birds flocking and develop PSO. The position of each agent is represented by $x\,,y$ axis position and velocity is represented by $vx\,,vy$ w.r.t $x\ and\ y$ axis respectively.

Every agent knows its best position represented by (*pbest*) , this is the personal experience of the particle .also every particle knows best position in group represented by (*gbest*) due to which particle knows how other agents perform surrounding it, which is other's experience.

Now every agent uses the following information to modify its position:

- Current position x, y
- Current velocity vx, vy
- Difference between current position and *pbest*
- Difference between current position and *gbest*

The following velocity equation is used for modifying velocity.

$$v_i^{k+1} = wv_i^k + c_1rand_1 \times (pbest_i - s_i^k) + c_2rand_2 \times (gbest - s_i^k)$$

Where v_i^k is the current velocity of ith agent at iteration k, w is weighting function $c_1$and c_2 are weighting coefficients , rand is random num between 0 and 1 and s_i^k is current position of agent. $pbest_i$ is best position of agent i and $gbest$ is best position in group.

Following weighting function is used in above eq.

$$w = w_{max} - \frac{w_{max} - w_{min}}{iter_{max}} \times iter$$

where

w_{max} =initial weight
w_{min} =final weightd
$iter_{max}$=maximum iteration number
$iter$ =current iteration number

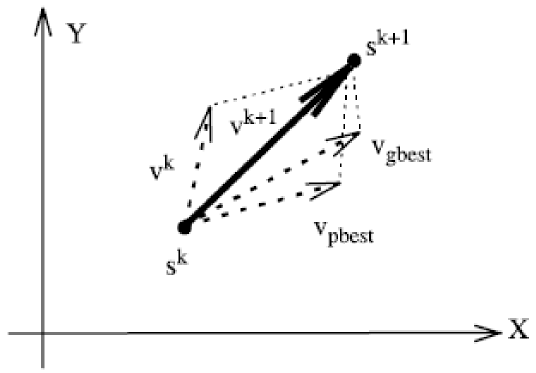

Fig.4. Modification of searching point.

Equation 1 can be explained as follow. RHS of equation 1 have three terms. First term shows the previous velocity and 2nd and 3rd terms use to modify the velocity of agent. Without 2nd and 3rd terms the agent will keep moving in the same direction so 1st term corresponds to diversification (as agent try to discover new areas). 2nd term corresponds to intensification as agent try to converge its $pbest_i$ and $gbest.$

PSO using (1) and (2) called inertia weights approach (IWA). The following equation use to modify the agents position.

$$s_i^{k+1} = s_i^k + v_i^{k+1}$$

The PSO can be summarize as

1. Generate initial condition of each agent. Initially searching point s_i^0 and velocity v_i^0 of every agent is set randomly within given range.Current searching point is set as pbest and best value among all pbest is set as gbest [15].
2. Evaluation of searching point .Calculate value of objective function for every agent. If this value is better than current *pbest* then *pbest* is replace by this value. And if the best value among new *pbest* is better than current *gbest* then *gbest* is replace by that value.

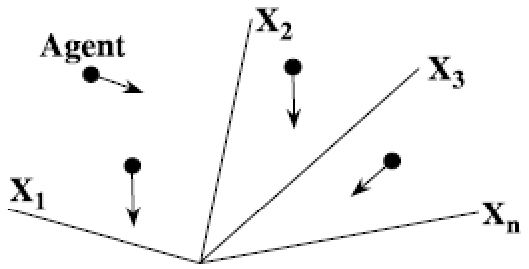

Fig.5. Searching concept with agent in solution space.

3. Modification of searching point. The searching point is is modified by using equation 1, 2 and 3.
4. Exit criteria. If current iteration reaches the pre determine positive number then stop the procedure [16].

Table-2: Parameter table for PSO

Description	Values
Maximum Number of Iterations	MaxIt=100
Population Size (Swarm Size)	nPop=10
Inertia Weight	w=1
Inertia Weight Damping Ratio	wdamp=0.99
Personal Learning Coefficient	c1=1.5
Global Learning Coefficient	c2=2.0

Differential Evolution

DE was first introduced by Storn and Price in (1994-1996).It is stochastic direct search optimization method. It is considered as fast and accurate method.

Population Structure: The versatile implementation of DE maintains a pair of vector population. The current population is represented byp_x that is composed of vectors $X_{i,g}$ and

$$p_{x,g} = X_{i,g} \quad ; i = 0,1,2, \dots\dots N_p - 1,$$
$$g = 0,1,2, \dots\dots. g_{max}$$
$$X_{i,g} = x_{j,i,g} \quad ; \ j = 0,1,2 \dots\dots\dots D - 1$$

Where $g = 0,1,2, \dots\dots. g_{max}$ indicate generation, i is population index run from 0 to $N_p - 1$ and j represents parameters within vectors run from 0 to -1 .

Initialization: Before initialization upper and lower bounds $b_U\ and\ b_L$ are specified .Once bounds are specified, a random number generator assign each parameter of every vector a value within given range. e.g. for generation $g = 0$

$x_{j,i,g} = rand_j(0,1).(b_{j,u} - b_{j,L}) + b_{j,L}$

The random number generator $rand_j(0,1)$ returns a uniformly distributed random number within range $[0,1)$.

Mutation: To produce new population of N_p vectors, DE mutates and recombines vectors. A weighted difference of two vectors is added to randomly select third vector. Following equation shows how three different vectors combine to produce mutant vector $v_{i,g}$ [17].

$$v_{i,g} = x_{r0,g} + F.(x_{r1,g} - x_{r2,g})$$

The scale factor, $F \in (0,1+)$, is a positive real number that controls the rate at which the population evolves. While there is no upper limit on F, effective values are seldom greater than 1. The *base vector* index, r_0 is assumed to be a randomly chosen vector index that is different from the *target vector* index, i. Except for being distinct from each other and from the base and target vector indices, the *difference vector* indices, r_1 and r_2, are also randomly selected once per mutant.

Crossover: DE crosses each vector with a mutant vector

$$U_{i,g} = u_{j,i,g} = \begin{cases} u_{j,i,g} & if\ (rand_j(0,1) \le Cr\ or\ j = j_{rand} \\ x_{j,i,g} & otherwise \end{cases}$$

$Cr \in (0,1)$ is user defined value that controls fraction of parameter value.

Crossover comprise Cr to uniform random no. generator.If random no. is less then or equal to Cr then it is selected otherwise parameter is copied by $x_{j,i,g}$[18].

Selection: If objective function value of $u_{i,g}$ is less then the objective function value of its target vector $x_{i,g}$ then it replace target vector in next generation otherwise $x_{i,g}$ will remain in the population for next generation.

$$x_{i,g+1} = \begin{cases} u_{i,g} & if\ f(u,g) \le f(x_{i,g}) \\ x_{i,g} & otherwise \end{cases}$$

Once new generation is establish the process is repeat again and again until optimum is located or pre-determined termination criterion is obtained [19].

Table-3: Parameter table for PSO

Description	Values
Maximum Number of Iterations	MaxIt=1000
Population Size	nPop=10
Lower Bound of Scaling Factor	beta_min=0.2
Upper Bound of Scaling Factor	beta_max=0.8
Crossover Probability	pCR=0.2

RESULTS AND DISCUSSIONS

Ackley 2 Function [20]

Function: $f_1 = -200 e^{-0.02\sqrt{x_1^2+x_2^2}}$

Range of function: $-32 \le x_i \le 32$

The global minimum is located at $= (0,0)$

Function value at global minimum = -200

Table-4: Compression results for Ackley 2 Function

Meta-heuristic Method	Calculated Value	Number of iterations	Rank
DE	-200	41th	2
PSO	-200	87th	3
ABC	-200	31st	1

We execute the all algorithms several times with the parameter setting given in table 1. In Ackley 2 function the optimum point is obtained is -200 by all three methods but the performance of ABC is outstanding. ABC found the optimum value in 31 iterations but DE performs 41 iterations and PSO performs 87 iterations. So we conclude that ABC is the best choice to solve Ackley2 function among other two functions. Final the ranking is also in favor of ABC.

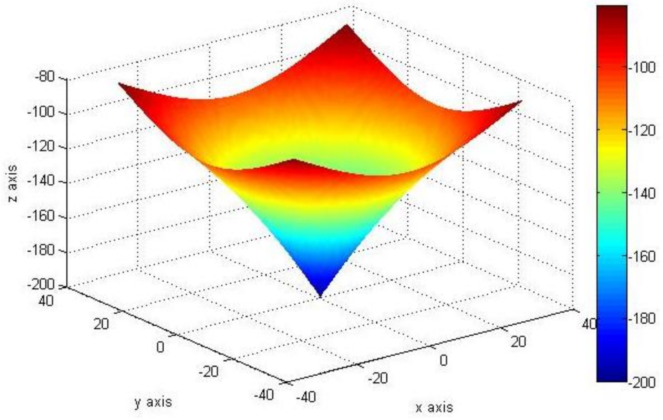

Fig 6: 3D surface plot for Ackley 2 Function

Ackley 3 Function [20]

Function: $f_2 = -200 e^{-0.02\sqrt{x_1^2+x_2^2}} + 5 e^{\cos(3x_1)+\sin(3x_2)}$

Range of function: $-32 \le x_i \le 32$

The global minimum is located at = $(0,-0.04)$

Function value at global minimum = -219.1418

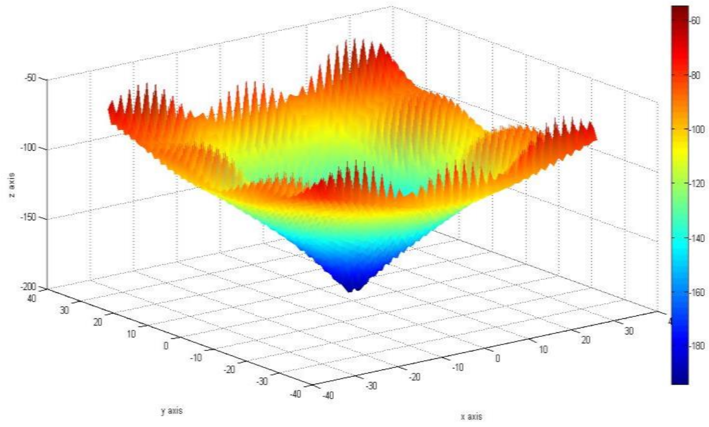

Fig. 7: 3D surface plot for Ackley 3 Function

Table-5: Compression results for Ackley 3 Function

Meta heuristic Method	Calculated Value	Number of iterations	Rank
DE	-195.23	9th	1
PSO	-195.629	56th	3
ABC	0.67676	22nd	2

We execute the all algorithms several times with the parameter setting given in table 1. In Ackley 3 function the optimum point is obtained is -195.23 by all three methods but the performance of DE is outstanding. DE found the optimum value in 9 iterations but ABC performs 22 iterations and PSO performs 56 iterations. So we conclude that DE is the best choice to solve Ackley3 function among other two functions. Final the ranking is also in favor of DE.

Beale function [20]

Function: $$f_3(x) = (1.5 - x_1 + x_1 x_2)^2 + (2.25 - x_1 + x_1 x_2^2)^2 + (2.625 - x_1 + x_1 x_2^3)^2$$

Range of function: $-4.5 \le x_i \le 4.5$

The global minimum is located at $=(3,0.5)$

Function value at global minimum = 0

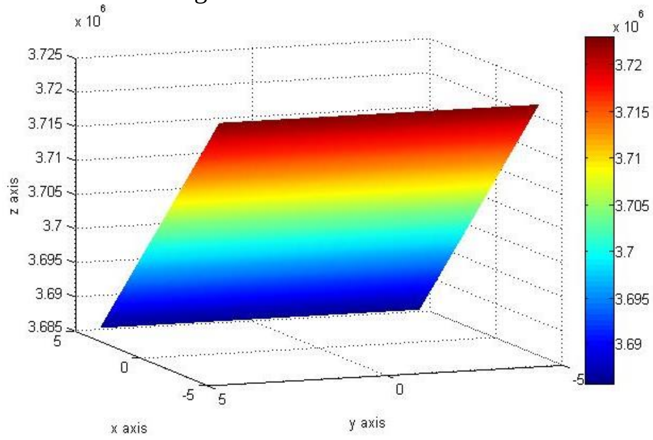

Fig 8: 3D surface plot for Beale function

Table-6: Compression results for Beale function

Meta heuristic Method	Calculated Value	Number of iterations	Rank
DE	0	490	3
PSO	0	201	2
ABC	$1.1785\text{x}10^{-9}$	57	1

We execute the all algorithms several times with the parameter setting given in table 1. In Beale function the optimum point is obtained is $1.1785\times e-009$ by all three methods but the performance of ABC is outstanding. ABC found the optimum value in 57 iterations but PSO performs 201 iterations and DE performs 490 iterations. So we conclude that ABC is the best choice to solve Beale function among other two functions. Final the ranking is also in favor of ABC.

Booth function [20]

Function: $f_4(x)=(x_1+2x_2-7)^2+(2x_1+x_2-5)^2$

Range of function: $-10 \le x_i \le 10$

The global minimum is located at $=(1,3)$

Function value at global minimum = 0

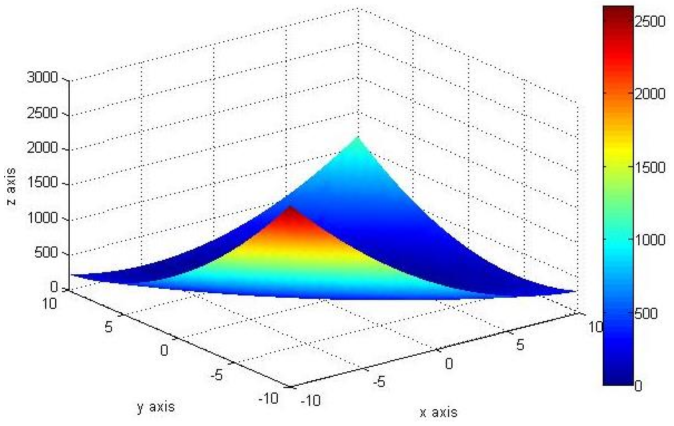

Fig 9: 3D surface plot for Booth function

Table-7: Compression results for Booth function

Meta heuristic Method	Calculated Value	Number of iterations	Rank
DE	0	337	3
PSO	0	159	2
ABC	$2.3827\times e-013$	80	1

We execute the all algorithms several times with the parameter setting given in table 1. In Booth function the optimum point is obtained is $2.3827\times e-013$ by all three methods but the performance of ABC is outstanding. ABC found the optimum value in 80 iterations but PSO performs 159 iterations and DE performs 337 iterations. So we conclude that ABC is the best choice to solve Booth function among other two functions. Final the ranking is also in favor of ABC.

Brent function [21]

Function: $f_5(x)=(x_1+10)^2+(x_2+10)^2+e^{-x_1^2-x_2^2}$

Range of function: $-10 \le x_i \le 10$

The global minimum is located at $=(0,\,0)$

Function value at global minimum = 0

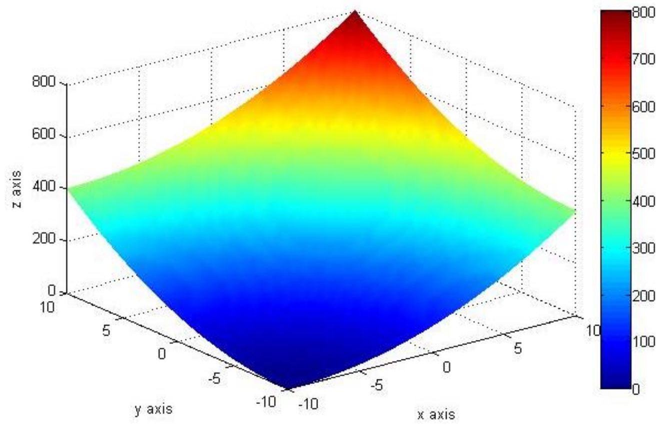

Fig 10: 3D surface plot for Brent function

Table-8: Compression results for Brent function

Meta heuristic Method	Calculated Value	Number of iterations	Rank
DE	$3.0055\times e-025$	100	2
PSO	0	171	3
ABC	$3.5391\times e-026$	84	1

We execute the all algorithms several times with the parameter setting given in table 1. In Brent function the optimum point is obtained is $3.5391\times e-026$by all three methods but the performance of ABC is outstanding. ABC found the optimum value in 84 iterations but PSO performs 171 iterations and DE performs 100 iterations. So we conclude that ABC is the best choice to solve Brent function among other two functions. Final the ranking is also in favor of ABC.

Cube function [22]

Function: $f_6(x)=100(x_2-x_1^{\,3})^2+(1-x_1)^2$

Range of function: $-10 \le x_i \le 10$

The global minimum is located at $=(0,\,0)$

Function value at global minimum = 0

Table-9: Compression results for Cube Function

Meta heuristic Method	Calculated Value	Number of iterations	Rank
DE	0	1365	3

PSO	0.00012446	142	2
ABC	0.00033169	99	1

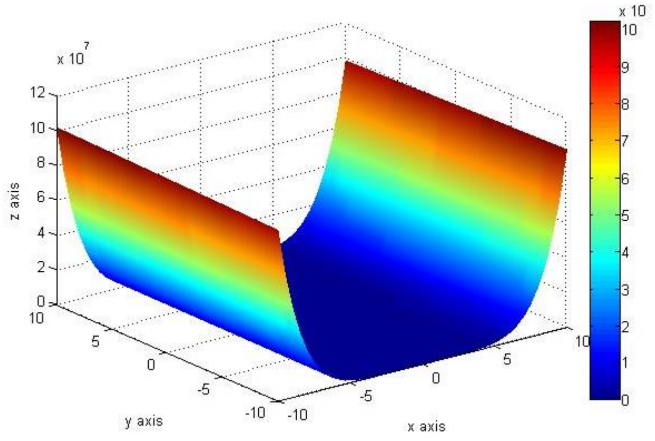

Fig 11: 3D surface plot for Cube Function

We execute the all algorithms several times with the parameter setting given in table 1. In Cube function the optimum point is obtained is 0.00012446 by all three methods but the performance of ABC is good with respect to iteration but not to function value. ABC found the optimum value in 99 iterations but PSO performs 142 iterations and DE performs 1365 iterations. So we conclude that PSO is the best choice to solve Cube function among other two functions. Final the ranking is also in favor of ABC.

El-Attar-Vidyasagar-Dutta function [23]

Function is: $f_7(x) = \left(x_1^2 + x_2 - 10\right)^2 + \left(x_1 - x_2^2 - 7\right)^2 + \left(x_1^2 + x_2^3 - 1\right)^2$

Range of function: $-500 \le x_i \le 500$

The global minimum is located at $= (2.842503, 1.920175)$

Function value at global minimum = 0.470427

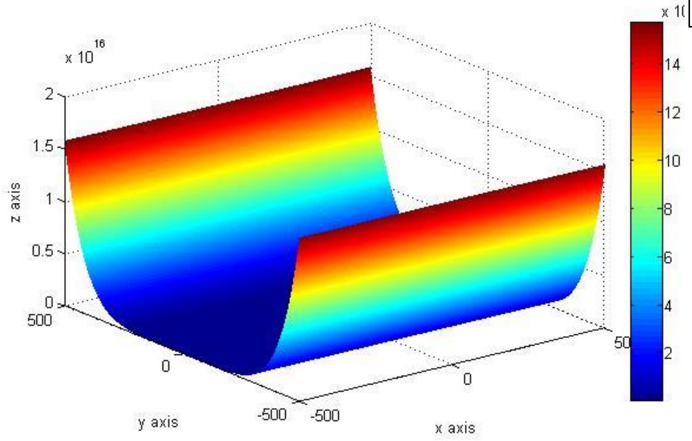

Fig 12: 3D surface plot for El-Attar Vidyasgar function

Table-10: Compression results for El-Attar Vidyasgar function

Meta heuristic Method	Calculated Value	Number of iterations	Rank
DE	1.7128	77th	2
PSO	1.7128	82th	3
ABC	1.7128	47th	1

We execute the all algorithms several times with the parameter setting given in table 1. In El-Attar Vidyasgar function the optimum point is obtained is 1.7128 by all three methods but the performance of ABC is outstanding. ABC found the optimum value in 47 iterations but PSO performs 82 iterations and DE performs 77 iterations. So we conclude that ABC is the best choice to solve El-Attar-Vidyasagar-Dutta function among other two functions. Final the ranking is also in favor of ABC.

Leon function [22]

Function is $f_8(x) = 100\left(x_2 - x_1^2\right)^2 + \left(1 - x_1\right)^2$

Range of function: $-1.2 \le x_i \le 1.2$

The global minimum is located at $= (1,1)$

Function value at global minimum = 0

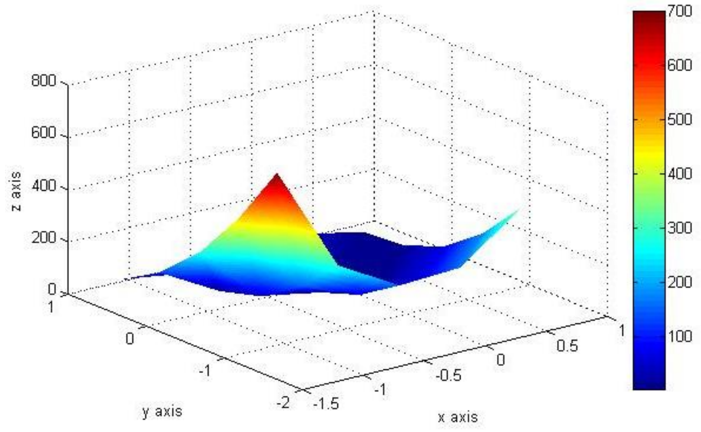

Fig 13: 3D surface plot for El-Attar Leon function

Table-11: Compression results for El-Attar Leon function

Meta heuristic Method	Calculated Value	Number of iterations	Rank
DE	0	1089	2
PSO	0	2235	3
ABC	0.00028426	165	1

We execute the all algorithms several times with the parameter setting given in table 1. In Leon function the optimum point is obtained is 0.00028426by all three methods but the performance of ABC is outstanding. ABC found the optimum value in 165 iterations but PSO performs 2235 iterations and DE performs 1089 iterations. So we conclude that ABC is the best choice to solve Leon function among other two functions. Final the ranking is also in favor of ABC.

Matyas function [24]

Function is: $f_9(x) = 0.26\left(x_1^2 - x_2^2\right)^2 - 0.48\,x_1 x_2$

Range of function: $-10 \le x_i \le 10$

The global minimum is located at $= (0,0)$

Function value at global minimum = 0

Table-12: Compression results for Matyas function

Meta heuristic Method	Calculated Value	Number of iterations	Rank
DE	$1.323\times e-011$	100	2
PSO	0	2853	3
ABC	$2.8248\times e-12$	65	1

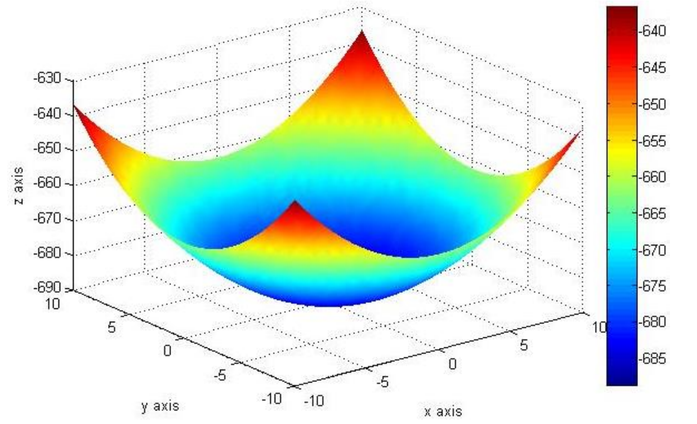

Fig 14: 3D surface plot for Matyas function

We execute the all algorithms several times with the parameter setting given in table 1. In Matyas function the optimum point is obtained is $2.8248\times e-012$ by all three methods but the performance of ABC is outstanding. ABC found the optimum value in 65 iterations but PSO performs 2853 iterations and DE performs 100 iterations. So we conclude that ABC is the best choice to solve Matyas function among other two functions. Final the ranking is also in favor of ABC.

Rotated Ellipse function [25]

Function is: $f_{10}(x)=7x_1^2-6\sqrt{3}x_1x_2+13x_2^2$

Range of function: $-500\le x_i \le 500$

The global minimum is located at $=(0,0)$

Function value at global minimum = 0

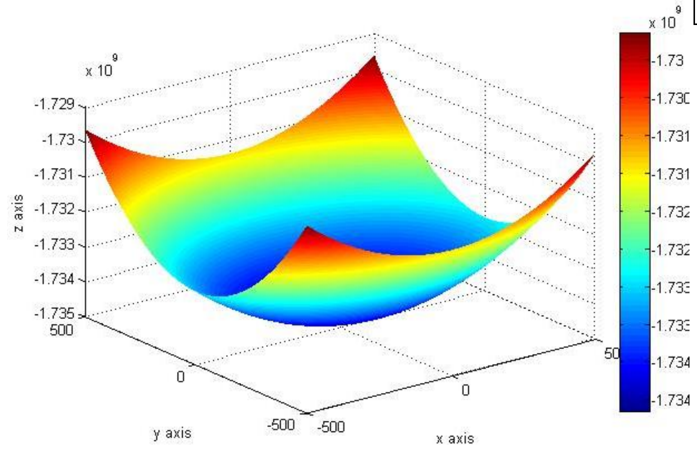

Fig 15: 3D surface plot for Rotated Ellipse Function

Table-13: Compression results for Rotated Ellipse Function

Meta heuristic Method	Calculated Value	Number of iterations	Rank
DE	0	1960	3
PSO	0	902th	2
ABC	$1.6747\times e-015$	77th	1

We execute the all algorithms several times with the parameter setting given in table 1. In Rotated Ellipse function the optimum point is obtained is $1.6747\times e-015$ by all three methods but the performance of ABC is outstanding. ABC found the optimum value in 77 iterations but PSO performs 902 iterations and DE performs 1960 iterations. So we conclude that ABC is the best choice to solve Rotated Ellipse function among other two functions. Final the ranking is also in favor of ABC.

Rotated Ellipse 2 function [25]

Function is $f_{11}(x)=x_1^2-x_1x_2+x_2^2$

Range of function: $-500\le x_i \le 500$

The global minimum is located at $=(0,0)$

Function value at global minimum = 0

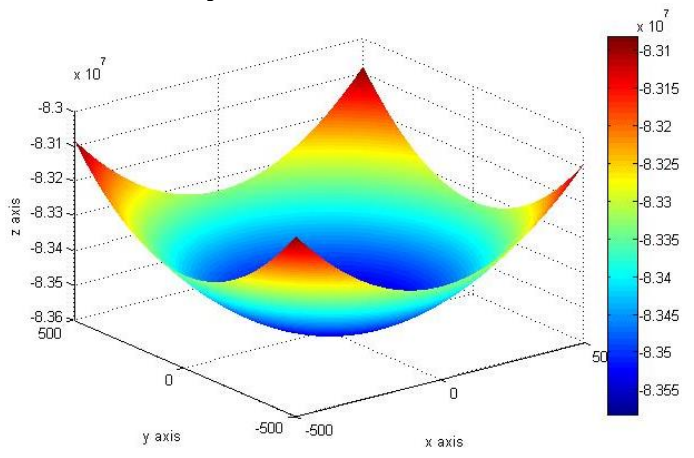

Fig 16: 3D surface plot for Rotated Ellipse 2

Table-14: Compression results for Rotated Ellipse 2

Meta heuristic Method	Calculated Value	Number of iterations	Rank
DE	0	1942	3
PSO	0	881	2
ABC	$2.5715\times e-019$	85th	1

We execute the all algorithms several times with the parameter setting given in table 1. In Rotated Ellipse function the optimum point is obtained is $2.5715\times e-019$ by all three methods but the performance of ABC is outstanding. ABC found the optimum value in 85 iterations but PSO performs 881 iterations and DE performs 1942 iterations. So we conclude that ABC is the best choice to solve Rotated ellipse 2 function among other two functions. Final the ranking is also in favor of ABC.

Rump function [26]

Function is

$$f_{12}(x)=(333.75-x_1^2)x_2^6+x_1^2\left(11x_1^2x_2^2-121x_2^4-2\right)+5.5x_2^8+\frac{x_1}{2x_2}$$

Range of function: $-500\le x_i \le 500$

The global minimum is located at $=(0,0)$

Function value at global minimum = 0

Table-15: Compression results for Rump function

Meta heuristic Method	Calculated Value	Number of iterations	Rank
DE	0	235	2
PSO	0	861	3
ABC	0	98	1

We execute the all algorithms several times with the parameter setting given in table 1. In Rotated Ellipse function the optimum point is obtained is $2.5715\times e-019$ by all three methods but the performance of ABC is outstanding.

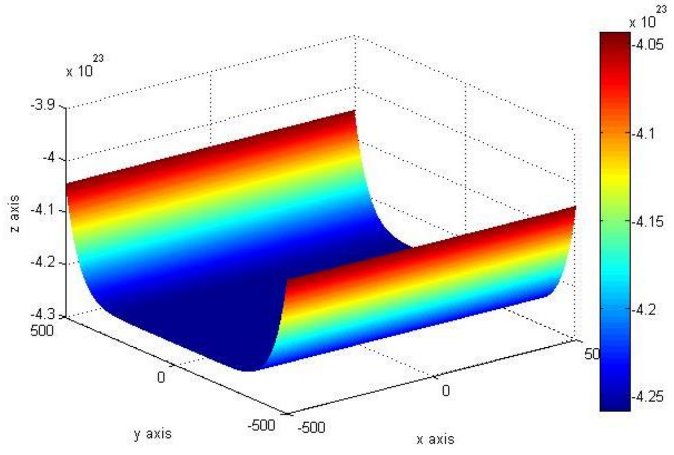

Fig 17: 3D surface plot for Rump function

ABC found the optimum value in 85 iterations but PSO performs 881 iterations and DE performs 1942 iterations. So we conclude that ABC is the best choice to solve Rump function among other two functions. Final the ranking is also in favor of ABC.

Zirilli or Aluffi-Pentini's function [27]

Function is: $f_{13}(x) = 0.25x_1^4 - 0.5x_1^2 + 0.1x_1 + 0.5x_2^2$

Range of function: $-10 \le x_i \le 10$

The global minimum is located at $(-1.0465,0)$

Function value at global minimum = -0.3523

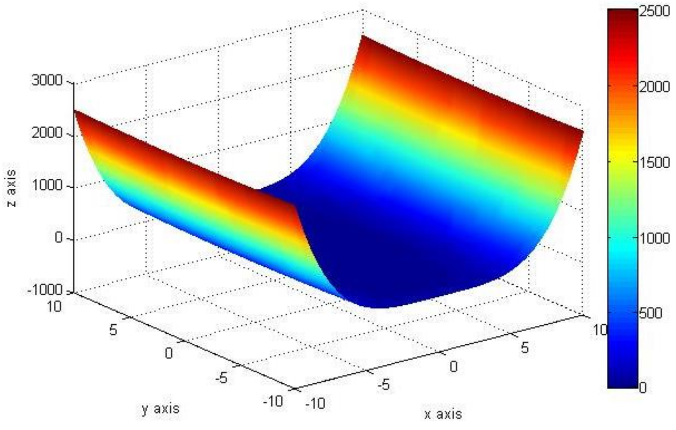

Fig 18: 3D surface plot for Zirilli or Aluffi-pentini's function

Table-16: Compression results for Aluffi-pentini's function

Meta heuristic Method	Calculated Value	Number of iterations	Rank
DE	-0.35232	18	-0.3523
PSO	-0.3523	51	-0.3523
ABC	-0.35239	12	-0.3523

We execute the all algorithms several times with the parameter setting given in table 1. In Rotated Ell Zirilli or Aluffi-Pentini's ipse function the optimum point is obtained is -0.35239 by all three methods but the performance of ABC is outstanding. ABC found the optimum value in 12 iterations but PSO performs 51 iterations and DE performs 18 iterations. So we conclude that ABC is the best choice to solve Zirilli or Aluffi-Pentini's function among other two functions. Final the ranking is also in favor of ABC.

Trecanni function [28]

Function is: $x_1^4 - 4x_1^3 + 4x_1 + x_2^2$

Range of function: $-5 \le x_i \le 5$

The global minimum is located at $(0,0)$

Function value at global minimum = 0

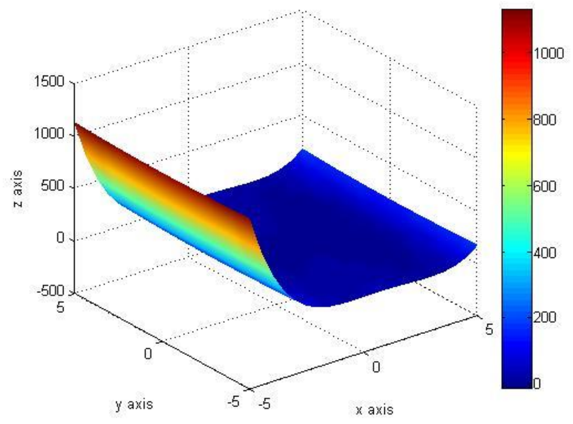

Fig 19: 3D surface plot for Trecanni function

Table-17: Compression results for Trecanni function

Meta heuristic Method	Calculated Value	Number of iterations	Rank
DE	-15.2344	23	2
PSO	-15.2344	43	3
ABC	-15.2344	12	1

We execute the all algorithms several times with the parameter setting given in table 1. In trecanni function the optimum point is obtained is -15.2344 by all three methods but the performance of ABC is outstanding. ABC found the optimum value in 12 iterations but PSO performs 43 iterations and DE performs 23 iterations. So we conclude that ABC is the best choice to solve trecanni's function among other two functions. Final the ranking is also in favor of ABC.

Wayburn Seader function [29]

Function is : $f_{15} = (x_1^6 + x_2^4 - 17)^2 + (2x_1 - x_2 - 4)^2$

Range of function: $-500 \le x_i \le 500$

The global minimum is located at $(0,0)$

Function value at global minimum = 0

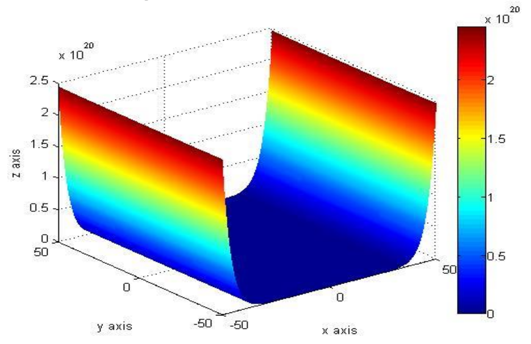

Fig 20: 3D surface plot for Wayburn Seader function

We execute the all algorithms several times with the parameter setting given in table 1. In wayburn function the optimum point is obtained is 0 by all three methods but the performance of DE is outstanding. DE found the optimum value in 580 iterations but PSO performs 1000 iterations and ABC performs 621 iterations. So we conclude that DE is the best choice to solve Wayburn function among other two functions. Final the ranking is also in favor of DE.

REFRENCES

[1] Averick, B.M., Carter, R.G., Moré, J.J., “The MINPACK-2 test problem collection (Preliminary version), Mathematics and Computer Science Division”, Argonne National Laboratory, (1991)

[2] Averick, B.M., Carter, R.G., Moré, J.J., Xue, G.L. “The MINPACK-2 test problem collection. Mathematics and Computer Science Division”, *Argonne National Laboratory, Preprint* MCS-P153-0692, (1992)

[3] Bongartz, I., Conn, A.R., Gould, N.I.M., Toint, P.L., “CUTE: constrained and unconstrained testing environments”, *ACM TOMS*, **21**: 123-160 (1995)

[4] Moré, J.J., Garbow, B.S., Hillstrom, K.E., “Testing unconstrained optimization software. ACM Trans”, *Math. Soft*. **7**:17-41 (1981)

[5] Himmelblau, D., “Applied nonlinear programming”. *McGraw-Hill*, New York, (1972)

[6] Chung, C. J., R. G. Reynolds, “CAEP: An Evolution-Based Tool for Real-Valued Function Optimization Using Cultural Algorithms”, *International Journal on Artificial Intelligence Tool*, **7**(3): 239-291 (1998)

[7] Winston, P. H. “Artificial Intelligence”, *Addison-Wesley*, (1992)

[8] Yao, X. Y. Liu, “Fast Evolutionary Programming,” *Proc. 5th Conf. on Evolutionary Programming*, (1996)

[9] Karaboga, D. and B. Basturk “On the performance of Artificial Bee Colony (ABC)”. *Applied Soft Computing* **8**(1): 687–697 (2008)

[10] Reynods C. “Flocks, herds, and schools: A behavioral model”, *Computer Graphics*; **21**(4):25–34. (1987)

[11] Boyd, R. Richerson P.J. “Culture and the evolutionary process.” *University of Chicago*, Chicago Press, (1985)

[12] Colorni, A. Dorigo M, Maniezzo V. “Distributed optimization by ant colonies”, *Proceedings of First European Conference on Artificial Life. Cambridge*, MA: MIT Press; p. 134–142 (1987)

[13] Kennedy, J. Eberhart R “Particle swarm optimization”. *Proceedings of IEEE International Conference on Neural Networks, IEEE Press*; **4**:1942–1948 (1995)

[14] Bonabeau, E. Dorigo M, Theraulaz G. “Swarm intelligence”: *From natural to artificial systems*. Oxford: Oxford University Press; (1999)

[15] Kennedy, J. Eberhart R. “Swarm intelligence”. *San Mateo, CA: Morgan Kaufmann*, (2001)

[16] Shi, Y. Eberhart R. “A modified particle swarm optimizer”, *Proceedings of IEEE International Conference on Evolutionary Computation* (ICEC’98) p. 69–73. Anchorage: IEEE Press (1998)

[17] Shi, Y. Eberhart R. “Parameter selection in particle swarm optimization”, *Proceedings of the 1998 Annual Conference on Evolutionary Programming*. San Diego: MIT Press, (1998)

[18] Kenneth V. price, Rainer M. Storn, Jouni A. Lampinen. “Differential Evolution”, *Springer-Verlag Berlin Heidelberg*, Germany (2005)

[19] Kwang, Y. Lee, Mohamed A. El-Sharkawi. “Modern Huristic Optimization Techniques”, Published *by John Wiley & Sons, Inc.*, Canada (2008)

[20] Ackley, D. H. “A Connectionist Machine for Genetic Hill-Climbing”, *Kluwer*, (1987)

[21] Branin, F. H. “Widely Convergent Method of Finding Multiple Solutions of Simultaneous Nonlinear Equations,” *IBM Journal of Research and Development*, **16**(5): 504-522, (1972)

[22] Lavi, A. T. P. Vogel, “Recent Advances in Optimization Techniques,” *John Wliley & Sons*, (1966).

[23] El-Attar, R. A. M. Vidyasagar, S. R. K. Dutta, “An Algorithm for II-norm Minimization” *SIAM Journal on Numerical Analysis*, **16**(1): 70-86, (1979)

[24] Hedar, A.-R. “Global Optimization Test Problems,” http://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar_files/TestGO.htm

[25] Price, K. V. R. M. Storn, J. A. Lampinen, “Differential Evolution: A Practical Approach to Global Optimization,” *Springer*, (2005)

[26] Moore, R. E. “Reliability in Computing,” *Academic Press*, (1998)

[27] Ali, M. M. C. Khompatraporn, Z. B. Zabinsky, “A Numerical Evaluation of Several Stochastic Algorithms on Selected Continuous Global Optimization Test Problems,” *Journal of Global Optimization*, 31: 635-672, (2005)

[28] Dixon, L. C., W. G. P. Szeg¨o.“Towards Global Optimization 2,” *Elsevier*, (1978)

[29] Wayburn, T. L. J. D. Seader, “Homotopy Continuation Methods for Computer-Aided Process Design,” *Computers Engineering*, **11**(1): 7-25, (1987)

[30] Tabassum, M. F., M. Saeed , A. Sana, Nazir Ahmad, “Solution of 7 Bar Tress Model using Derivative Free Methods”, *Proceedings of the Pakistan Academy of Sciences*, **52**(3): 265-271 (2015).

[31] Tabassum, M.F., M.Saeed, Nazir Ahmad and A.Sana, “Solution of War Planning Problem Using Derivative Free Methods”. Sci.Int., **27**(1): 395-398 (2015)

[32] Ali J., M. Saeed, N. A. Chaudhry and M. F. Tabassum. New Mathematical Models of N-Queens Problem and its Solution by a Derivative-Free Method of Optimization, Sci.Int., **27**(3): 2005-2008 (2015).

[33] Tabassum, M. F., M. Saeed, N. A. Chaudhry, A. Sana and Z. Zafar., Optimal Design of Oxygen Production System by Pattern Search Methods. Pakistan Journal of Science, **67**(4): 371-376 (2015).

[34] Sana, A., N. A. Chaudhry, M. Saeed, M. F. Tabassum, and M. Rafiq, Optimal Design of 16 Bar Truss Structure by Pattern Search Methods. Pakistan Journal of Science, **68**(1): 74-81 (2015)

[35] Tabassum, M.F., M.Saeed, N. A. Chaudhry, J. Ali and A. Sana, “Modeling and optimizing oxygen production system using differential evolution”. Sci.Int., **28**(1): 1-6 (2016)