CCET JOURNAL OF SCIENCE AND ENGINEERING EDUCATION
(ISSN 2455-5061) *Vol. – 4, Page-36-47, Year-2019*

# Comparison of Evolutionary Optimization Techniques for Unconstrained Continuous Optimization Problems

**Joji Thomas[1],Ganpat lal Rakesh[2] ,S. S. Mahapatra[3]**
[1]*Associate Professor, Mechanical Engineering, CCET Bhilai, Chhattisgarh, india,* ***joji.siji@gmail.com***
[2]*Assistant Professor, Mechanical Engineering, CCET Bhilai, Chhattisgarh, india,* ***ganpat123@gmail.com***
[3]*Professor, Mechanical Engineering, NIT Rourkela, Odisha, india,* ***ssm@nitrkl.ac.in***

**Abstract**

*This paper represents, a comparison between four Evolutionary Algorithms (EAs) i.e. Particle swarm optimization(PSO), Artificial Bee Colony algorithm(ABC), Shuffled frog leaping algorithm(SFL) and Imperialistic competitive algorithm (ICA) for solving optimization problems is made. These techniques can be useful to solve complicated real-world problems. Testing of these algorithms with standard problems is necessary to check their effectiveness. The basic versions of four algorithms are implemented in MATLAB and are applied to twenty-five unconstrained continuous optimization problems available in literature.*

***Index Terms:*** *EAs, Optimization, PSO, ABC, SFL, ICA*

## 1. Introduction

In computational science, optimization refers to the selection of a best element from some set of available alternatives. In the simplest case, this means solving problems in which one seeks to maximize (or to minimize) a real function by systematically choosing the values of real or integer variables from within an allowed set. Evolutionary optimization techniques can be used for getting near optimal solutions of difficult optimization problems. There are different types of Evolutionary techniques in use, natural evolutionary techniques e.g. GA,DE etc., swarm intelligence-based techniques e.g. PSO,ABC etc.andcultural algorithms e.g. ICA. Evolutionary algorithms are stochastic search methods that mimic the metaphor of natural biological evolution and/or the social or cultural behaviour of species. Researchers have developed computational systems that mimic the efficient behaviour of species such as birds, bees, and frogs as a means to seek faster and more robust solutions to complex optimization problems. The first evolutionary based technique introduced in the literature was the genetic algorithm. GAs was developed based on the Darwinian principle of the ‘survival of the fittest’ and the natural process of evolution through reproduction [1]. A popular swarm intelligence-based algorithm is the particle swarm optimization algorithm which was developed by Eberhart and Kennedy in 1995[2]. It models the social behaviour of bird flocking or fish schooling. Another swarm intelligence-based algorithm is artificial bee colony algorithm proposed by Karaboga in 2005[3], which mimic the foraging behaviour of a honeybee colony. Shuffled Frog Leaping algorithm developed by Eusuff and Lansey in 2003[4] is a mementicmetaheuristic based on the frog behaviour. In the SFL, the population consists of a set offrogs (solutions) that is partitioned into subsets referred to as memeplexes. The different memeplexes are considered as different cultures of frogs, each performing a local search. Within each memeplex, the individual frogs hold ideas, that can be influenced by the ideas of other frogs, and evolve through a process of mementic evolution. After a defined number of mementic evolution steps, ideas are passed among memeplexes in a shuffling process. The Imperialist Competitive Algorithm proposed by Atashpazet al. [5]is based on a socio-politically inspired optimization strategy. In this paper, the four EAs are reviewed and a flowchart for each algorithm is presented to facilitate its implementation. Performance comparison among the four algorithms is then presented. The standard versions of these techniques use pseudo-random numbers which cannot ensure the optimization’s ergodicity in solution space because they are absolutely random [6]therefore, a slight modification is done in each of the mentioned EAs by employing the chaotic operator (with *Logistic map)* in place of random number generator. Chaos method has the particular characteristics, such as the randomicity and ergodicity, which can enhance the diversity of the particles and actuate the particles to move out from the local near-optimal solutions[7].Comparison is also made between these modified EAs.

In each of these EAs a population of candidate solutions is generated randomly. These populations are called with different names in different techniques.

CCET JOURNAL OF SCIENCE AND ENGINEERING EDUCATION
(ISSN 2455-5061) *Vol. – 4, Page-36-47, Year-2019*

## 2. Evolutionary algorithms(EAs):

### 2.1 Particle Swarm Optimization(PSO)

PSO optimizes a problem by having a population of candidate solutions, here dubbed particles, and moving these particles around in the search-space according to simple mathematical formulae over the particle's position and velocity. Each particle's movement is influenced by its local best-known position called **pBest** and the best-known position of the entire swarm called **gBest**. In this way it is expected that swarm moves toward the best solution[2].

The position of a particle refers to a possible solution of the function to be optimized, which is updated in each iteration using formula 2. Here velocity of particle in each iteration is calculated using formula 1. If **R** is range of vector **x** then velocity is normally initialized randomly in the range [-R, +R].

$$\mathbf{v}_i = \omega\ \mathbf{v}_i + \varphi_p r_p\ (\mathbf{pBest}_i - \mathbf{x}_i) + \varphi_g r_g\ (\mathbf{gBest} - \mathbf{x}_i) \quad \text{..........}\ (1)$$

$$\mathbf{x}_i = \mathbf{x}_i + \mathbf{v}_i \quad \text{..........}\ (2)$$

The parameter ω is called the inertia weight and controls the magnitude of the old velocity in the calculation of the newvelocity, whereas $\varphi_p$ and $\varphi_g$ determine the significance of **pBest**and**gBest**respectively,$r_p$ and $r_g$are the random numbers generated in the range [0,1]. Furthermore, $\mathbf{v}_i$ at any iteration is constrained by the parameter $\mathbf{v}_{max}$which is normally taken about 20% of the range of **v.** If in any iteration position of the particle crosses the boundary then velocity is adjusted so that particles position reaches to the boundary which is called clamping of velocity.

In the modified version $r_p$and$r_g$are the chaotic numbersin the range [0,1].

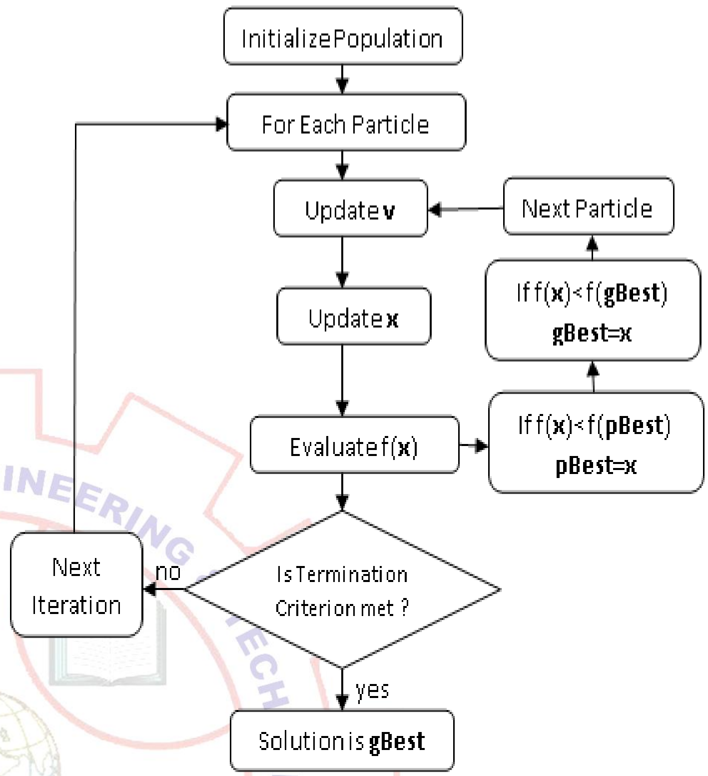


**Fig.1** The flowchart of PSO algorithm

### 2.2Artificial Bee Colony algorithm (ABC)

In ABC algorithm, the position of a food source represents a possible solution to the optimization problem. At initialization, a set of food source positions are randomly produced. The nectar amount retrievable from food source corresponds to the quality of the solution (fitness value) represented by that food source. Each cycle of the search consists of three steps after initialization stage: placing the employed bees onto the food sources and calculating their nectar amounts; placing the onlookers onto the food sources and calculating the nectar amounts and determining the scout bees and placing them onto the randomly determined food sources.
Each employed bee searches a nearby food source and checks its nectar amount, if new food source is having higher nectar(better fitness) then it forgets the previous food source and remembers only new one. At the second step, an onlooker prefers a food source area depending on the nectar information distributed by the employed bees. As the nectar amount of a food source increases, the probability of that foodsource chosen also increases. After selecting a food source onlooker bee searches, a nearby source and checks itsnectar amount, if new foodsource is having higher nectar(better fitness) then it forgets the previous food source and remembers only new one.

CCET JOURNAL OF SCIENCE AND ENGINEERING EDUCATION
(ISSN 2455-5061) *Vol. – 4, Page-36-47, Year-2019*

Searching of nearby food source by employed and onlooker bees is done according to following equation[3].

$$v_{ij} = x_{ij} + \emptyset_{ij}\,(x_{ij} - x_{kj})$$

Where $i, k \in \{1,2 \ldots\ldots NS\}$ $NS is number of food sources$

And $j \in \{1,2 \ldots\ldots D\}$ $D is dimension of the problem$. $k$ and $j$ are randomly chosen indexes. Although $k$ is determined randomly it should be different from $i$. $\emptyset_{ij}$ is a random number between [1,-1]. If a parameter value produced by this operation exceeds its predetermined limit, the parameter can be set to an acceptable value. Normally the value of the parameter exceeding its limit is set to its limit value (clamping).

Selection of a food source by an onlooker bee is done on the basis of probability value associated with that food source, $p_i$ calculated by the following expression [3].

$$p_i = \frac{fit_i}{\sum_{i=1}^{NS} fit_i}$$

where $fit_i$ is the fitness of $i^{th}$ food source.

For minimization type problems $fit_i$=1/(objective function value for $i^{th}$ food source).

In this work, after calculating probabilities selection of food source is done on the basis of Roulette Wheel Selection.

The food source which is abandoned by the bees is replaced with a new food source by the scouts. In ABC, if a position cannot be improved further through a predetermined number of cycles, then that food source is assumed to be abandoned. Suppose a food source $x_i$ is abandoned, then the scout discovers a new food source randomly to replace the $x_i$.

In the modified version chaotic number can be used to generate initial population, to search neighbouring food source by employed and onlooker bees, and to discover a new food source by scout bees. In this work chaotic numbers are used only in searching of neighbourhood solutions.

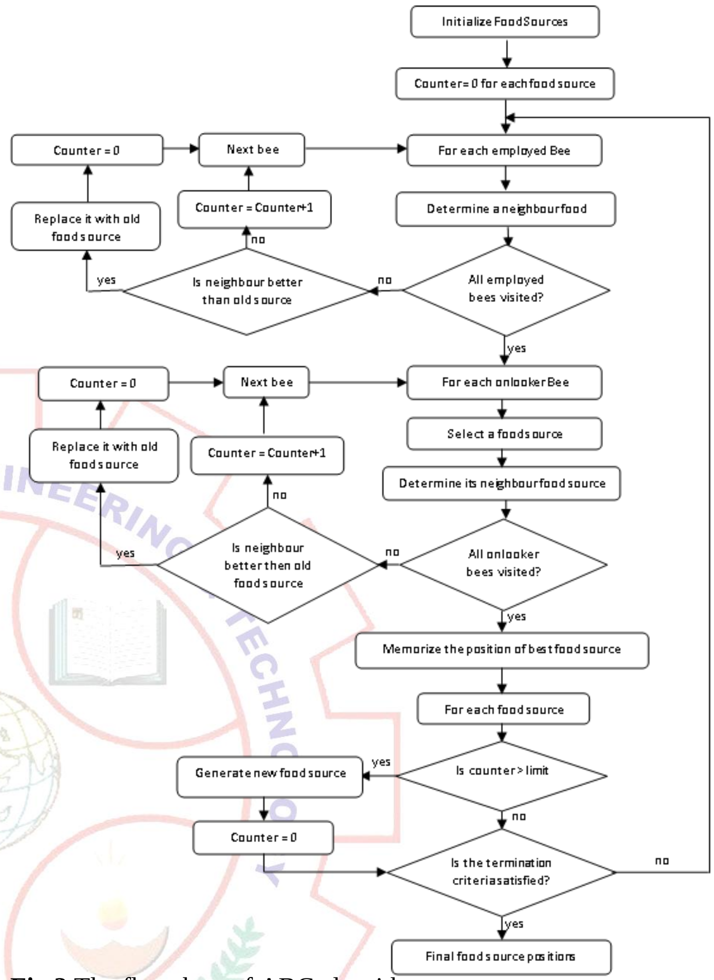


**Fig.2** The flowchart of ABC algorithm

## 2.3Shuffled frog leaping algorithm(SFL)

In the SFL, the population consists of a set of frogs (solutions) that is partitioned into subsets referred to as memplexes. The different memplexes are considered as different cultures of frogs, each performing a local search. Within each memplex, the individual frogs hold ideas, that can be influenced by the ideas of other frogs, and evolve through a process of mementic evolution. After a defined number of mementic evolution steps, ideas are passed among memplexes in a shuffling process [8]. The local search and the shuffling processes continue until defined convergence criteria are satisfied [4].

In SFL an initial population of *p* frogs is created randomly. Each frog ***x*** is a *D* dimensional vector and is a potential solution to the problem. Afterwards fitness value of each frog is calculated and frogs are sorted in descending order of their fitness. Then the entire population is divided into *m*memplexes with *n=p/m* frogs in each memplex. In this process first frog goes to first

CCET JOURNAL OF SCIENCE AND ENGINEERING EDUCATION
(ISSN 2455-5061) *Vol. – 4, Page-36-47, Year-2019*

memplex, second to second and $m$ to $m^{th}$memplex then $(m+1)^{th}$ frog will go to $1^{st}$memplex and so on.
Within each memplex best and worst frogs according to their fitness are termed as $x_b$ and $x_w$. Also, the best frog in entire population is termed as $x_g$. In the evolution process only, the worst frog changes its position according to following equations [4]
Change in frog position $S = rand().(x_b-x_w)$
New position of worst frog$x_w= x_w+SS_{max}\geq S\geq -S_{max}$
Where *rand()* is a rand number between 0 and 1; and $S_{max}$ is the maximum allowed change in a frog's position.
If fitness of new frog is better than the old frog then the worst frog is replaced with the new one, else the same equations is applied with $x_b$replaced by $x_g$. Even after applying above equations if solution is not improved then the worst frog is replaced by a randomly generated new frog. The calculations then continue for aspecific number of iterations. After this, all the memplexes are combined together and a new set of memplexes created after reshuffling. Process continues till the convergence criteria are not met.
In modified version, chaotic numbers in place of random numbers are used to change the frog position.

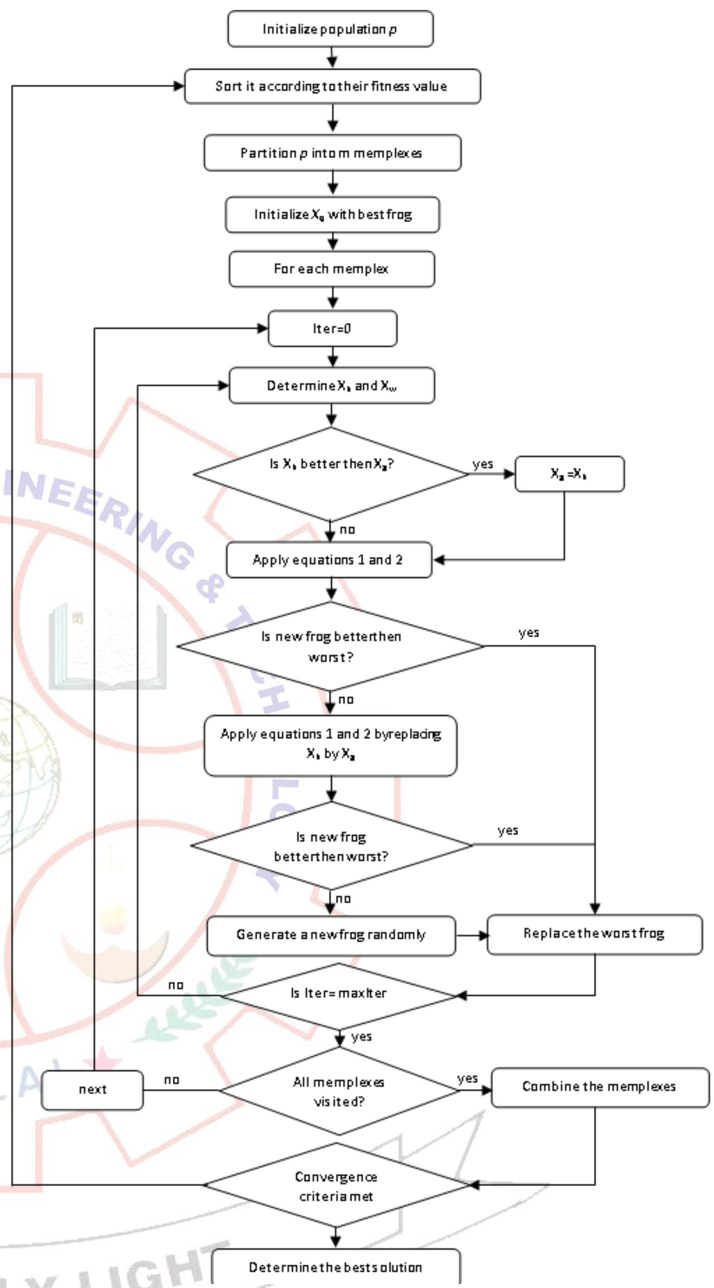


**Fig.3** The flowchart of SFLA algorithm

## 2.4Imperialistic competitive algorithm (ICA)

Like other evolutionary algorithm ICA starts with an initial population of potential solutions called countries of the world. Some of the best countries in the population are selected as imperialists and rest form the colonies of these imperialists. All the colonies are distributed among imperialists according to their power. More the power of an imperialist more number of colonies it possess. After dividing all the colonies among imperialists these colonies

CCET JOURNAL OF SCIENCE AND ENGINEERING EDUCATION
(ISSN 2455-5061) Vol. – 4, Page-36-47, Year-2019

start moving towards their relevant imperialist. Total power of an empire depends on both the power of the imperialist and power of its colonies. Thereafter an imperialistic competition begins amongst the empires. During competition any empire with no colony in it is eliminated from the competition. The movement of colonies towards their imperialist and competition among empires hopefully cause all the countries to converge to a state in which there exists only one empire in the world with all the countries having same power as of its imperialist [5].

In ICA an initial population of $p$ countries is generated. Each country $\boldsymbol{x}$is a $D$ dimensional vector and is a potential solution to the problem. Out of these $p$countries$N_{imp}$ of the most powerful countries are selected as the imperialist. The remaining countries $N_{col}$ will be colonies of the empires.

Initially colonies are divided among imperialists according to their power. Power of each imperialist is calculated according to the cost of the country the term cost is similar to the term fitness as used in other EAs.

Normalized cost of an imperialist $C_n$=$c_n$-max{$c_i$} where $c_n$ is cost(fitness) of $n^{th}$ imperialist.

Normalized power of each imperialist is defined by

$p_n = \left|\frac{C_n}{\sum C_i}\right|$, more the power of an imperialist more number of colonies will be allotted to it.

Initial number of colonies to $n^{th}$ empire will be

$$NC_n = round(p_n.N_{col})$$

out of $N_{col}$ ,the $NC_n$colonies are selected randomly and allotted to the nth empire.

In the second stage, colonies move towards their imperialist. A country$x_i^n$ of $n^{th}$ imperialist move towards its imperialist according to following equations [5]

$d = x_{imp}^n - x_i^n$ $d$is a vector representing distance between the $n^{th}$ imperialist and $i^{th}$ country of $n^{th}$imperialist

New position of the country will be given by

$x_i^n = x_i^n + rand.\beta.d$ (3)

Where $rand$ is a random number between [0,1] and $\beta$is a number greater than 1. In this equation $x_i^n$ and $d$ both are $D$ dimensional vector, multiplying a single random number to $d$ causes movement of colony towards imperialist along the line joining colony to imperialist. To search different points around the imperialist a random amount of deviation to the

direction of movement is given. This is done by multiplying different random numbers to different dimensions of $d$ [10].

If cost of the colony so modified is higher than its imperialist then both will exchange their position in the empire.

Total power of an empire is calculated according to the total cost of an empire which is defined as

$TC_n = C_{n,imp} + \xi.mean\{C_i^n\}$Where$\xi$ is a positive number which is considered to be less than one.

In imperialistic competition the weakest colony of the weakest empire is allotted to another empire. This allotment is done based on the total power of the empires. Each empire has a probability of getting the colony. Probability of each empire is calculated based on the normalized total cost of the empire given by

$NTC_n = TC_n - \max\{TC_i\}$where$TC_n$ is the total cost of the nth empire. Having the normalized total cost, the possession probability of each empire is given by

$p_{pos} = \left|\frac{NTC_n}{\sum NTC_i}\right|$now form a vector containing possession probability of all the empires.

$$P = [p_{pos\,1}, p_{pos\,2}, p_{pos\,3} \dots p_{posNimp}]$$

Then create a vector of random numbers in a range [0,1] having a size of $P$.

$$R = [r_1, r_2, r_3, \dots \dots \dots . r_{Nimp}]$$

Then calculate

$$PROB = P - R$$

Referring to vector$PROB$, the weakest colony will be given to an empire whose relevant index in $PROB$ is maximum.

After imperialistic competition if any empire has no colony then the empire is eliminated and its imperialist is allotted to an empire with highest power.

Iterations will be terminated when only one empire remains or if cost of all empires becomes same.

In chaotic version, chaotic number in place of random number is used in equation (3)[11].

CCET JOURNAL OF SCIENCE AND ENGINEERING EDUCATION
(ISSN 2455-5061) *Vol. – 4, Page-36-47, Year-2019*

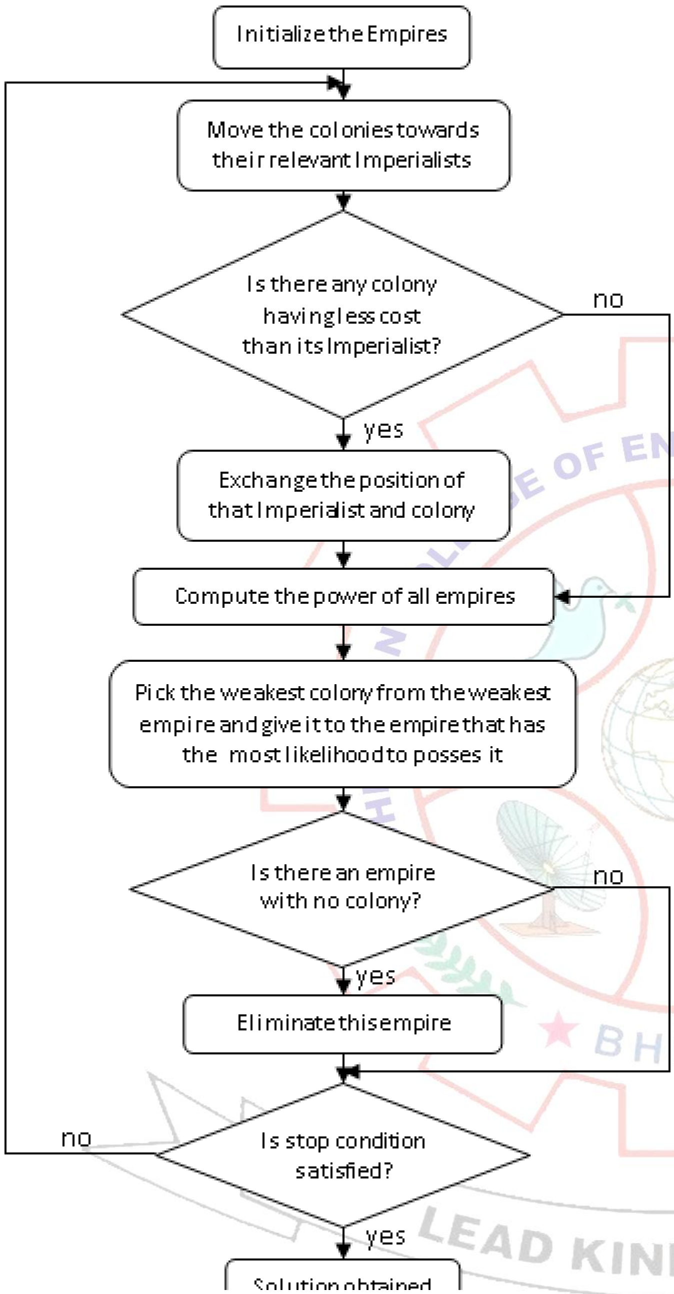


**Fig.4**The flowchart of ICA algorithm

## Settings:

Each algorithm has its own parameters that affect its performance in terms of solution quality and processing time. To obtain the best solutions from each algorithm initial settings are made according to previously reported values in the literature [2, 3, 4, 5]. Then these values are altered to get best solutions possible by these algorithms.

In all the algorithms random numbers are generated using same seed. All problems are of minimization type therefore fitness value is equal to reciprocal of objective function value. Maximum number of cycles used for any algorithm is 80000. During the iterations if any

solution generated is violating the bound constraints then it is reset to its nearest boundary [12].
In modified version of each method chaotic numbers are generated using the formula [9]

$$N_{k+1} = \mu . N_k . (1 - N_k)$$

in this paper value of $\mu$ is taken as 0.4 and initial chaotic number $N_0$ is taken as 0.1.

**PSO settings[2,13]**:
Cognitive and social components ($\varphi_p$and$\varphi_g$in (2))are constants that can be used to change the weighting between personal and population experience, respectively. In our experiments value of $\varphi_p$ and $\varphi_g$ are set to in between 1.4 to 2. Inertia weight ω was taken in between 0.8 to1.2. Initial population size is taken as 20. Maximum velocity is clamped to 20% of the total range of velocity.
**ABC setting[3]:**
Size of initial population is taken as 25. Parameter limit is the maximum number of cycles for which a food source. If a food source does not improve in a predetermined number of cycles then it is abandoned. This number of cycles is called limit and is set to *SN*D*, where *SN* is number of food sources and *D* is dimension of the problem.
**FLA setting[4]:**
Size of initial population is taken as 200. Number of memplexes is equal to 20. Number of iterations within each memplex is taken as 10. Maximum allowed change in the frog position $S_{max}$ is taken about 25% of the range of frog position.
**ICA setting[5]:**
Number of countries is taken as 80. Number of imperialists is either 8 or 9. Value of $\beta$is taken as 2 and value of $\xi$ is taken as 0.1.

**Benchmark functions:**
To compare the performance of four EAs, 25 benchmark problems for continuous optimization are used[3]. A description ofthese test problems is given in the table 1.

## 3.Results and discussion.

the results obtained by solving these test problems are summarized in tables 2,3 and 4. The tests are performed to check whether actual solution is obtained within specified number of iterations. Also processing time for each algorithm is calculated to measure the speed of each EA, because the number of generations in each

CCET JOURNAL OF SCIENCE AND ENGINEERING EDUCATION
(ISSN 2455-5061) *Vol. – 4, Page-36-47, Year-2019*

evolutionary cycle is different from one algorithm to another.

**Table 1**: Benchmark functions used in experiments *D*:dimension,*C*:characteristics,*U*:unimodal,*M*:multimodal, *S*:separable,*N*:non-separable

| Funct | FunctionName | Fuction | D | C | Range |
|---|---|---|---|---|---|
| $f_1$ | Easom | $f(x) = -\cos(x_1)\cos(x_2)\exp(-(x_1-\pi)^2-(x_2-\pi)^2)$ | 2 | *UN* | $[-100,100]$ |
| $f_2$ | Matyas | $f(x) = 0.26(x_1^2+x_2^2) - 0.48x_1x_2$ | 2 | *UN* | $[-10,10]$ |
| $f_3$ | Quartic | $f(x) = \sum_{i=1}^{D} ix_i^4 + random[0,1]$ | 30 | *US* | $[-1.28,1.28]$ |
| $f_4$ | sphere | $f(x) = \sum_{i=1}^{D} x_i^2$ | 30 | *US* | $[-100,100]$ |
| $f_5$ | StepInt | $f(x) = 25 + \sum_{i=1}^{D} x_i$ | 5 | *US* | $[-5.12,5.12]$ |
| $f_6$ | Step | $f(x) = \sum_{i=1}^{D} ([x_i+5])^2$ | 30 | *US* | $[-100,100]$ |
| $f_7$ | SumSquares | $f(x) = \sum_{i=1}^{D} ix_i^2$ | 30 | *US* | $[-10,10]$ |
| $f_8$ | Trid10 | $f(x) = \sum_{i=1}^{D} (x_i-1)^2 - \sum_{i=2}^{D} x_i x_{i-1}$ | 10 | *UN* | $[-D^2,D^2]$ |
| $f_9$ | Trid6 | $f(x) = \sum_{i=1}^{D} (x_i-1)^2 - \sum_{i=2}^{D} x_i x_{i-1}$ | 6 | *UN* | $[-D^2,D^2]$ |
| $f_{10}$ | Zakharov | $f(x) = \sum_{i=1}^{D} x_i^2 + (\sum_{i=1}^{D} 0.5ix_i)^2 + (\sum_{i=1}^{D} 0.5ix_i)^4$ | 10 | *UN* | $[-5,10]$ |
| $f_{11}$ | Bohchevsky1 | $f(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0.7$ | 2 | *MS* | $[-100,100]$ |
| $f_{12}$ | Bohchevsky2 | $f(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1)(4\pi x_2) + 0.3$ | 2 | *MN* | $[-100,100]$ |
| $f_{13}$ | Bohchevsky3 | $f(x) = x_1^2 + 2x_2^2 - 0.3\cos((3\pi x_1) + (4\pi x_2)) + 0.3$ | 2 | *MN* | $[-100,100]$ |
| $f_{14}$ | CamelBack | $f(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$ | 2 | *MN* | $[-5,5]$ |
| $f_{15}$ | Colville | $f(x) = 100(x_1^2-x_2)^2 + (x_1-1)^2 + (x_3-1)^2 + 90(x_3^2-x_4)^2 + 10.1((x_2-1)^2 + (x_4-1)^2) + 19.8(x_2-1)(x_4-1)$ | 4 | *UN* | $[-10,10]$ |
| $f_{16}$ | DixonPrice | $f(x) = (x_i-1)^2 + \sum_{i=2}^{D} i(2x_i^2 - x_{i-1})^2$ | 30 | *UN* | $[-10,10]$ |
| $f_{17}$ | Michalewicz10 | $f(x) = -\sum_{i=1}^{D} \sin(x_i)(\sin\left(\frac{ix_i^2}{\pi}\right))^{20}$ | 10 | *MS* | $[0,\pi]$ |
| $f_{18}$ | Michalewicz2 | $f(x) = -\sum_{i=1}^{D} \sin(x_i)(\sin\left(\frac{ix_i^2}{\pi}\right))^{20}$ | 2 | *MS* | $[0,\pi]$ |
| $f_{19}$ | Michalewicz5 | $f(x) = -\sum_{i=1}^{D} \sin(x_i)(\sin\left(\frac{ix_i^2}{\pi}\right))^{20}$ | 5 | *MS* | $[0,\pi]$ |
| $f_{20}$ | Rastrigin | $f(x) = \sum_{i=1}^{D} [x_i^2 - 10\cos(2\pi x_i) + 10]$ | 30 | *MS* | $[-5.12,5.12]$ |
| $f_{21}$ | Rosenbrock | $f(x) = \sum_{i=1}^{D-1} [100\ (x_{i+1} - x_i^2)^2 + (x_i-1)^2]$ | 30 | *UN* | $[-30,30]$ |
| $f_{22}$ | Schwefel | $f(x) = \sum_{i=1}^{D} -x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)$ | 30 | *MS* | $[-500,500]$ |
| $f_{23}$ | Akley | $f(x) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)) + 20 + e$ | 30 | *MN* | $[-32,32]$ |
| $f_{24}$ | Griewank | $f(x) = 1 + \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos(\frac{x}{\sqrt{i}})$ | 30 | *MN* | $[-600,600]$ |
| $f_{25}$ | Powell | $f(x) = \sum_{i=1}^{n/4} (x_{4i-3} + 10x_{4i-2})^2 + 5(x_{4i-1} - x_{4i})^2 + (x_{4i-2} - x_{4i-1})^4 + 10(x_{4i-3} - x_{4i})^4$ | 24 | *UN* | $[-4,5]$ |

PSO with other algorithms: results obtained shows that PSO gives fairly good solutions to the almost all problems except for functions $f_{16}$,$f_{17}$,$f_{19}$, $f_{20}$, $f_{21}$ and $f_{22}$ where it is failed to reach optimum solution, solution obtained for $f_{17}$ and$f_{22}$ is better than other EAs. Out of these functions $f_{16}$,$f_{21}$ are unimodal no separable while other functions are multimodal separable functions. Main advantage of PSO algorithm is its simplicity and it reaches to the solution quickly as compared to other algorithms. Drawback of this algorithm is that it is verysensitive to its control parameters even a slight variation in any of these parameters may prevent the algorithm to reach the actual solution. It is also observed that same parameter setting cannot work for all problems.

ABC with other algorithms: ABC algorithm is unable to reach solution in functions $f_{17}$,$f_{19}$ and $f_{22}$ but it reached to the exact solution of function $f_{20}$ and fairly good solution to $f_{16}$ and$f_{21}$ . All other EAs are unable to reach the solution of these three functions. Main disadvantage of ABC algorithm is that it takes more CPU time to reach the solution.

FLA with other algorithms: basic FLA algorithm do not give as good solution as compared to PSO and ABC. It is unable to reach solution in functions $f_5$, $f_{16}$, $f_{17}$, $f_{20}$, $f_{21}$, $f_{22}$, $f_{23}$,$f_{24}$. In case of function $f_{19}$ it gives best solution as compared to all other EAs.

ICA with other algorithms: similar to FLA it also does not give good solutions as compared to PSO and ABC. Advantage of this algorithm is that it reaches to optimum quickly. In less number of iterations one can know whether optimum value can be obtained or not.

Effect of using chaotic numbers in place of random numbers is also studied. Chaotic numbers can be used in initializing the population and/or movement of population towards optimum. Conducting large numbers of test indicate that using chaotic numbers in movement of population with initialization of population by random numbers gives better result.

PSO with chaotic number gives better result in almost all problems. PSO with chaotic number reaches to near optimum value in case of functions $f_{20}$, $f_{21}$. For these two functions basic PSO unable to reach near optimum value.

ABC with chaotic numbers gives almost same result as given by the ABC with random numbers but for some functions it reaches to optimum value quickly as compared to ABC with random numbers.

In case of FLA and ICA not much better effect is observed in using chaotic numbers in place of random numbers.

In this work three representative cases of function $f_1$,$f_2$,$f_{18}$ are taken. These functions are selected because they are two dimensional functions and therefore, they can be represented by surface plots. Also, all EAs in study reached to the final solution in each case. Figs 5,8,11 represent the surface plot of $f_1$,$f_2$,$f_{18}$ functions respectively. Figs 6,9,12 shows the convergence curves for these three functions. Figs 7,10,13 shows CPU time taken by these algorithms to reach final solution. Comparing figs 6 and 7 shows that FLA and ICA reached to the solutions before

CCET JOURNAL OF SCIENCE AND ENGINEERING EDUCATION
(ISSN 2455-5061) *Vol. – 4, Page-36-47, Year-2019*

10 iterations and PSO took about 40 iterations but time taken to reach final solution by PSO is less than time taken by FLA and ICA. Same is the case with other functions. Which shows PSO is computationally less expensive as compared to other algorithms.

Figs 14 and 15 shows the surface plot of functions$f_{20}$ and $f_{21}$ with two variables respectively. Surface plots of these functions shows how complex these functions are.

**Table 2**: Function values obtained with random numbers

| Funct | Actual value | PSO | ABC | FLA | ICA |
|---|---|---|---|---|---|
| $f_1$ | -1 | -0.999803974 | -0.996252324 | -1 | -1 |
| $f_2$ | 0 | 3.02E-12 | 2.37E-05 | 2.05E-17 | 1.62E-96 |
| $f_3$ | 0 | 1.94E-06 | 0.032402517 | 0.00096448 | 0.30682399 |
| $f_4$ | 0 | 0.00E+00 | 2.31E-10 | 1.20E+02 | 1.50E+02 |
| $f_5$ | -0.6 | -0.6 | -0.6 | 3.937436149 | -0.6 |
| $f_6$ | 0 | 8.79E-07 | 1.44E-05 | 1.02E+00 | 9.22E-01 |
| $f_7$ | 0 | 0 | 2.82E-08 | 5.01E+00 | 5.60E+01 |
| $f_8$ | -210 | -209.9999953 | -208.3080852 | -161.6337868 | -208.7890396 |
| $f_9$ | -50 | -50 | -49.68230577 | -48.14794998 | -49.99994878 |
| $f_{10}$ | 0 | 6.08E-08 | 1.60E-05 | 4.05E-01 | 2.88E-08 |
| $f_{11}$ | 0 | 0 | 0 | 0 | 0 |
| $f_{12}$ | 0 | 0 | 0 | 0 | 0 |
| $f_{13}$ | 0 | 0 | 5.00E-16 | 0 | 0 |
| $f_{14}$ | -1.0316 | -1.031628453 | -1.031597588 | -1.031628453 | -1.031628453 |
| $f_{15}$ | 0 | 1.92E-09 | 0.03580093 | 0.094817296 | 0.155896041 |
| $f_{16}$ | 0 | 3.85E-05 | 1.41E-15 | 1.50E+00 | 3.86E+02 |
| $f_{17}$ | -9.6602 | -8.868615341 | -4.765984836 | -7.455034029 | -8.604502165 |
| $f_{18}$ | -1.8013 | -1.801302254 | -1.7988017 | -1.80130341 | -1.80130341 |
| $f_{19}$ | -4.6877 | -4.619018042 | -3.45008875 | -4.594201266 | -4.687658179 |
| $f_{20}$ | 0 | 0.000205978 | 0 | 8.600078383 | 179.147263 |
| $f_{21}$ | 0 | 0.054097263 | 0.000381093 | 2755.171552 | 14887.80809 |
| $f_{22}$ | -12570 | -10185.54554 | -4120.914826 | -5525.188892 | -7832.116382 |
| $f_{23}$ | 0 | 3.02E-05 | 3.11E-14 | 4.96E+00 | 1.82E+01 |
| $f_{24}$ | 0 | 0.000240859 | 0 | 6.150562692 | 1.6339931 |
| $f_{25}$ | 0 | 0 | 0.001155778 | 0.741950855 | 0.972720905 |

**Table 3**: Function values obtained with chaotic numbers

| Funct | Actual value | PSO | ABC | FLA | ICA |
|---|---|---|---|---|---|
| $f_1$ | -1 | -0.999999 | -0.96959 | -1 | -1 |
| $f_2$ | 0 | 4.23E-18 | 1.04E-07 | 4.09E-36 | 1.04E-22 |
| $f_3$ | 0 | 7.35E-05 | 0.003638 | 0.001169 | 0.552213 |
| $f_4$ | 0 | 1.95E-06 | 6.36E-07 | 0.261104 | 184.2378 |
| $f_5$ | -0.6 | -0.6 | -0.6 | 5.333682 | -0.6 |
| $f_6$ | 0 | 1.02E-06 | 1.11E-06 | 0.020096 | 2.769053 |
| $f_7$ | 0 | 1.88E-05 | 2.47E-10 | 0.050441 | 61.70511 |
| $f_8$ | -210 | -209.91 | -208.572 | -198.6797 | -146.15 |
| $f_9$ | -50 | -50 | -49.7213 | -49.93849 | -49.8903 |
| $f_{10}$ | 0 | 1.69E-06 | 3.19E-07 | 0.040994 | 0.046523 |
| $f_{11}$ | 0 | 0 | 0 | 0 | 0 |
| $f_{12}$ | 0 | 7.94E-15 | 0 | 0 | 0 |
| $f_{13}$ | 0 | 1.14E-13 | 8.33E-16 | 0 | 0 |
| $f_{14}$ | -1.0316 | -1.031628 | -1.03142 | -1.031628 | -1.03163 |
| $f_{15}$ | 0 | 0 | 0.003742 | 0.023713 | 0.000762 |
| $f_{16}$ | 0 | 0.1657129 | 1.65E-15 | 1.354628 | 156.7547 |
| $f_{17}$ | -9.6602 | -8.462056 | -4.20624 | -8.10614 | -8.38376 |
| $f_{18}$ | -1.8013 | -1.801303 | -1.80091 | -1.801303 | -1.8013 |
| $f_{19}$ | -4.6877 | -3.537656 | -3.58046 | -4.641834 | -4.5249 |
| $f_{20}$ | 0 | 49.460675 | 0 | 20.44472 | 173.3691 |
| $f_{21}$ | 0 | 119.45312 | 0.017183 | 37.02363 | 59523.26 |
| $f_{22}$ | -12570 | -9349.24 | -3725.55 | -6135.667 | -7798.55 |
| $f_{23}$ | 0 | 0.0004139 | 3.11E-14 | 3.478317 | 18.54469 |
| $f_{24}$ | 0 | 0.0078345 | 0 | 4.711405 | 2.075595 |
| $f_{25}$ | 0 | 0 | 0.000886 | 0.175554 | 1.747874 |

CCET JOURNAL OF SCIENCE AND ENGINEERING EDUCATION
(ISSN 2455-5061) *Vol. – 4, Page-36-47, Year-2019*

**Table 4**: CPU Time required to solve the functions

| Funct | RANDOM NUMBERS | | | | CHAOTIC NUMBERS | | | |
|---|---|---|---|---|---|---|---|---|
| | PSO | ABC | FLA | ICA | PSO | ABC | FLA | ICA |
| $f_1$ | 0.016502 | 33.83684 | 0.893104 | 0.096373 | 0.066048 | 50.53339 | 0.855622 | 0.097648 |
| $f_2$ | 0.019879 | 0.867564 | 0.811713 | 0.161606 | 0.013653 | 1.672332 | 0.795861 | 0.287446 |
| $f_3$ | 68.5021 | 60.69918 | 9.187863 | 0.318333 | 6.842619 | 60.48162 | 9.49641 | 0.344658 |
| $f_4$ | 6.05692 | 1.120752 | 7.770823 | 0.246303 | 0.021853 | 1.116779 | 8.149328 | 0.301078 |
| $f_5$ | 0.0114 | 0.113665 | 1.389757 | 0.111874 | 0.011119 | 0.077554 | 1.449402 | 0.110728 |
| $f_6$ | 8.490175 | 1.128224 | 15.56196 | 0.261273 | 4.043949 | 1.119945 | 16.09934 | 0.290571 |
| $f_7$ | 25.33892 | 1.824721 | 15.31768 | 0.284447 | 0.136936 | 0.925001 | 15.82429 | 0.274917 |
| $f_8$ | 31.41636 | 124.9622 | 70.79752 | 0.21464 | 21.09176 | 17.83803 | 72.96034 | 0.57861 |
| $f_9$ | 0.362898 | 142.2193 | 13.95108 | 0.200398 | 0.296376 | 17.5011 | 14.41284 | 0.438329 |
| $f_{10}$ | 50.38727 | 37.78305 | 73.10906 | 0.427847 | 11.38864 | 93.43288 | 75.61987 | 0.836852 |
| $f_{11}$ | 0.028142 | 6.062968 | 0.825795 | 0.136575 | 0.011914 | 0.646761 | 0.833336 | 0.166509 |
| $f_{12}$ | 0.027257 | 6.0807 | 0.831346 | 0.134025 | 0.011009 | 0.643936 | 0.828913 | 0.16151 |
| $f_{13}$ | 0.027101 | 6.025627 | 0.821557 | 0.161276 | 0.011028 | 6.094848 | 0.821313 | 0.189372 |
| $f_{14}$ | 0.136366 | 12.5519 | 0.842671 | 0.187172 | 0.132276 | 62.51955 | 0.831682 | 0.215261 |
| $f_{15}$ | 0.140751 | 58.80686 | 0.7416 | 0.206682 | 7.341341 | 59.77813 | 0.759008 | 0.6792 |
| $f_{16}$ | 55.37082 | 6.225945 | 15.76635 | 0.296448 | 55.37811 | 6.215984 | 79.45357 | 0.276883 |
| $f_{17}$ | 42.22675 | 68.53247 | 7.569207 | 0.196749 | 67.93186 | 68.68522 | 7.605441 | 0.222201 |
| $f_{18}$ | 0.289486 | 64.55791 | 0.856337 | 0.171559 | 0.289207 | 64.47482 | 0.844151 | 0.186828 |
| $f_{19}$ | 24.38759 | 65.40774 | 0.802471 | 0.174577 | 24.27668 | 65.69026 | 0.811212 | 0.16401 |
| $f_{20}$ | 107.1953 | 4.025435 | 77.89875 | 0.246993 | 104.7837 | 3.956159 | 80.22436 | 1.269086 |
| $f_{21}$ | 29.95885 | 37.27963 | 7.818207 | 0.353399 | 29.75106 | 61.81118 | 7.999381 | 0.301325 |
| $f_{22}$ | 20.51256 | 72.62486 | 8.307441 | 0.248584 | 20.83316 | 71.54258 | 8.489359 | 1.367079 |
| $f_{23}$ | 42.45895 | 6.774194 | 16.05082 | 0.101525 | 42.66888 | 6.801612 | 1.752358 | 0.286846 |
| $f_{24}$ | 41.71973 | 7.053963 | 81.65986 | 0.32731 | 27.70661 | 7.130018 | 83.14513 | 1.293629 |
| $f_{25}$ | 0.493707 | 90.4273 | 15.59134 | 0.42767 | 0.495026 | 65.26951 | 15.86635 | 0.314041 |

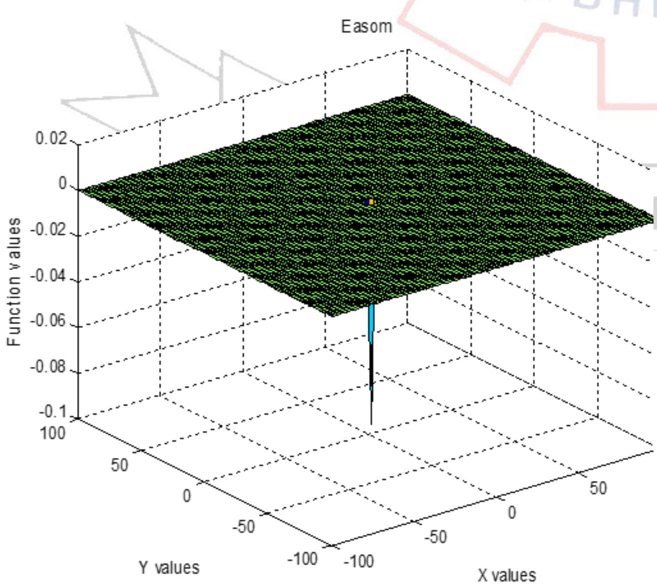


**Fig.5** Surface plot of Easomfunction($f_1$)

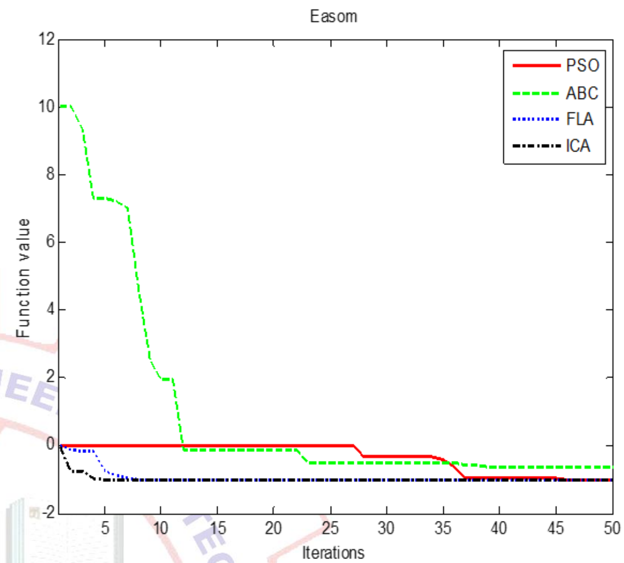


**Fig.6** Convergence curve for Easomfunction($f_1$)

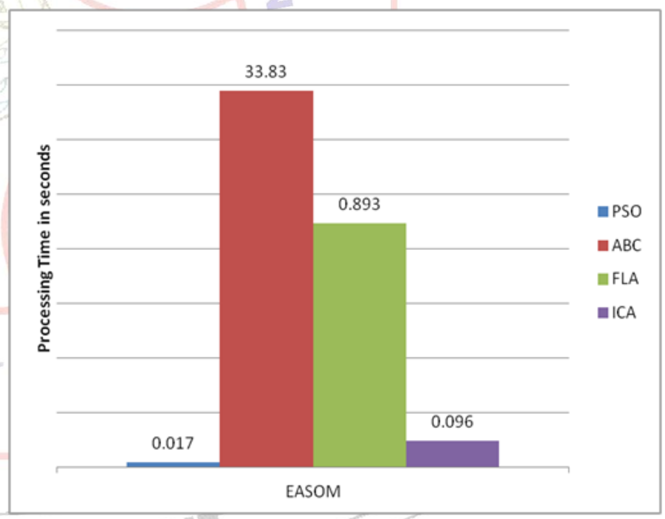


**Fig.7** Time taken to reach solution of Easomfunction($f_1$)

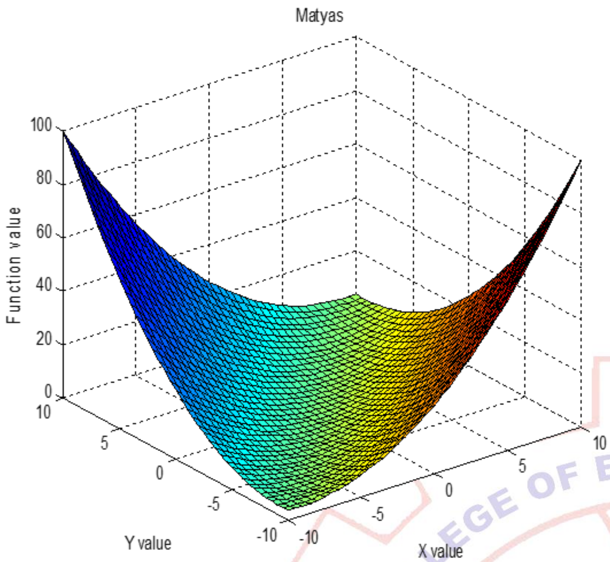


**Fig.8** Surface plot of Matyasfunction($f_2$)

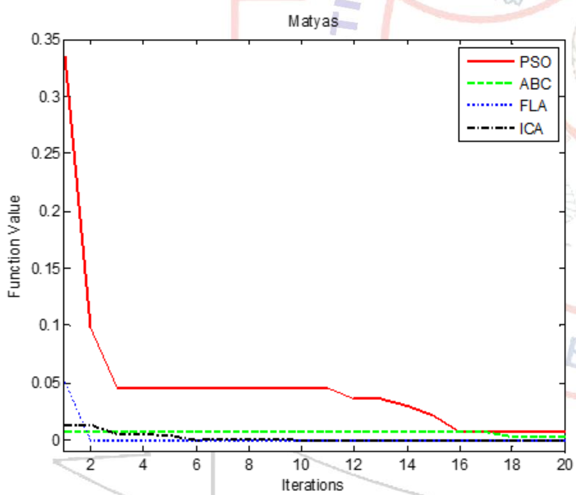


**Fig.9**Convergence curve for Matyasfunction($f_2$)

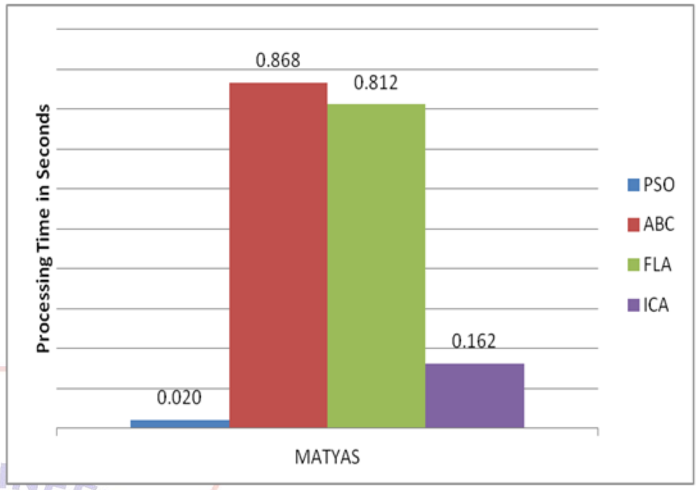


**Fig.10**Time taken to reach solution of Matyasfunction($f_2$)

**Fig.11**Surface plot of Michalewicz2 function($f_{18}$)

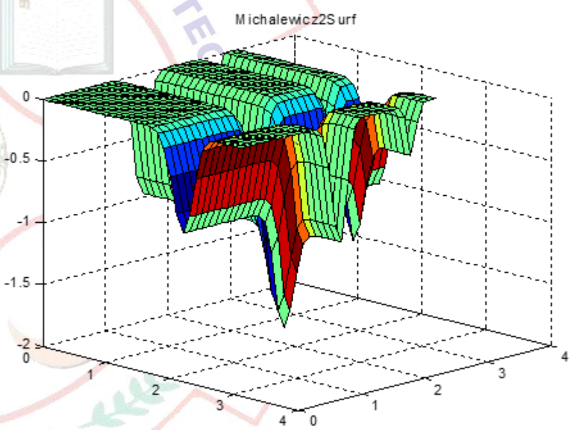


**Fig.11**Surface plot of Michalewicz2 function($f_{18}$)

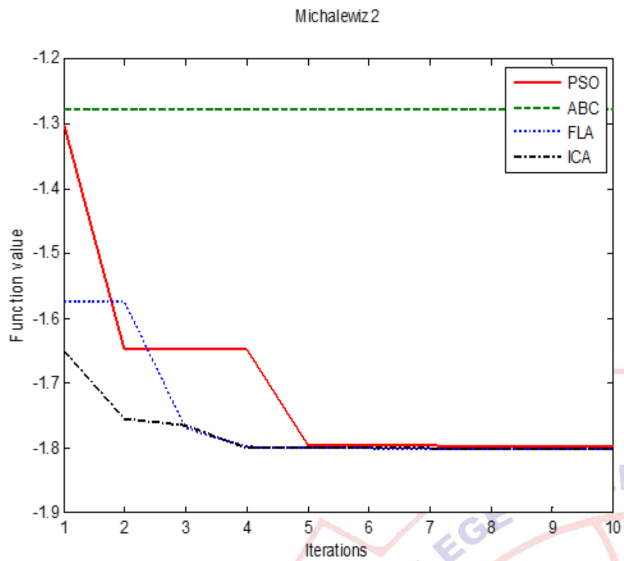


**Fig.12**Convergence curve for Michalewicz2 function($f_{18}$)

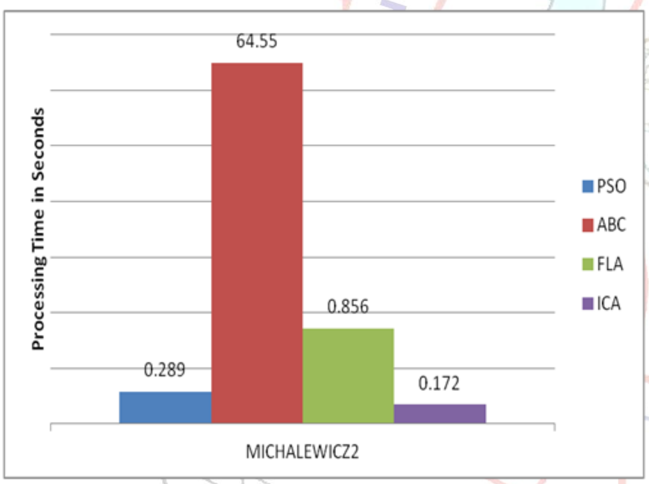


**Fig.13**Time taken to reach solution of Michalewicz2 function($f_{18}$)

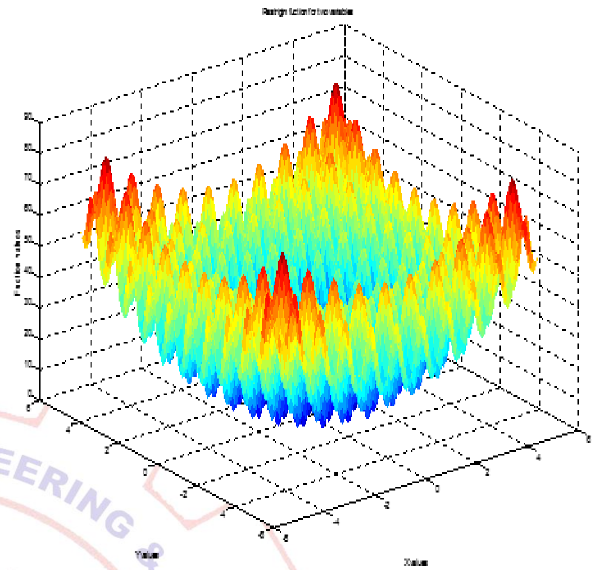

**Fig.14**Surface plot of Rastrigin function for two variables

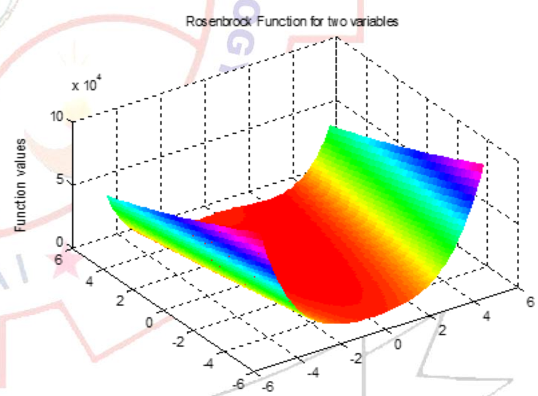


**Fig.15**Surface plot of Rosenbrock function with two variables

## 4. CONCLUSION

In this paper a comparison among four different EAs were presented. A brief description of each method along with flow chart is presented to facilitate their implementation. Programs are written in MATLAB to implement each algorithm. Twenty-five continuous optimization problems were solved using these algorithms. To explore the effect of using chaotic number inplace of random number, modificationswere done in these algorithms. Comparison indicates that a single technique cannot be used to get solution of all types of problems. PSO with chaotic number gives solution to almost all problems but parameter setting required to get these solutions are different for different problems. For a new problem one has to solve same problem with different settings to get reasonable solution.

## REFERENCES

[1]. D E Goldberg, "Genetic Algorithms in Search, Optimization, and Machine Learning "Addison-Wesley(1989).

[2]. J. Kennedy, R.C. Eberhart, in: Particle Swarm Optimization, 1995 IEEE International Conference on Neural Networks, vol. 4, 1995, pp. 1942–1948.

[3]. Karaboga, D. andAkay, B. (2009), "A comparative study of Artificial Bee Colony algorithm", *Applied Mathematics and Computation*, Vol. 214, No. 1, pp. 108-132.

[4]. Eusuff, M.M. and Lansey, K.E., Optimization of water distribution network design using the shuffled frog leaping algorithm. Journal of WaterResources Planning and Management ASCE, 2003, 129, 210 – 225.

[5]. E. AtashpazGargari & Caro Lucas, "Imperialist Competitive Algorithm: An Algorithm for Optimization Inspired by Imperialistic Competition", IEEE Congress on Evolutionary Computation, Singapore, 2007, pp 4661-4667.

[6]. Zheng Zhang, Fujun Shi and XinjiaGu, "A Rule-based Classifier by Adaptive Chaotic PSO"International Journal of Research and Reviews in Soft and Intelligent Computing Vol. 1, No. 1, March 2011

[7]. Leandro dos Santos Coelho, VivianaCoccoMariani, "Use of chaotic sequences in a biologically inspired algorithm for engineering design optimization", Expert Systems with Applications 34 (2008) 1905–1913

[8]. Liong S-Y, Atiquzzaman Md. Optimal design of water distribution network using shuffled complex evolution. Journal of The Institution of Engineers, Singapore 2004;44(1):93–107.

**[9].** Bilal Alatas, Chaotic bee colony algorithms for global numerical optimization, Expert Systems with Applications, Volume 37, Issue 8, August 2010, Pages 5682-5687

**[10].** Yang Zhang, Yong Wang, Cheng Peng "Improved Imperialist Competitive Algorithm for Constrained Optimization"2009 International Forum on Computer Science-Technology and Applications,25-27 Dec. 2009, pg 204 – 207,

**[11].** S. Talatahari, B. FarahmandAzar, R. Sheikholeslami, A.H. Gandomi, Imperialist competitive algorithm combined with chaos for global optimization, Communications inNonlinear Science and Numerical Simulation, Volume 17, Issue 3, March 2012, Pages 1312-1319,

**[12].** Sabine Helwig and Rolf Wanka, Particle Swarm Optimizationin High-Dimensional Bounded Search Spaces, Proceedings of the 2007 IEEE Swarm Intelligence Symposium, pp. 198–205.

**[13].** F. Bergh, A.P. Engelbrecht, A study of particle swarm optimization particle trajectories, Information Sciences 176 (2006) 937–971.

**[14].** K DebOptimization for engineering design,PHI Learning Pvt. Ltd,2004