HYBRID BACTERIAL FORAGING WITH FIREFLY ALGORITHM

MAJOR PROJECT SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF DEGREE OF

Master of Technology

In

Information Systems

Submitted By:

TEJNA PATODI

(2K12/ISY/25)

Under the Guidance of

Dr. O. P. Verma

(Prof. and Head, Department of IT)

DEPARTMENT OF INFORMATION TECHNOLOGY
DELHI TECHNOLOGICAL UNIVERSITY
(2012-2014)

CERTIFICATE

This is to certify that **Tejna Patodi(2K12/ISY/25)** has carried out the major project titled **“Hybrid Bacterial Foraging with Firefly Algorithm”** in partial fulfillment of the requirements for the award of Master of Technology degree in Information Systems by **Delhi Technological University**.

The major project is a bonafide piece of work carried out and completed under my supervision and guidance during the academic session **2012-2014**. To the best of my knowledge, the matter embodied in the thesis has not been submitted to any other University / Institute for the award of any Degree or Diploma.

Dr. O. P. Verma

Prof. and Head

Department of Information Technology

Delhi Technological University

Delhi-110042

ACKNOWLEDGEMENT

I take the opportunity to express my sincere gratitude to my project mentor **Dr. O.P. Verma**, Prof. and Head, Department of Information Technology, Delhi Technological University, Delhi for providing valuable guidance and constant encouragement throughout the project .It is my pleasure to record my sincere thanks to him for his constructive criticism and insight without which the project would not have shaped as it has.

I humbly extend my words of gratitude to other faculty members of this department for providing their valuable help and time whenever it was required.

Tejna Patodi

Roll No. 2K12/ISY/25

M.Tech (Information Systems)

E-mail: **patoditejna@gmail.com**

ABSTRACT

In artificial intelligence, an evolutionary algorithm (EA) is a subset of evolutionary computation, a generic population-based metaheuristic optimization algorithm. An EA uses mechanisms inspired by biological evolution, such as reproduction, mutation, recombination, and selection. Candidate solutions to the optimization problem play the role of individuals in a population, and the fitness function determines the quality of the solutions. EAs are well-known optimization approaches to deal with nonlinear and complex problems. EAs often perform well approximating solutions to all types of problems because they ideally do not make any assumption about the underlying fitness landscape. The computational complexity depends on the Fitness function evaluation. One such example of EA is Bacterial Foraging Algorithm. Bacterial Foraging Optimization (BFO) [1] is a bio-inspired optimization technique, proposed by K.M. Passion. It is based on the Escherichia Coli (E. Coli) bacteria's foraging behavior i.e. the food seeking and reproductive behavior of bacteria. The classical BFO has three main mechanisms, viz, chemo taxis step, reproduction step and elimination dispersal step. The chemo taxis gives local optima whereas the reproduction, mutation and elimination-dispersal gives the global optima. Another nature inspired metaheuristic algorithm is the Firefly Algorithm (FA). It was developed by Xin-She Yang in 2008 at Cambridge University [19]. In this, the search algorithm is inspired by the social behavior of Fireflies. There are two important issues in this algorithm, namely, variation of light intensity and formulation of attractiveness. This thesis presents the hybrid of BFO with FA. Two modifications are made to the original BFO algorithm. Firstly, the position of bacteria in Bacterial Foraging Algorithm (BFO) is updated after all fitness evaluation calculations rather than each fitness evaluation calculation in chemo taxis step. Secondly, the bacteria positions are updated according to the position updation equation of Firefly Algorithm (FA). This step is defined as Mutation in the proposed algorithm. In this way, more accurate values of global optima are obtained using BFO and fast convergence is ensured using Firefly Algorithm. The technique has been applied on various benchmark functions to validate claims and results are compared with traditional BFO and FA. The results show that the proposed technique gives efficient results compared to the traditional algorithms.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

INTRODUCTION

1.1 Background

Optimization is the process of selecting the optimum solution from the set of alternative ones. We have to either maximize or minimize the objective function by calculating the value of function using several input values from the given range of values. More generally, optimization includes finding "best available" values of some objective function given a defined domain (or a set of constraints), including a variety of different types of objective functions and different types of domains. Evolutionary algorithms are being widely used in optimization problems. Reproduction, mutation, crossover, recombination, etc., mechanisms are used in such algorithms. Population is formed by the candidate solutions of the given problem and in every generation evolution of the population takes place by applying above mentioned mechanisms.

Swarm intelligence (SI) and bio-inspired computing in general have attracted great interest in almost every area of science, engineering, and industry over the last two decades. Biology-inspired algorithms have many advantages over traditional optimization methods such as steep descent and hill climbing and calculus based techniques due to the parallelism and the ability of locating the very good approximate solutions in extremely very large search spaces. Furthermore, more powerful new generation algorithm can be formulated by combining existing and new evolutionary algorithms with classical optimization methods. [14]

1.2 Motivation

The motivation is to develop an algorithm which increases the efficiency of achieving the global optima while ensuring faster convergence. Bacterial Foraging Optimization (BFO) algorithm [1] is an Evolutionary algorithm (EA) that is based on the Escherichia Coli (E. Coli) bacteria's foraging behavior i.e. the food seeking and reproductive behavior of bacteria. However, it has poor convergence nature in multi-modal and high-dimensional problems. Also, other EAs, like

Genetic Algorithm [2], Particle Swarm Optimization (PSO) [2], Differential Evolution [3] do not work well with increased complexity and dimension due to their stochastic nature. Various adaptive bacterial foraging strategy has been proposed [4][5][6] but they do not work well with multimodal or high dimensional functions. The hybrid of BFO with Parameter free PSO and the hybrid of BFO with Genetic Algorithm has been validated on benchmark functions but the problem of quick convergence is still there. Another EA called Firefly Algorithm (FA) [7], inspired by the social behavior of fireflies was developed. It has high convergence speed along with less computation rate, but optima obtained are not accurate. Therefore, it is clear that it is significantly harder to find correct optima with high convergence rate, ensuring fewer computations.

Hence, there is a need to design an algorithm that gives the correct optima along with high convergence rate.

The algorithm should do the following:

1. It gives the efficiency better than original BFO and FA
2. It gives the accuracy better than original BFO and FA
3. It provides global solution with less time complexity and less computation.

1.3 Present Work

To remove the limitations of poor convergence rate in BFO and accurate optima in FA, a new algorithm is proposed. The approach is a hybrid of BFO and FA. BFO gives the correct optima for the functions, but poor convergence rate, whereas, FA has high convergence speed with less computation rate, but optima is not accurate. Hence, the hybrid BFO algorithm with FA will have features of both, BFO and FA. Two major modifications are proposed to the BFO algorithm. Firstly, the position of bacteria in BFO is updated after all fitness evaluation calculations rather than each fitness evaluation calculation in chemo taxis step. Secondly, the bacteria positions are updated according to the position updation equation of Firefly Algorithm (FA). This step is defined as Mutation in the proposed algorithm. In this way, more accurate values of global optima are obtained using BFO and fast convergence is ensured using Firefly Algorithm. The technique has been applied on various benchmark functions to validate claims and results are compared with traditional BFO and FA. The results show that the proposed technique gives efficient results compared to the traditional algorithms.

1.4 Thesis Organization

The structure of thesis is as follows:

Chapter 2 provides literature review which includes brief explanation of optimization and evolutionary algorithms, Basic evolutionary process, biology involved in it and advantages of evolutionary computation is also explained in it. It also gives the explanation of some evolutionary algorithms which include Particle Swarm Optimization and Differential Evolution algorithm. Brief introduction of Firefly Algorithm and Bacterial Foraging Algorithm is also given in this chapter.

Chapter 3 gives the explanation of proposed methodology. It explains the proposed hybrid BFO algorithm with Firefly Algorithm

Chapter 4 shows the Experimental results which includes the comparison of original Firefly Algorithm, Bacterial Foraging Algorithm and proposed approach. It also includes the graphs showing the global optima obtained by proposed approach.

Finally, chapter 5 concludes the thesis.

Chapter 2

LITERATURE REVIEW

2.1 Optimization Problem

An optimization problem [15] is a problem of finding values for the variables of a function to optimize the function. These kinds of problems exist in many disciplines. Whenever a decision needs to be made and the problem is formulated using mathematical terms, optimization solution methods will be used to solve the formulated problem. The solution methods exist depending on the behavior of the problem. For example, if both the objective function and the functions which construct the feasible region are linear, it is called a linear programming problem, and methods like simplex algorithm will be used to solve it. Some of the solution methods depend on the derivatives of the objective function. Even though there are many solution methods, there are many problems which need special attention and are hard to solve using the deterministic solution methods. Metaheuristic algorithms are optimization algorithms which try to improve the quality of solution members iteratively with some randomness properties. Most of these algorithms are inspired by biological aspects. Unlike deterministic solution methods metaheuristic algorithms are not affected by the behavior of the optimization problem. This makes the algorithm to be used widely in different fields. Since the introduction of genetic algorithm in 1975, many metaheuristic algorithms are introduced. An optimization problem has basically three components: a function to optimize, possible solution set to choose a value for the variable from, and the optimization rule, which will be either maximized or minimized. Since one can switch between minimization and maximization problems by multiplying the objective function by negative one, analyzing either minimization or maximization problem is enough.

2.2 Evolutionary Algorithm

In computer science, **evolutionary computation** is a subfield of artificial intelligence (more particularly computational intelligence) that involves combinatorial optimization problems. Evolutionary computation uses iterative progress, such as growth or development in a population. Given a population of individuals the environmental pressure causes natural selection (survival of the fittest) and this causes a rise in the fitness of the population. Given a quality function to be maximized or minimized, we can randomly create a set of candidate solutions i.e. elements of the function’s domain, and apply the quality function as an abstract fitness measure – the higher the better. Based on this fitness, some of the better candidates are chosen to seed the next generation by applying recombination and/or mutation to them. Recombination is an operator applied to two or more selected candidates (the so-called parents) and results one or more selected candidates (the children). Mutation is applied to one candidate and results in one new candidate. Executing recombination and mutation leads to a set of new candidates (the offspring) that compete- based on their fitness (and possibly age) – with the old ones for a place in the next generation. This process can be iterated until a candidate with sufficient quality (a solution) is found or a previously set computational limit is reached.

In this process, there are two fundamental forces that form the basis of evolutionary systems. [16]

1. Variation operators (recombination and mutation) create the necessary diversity and thereby facilitate novelty, while
2. Selection acts as a force pushing quality.

The combined application of variation and selection generally leads to improving fitness values in consecutive populations. The evolutionary process makes the population adapt to the environment better and better. Since the 1990s, evolutionary computation has largely become swarm-based computation, and nature-inspired algorithms are becoming an increasingly significant part.

2.2.1 The general scheme of an Evolutionary Processes

Let us note that many components of such an evolutionary process are stochastic. During selection, fitter individuals have a higher chance to be selected than less fit ones, but typically even the weak individuals have a chance to become parents or to survive. For recombination of individuals the choice of which pieces will be recombined is random. Similarly for mutation, the pieces that will be mutated within the candidate solution, and the new pieces replacing them, are chosen randomly. The general scheme of an Evolutionary Algorithm is given in Figure 1 in a pseudo-code fashion; Figure 2 shows the diagram.

```
BEGIN
    INITIALISE population with random candidate solutions;
    EVALUATE each candidate;
    REPEAT UNTIL (TERMINATION CONDITION is satisfied) DO
    1. SELECT parents;
    2. RECOMBINE pairs of parents;
    3. MUTATE the resulting offspring;
    4. EVALUATE new candidates;
    5. SELECT individuals for the next generation;
    OD
END
```

Fig. 1 The general scheme of an Evolutionary Algorithm in pseudo-code

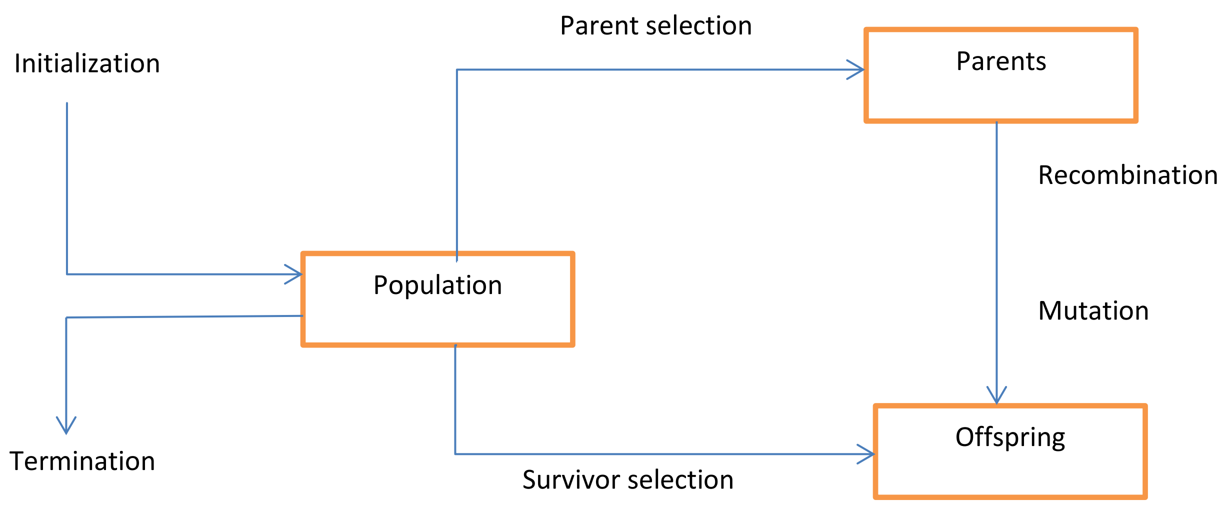


Fig. 2 The general scheme of an Evolutionary Algorithm as a flow chart

2.2.2 Components of Evolutionary Algorithms

In this section, we discuss Evolutionary Algorithms in detail. EAs have a number of components, procedures or operators that must be specified in order to define a particular EA. The most important components are,

- Representation (definition of individuals)
- Evaluation function (or fitness function)
- Population
- Parent selection mechanism
- Variation operators, recombination, and mutation
- Survivor selection mechanism (replacement)

Representation (Definition of Individuals)

The first step in defining an EA is to link the "real world" to the "EA world", that is to set up a bridge between the original problem context and the problem solving space where evolution will take place. Objects forming possible solutions within original problem context are referred to as **phenotypes,** their encoding, the individuals within the EA, are called **genotypes.** The first step is commonly called representation, as it amounts to specifying a mapping form the phenotypes onto a set of genotypes that are said to represent these phenotypes.

On the side of the original problem context, **candidate solution**, phenotype, and **individual** are used to denote points of the space of possible solutions. This space itself is commonly called the **phenotype space**. On the side of the EA, genotype, **chromosome,** and again individual can be used in the space where the evolutionary search will actually take place. This space is often termed the **genotype space.** Also, for the elements of the individuals, there are many synonymous terms. A place holder is commonly called a variable, a **locus**, a position, or in a biologically oriented terminology, a **gene.** An object on such a place can be called a value or an **allele.**

It should be noted that the "representation" is used in two slightly different ways. Sometimes, it stands for the mapping from the phenotype to the genotype space. In this sense, it is synonymous with **encoding**, e.g., one could mention binary representation or binary encoding of candidate solutions. The inverse mapping from genotypes to phenotypes is usually called decoding and it is required that the representation be invertible: to each genotype there has to be at most one corresponding phenotype. The word representation can also be used in a slightly different sense, where the emphasis is not on the mapping itself, but on the "data structure" of the genotype space.

Evaluation Function (Fitness Function)

The role of the **evaluation function** is to represent the requirements to adapt to. It forms the basis for selection, thereby it facilitates improvements. More accurately, it defines what improvement means. From the problem solving perspective, it represents the task to solve in the evolutionary context. Technically, it is a function or procedure that assigns a quality measure to genotypes. Typically, this function is composed from a quality measure in the phenotype space and the inverse representation.

The evaluation function is commonly called the **fitness function.** This might cause a counterintuitive terminology if the original problem requires minimization for fitness is usually associated with maximization.

Population

The role of **population** is to hold possible solutions. A population is a multiset of genotypes. The population forms the unit of evolution. Individuals are static objectsnot changing or adapting, it is the population that does. Given a representation, defining a population can be as simple as specifying how many individuals are in it, that is, setting the population size. In some sophisticated EAs a population has an additional spatial structure, with a distance measure or a neighborhood relation. In such cases, the additional structure has to be defined to fully specify a population. For instance, the best individual of the given population s chosen to seed the next generation, or the worst individual of the given population is chosen to be replaced by the new one. In almost all EA applications the population size is constant, not changing during the evolutionary search.

The **diversity** of a population is a measure of the number of different solutions present. No single measure for diversity exists, typically people might refer to the number of different fitness values present, the number of different phenotypes present, or the number of different genotypes. One genotype implies only one phenotype and fitness value.

Parent Selection Mechanism

The role of **parent selection** or **mating selection** is to distinguish among individuals based on their quality, in particular, to allow the better individuals to become parents of the next generation. An individual is a **parent** if it has been selected to undergo variation in order to create offspring. Together with the survivor selection mechanism, parent selection is responsible for pushing quality improvements. In Evolutionary Computing, parent selection is totally probabilistic. Thus, high quality individuals get a higher chance to become parents than those with low quality.

Variation Operators, Mutation and Recombination

The role of **Variation Operators** is to create new individuals from old ones. In the corresponding phenotype space, this amounts to generating new candidate solutions. From the generate-and-test search perspective, variation operators perform the “generate” step. Variation operators in Evolutionary computing are divided into two types based in their **arity**.

A unary variation operator is commonly called **mutation**. It is applied to one genotype and delivers a slightly modified mutant, the **child** or **offspring** of it. A mutation operator is always stochastic, its output –the child-depends on the outcomes of a series of random choices. A problem specific heuristic operator acting on one individual could be termed as mutation for being unary.

A binary variation operator is called **recombination** or **crossover**. As the names indicate such operator merges information from two parent genotypes into one or two offspring genotypes. Similarly to mutation, recombination is a stochastic operator: the choice of what parts of each parent are combined, and the way these parts are combined, depends on random drawings. The principle behind recombination is simple- that by mating two individuals with different but desirable features, we can produce an offspring which combines both of those features.

Survivor Selection mechanism (Replacement)

The role of **survivor selection** or **environmental selection** is to distinguish among individuals based on their quality. In that it is similar to parent selection, but it is used in a different stage of the evolutionary cycle. The survivor selection mechanism is called after having created the offspring of the selected parents. Survivor selection mechanism is also called **replacement** or replacement strategy. In many cases the two terms can be used interchangeably. The choice between the two is thus often arbitrary. A good reason to use the name survivor selection is to keep terminology consistent. A preference for using replacement can be motivated by the skewed proportion of the number of individuals in the population and the number of newly created children. In particular, if the number of children is very small with respect to the population size, e.g., 2 children and a population of 100. In this case, the survivor selection step is as simple as to choose the two old individuals that are to be deleted to make place for the new ones. In other words, it is more efficient to declare that everybody survives unless deleted, and to choose whom to replace. If the proportion is not skewed like this, e.g., 500 children made from a population of 100, then this is not an option, so using the term survivor selection is appropriate.

2.2.3 Some Evolutionary Algorithms

1. Particle Swarm Optimization

In computer science, **Particle Swarm Optimization** (**PSO**) [8] is a computational method that optimizes a problem by iteratively trying to improve a candidate solution with regard to a given measure of quality. 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 and is also guided toward the best known positions in the search-space, which are updated as better positions are found by other particles. This is expected to move the swarm toward the best solutions. PSO is originally attributed to Kennedy, Eberhart and was first intended for simulating social behavior, as a stylized representation of the movement of organisms in a bird flock or fish school. The algorithm was simplified and it was observed to be performing optimization. The basic concept of PSO lies in accelerating each particle toward its pbest and the gbest locations, with a random weighted acceleration at each time step.

A particle (individual) is composed of:

Three vectors:

- The **x-vector** records the current position (location) of the particle in the search space,
- The **p-vector** records the location of the best solution found so far by the particle, and
- The **v-vector** contains a gradient (direction) for which particle will travel in if undisturbed.

Two fitness values:

- The **x-fitness** records the fitness of the x-vector, and
- The **p-fitness** records the fitness of the p-vector.

Basic algorithm

```
For each particle
Initialize particle
END
Do
  For each particle
    Calculate fitness value
    If the fitness value is better than the best fitness value (pBest) in history
      set current value as the new pBest
  End
  Choose the particle with the best fitness value of all the particles as the gBest
  For each particle
```

```
        Calculate particle velocity according Eq. 1
        Update particle position according Eq. 2
    End
While maximum iterations or minimum error criteria is not attained.
```

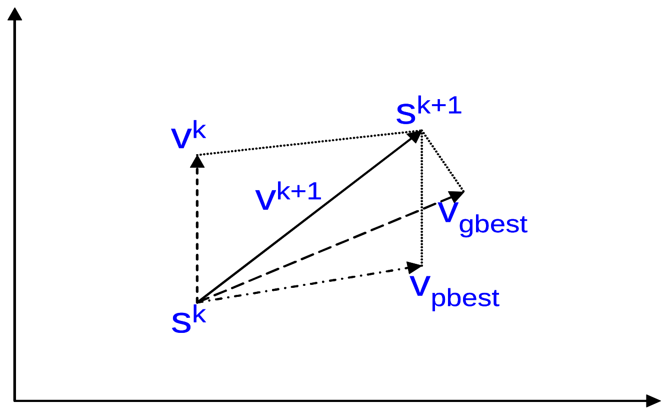
Each particle tries to modify its current position and velocity according to the distance between its current position and *pbest*, and the distance between its current position and *gbest*.

$$\boldsymbol{v(t+1) = (w * v(t)) + (c_1 * r_1 * (p(t) - x(t)) + (c_2 * r_2 * (g(t) - x(t))} \quad \textbf{(1)}$$

$$\boldsymbol{x(t+1)=v(t+1)+x(t)} \quad \textbf{(2)}$$

where,

v(t)	**: velocity of agent at iteration t,**
w	**: weighting function,**
c_1	**: weighting factor,**
r_1	**: uniformly distributed random number between 0 and 1,**
x(t)	**: current position of agent at iteration t,**
p(t)	**: pbest of agent i,**
g(t)	**: gbest of the group.**



$^s k$	**: current searching point.**
s^{k+1}	**: modified searching point.**
v^k	**: current velocity.**
v^{k+1}	**: modified velocity.**

v_{pbest} : velocity based on pbest.

v_{gbest} : velocity based on gbest

2. Differential Evolution Algorithm

DE was designed to be a stochastic direct search method [9]. The initial vector population is chosen randomly and should cover the entire parameter space. DE generates new parameter vectors by adding the weighted difference between two population vectors to a third vector. Let this operation be called mutation. The mutated vector's parameters are then mixed with the parameters of another predetermined vector, the target vector, to yield the so-called trial vector. Parameter mixing is often referred to as "crossover" vector, to yield the so-called trial vector. If the trial vector yields a lower cost function value than the target vector, the trial vector replaces the target vector in the following generation. This last operation is called selection. Each population vector has to serve once as the target vector so that NP competitions take place in one generation.

Basic strategy is as follows:

1. Mutation

For each target vector $x_{i,G}, i = 1,2,3,\ldots \text{NP}$, a mutant vector is generated according to

$$v_{i,G+1} = x_{r_1,G} + F.(x_{r_2,G} - x_{r_3,G}) \qquad (3)$$

with random indexes $r_1, r_2, r_3 \in \{1,2,\ldots.\text{NP}\}$, integer, mutually different. The randomly chosen integers r_1, r_2, r_3 are also chosen to be different from the running index i , so that NP must be greater or equal to four to allow for this condition.

2. Crossover

In order to increase the diversity of the perturbed parameter vectors, crossover is introduced. To this end, the trial vector:

$$u_{i,G+1} = (\text{u}_{1i,G+1}, \text{u}_{2i,G+1}, \ldots, \text{u}_{Di,G+1}) \qquad (4)$$

is formed, where,

$$u_{ji,G+1} = \begin{cases} v_{ji,G+1} & \text{if } (randb(j) \le CR) \text{ or } j = rnbr(i) \\ x_{ji,G} & \text{if } (randb(j) > CR) \text{ and } j \ne rnbr(i) \end{cases},$$
$$j = 1,2,\ldots,D. \qquad (5)$$

3. Selection

To decide whether or not it should become a member of generation G+1, the trial vector $u_{i,G+1}$ is compared to the target vector $x_{i,G}$ using the greedy criterion. If vector $u_{i,G+1}$ yields a smaller cost function value than $x_{i,G}$, then $x_{i,G+1}$ is set to $u_{i,G+1}$; otherwise, the $x_{i,G}$ old value is retained.

2.3 Firefly Algorithm

2.3.1 Description

Nature-inspired algorithms such as Particle Swarm Optimization and Firefly Algorithm are among the most powerful algorithms for optimization. The Firefly algorithm (FA) may also be considered as a typical swarm-based approach for optimization, in which the search algorithm is inspired by social behavior of Fireflies. There are two important issues in the Firefly algorithm that are the

- Variation of light intensity, and
- Formulation of attractiveness.

FA is used in variety of applications, namely, Job Shop Scheduling [28], Clustering [29], PID Controller tuning [30], etc .FA has high convergence speed along with less computation rate as compared to other EA. Therefore, it reduces the total runtimes and gives good results.

What are fireflies?

Lampyridae is a family of insects in the beetle order Coleoptera. They are winged beetles, and commonly called fireflies or lightning bugs for their ability to emit light. Light production in fireflies is due to a type of chemical reaction called bioluminescence. This process occurs in specialized light-emitting organs, usually on a firefly's lower abdomen. Light in adult beetles was originally thought to be used for warning purposes, but its primary purpose is now thought to be used in mate selection. Fireflies are a classic example of an organism that uses bioluminescence for sexual selection. They have evolved a variety of ways to communicate with mates in courtships: steady glows, flashing, and the use of chemical signals unrelated to photonic systems. The unique patterns of male flashes attract females of the same species, but there are examples where females mimic these patterns to lure other species in order to eat them. The large groups of fireflies are also known to synchronize their flashes. This phenomenon is explained as phase synchronization.

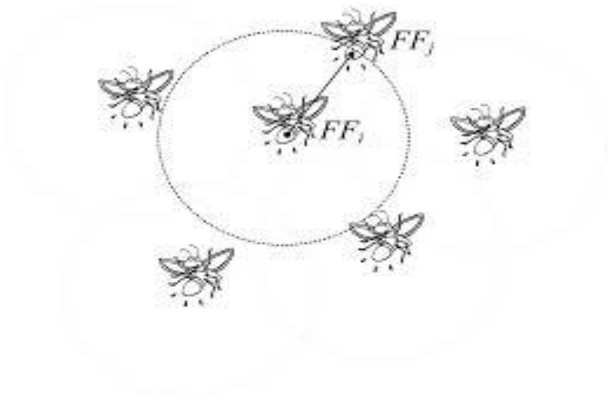
The intensity of light drops exponentially as the distance increases between the emitter and the receiver. That is, the light intensity I decreases with the increase in distance r in terms I. Also, the environment can absorb part of the light and thus further decrease the intensity of the emitted light. These properties influence on communicating abilities of fireflies and are used to simulate behavior of fireflies in our algorithm.

Rules of Firefly Algorithm

In order to simulate light communication of fireflies with an algorithm we must simplify this phenomenon and disregard parts which are too complicated or which are part of some broader feature. There are three guiding rules for construction of such algorithm which is known as Firefly Algorithm or abbreviated FA.

- First, we must think of all fireflies as if they have only one sex and are attracted to each other.
- Second, attractiveness is associated with the intensity of the light being emitted by fireflies which also means that the brighter bug will attract the less capable bug to emit light which will move her toward the first bug. We should also consider the physical property of intensity and distance, in other words if the flies are far apart there will be low attraction. The brightest firefly will move randomly since it has no other bug to attract her.
- Third, the brightness of a firefly is affected or determined by the distribution of the objective function.

Attractiveness



Intuitive step would be to use an objective function f(x) which would encode the brightness of a given firefly. Then we can think of it as the intensity at the location x as I(x) = f(x). But there are issues with the distance and the point of view. The brightness is differently perceived from the source of the flashes where it is the brightest and from some distant point watching those flashes. Then there is also capability of a medium to absorb part of that emitted light and thus decreased intensity of the watched firefly. We concluded that the attractiveness of a firefly which depends on intensity is relative. We know that the light intensity I(r) varies according to inverse square law, $I(\mathrm{r}) = \frac{I_0}{r^2}$, where I_0 is the intensity at the source of the emittance [19]. Next step is to add light absorption coefficient
 to equation.

$$I(\mathrm{r}) = \frac{I_0}{1+\gamma r^2} \qquad (6)$$

Note that we added 1 to denominator just to avoid singularity of the term at the source (r = 0). As written earlier attractiveness is proportional to intensity so we can use the equation:

$$\beta(\mathrm{r}) = \frac{\beta_0}{1+\gamma r^2} \qquad (7)$$

We could approximate given attractiveness function with Gaussian form:

$$\beta(r) = \beta_0 e^{-\gamma r^2} \qquad (8)$$

Movement of fireflies

The distance between any two Fireflies *i* and *j* whose positions are x_i and x_j is given by the Cartesian distance as follows:

$$r_{ij} = \sqrt{\sum_{m=1}^{D}\left(x_{i,\mathrm{m}} - x_{j,\mathrm{m}}\right)^2} \qquad (9)$$

The movement of a Firefly i, is attracted to another more attractive Firefly j is determined by:

$$x_i = x_i + \beta_0 e^{-\gamma r_{ij}{}^2}(\mathrm{x}_i - x_j) + \alpha(rand - \frac{1}{2}) \qquad (10)$$

Where,

- the second term is due to the attraction
- The third term is a randomization with use of randomization parameter (α), where α∈ [0, 1] rand is random number generator of Uniform distribution between 0 and 1.

Asymptotic cases

There are two important limiting cases

- when , $\gamma -> 0, \beta(r) = \beta_0$

  This is equivalent to say that the light intensity does not decrease in an idealized sky. Thus, a flashing firefly can be seen anywhere in the domain. Thus, a single (usually global) optimum can easily be reached. This corresponds to a special case of particle swarm optimization (PSO). Subsequently, the efficiency of this special case is the same as that of PSO.
- On the other hand, the limiting case, $\gamma -> \infty, \beta(r) = \delta(\mathrm{r})$

  (The Dirac delta function), which means that the attractiveness is almost zero in the sight of other fireflies or the fireflies are short-sighted. This is equivalent to the case where the fireflies fly in a very foggy region randomly. No other fireflies can be seen, and each firefly roams in a completely random way. Therefore, this corresponds to the completely random search method.

2.3.2 Algorithm

```
        Let Fitness function be $f(x)$ where $x = (x_1, x_2, ..., x_D)$
        Generate an initial population of fireflies $x_i$ $(i = 1, 2, ..... \text{n})$
        Light intensity $I_i$ at $x_i$ is determined by $f(x_i)$
        Define the light absorption coefficient $\gamma$
        While (t <MaxGeneration)
            For $i = 1$ to $n$ all n fireflies
                For $j = 1$ to $n$ all n fireflies
                    If $(I_j > I_i)$
                        Move firefly $i$ towards $j$ in d-dimension
                    End if
                Attractiveness varies with distance r via $\beta_0 e^{-\gamma r^2}$
                Evaluate new solutions and update light intensity
            End for j
        End for i
        Rank the fireflies and find the current best
    End while
    Post process results and visualization
    End
```

Fig. 3 Pseudocode of Firefly Algorithm

2.3.3 Flow chart

The flowchart of complete algorithm is as follows:

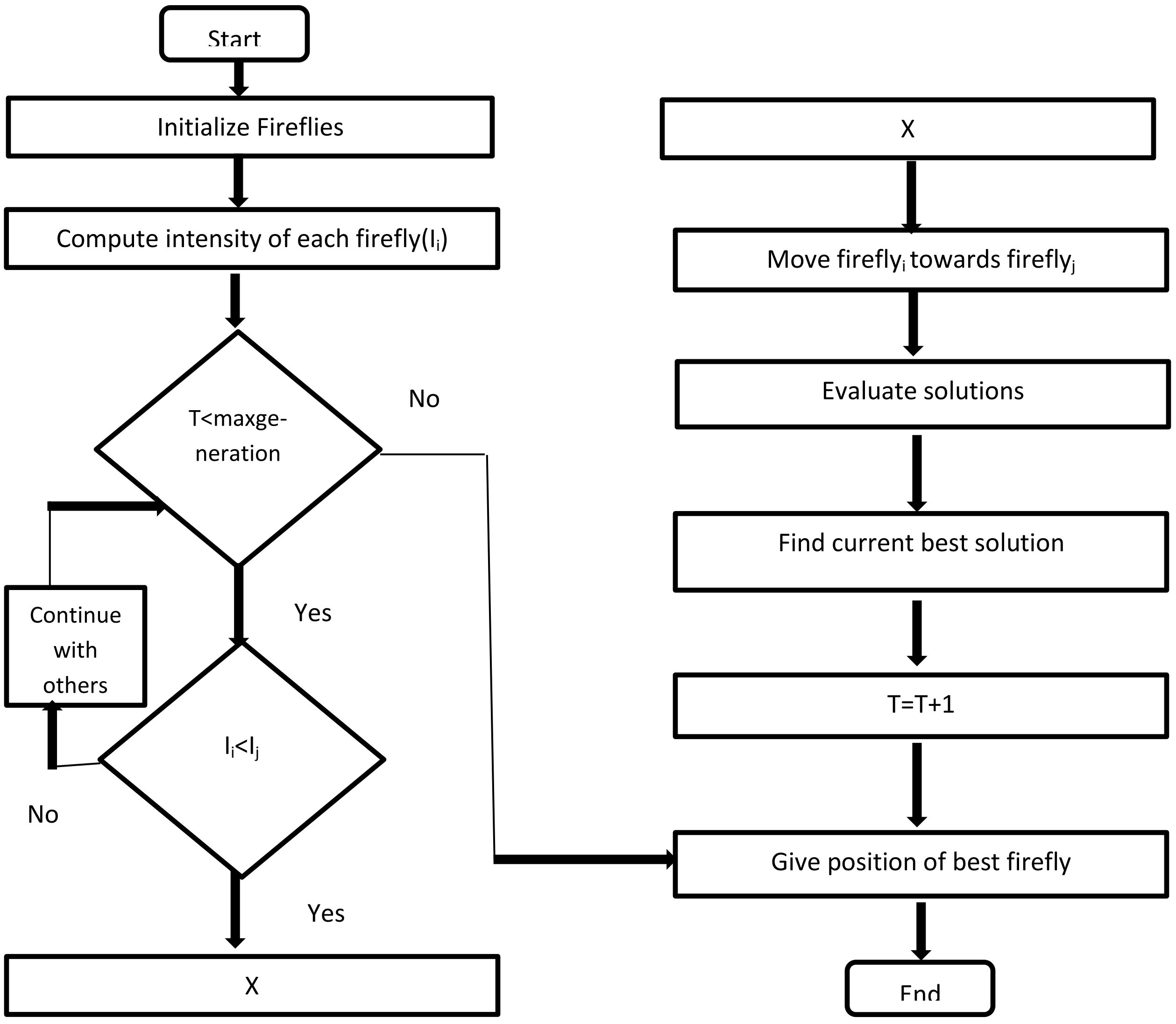

Start
Initialize Fireflies
Compute intensity of each firefly(I_i)
T<maxgeneration
No
Yes
Continue with others
$I_i<I_j$
No
Yes
X
X
Move firefly$_i$ towards firefly$_j$
Evaluate solutions
Find current best solution
T=T+1
Give position of best firefly
End

Fig. 4 Flowchart of Firefly Algorithm

2.4 Bacterial Foraging Algorithm

2.4.1 Description

Bacterial foraging optimization algorithm (BFO), proposed by Passino [10], has been broadly acknowledged as a global optimization algorithm. BFO has drawn the interest of researchers from various fields of knowledge since its origin, due to its biological motivation and refined structure [18]. The key idea of this algorithm is inspired by the social foraging behavior of Escherichia coli bacteria in multi-optimal function optimization [20]. BFO has been applied successfully in numerous problems, such as harmonic estimation [21], PID control [22], resonant frequency calculation for antenna [23] and Neural Network Fuzzy learning [24], FACTS based transmission loss reduction [25], color image enhancement [26], edge detection [27] etc.

The aim of bacteria in BFO is to maximize energy attained per unit time in search for nutrients. The locomotion of bacteria is achieved with the help of a set of tensile flagella during foraging. The tumble or swim are the two basic operation performed by the bacteria at the time of foraging. The movement of the flagella can be in the clockwise or counterclockwise direction depending upon the operation (tumble or swim) performed by the bacteria [11].

In BFO algorithm, the bacteria like to move towards a nutrient gradient and avoid unfavorable environment, this process is known as Chemotaxis. The bacteria move for a longer duration in the favorable environment. They are enlarged if they get sufficient food whereas in the presence of suitable temperature they replicate themselves. This phenomenon motivated Passino to introduce a step of reproduction in BFO. The chemotactic progress may be terminated due to occurrence of abrupt environmental conditions and a group of bacteria may be traversed to a new location. This comprises the event of elimination-dispersal BFO.

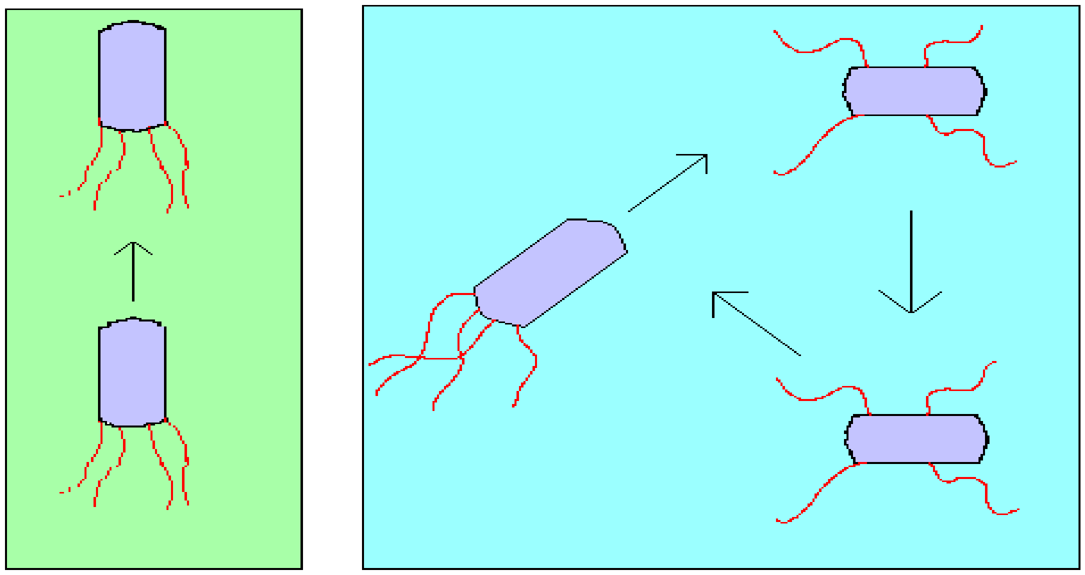


Fig. 5 Swim and tumble of a bacterium

Hence, a bacterium goes through four phases during its entire lifetime as explained below:

Chemotaxes: The movement of the bacteria in search of nutrients is termed as chemotaxes. It consists of two modes of locomotive behavior, namely tumble and swim. The bacteria tumble when the flagella rotate clockwise. In this motion, there is least displacement and after completion the bacteria is aligned along a random direction. On the other hand, the bacteria swim in counter-clockwise rotation. In this motion bacteria move forward in a particular direction. During its lifetime the bacteria alternates between these modes of motion. In locations with high nutrient concentration the bacteria swims more often than it tumbles and vice versa in areas with less favorable conditions. Thus, by continuously swimming and tumbling the bacteria effectively conducts the foraging process [12]. The tumbling process can be mathematically represented as

$$\theta^i(j+1,k,l) = \theta^i(j,k,l) + C(i)\frac{\Delta(i)}{\sqrt{\Delta^T(i)\Delta(i)}} \tag{11}$$

Where θ represents ith bacterium at jth chemotactic, kth reproductive and lth elimination-dispersal step, C(i) is the step size taken in the random direction specified by the tumble (run length unit) and Δ indicates a vector in the random direction whose values lie in the range [-1, 1]. [17]

Swarming: It has been noticed that several bacteria species including E.Coli form stable spatio temporal ring shaped swarms in the presence of a semi solid nutrient medium. The E.Coli cells when stimulated by a high level of succinate release an attractive called aspertate that helps them aggregate in such groups. This cell-cell attraction can be mathematically represented as [12]

$$J_{cc}(\theta, P(j,k,l)) = \sum_{i=1}^{S} J_{cc}(\theta, \theta^i(j,k,l))$$
$$= \sum_{i=1}^{S}[-d_{\text{attractant}}\exp(-w_{\text{attractant}}\sum_{m=1}^{p}(\theta_m - \theta_m^i)^2)] + \sum_{i=1}^{S}[h_{\text{repellant}}\exp(-w_{\text{repellant}}\sum_{m=1}^{p}(\theta_m - \theta_m^i)^2)] \tag{12}$$

where p is the dimension of search space, $J_{cc}(\theta, P(i, j,k, l))$ is the cost function that is to be added to the original cost function, d_{attract}, w_{attractt}, $h_{\text{repellant}}$, $w_{\text{repellant}}$ are the coefficients which determine the depth and width of attractant and height and width of repellant, which are to be selected properly. S is the total number of bacteria.

Reproduction: In this step, the least healthy bacteria die out. The healthiest bacteria (based on the nutrient function) then asexually split into two (without mutation and crossover) at the locations replacing the dead bacteria. This keeps the swarm size constant.

Elimination and Dispersal: Environmental factors such as running water or extreme temperatures may cause a group of bacteria to die out or get transferred to some other location. To incorporate this in the algorithm, on the basis of a very small probability some bacteria are killed off and their replacements are placed at random points in the solution space.

2.4.2 Algorithm

Parameter definition

p: the dimension of the search space

S: the number of bacteria in the population iterated by counter i

N_c: the number of chemotactic steps iterated by counter j

N_s: the number of swims after tumble iterated by the counter m

N_{re}: the number of reproductive steps iterated by counter k

N_{ed}: the number of elimination dispersal events iterated by the counter l

p_{ed}: elimination dispersal probability

C(i,k): the size of step taken in a random direction specified by tumble

[Step 1] *Initialize all the parameters defined above*

[Step 2] *Elimination dispersal loop: l=l+1*

[Step 3] *Reproduction loop: k=k+1*

[Step 4] *Chemotaxis loop: j=j+1*

For *i=1,2...,S perform a chemotactic step for bacterium i as follows*

1. *Calculate fitness function J (i,j,k,l).*
2. *Assign J_{last} = J(i,j,k,l) to update the value of the fitness function in case of better solution*
3. *Tumble: generate a random vector Δ(i) with each element $Δ_m$(i), m=1,2,...,p. The value of $Δ_m$(i) is a random number in the range [-1,1].*
4. *Move:Let*

$$\theta^i(j+1,k,l) = \theta^i(j,k,l) + C(i)\frac{\Delta(i)}{\sqrt{\Delta^T(i)\Delta(i)}}$$

Where θ_i is the chemotactic step size C(i) in the direction of the tumble for bacterium i.

5. *Calculate J(i,j+1,k,l)*

6. Swim

i) Let m=0 (counter for swim length)

ii) while $m<N_s$ *(if have not climber down too long)*

- *Let m=m+1*
- If $J(i,j+1,k,l)<J_{last}$, *let* $J_{last} = J(i,j+1,k,l)$ *and let*

  $$\theta^i(j+1,k,l) = \theta^i(j,k,l) + C(i)\frac{\Delta(i)}{\sqrt{\Delta^T(i)\Delta(i)}}$$

  and use this $\theta^i(j+1,k,l)$ *to calculate new* $J(i,j+1,k,l)$
- Else, *let* $m=N_s$.

This is the end of the while statement.

7. Go to the next bacterium (i+1), if i is not equal to S (i.e., go to 1. to process the next bacterium)

[Step 5] If $j<N_c$, *go to step 4 and continue chemotaxis process since the life of the bacteria has not ended.*

[Step 6] *Reproduction*

1. *For the given k and l, and for each i = 1,2,..., S , let*

   $$J^i_{health} = \sum_{i=1}^{N_c+1} J(i,j,k,l)$$

   be the health of the bacterium i. Sort bacteria and chemotactic parameters C(i) in order of increasing cost J_{health}.
2. *The* S_r *bacteria with the maximum* J_{health} *values die and the remaining* S_r *bacteria with the lowest values split by replicating themselves.*

[Step 7] If $k<N_{re}$, *go to step 3. The number of specified reproduction steps has not been reached, so we start the next generation of the chemotactic loop.*

[Step 8] *Elimination-dispersal:* For $i=1, 2... S$ *with probability* P_{ed}, *eliminate and disperse each bacterium with insufficient nutrient. To perform this task if bacterium is eliminated simply disperses another bacterium to a random location in the optimization domain.* If $l<N_{ed}$, *then go to step 2; otherwise* end.

2.4.3 Flow chart

The flowchart of the complete algorithm is illustrated below in fig. 6

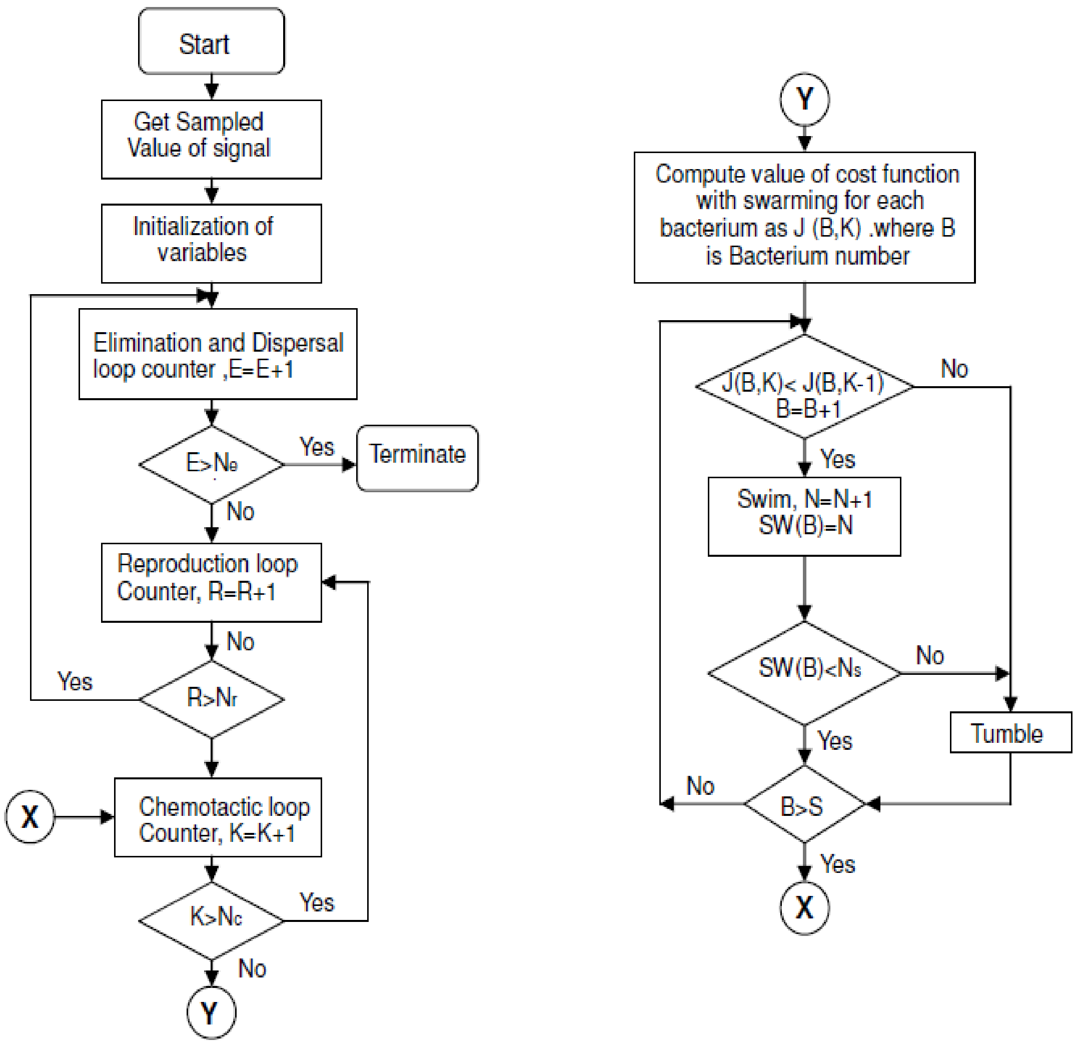

Start
Get Sampled Value of signal
Initialization of variables
Elimination and Dispersal loop counter ,E=E+1
E>Ne
Yes
Terminate
No
Reproduction loop Counter, R=R+1
No
Yes
R>Nr
X
Chemotactic loop Counter, K=K+1
K>Nc
Yes
No
Y
Y
Compute value of cost function with swarming for each bacterium as J (B,K) .where B is Bacterium number
J(B,K)< J(B,K-1) B=B+1
No
Yes
Swim, N=N+1 SW(B)=N
SW(B)<Ns
No
Yes
Tumble
No
B>S
Yes
X

Fig. 6 Flowchart of Bacterial Foraging Algorithm

Chapter 3

PROPOSED METHODOLOGY

3.1 Introduction

In BFO, the local search can be done through chemo taxis, whereas the reproduction step speeds up the process of convergence. In Elimination and dispersal, global optima can be achieved and premature convergence can be avoided. It has been observed, that for global optima searching, only chemo taxis and reproduction will not be enough steps. Since the dispersion events occur after some number of reproduction processes, the bacteria may get stuck into local optima. So, to avoid such condition, a mutation operator is introduced. The mutation operator is responsible to avoid premature convergence and for faster convergence.

In this algorithm, we introduce a new strategy for updating the position of bacteria. It helps to improve the convergence and accuracy of result. It consists of two steps. The position of bacteria, according to the traditional BFO algorithm, is updated after each fitness evaluation. However, it increases the probability of local optima. Hence, in the proposed algorithm, the position of bacteria has been updated after all fitness evaluations and not after each fitness evaluation. Also, the updation of bacteria positions is done using Firefly Algorithm. In this way, more accurate values of global optima are obtained using BFO and fast convergence is obtained using Firefly Algorithm.

1. **Chemo taxis**

   An Escherichia-coli bacterium can move in two different ways either swim or tumble. A tumble means a walk of one unit in random direction whereas a swim means a walk of one unit in the same direction. In the original BFO algorithm, the bacteria positions are updated after each fitness evaluation. In the proposed algorithm, all the bacteria positions and directions are updated after all fitness evaluations.

Let us consider the q^{th} chemotactic step for p^{th} bacteria of r^{th} reproduction step of the k^{th} elimination-dispersion of BFO. After each movement, the bacteria changes its position according to Eq. 4 as given below

$$\mathrm{pos}(\mathrm{p},\mathrm{q}+1,\mathrm{r},\mathrm{k}) = pos(\mathrm{p},\mathrm{q},\mathrm{r},\mathrm{k}) + C(\mathrm{p})\frac{\Delta(\mathrm{p})}{\sqrt{\Delta^T(\mathrm{p})\Delta(\mathrm{p})}} \qquad (13)$$

Where $\Delta(\mathrm{p})$ is the direction vector of the p^{th} bacterium movement in the current chemotactic step, and $C(\mathrm{p})$ is the step size in each chemotactic swim or tumble.

2. **Mutation**

We now introduce a new step to the classical BFO algorithm known as mutation. In this step, the bacteria position is updated using mutation operator in order to achieve the global optima. The mutation operator uses the FA for updating the positions. It helps in achieving the results with good accuracy and precision. The position of bacterium is updated as per the following equation:

$$\mathrm{pos}(\mathrm{p},\mathrm{q}+1,\mathrm{r},\mathrm{k}) = pos(\mathrm{p},\mathrm{q}+1,\mathrm{r},\mathrm{k}) + \beta_0 e^{-\gamma r_{pq}^{2}}(\mathrm{pos}(\mathrm{p},\mathrm{q}+1,\mathrm{r},\mathrm{k}) - pos(\mathrm{p},\mathrm{q},\mathrm{r},\mathrm{k})) + \alpha(rand - \frac{1}{2}) \qquad (14)$$

where $\mathrm{pos}(\mathrm{p},\mathrm{q},\mathrm{r},\mathrm{k})$ is the Position vector of p^{th} bacterium in q^{th} chemotaxis step, r^{th} reproduction steps with k^{th} elimination- dispersion and $\mathrm{pos}(\mathrm{p},\mathrm{q}+1,\mathrm{r},\mathrm{k})$ is the Position vector of p^{th} bacterium in $(q+1)^{th}$ chemotaxis step, r^{th} reproduction steps with k^{th} elimination- dispersion.

3. **Reproduction**

In this step, the healthy bacteria sustain whereas the least healthy bacteria die. Global fitness value is updated in each reproduction step. Each bacteria that survives splits into two so that the total number of bacteria remains constant. The criteria for deciding the bacteria that will survive is decided by calculating the total health compared to the preceding reproduction, given by

$$J_{health} = \Sigma F\left(p,q,r,k\right) \qquad (15)$$

The bacteria with minimum accumulated health function get selected for elimination.

4. **Elimination-dispersion**

This step occurs after a predefined number of reproduction steps with the intent of improving the global search. In this step, the bacteria are eliminated and dispersed in random positions in order to avoid getting trapped in the local optima. This happens according to the probability N_{eldis}, probability of elimination and dispersion.

3.2 Algorithm

1. *Initialize Parameters,* $s, N_{chemo}, N_{swim}, N_{repd}, N_{eldis}$ *and C (p), p= 1,2…s .*

   Where,

   s = *Population of Bacteria*

   N_{chemo} = *Chemotaxis steps*

   N_{swim} = *Swimming steps*

   N_{repd} = *Reproduction steps*

   C (p) = Step size specified by the tumble in any random direction

   $F(p,q,r,k)$ = *Fitness value or cost of* p^{th} *bacteria in the* q^{th} *chemo taxis and* r^{th} *reproduction steps.*

   F_{last} = *Fitness value or cost of best position in the* q^{th} *chemo taxis and* r^{th} *reproduction steps*

2. *Initialize the positions of bacteria*
3. *Initialize Elimination and dispersal loop: t = t+1*
4. *Initialize Reproduction loop: r=r+1*
5. *Initialize Chemo taxis loop: q=q+1*
   a) *Compute the cost or fitness function* $F(p,q,r,k)$ *for p = 1, 2, 3…s. Update* F_{last}.
   b) *Tumble: Generation of a random vector* $\Delta(\mathrm{p}) \in R^p$ *with each element* $\Delta_m(\mathrm{p})$ *, m = 1,2,…p, a random number [-1, 1]*
   c) *Computing pos for p=1,2,…,s (number of bacteria)*

$$\mathrm{pos(p,q+1,r,k)} = pos\mathrm{(p,q,r,k)} + C\mathrm{(p)}\frac{\Delta\mathrm{(p)}}{\sqrt{\Delta^T\mathrm{(p)}\Delta\mathrm{(p)}}}$$

d) Swim

```
i)   Set swim length counter, m=0
ii)  While $m < N_{swim}$
     m=m+1
     For p=1,2,3,...,s(number of bacteria)
     calculate fitness function F(p,q+1,r,k)
     update $F_{last}$
       • If $F(p,q+1,r,k) < F_{last}$
         $F_{last} = F(\mathrm{p,q+1},r,k)$
         Computing pos for p=1,2,...,s(number of bacteria)
         $\mathrm{pos(p,q+1,r,k)} = pos\mathrm{(p,q,r,k)} + C\mathrm{(p)}\frac{\Delta\mathrm{(p)}}{\sqrt{\Delta^T\mathrm{(p)}\Delta\mathrm{(p)}}}$
         Use $\mathrm{pos(p,q+1,r,k)}$ to calculate new fitness function $F(p,q,r,k)$
       • Else take $m = N_{swim}$
       End of while statement
```

e) Mutation: Updating the bacteria position using Firefly Algorithm.

Computing pos for p=1,2,...,s (number of bacteria)

$$\mathrm{pos(p,q+1,r,k)} = pos\mathrm{(p,q+1,r,k)} + \beta_0 e^{-\gamma r_{pq}^2}(\mathrm{pos(p,q+1,r,k)} - pos\mathrm{(p,q,r,k)}) + \alpha(rand - \frac{1}{2})$$

6. If $q < N_{chemo}$ *continue in step 4 chemo taxis, as the life of bacteria is still left.*

7. $S_r = s/2$ *here bacteria with the highest cost function value die. Update* F_{last}.

8. If $r < N_{repd}$ *, continue in step .*

9. For *m=1,2,3,....,s (number of bacteria)*

```
If $N_{eldis} > rand$, eliminate the bacteria
 Else $t=t+1$, and update bacteria position for the bacteria that are not dispersed.
 End.
```

Chapter 4

RESULTS & DISCUSSIONS

This section shows the evaluation of the proposed hybrid BFO algorithm with Firefly Algorithm. We have considered minimization of a set of nonlinear mathematical benchmark functions [13]. The benchmark functions are given in Appendix A along with their global minimum values and search domain. We have used both multimodal and uni-modal functions for testing the efficiency of the proposed technique. The Uni-modal functions like Rosenbrock, Matyas and Multimodal functions like Ackley, Rastrigin, Levy, Goldstein, Cross-in-tray, Sphere are considered to validate the performance of the proposed technique.

The experiments are performed on MATLAB, 2.50GHz Intel i5 processor.

4.1 Experimental Results

Table 1 shows the coordinates, the value of fitness function f(.) and time complexity obtained by BFO, FA and BFO-FA for several standard functions. It is observed from the results that BFO-FA outperforms BFO and FA for all the benchmark functions. The fitness value of all the benchmark functions is almost accurate and much better than other two algorithms. The time complexity of BFO-FA, BFO and FA shows that in almost same time, BFO-FA converges with more optimum results.

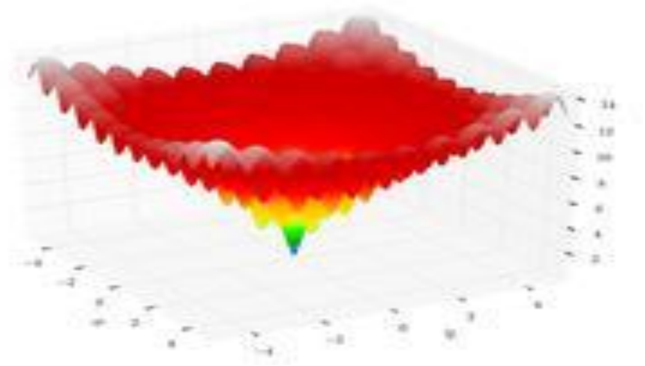
Table 1: Coordinates, fitness function f(.) and time

Function	Coordinate Value			Optimum Value, f(.)			Time (sec)		
	BFO	**FA**	**BFO-FA**	**BFO**	**FA**	**BFO-FA**	**BFO**	**FA**	**BFO-FA**
Ackley	0.0985, 0.0276	-0.0018 ,0.0014	0, 0	0.0066	0.0065	8.8818e-16	58.44	57.58	56.36
Rastrigin	-0.032, -0.0372	0.9952, -0.0042	0, 0	8.1553e-4	0.9986	0	56.49	57.45	57.78
Levy	0.9630, 1.0214	0.9973, 0.9938	1, 1	9.8998e-6	1.0276e-005	1.4998e-32	56.69	58.97	57.32
Goldstein–Price	-0.035, -0.9650	-0.0041, -1.0022	0.0908, -0.9709	3.0002	3.0044	3.0076	55.82	57.85	56.21
Rosenbrock	1.0015, 0.9304	0.8985, 0.8116	1, 1	1.5393e-4	0.0122	0	56.58	58.75	58.63
Matyas	0.0534, -0.0030	0.0014, 0.0013	0, 0	5.8275e-7	7.2783e-008	0	56.68	57.45	56.67
Sphere	0.0611, 0.0779	0.0051, 0.0010	0, 0	7.8730e-6	2.6951e-005	0	56.69	57.22	57.40
Cross-in-tray	1.4481, 1.3574	-1.35, -1.3474	-1.374, 1.3755	-2.0626	-2.0626	-2.0626	56.25	57.08	57.44

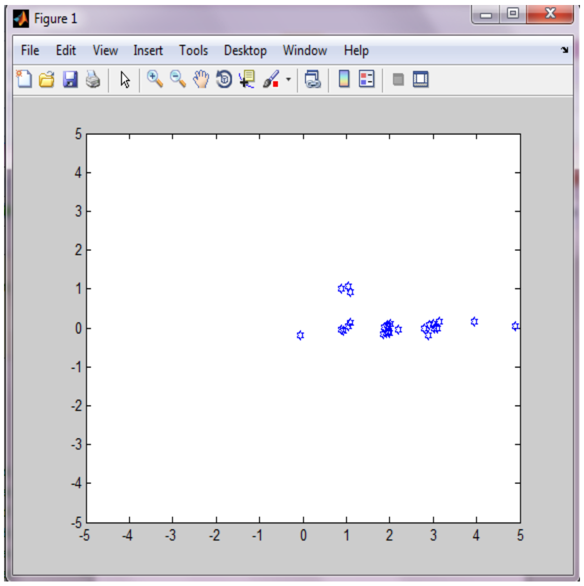
4.2 Graphs showing the global minima of functions

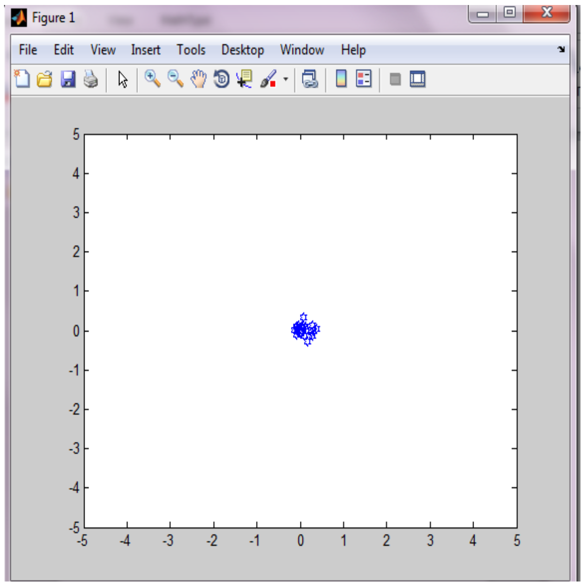
4.2.1 Ackley function

Fig. 7 shows the Global optimal solution of the Ackley function. (a) Shows the plot of Ackley function. (b) Shows the distributed bacteria in the whole search space and their movement towards the optimal solution which is the global minimum of Ackley function. (c) Shows the convergence of all bacteria to the optimal solution of the Ackley function after the last generation thus giving the optimal solution.



(a)



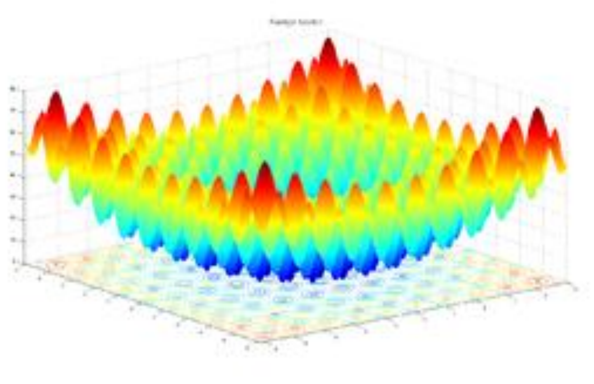


(b) (c)

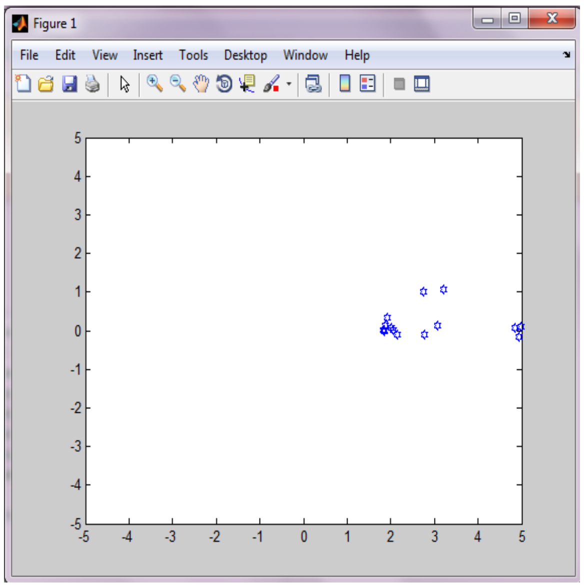
Fig. 7 Graphs showing convergence of bacteria towards optimal solution of Ackley function (a) plot of the function (b) distributed bacteria in the whole search space and their movement towards the optimal solution (c) convergence of all bacteria to the optimal solution

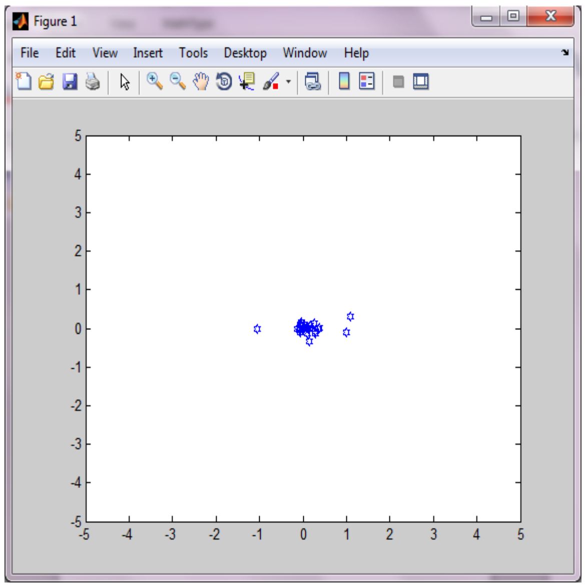
4.2.2 Rastrigin function

Fig. 8 shows the Global optimal solution of the Rastrigin function. (a) Shows the plot of Rastrigin function. (b) Shows the distributed bacteria in the whole search space and their movement towards the optimal solution which is the global minimum of Rastrigin function. (c) Shows the convergence of all bacteria to the optimal solution of the Rastrigin function after the last generation thus giving the optimal solution.



(a)





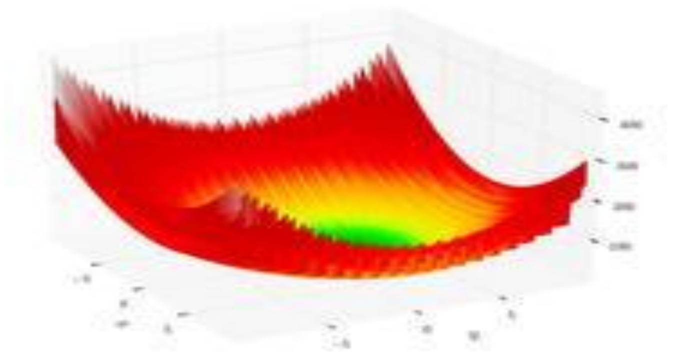
(b) (c)

Fig. 8 Graphs showing convergence of bacteria towards optimal solution of Rastrigin function (a) plot of the function (b) distributed bacteria in the whole search space and their

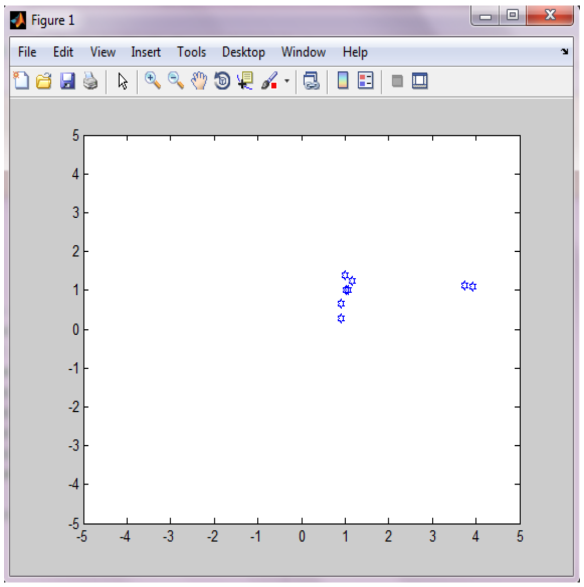
movement towards the optimal solution (c) convergence of all bacteria to the optimal - solution

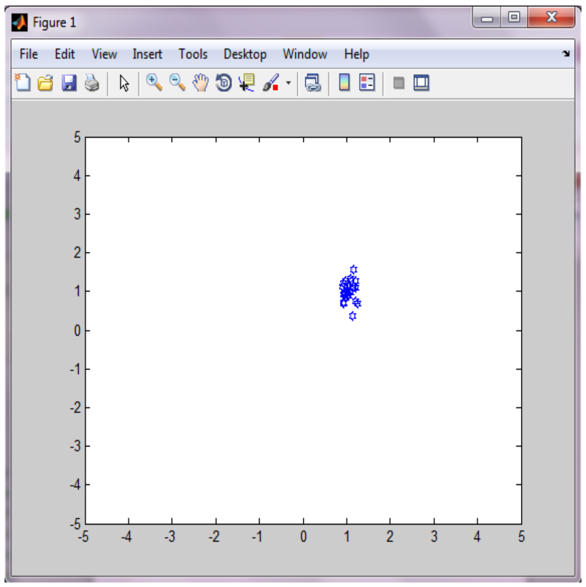
4.2.3 Levy function

Fig. 9 shows the Global optimal solution of the Levy function. (a) Shows the plot of Levy function. (b) Shows the distributed bacteria in the whole search space and their movement towards the optimal solution which is the global minimum of Levy function. (c) Shows the convergence of all bacteria to the optimal solution of the Levy function after the last generation thus giving the optimal solution.



(a)



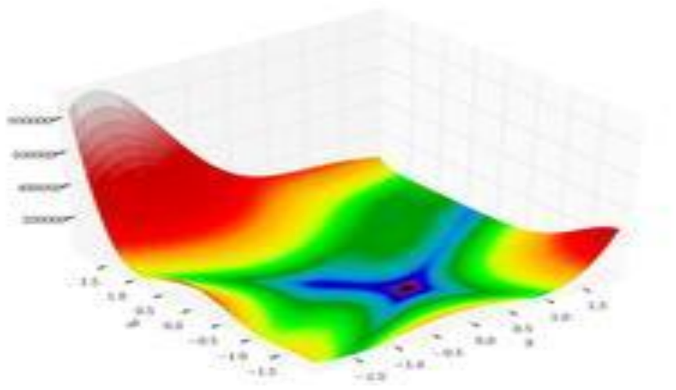


(b) (c)

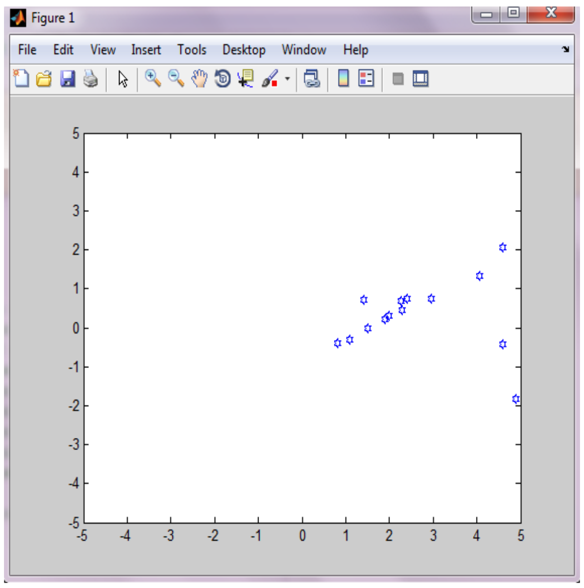
Fig. 9 Graphs showing convergence of bacteria towards optimal solution of Levy function (a) plot of the function (b) distributed bacteria in the whole search space and their movement towards the optimal solution (c) convergence of all bacteria to the optimal solution

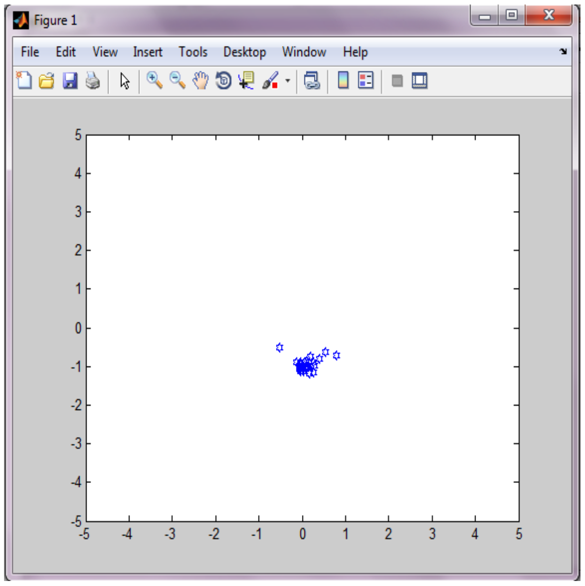
4.2.4 Goldstein Price function

Fig. 10 shows the Global optimal solution of the Goldstein Price function. (a) Shows the plot of Goldstein Price function. (b) Shows the distributed bacteria in the whole search space and their movement towards the optimal solution which is the global minimum of Goldstein Price function. (c) Shows the convergence of all bacteria to the optimal solution of the Goldstein Price function after the last generation thus giving the optimal solution.



(a)



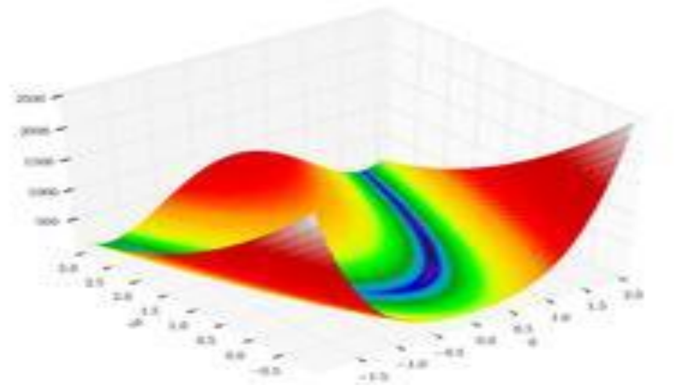


(b) (c)

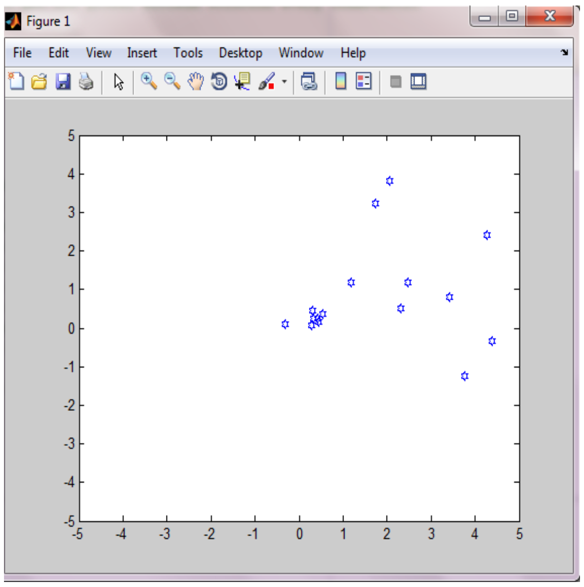
Fig. 10 Graphs showing convergence of bacteria towards optimal solution of Goldstein-Price function (a) plot of the function (b) distributed bacteria in the whole search space and their movement towards the optimal solution (c) convergence of all bacteria to the optimal solution

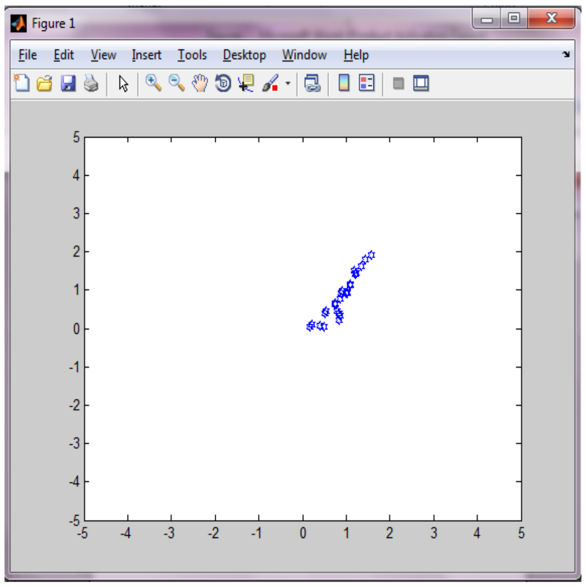
4.2.5 Rosenbrock function

Fig. 11 shows the Global optimal solution of the Rosenbrock function. (a) Shows the plot of Rosenbrock function. (b) Shows the distributed bacteria in the whole search space and their movement towards the optimal solution which is the global minimum of Rosenbrock function. (c) Shows the convergence of all bacteria to the optimal solution of the Rosenbrock function after the last generation thus giving the optimal solution.



(a)



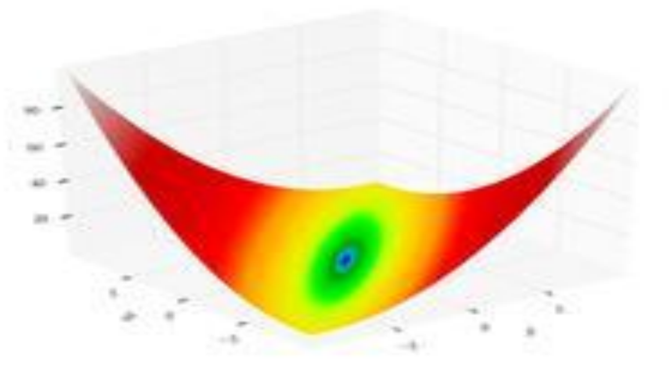


(b) (c)

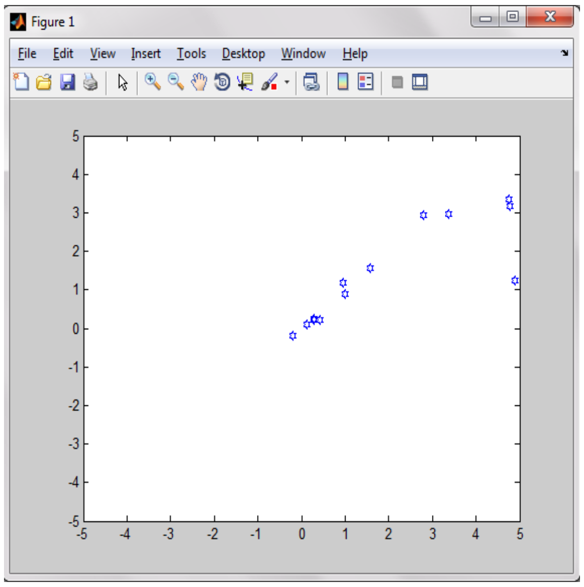
Fig. 11 Graphs showing convergence of bacteria towards optimal solution of Rosenbrock function (a) plot of the function (b) distributed bacteria in the whole search space and their movement towards the optimal solution (c) convergence of all bacteria to the optimal solution

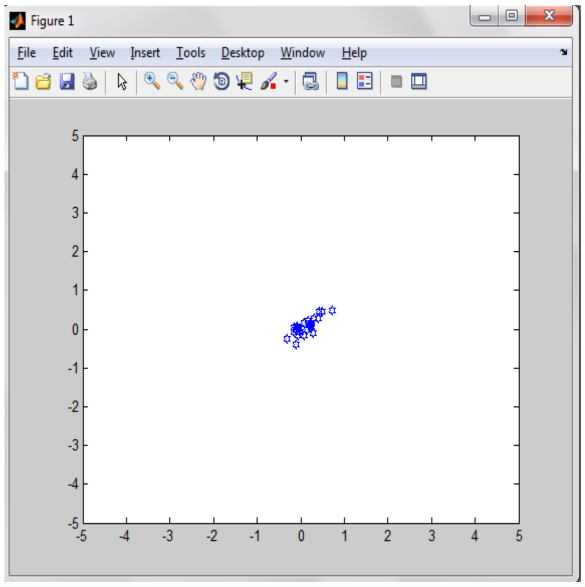
4.2.6 Matyas function

Fig. 12 shows the Global optimal solution of the Matyas function. (a) Shows the plot of Matyas function. (b) Shows the distributed bacteria in the whole search space and their Matyas function. (c) Shows the convergence of all bacteria to the optimal solution of the Matyas function after the last generation thus giving the optimal solution.



(a)



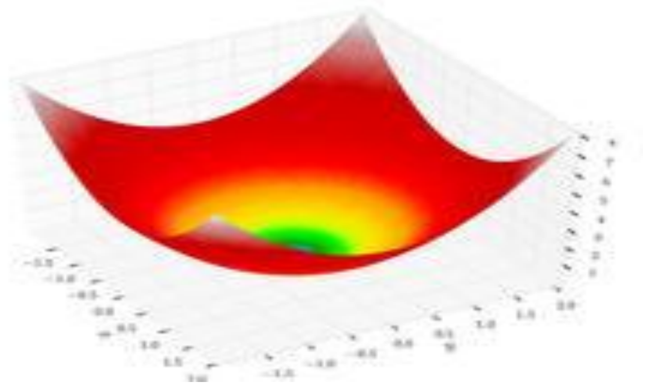


(b) (c)

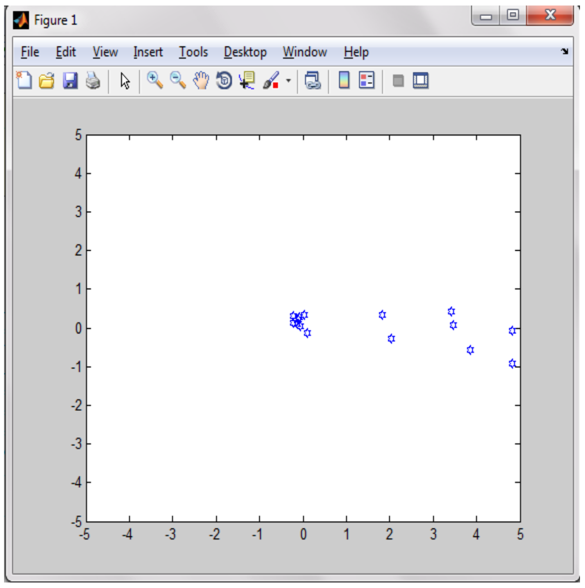
Fig. 12 Graphs showing convergence of bacteria towards optimal solution of Matyas function (a) plot of the function (b) distributed bacteria in the whole search space and their movement towards the optimal solution (c) convergence of all bacteria to the optimal solution

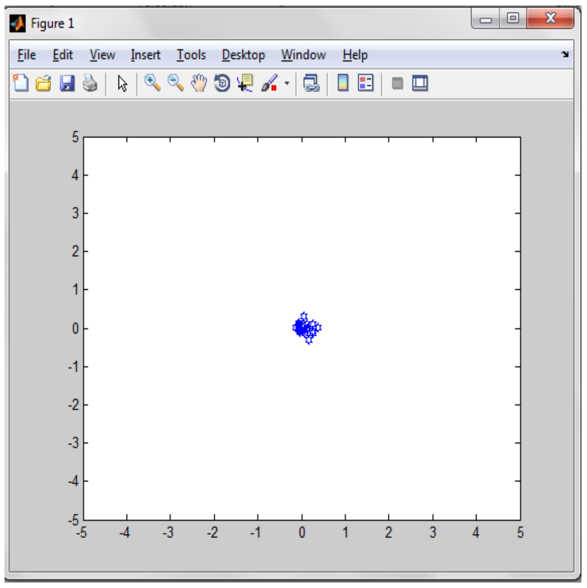
4.2.7 Sphere function

Fig. 13 shows the Global optimal solution of the Sphere function. (a) Shows the plot of Sphere function. (b) Shows the distributed bacteria in the whole search space and their Sphere function. (c) Shows the convergence of all bacteria to the optimal solution of the Sphere function after the last generation thus giving the optimal solution.



(a)





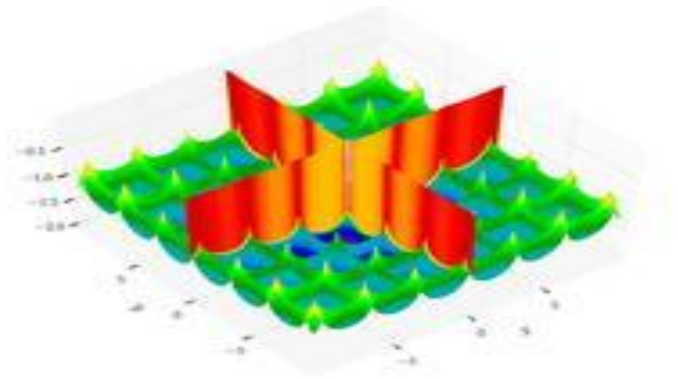
(b) (c)

Fig. 13 Graphs showing convergence of bacteria towards optimal solution of Sphere function (a) plot of the function (b) distributed bacteria in the whole search space and their movement towards the optimal solution (c) convergence of all bacteria to the optimal solution

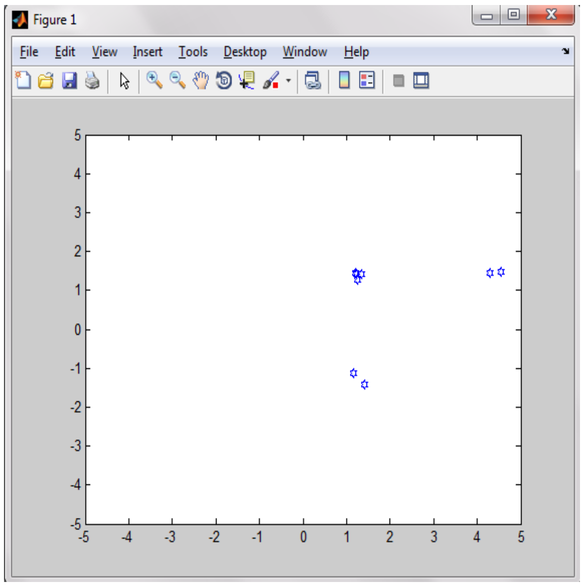
4.2.8 Cross-in tray function

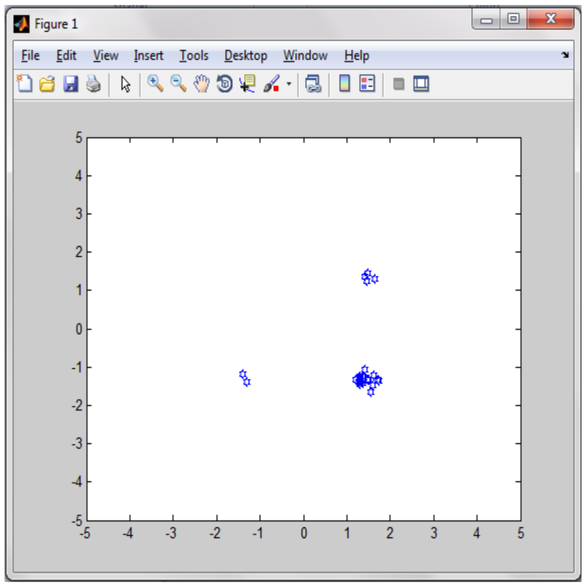
Fig. 14 shows the Global optimal solution of the Cross-in tray function. (a) Shows the plot of Cross-in tray function. (b) Shows the distributed bacteria in the whole search space and their

Cross-in tray function. (c) Shows the convergence of all bacteria to the optimal solution of the Cross-in tray function after the last generation thus giving the optimal solution.



(a)





(b) (c)

Fig. 14 Graphs showing convergence of bacteria towards optimal solution of Cross-in tray function (a) plot of the function (b) distributed bacteria in the whole search space and their movement towards the optimal solution (c) convergence of all bacteria to the optimal solution

Chapter 5

CONCLUSIONS

The hybrid of BFO and FA is presented in this paper and after using various benchmark functions, we have validated its performance. The position of bacteria is updated at each step after fitness function evaluation and the updation happens using Firefly Algorithm. The proposed hybrid approach can be used to solve unimodal or multimodal optimization problems ensuring accurate optima and fast convergence.

The following are the main contribution of proposed approach:

- The proposed method introduces mutation operator to update the position of bacteria. The mutation operator uses FA equations. The FA catalyze the global performance and avoid premature convergence.
- The bacteria positions are updated after all fitness evaluations rather than each fitness evaluation. Initially, the step size may be large, but it decreases as the bacteria starts reaching its global position.
- From the experimentation results, it is shown that the BFO-FA algorithm produces good quality and more accurate optima with faster convergence as compared to other EA techniques.
- This approach can be used to solve unimodal or multimodal optimization problems.

Appendix A

The functions used for verification of proposed algorithm

- Ackley function

$$f(x) = -\mathrm{a}\mathrm{e}^{-\mathrm{b}\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}} - \mathrm{e}^{\frac{1}{d}\sum_{i=1}^{d}\cos(\mathrm{c}\, x_i)} + \mathrm{a} + \mathrm{e}^1$$

Where, a = 20, b = 0.2 and c = 2π

Minimum, $f(x) = 0$, at $x = (0,\ldots..,0)$

Search domain, $x_i \in [-32.768\ 32.768]$ for all $\mathrm{i} = 1,\ldots\ldots,\mathrm{d}$

- Rastrigin function

$$f(x) = An + \sum_{i=1}^{n} [\mathrm{x}_i^2 - \mathrm{A}\cos(2\pi\, \mathrm{x}_i)] \quad \textit{where } A = 10$$

Minimum, $f(0) = 0$

Search domain, $-5.12 \le x_i \le 5.12$

- Levy function

$$f(\mathrm{x}, \mathrm{y}) = \sin^2(3\pi\, \mathrm{x}) + (\mathrm{x}-1)^2(1 + \sin^2(3\pi\, \mathrm{y})) + (\mathrm{y}-1)^2(1 + \sin^2(2\pi\, \mathrm{y}))$$

Minimum, $f(1,1) = 0$

Search domain, $-10 \le x, y \le 10$

- Goldstein Price function

$$\begin{aligned} f(\mathrm{x}, \mathrm{y}) = {} & (1 + (\mathrm{x} + \mathrm{y} + 1)^2(19 - 4\mathrm{x} + 3\mathrm{x}^2 - 14\mathrm{y} + 6\mathrm{xy} + 3\mathrm{y}^2))(30 + (2\mathrm{x} - 3\mathrm{y})^2 \\ & (18 - 32\mathrm{x} + 12\mathrm{x}^2 + 48\mathrm{y} - 36\mathrm{xy} + 27\mathrm{y}^2)) \end{aligned}$$

Minimum, $f(0,-1) = 3$

Search domain, $-2 \le x, y \le 2$

- Rosenbrock function

$$f(x) = \sum_{i=1}^{n-1} [100(\mathrm{x}_{i+1} - \mathrm{x}_i^2)^2 + (\mathrm{x}_i - 1)^2]$$

$$\text{Minimum, } Min = \begin{array}{l} n = 2 \to f(1,1) = 0, \\ n = 3 \to f(1,1,1) = 0, \\ n > 3 \to f(\underbrace{1,.....,1}_{(n)times}) = 0. \end{array}$$

$$\text{Search domain, } \begin{array}{c} -\infty \le x_i \le \infty, \\ 1 \le i \le n \end{array}$$

- Matyas function

$$f(x, y) = 0.26(\mathrm{x}^2 + \mathrm{y}^2) - 0.48\,\mathrm{xy}$$

Minimum, $f(0,0) = 0$

Search domain, $-10 \le x, y \le 10$

- Sphere function

$$f(x) = \sum_{i=1}^{n} x_i^2$$

Minimum, $f(\mathrm{x}_1, ..., \mathrm{x}_n) = \mathrm{f}(0,...,0) = 0$

$$\text{Search domain, } \begin{array}{c} -\infty \le x_i \le \infty, \\ 1 \le i \le n \end{array}$$

- Cross-in tray function

$$f(x, y) = -0.0001\left(\left| \begin{array}{l} \sin(x)\sin(y) \\ \exp\left(\left|100 - \dfrac{\sqrt{x^2 + y^2}}{\pi}\right|\right) \end{array} \right| + 1\right)^{0.1}$$

Minimum,

$$Min = \begin{cases} f(1.34941, -1.34941) = -2.06261 \\ f(1.34941, 1.34941) = -2.06261 \\ f(-1.34941, 1.34941) = -2.06261 \\ f(-1.34941, -1.34941) = -2.06261 \end{cases}$$

Search domain, $-10 \leq x, y \leq 10$

References

[1] Liu , K.M. Passino, M.A. Simaan, “Biomimircy of social foraging bacteria for distributed optimization: models, principles, and emergent behaviors”, Journal of Optimization Theory and Applications, Vol. 115, No.3, December 2002, pp 603-628.

[2] S. Mishra “Hybrid least-square fuzzy bacterial foraging strategy for harmonic estimation”, IEEE Transactions on Evolutionary Computation, Vol. 9, No.1, 2005, pp. 61-73.

[3] Kim, D. H., Cho, J. H., “Adaptive tuning of PID controller for multivariable system using bacterial foraging based optimization”, AWIC 2005, LNAE 3528, 2005, pp.231-235.

[4] Kim, D. H., Cho, J. H., “Adaptive tuning of PID controller for multivariable system using bacterial foraging based optimization”, AWIC 2005, LNAE 3528, 2005, pp.231-235.

[5] S. Mishra, “Hybrid least-square adaptive bacterial foraging strategy for harmonic estimation”, IEEE Proceedings-Generation, Transmission, Distribution, 2005, Vol. 152, pp. 379-389.

[6] T. Datta , I.S. Misra, “ Improved adaptive bacteria foraging algorithm in optimization of antenna array for faster convergence” Electromagnetic Research C, Vol.1, 2008, pp. 143-157.

[7] Tahereh , Hakimeh Vojodi, Amir Masoud Eftekhari Moghadam “An image segmentation approach based on maximum variance intra-cluster method and firefly algorithm”, IEEE 2011, Vol. 3, July 2011, pp. 1817-1821

[8] Kennedy, J., Eberhart, R., “Particle swarm optimization”, Neural Networks, 1995. Proceedings., IEEE International Conference on Vol. 4 , pp. 1942 – 1948.

[9] Rainer storn, Kenneth price, “Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces”, Journal of Global Optimization, 1997, pp. 341–359.

[10] K. M. Passino, ‘Biomimicry of bacterial foraging for distributed optimization and control’, IEEE Control Systems 22(3), vol. 22, pp. 52-67, 2002.

[11] Swagatam Das, Arijit Biswas, Sambarta Dasgupta1, and Ajith Abraham, “Bacterial Foraging Optimization Algorithm: Theoretical Foundations, Analysis, and Applications”

[12] L.Aurdal, ‘Image Segmentation beyond thresholding’, Norsk Regnescentral, 2006.

[13] Sotirios P. Chatzis, “Numerical optimization using synergetic swarms of foraging bacterial populations”, Elsevier, Expert Systems with Applications 38.

[14] Yang, “Genetic Algorithms”, Engineering Optimization: An Introduction with Metaheuristic Applications, 2010.

[15] Tilahun, Surafel Luleseged, and Hong Choon Ong., “Modified Firefly Algorithm”, Journel of Applied Mathematics, 2012.

[16] Tran Trong Dao, “Investigation on Evolutionary Computation Techniques of a Nonlinear system”, Modelling and Simulation in Engineering, 2011.

[17] Xu, Xin, Yan-heng Liu, Ai-min Wang, and Hui-ling Chen, “A new adaptive bacterial foraging optimizer based on field”, 2012, 8th International Conference on Natural Computation, 2012.

[18] Gao, Fei, Hongrui Gao , Yibo Qui, “Bacterial Foraging oriented by Differential Evolution Strategy”, 2010, 2nd International Conference on Information Engineering and Computer Science, 2010.

[19] Xin-She Yang, “Firefly Algorithm, Levy Flights and Global Optimization”, Research and Development in Intelligent Systems XXVI, 2010.

[20] Mingru Zhao, “An improved K-means Algorithm based on Bacterial Foraging”, International Review on Computers and Software, 2012.

[21] S. Mishra “Hybrid least-square fuzzy bacterial foraging strategy for harmonic estimation”, IEEE Transactions on Evolutionary Computation, Vol. 9, No.1, 2005, pp. 61-73.

[22] Kim, D. H., Cho, J. H., “Adaptive tuning of PID controller for multivariable system using bacterial foraging based optimization”, AWIC 2005, LNAE 3528, 2005, pp.231-235.

[23] Sastry V.R.S. Gollapudi, Shyam S. Pattnaik, O. P. Bajpai, Swapna Devi, Ch. Vidya Sagar, Patra K. Pradyumna, K.M.Bakwad, “Bacterial foraging optimization technique to calculate resonant frequency of rectangular microstrip antenna”, International Journal of RF and Microwave Computer-Aided Engineering, Vol.18 , No.4, 2008, pp.383-388.

[24] Kim, D. H. , Cho, C. H. , “Bacterial foraging based neural network fuzzy learning” , Indian International Conference on Artificial Intelligence, 2005, pp. 2030-2036.

[25] Tripathy, M. , Mishra, S., Lai, L. L., Zhang, Q. P, “Transmission loss reduction based on FACTS and bacteria foraging algorithm”, Proceedings of the 2006 Parallel Problem Solving from Nature, 2006. pp. 4139: 222-231.

[26] Madasu Hanmandlu, Om Prakash Verma, Nukala Krishna Kumar, Muralidhar Kulkarni, “A novel optimal fuzzy system for color image enhancement using bacterial foraging”, IEEE Transactions on Instrumentation and Measurement, 2009, pp 2867-2879.

[27] Om Prakash Verma, Madasu Hanmandlu, Puneet Kumar, Sidharth Chhabra, Akhil Jindal, “A novel bacterial foraging technique for edge detection”, Pattern recognition letters, 2011, pp 1187-1196.

[28] Aphirak Khadwilard, “Application of Firefly algorithm and its parameter setting for job shop scheduling”, The Journal of Industrial Technology, Vol. 8, No. 1 January – April 2012.

[29] J. Senthilnath, “Clustering using firefly algorithm: performance study”, Elsevier Swarm and Evolutionary Computation, Volume 1, Issue 3, September 2011, pp. 164–171.

[30] Olympia Roeva, “Firefly algorithm tuning of PID controller for glucose concentration control during E. coli fed-batch cultivation process”, IEEE Proceedings of the Federated Conference on Computer Science and Information Systems pp. 455–462.