OPPOSITION & DIMENSIONAL BASED 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:

DEEPTI AGGARWAL

(2K12/ISY/09)

Under the Guidance of

Dr. O. P. Verma

(HOD, Department of IT)

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

ABSTRACT

Evolutionary algorithms (EAs) are well-known optimization approaches to deal with nonlinear and complex problems. Evolutionary algorithms often perform well approximating solutions to all types of problems because they ideally do not make any assumption about the underlying fitness landscape. Firefly algorithm is one of the new metaheuristic algorithms for optimization problems. It was originally proposed by X. S. Yang [1]. It is one of the latest additions to the family of swarm intelligence metaheuristics for hard optimization problems. The algorithm is inspired by the flashing behavior of fireflies. In the algorithm, randomly generated solutions will be considered as fireflies, and brightness is assigned depending on their performance on the objective function. One of the rules used to construct the algorithm is, a firefly will be attracted to a brighter firefly, and if there is no brighter firefly, it will move randomly. This thesis presents the modified Firefly Algorithm (FA). The proposed approach gives more efficient solution with reduced time complexity in comparison to original FA. Two modifications made are; 1) Opposition-based methodology is deployed where initialization of candidate solutions is done using opposition based learning to improve convergence rate of original FA. In this opposite numbers are used which helps in generating the initial solutions from both ends hence explores the search space more efficiently. 2) The dimensional-based approach is employed in which optimization of each dimension of the solution is done separately. The dimensional method is specifically employed to conquer the "curse of dimensionality," by splitting a firefly with composite high dimensionality into several 1-D subparts. Then, the firefly makes contribution to the population not only as a whole item but also in each dimension. This ensures searching the optimal value of each dimension efficiently and hence gives more optimal solution. Several complex multidimensional standard functions are employed for experimental verification. Experimental results show that the ODFA (Opposition and Dimensional based FA) gives more accurate optimal solution with high convergence speed than the original FA.

ACKNOWLEDGEMENT

I take the opportunity to express my sincere gratitude to my project mentor **Dr. O.P. Verma**, Head of Department, 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.

Deepti Aggarwal

Roll No. 2K12/ISY/09

M.Tech (Information Systems)

E-mail: deepti.deeps.aggarwal18@gmail.com

CERTIFICATE

This is to certify that **Deepti Aggarwal(2K12/ISY/09)** has carried out the major project titled **"Opposition & dimensional based 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

Head of Department

Department of Information Technology

Delhi Technological University

Delhi-110042

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

1. Introduction

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.

1.1 Problem statement

The motivation is to develop an algorithm which increase the efficiency of original FA. Firstly, the position of the fireflies in D-dimensional space represents the candidate solution of the optimization problem. The initialization of these fireflies in original FA is purely random so it can be possible that all fireflies initially take their random positions in the same direction whereas the global solution lies in the other direction due to which the time taken to reach the global solution increases and there can exist a chance that fireflies stuck in the local optima. Secondly, In the FA, Each update step is performed on a full D-dimensional particle. This leads to the possibility of some components in the particle having been moved closer to the solution, while others have actually been moved away from the solution. Although the effect of the improvement outperforms the effect of the components that deteriorated, the FA will consider the updated particle as overall improvement but neglect the deteriorated

components in the firefly. Therefore, it is clear that it is typically significantly harder to find the global optimum of a high-dimensional problem. These two limitations of FA has been overcome in the proposed work.

1.2 Proposed solution

To remove the limitations of the original FA, two modifications have been employed in it. Firstly, opposition based learning is employed. The approach of opposition-based learning, OBL was given by Tizhoosh [16]. Here, whole search space is searched efficiently by considering the corresponding opposite estimate simultaneously along with the estimate. So, the current estimate is searched in two directions and the search space is searched more efficiently. The opposition based optimization helps the solution to converge faster hence reduces the time complexity. The comparison between randomness and opposition based approach has been done in [16] which proves that opposition based learning gives better results in less time. OBL has been applied in image segmentation [13], management of water resources [14], learning in neural network [15] etc. In this work, OBL has been used at the time of population initialization. This gives the better approximation of the initial values of the particles and hence the solution converges faster. Secondly, dimensional based approach has been employed. Since the original FA uses a population of D-dimensional fireflies, we present a new approach in which each firefly attempts to also optimize a single dimension of the solution vector instead of just as a whole item. It means that we test not only the particle as a whole item, but also the particle in each dimension. The accuracy of the algorithm increases by giving importance to each dimension to independently participate in global solution.

1.3 Justification and need of the algorithm

1. It improves the efficiency of the original FA
2. It increases the accuracy of the original FA
3. It provides global solution with less time complexity.

1.4 Thesis Organization

The structure of thesis is as follows:

Chapter 2 provides literature review which includes brief explanation of optimization and how evolutionary algorithms helps in optimization. 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 Bacterial Foraging, Particle Swarm Optimization and Differential Evolution algorithm. Brief introduction of Firefly Algorithm and Opposition based Learning is also given in this chapter.

Chapter 3 gives the explanation of proposed methodology. It explains the proposed modified Firefly Algorithm which include two modifications: one is dimensional based approach and other is opposition based learning.

Chapter 4 shows the Experimental results which includes the comparison of original Firefly 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 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

2.2.1 What are evolutionary algorithms?

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. This population is then selected in a guided random search using parallel

processing to achieve the desired end. Such processes are often inspired by biological mechanisms of evolution. As evolution can produce highly optimized processes and networks, it has many applications in computer science. The use of Darwinian principles for automated problem solving originated in the 1950s. It was not until the 1960s that three distinct interpretations of this idea started to be developed in three different places.

Evolutionary programming was introduced by Lawrence J. Fogel in the US, while John Henry Holland called his method a genetic algorithm. In Germany Ingo Rechenberg and Hans-Paul Schwefel introduced evolution strategies. These areas developed separately for about 15 years. From the early nineties on they are unified as different representatives ("dialects") of one technology, called evolutionary computing. Also in the early nineties, a fourth stream following the general ideas had emerged genetic programming. Since the 1990s, evolutionary computation has largely become swarm-based computation, and nature-inspired algorithms are becoming an increasingly significant part.

2.2.2 Basic Evolutionary Processes

A good place to start the discussion is to ask what the basic components of an evolutionary system are. The first thing to note is that there are at least two possible interpretations of the term evolutionary system. It is frequently used in a very general sense to describe a system that changes incrementally over time, such as the software requirements for a payroll accounting system. The second sense, and the one used throughout this book, is the narrower use of the term in biology, namely, to mean a Darwinian evolutionary system. In order to proceed, then, we need to be more precise about what constitutes such a system. One way of answering this question is to identify a set of core components such that, if any one of these components were missing, we would be reluctant to describe it as a Darwinian evolutionary system. Although there is by no means a consensus on this issue, there is fairly general agreement that Darwinian evolutionary systems embody:

• One or more populations of individuals competing for limited resources,

• The notion of dynamically changing populations due to the birth and death of individuals,

• A concept of fitness which reflects the ability of an individual to survive and reproduce, and

•Concept of variational inheritance: offspring closely resemble their parents, but are not identical.

Such a characterization leads naturally to the view of an evolutionary system as a process that, given particular initial conditions, follows a trajectory over time through a complex evolutionary state space. One can then study various aspects of these processes such as their convergence properties, their sensitivity to initial conditions, their transient behavior, and so on. Depending on one's goals and interests, various components of such a system may be fixed or themselves subject to evolutionary pressures.

2.2.3 A Little Biology

Evolution is a change in the gene pool of a population over time. A gene is a hereditary unit that can be passed on unaltered for many generations. The gene pool is the set of all genes in a species or population.

Natural selection is the gradual, non-random process by which biological traits become either more or less common in a population as a function of differential reproduction of their bearers. It is a key mechanism of evolution. The term "natural selection" was popularized by Charles Darwin who intended it to be compared with artificial selection, what we now call selective breeding.

Variation exists within all populations of organisms. This occurs partly because random mutations cause changes in the genome of an individual organism, and these mutations can be passed to offspring. Throughout the individuals' lives, their genomes interact with their environments to cause variations in traits. (The environment of a genome includes the molecular biology in the cell, other cells, other individuals, populations, species, as well as the abiotic environment.) Individuals with certain variants of the trait may survive and reproduce more than individuals with other variants.

Major Agents of Genetic Change in Individuals:

Mutation

A **Mutation** occurs when a DNA gene is damaged or changed in such a way as to alter the genetic message carried by that gene. A **Mutagen** is an agent of substance that can bring about a permanent alteration to the physical composition of a DNA gene such that the genetic message is changed.

It may be due to various sources (e.g. UV rays, chemicals, etc.)

Start:

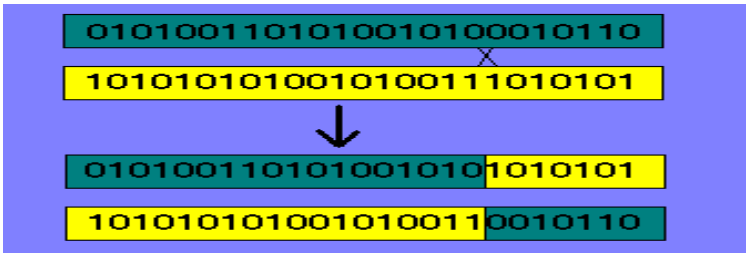
100100**1**001001001001001

After Mutation:

100100**0**001001001001001

Recombination:

Sections of genetic material exchanged between two chromosomes.



Issues to Consider

- The representation used in a given example of evolutionary computation is the data structure used together with the choice of variation operators.
- The fitness function is the method of assigning a heuristic numerical estimate of quality to members of the evolving population. It may only be necessary to decide which of two structures is better rather than to assign an actual numerical quality

We already know that evolutionary computation uses algorithms that operate on populations of data structures by selection and variation.

2.2.4 Basic evolutionary algorithm

In an evolutionary algorithm (EA), we search the **solution space** (the set of all possible inputs) of a difficult problem for the best solutions, but not naïvely like in a random or brute-force search. We use some biological principles to help guide the search:

Reproduction

New solutions don't just appear out of thin air (or our random number generator) -- we combine existing good solutions to come up with new (hopefully even better) solutions.

Mutation

Just like in real life, mutations can help or hurt. But they keep the gene pool from becoming stagnant and homogenous -- preventing EA "inbreeding".

Survival of the fittest

The "good" solutions in the population are the ones we pick to pass down their genes. We eliminate the poor solutions from consideration.

Fitness:

The fitness measure need not be a true function in the mathematical sense. It might be probabilistic, or it might depend also on other members of the population. It also often involves a model or simulation of the problem, executed with the individuals of the population.

The Process:

- Create an initial population (usually at random)
- Until "done": (**exit criteria**)
- Select some pairs to be parents (**selection**)
- Combine pairs of parents to create offspring (**recombination**)
- Perform some mutation(s) on the offspring (**mutation**)
- Select some population members to be replaced by the new offspring (**replacement**)
- Repeat

The **exit criteria** sets the target for the fitness measure, but also usually includes an upper limit on the number of iterations, in case the evolution gets "stuck." A typical exit criteria might be: "stop when some individual achieves a fitness of 100, or when we have iterated 10,000 times." We'll talk more about evolution getting "stuck" later. Sticking with the biology jargon, each iteration of the loop is called a **generation**.

Selection and **replacement** grant breeding rights and cause extinction within the population, respectively. They are independent of the representation scheme, and should only rely on your choice of fitness measure. Usually a small fraction of the population are chosen for breeding or replacement each generation. For simplicity, often the same number of individuals are chosen for breeding and replacement, although this is not required (causing the population to change in size). Here are a few of the most common selection and replacement methods:

- **Random**: Choose completely at random, with uniform probability given to each individual (regardless of fitness).
- **Absolute**: Always breed the *n* best-fit individuals, and replace the *n* least-fit individuals. (No randomness, always a deterministic choice)
- **Roulette**: Pick randomly, but with relative weights proportional to fitness. Higher-fit individuals have a better chance of getting chosen for breeding, and less-fit individuals have a better chance of getting chosen for replacement
- **Rank**: Same as roulette, but make the relative weights proportional to an individual's rank within the population, not fitness. The least-fit individual has rank 1, while the most-fit has rank *N* (the size of the population).

Recombination (or breeding) is the process of using existing pairs of "parent" genes to produce new "offspring" genes. The details of this operation depend on your representation scheme, but by far the most common recombination operation is called**crossover**. Crossover can be used with string and array representations. It involves making copies of the parents and then swapping a chunk between the copies.

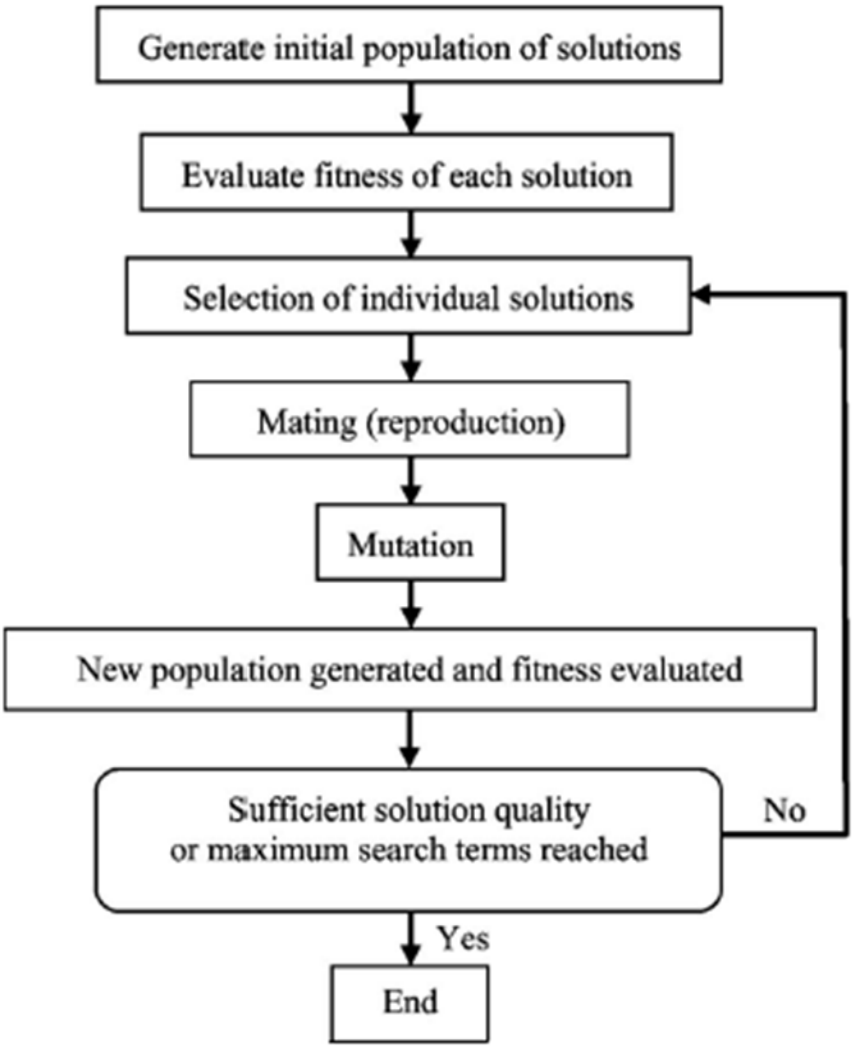


Fig 1. Flowchart of Basic Evolutionary Process

2.2.5 Advantages of evolutionary computing

- **Conceptual simplicity**: A primary advantage of evolutionary computation is that it is conceptually simple. The algorithm consists of initialization, which may be a purely random sampling of possible solutions, followed by iterative variation and selection in light of a performance index. This figure of merit must assign a numeric value to any possible solution such that two competing solutions can be rank ordered. Finer granularity is not required. Thus the criterion need not be specified with the precision that is required of some other methods. In particular, no gradient information needs to

be presented to the algorithm. Over iterations of random variation and selection, the population can be made to converge to optimal solutions.

- **Broad Applicability:** Evolutionary algorithms can be applied to virtually any problem that can be formulated as a function optimization task. It requires a data structure to represent solutions, a performance index to evaluate solutions, and variation operators to generate new solutions from old solutions (selection is also required but is less dependent on human preferences). The state space of possible solutions can be disjoint and can encompass infeasible regions, and the performance index can be time varying, or even a function of competing solutions extant in the population.
- **Outperform Classic Methods on Real Problems:** Real-world function optimization problems often (1) impose nonlinear constraints, require payoff functions that are not concerned with least squared error, (3) involve nonstationary conditions, (4) incorporate noisy observations or random processing, or include other vagaries that do not conform well to the prerequisites of classic optimization techniques. The response surfaces posed in real-world problems are often multimodal, and gradient-based methods rapidly converge to local optima (or perhaps saddle points) which may yield insufficient performance.

2.2.6 Examples of evolutionary algorithms

2.2.6.1 Bacterial Foraging Algorithm

Bacterial foraging optimization algorithm (BFOA), proposed by Passino [19], has been broadly acknowledged as a global optimization algorithm. The key idea of this algorithm is inspired by the social foraging behavior of *Escherichia coli* bacteria in multi-optimal function optimization.

In BFOA, [20] 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 BFOA. 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 BFOA.

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

The Algorithm:

[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

1. For i=1,2...,S perform a chemotactic step for bacterium i as follows
2. Calculate fitness function J (i,j,k,l).
3. Assign J_{last} = J(i,j,k,l) to update the value of the fitness function in case of better solution

4. Tumble: generate a random vector Δ(i) with each element $\Delta_m(i)$, m=1,2,...,p. The value of $\Delta_m(i)$ is a random number in the range [-1,1].
5. Move:Let

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

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

6. Calculate J(i,j+1,k,l)
7. 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

$$\theta^i(j+1,k,l) = \theta^i(j,k,l) + C(i)\frac{\Delta(i)}{\sqrt{\Delta^T(i)\Delta(i)}} \qquad (2)$$

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.

8. Go to the next bacterium (i+1), if i is not equal to S (i.e., go to 2. 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

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

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.

2.2.6.2 Particle Swarm Optimization

In computer science, **Particle Swarm Optimization** (**PSO**) [18] 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.

2.2 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 equation (a)
Update particle position according equation (b)

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*.

$$v(t+1) = (w * v(t)) + (c_1 * r_1 * (p(t) - x(t)) + (c_2 * r_2 * (g(t) - x(t)) \quad (3)$$

$$x(t+1)=v(t+1)+x(t) \quad (4)$$

where, **v(t)** **: velocity of agent at iteration t,**

w **: weighting function,**

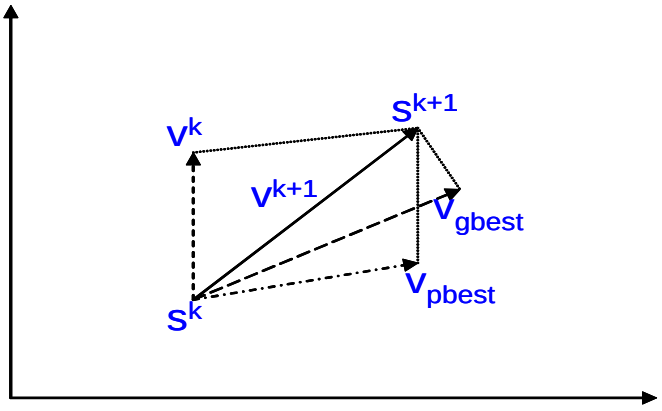
c_1 **: weighting factor,**

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.**



^sk **: 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**

2.2.6.3 Differential Evolution Algorithm

DE was designed to be a stochastic direct search method [17]. 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 $x_{i,G}, i = 1,2,3,...\text{NP}$, a mutant vector is generated according to

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

with random indexes $r_1, r_2, r_3 \in \{1,2,....\text{NP}\}$, 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:

$u_{i,G+1} = (u_{1i,G+1}, u_{2i,G+1}, ..., u_{Di,G+1})$

is formed, where,

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

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

Nature-inspired algorithms such as Particle Swarm Optimization and Firefly Algorithm are among the most powerful algorithms for optimization. The Firefly algorithm 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.

2.3.1 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.

2.3.2 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.

Let Fitness function be $f(x)$ *where* $x = (x_1, x_2, ..., x_D)$

Generate an initial population of fireflies x_i $(i = 1, 2, ...n)$

Light intensity I_i *at* x_i *is determined by* $f(x_i)$

Define the light absorption coefficient γ

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. 2 Pseudocode of Firefly Algorithm

The flowchart of complete algorithm is as follows:

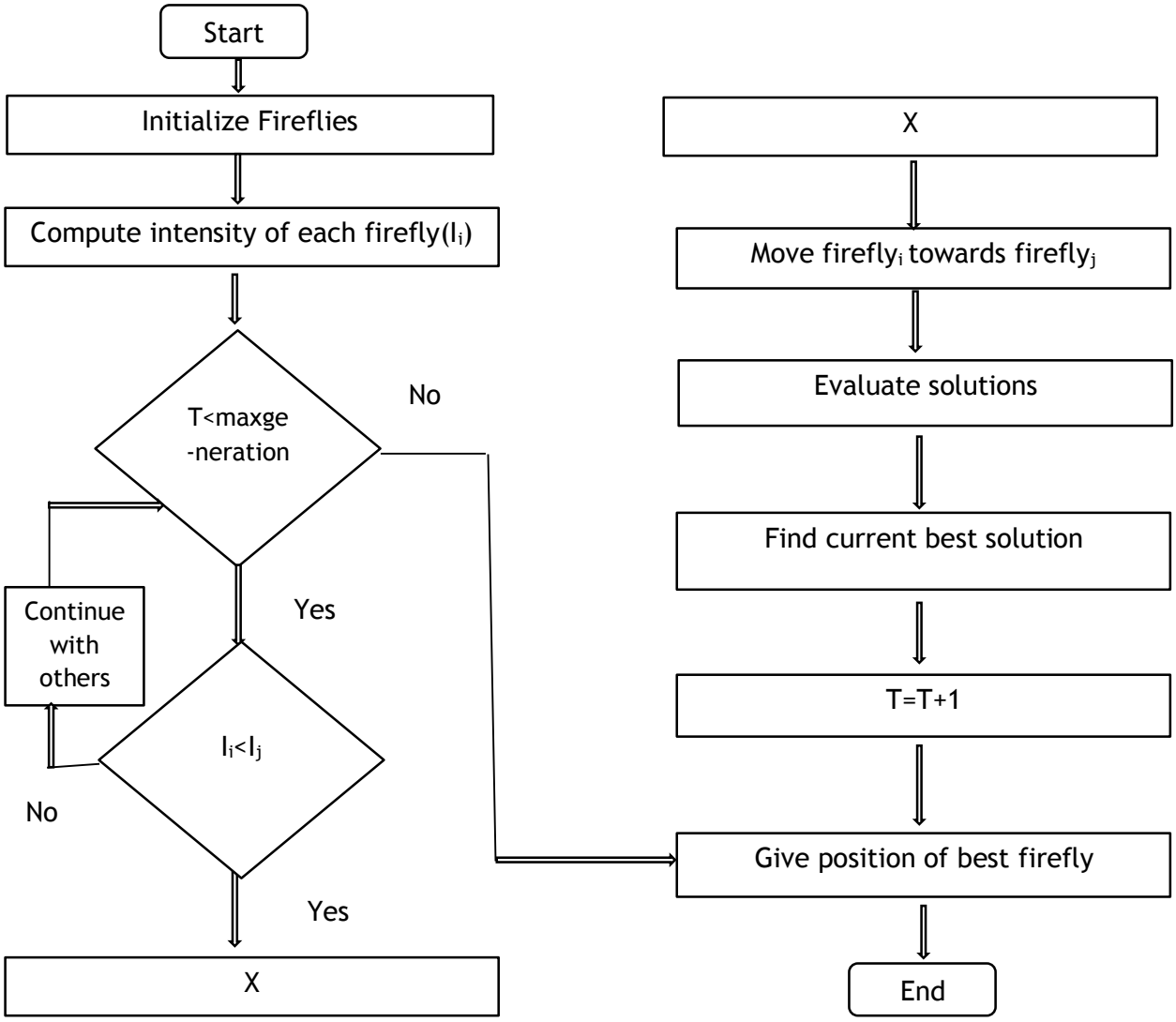
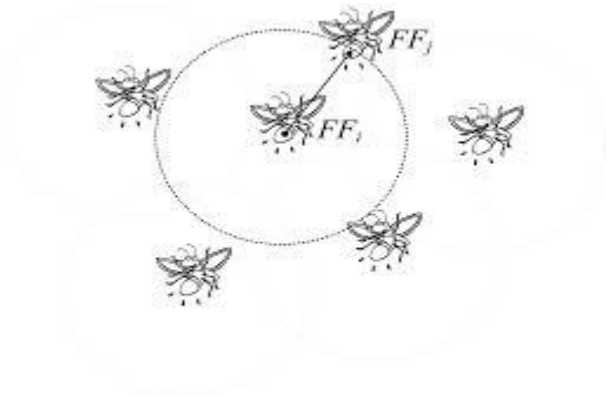


Fig. 3 Flowchart of Firefly Algorithm

2.3.3 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. 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:

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

2.3.4 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.

2.3.5 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.

2.4 Opposition based learning

This section reviews OBL introduced by Tizhoosh [10]. The reinforcement learning based upon opposition based approach has been explained in [11], [12]. Many machine intelligence algorithms are inspired by different natural systems. Genetic algorithms, neural nets, reinforcement agents, and ant colonies are, to mention some examples, well established methodologies motivated by evolution, human nervous system, psychology and animal intelligence, respectively. The learning in natural contexts such as these is generally sluggish. Genetic changes, for instance, take generations to introduce a new direction in the biological development. Behavior adjustment based on evaluative feedback, such as reward and punishment, needs prolonged learning time as well. Learning, optimization and search are fundamental tasks in the machine intelligence research. Algorithms learn from past data or instructions, optimize estimated solutions and search in large spaces for an existing solution. The problems are from different nature, and algorithms are inspired by diverse biological, behavioral and natural phenomena.

To get the optimum solution of a given problem we have to make some initial estimates. This estimate can be made by having some prior information about the solution or it can be completely random. In absence of prior information, convergence of solution depends upon the distance of the optimal solution from the random estimate. In worst case, if random guess is very much far away from the initial random guess than sometimes the solution could not be reached. The solution to this problem could be to look for solution in all directions or at least in the opposite direction. In the initialization step, along with the random guess of the solution x, the opposite of the x, $\breve{x}$ should be considered simultaneously. This leads to searching of search space more thoroughly and increase in rate of convergence.

2.4.1 Opposite number —Suppose x is a real number which lies in the interval: $x \in [\mathrm{m}, \mathrm{n}]$. The opposite number of x is given by $\breve{x}$ as follows:

$$\breve{x} = m + n - x \qquad (11)$$

In multidimensional space, the opposite number can be given as.

Let $x = (x_1, x_2, ..., x_D)$ be a point in D-dimensional space, where $x_1, x_2, ..., x_D \in R$ and $x_i \in [m_i, n_i] \forall i \in \{1, 2, ..., D\}$. The opposite point x is defined by $x_1, ..., x_D$ in D-dimensional space where

$$x_i = m_i + n_i - x_i \quad i = 1, ..., D. \qquad (12)$$

2.4.2 Opposition-Based Optimization—let $f(\bullet)$ be the fitness function, x be a candidate solution in D-dimensional space and x be the opposite point of x . Replace point x with x if $f(x) \geq f(x)$ else continue with x . Hence, the point and its corresponding opposite point are assessed simultaneously to determine the more appropriate point for the given problem.

2.4.3 Extending Genetic Algorithms with OBL

For every selected chromosome a corresponding anti-chromosome can be generated. The initial chromosomes are generally generated randomly meaning that they can possess high or low fitness. However, in a complex problem it is usually very likely that the initial populations are not the optimal ones. In lack of any knowledge about the optimal solution, hence, it is reasonable to look at anti-chromosomes simultaneously. Considering the search direction and its opposite at the same time will bears more likelihood to reach the best population in a shorter time. Especially at the beginning of the optimization, either the chromosome or the anti-chromosome may be fitter (in some cases both may be fit solutions!). Considering a population member and its genetic opponent should accelerate finding the fittest member.

CHAPTER 3

PROPOSED METHODOLOGY

3. Proposed Methodology

The original FA stuck in the local optima for high-dimensional problems as it doesn't consider every dimension of each firefly separately for finding the best solution due to which some dimensions move towards the better solution but some dimensions are moves away from the global solution therefore global optima is not reached. The proposed dimensional FA helps FA to not to stuck in the local optima as in this, in every generation global best solution is formed whose each dimension represents the best value of that dimension among all the fireflies and then position of all fireflies are updated according to the global best firefly. This leads to optimization of each dimension and hence we got the global optima. The opposition based FA helps in initialization of the fireflies more efficiently so that fireflies converge faster and hence the time complexity is reduced. The proposed approach is explained as follows.

Let there are N-number of fireflies represented by $x = (x_1, x_2, ..., x_N)$ in D-dimensional space such that, i_{th} firefly, $x_i = (x_{i1}, x_{i2}, ..., x_{iD})$.

A. In the proposed approach, at the time of initialization phase, OBL is employed in which the opposite number of the positions of each firefly is calculated. Let the population generated after calculating the opposite number is $\tilde{x}$. From x and $\tilde{x}$, the best N fireflies are selected. Steps are as follows:

   1) Initialize the position of fireflies x randomly.

   2) Generate N more fireflies by calculating the opposite population, $\tilde{x}$. This is done using the multi-dimensional opposite number of each firefly in the set x

   $\tilde{x}_{ij} = a_j + b_j - x_{ij}$, where $i = 1, 2, ...N$ and $j = 1, 2, ...D$

   Where, x_{ij} and $\tilde{x}_{ij}$ represents j_{th} dimension of the i_{th} firefly of the population and the opposite-population, respectively.

   3) Select the best N fireflies from $x \cup \tilde{x}$. These fireflies constitutes the initial population.

B. Each firefly in FA represents a candidate solution of the optimization problem. In original FA, the updation of the D-dimensional firefly takes place together which leads to some dimensions move towards the optimal solution and other dimensions move away from it. The deteriorated dimensions are neglected hence it becomes difficult to find the global optimum in high-dimensional problem. The time taken to reach every dimension at their optimum value is high so Dimensional FA is introduced in which every dimension is updated separately. This is made possible by introducing the concept of Gbest in FA. Gbest represents the best firefly in the current generation. This Gbest is further improved by changing the value of one dimension at a time with the corresponding dimension value of other fireflies one by one and then if the fitness value improves, the Gbest is updated. Gbestpos is the position of the Gbest. The context vector is used to represent the vector formed by putting the each dimension of each firefly one by one in the Gbestpos. Context vestor is first initialized with the Gbestpos. The fitness of the firefly in dimension j is calculated by replacing the j_{th} component of context vector by j_{th} of this firefly. Then, if $f(\text{Contextvector})$ is better than Gbest, Gbestpos is replaced with that vector. Therefore, this method helps the firefly to contribute its merits in each dimension. This process is repeated for every dimension of every firefly in each generation. At the end of each generation, all the fireflies move towards the Gbest whereas in original firefly, the fitness value of each firefly is compared with all other fireflies and if the fitness value of that firefly is less than any other firefly, the firefly moves towards that better firefly. This concept of original firefly consumes time whereas in modified algorithm all fireflies move towards Gbest and no comparison is required. This helps in saving time.

Steps of Dimensional FA are as follows:

1. Initialize context vector with the global best firefly.

2. For each dimension j in D-dimensional space, repeat steps 3 and 4.

3. The fitness of the firefly in dimension j is calculated by replacing the j_{th} component of context vector by j_{th} of this firefly.

4. Repeat step 3 for each firefly

5. Move all fireflies towards Gbest.

6. Repeat all above steps for each generation.

C. The modified Firefly Algorithm constitutes above two modifications.

Steps are as follows:

1. Generate initial population of fireflies, x.

2. Generate opposite population by applying Opposition based learning, $\breve{x}$.

3. Select N fitted fireflies from the set $\{ x , \tilde{x}_{ij} \}$. This constitutes the initial position of fireflies.

4. for each generation

5. Repeat the steps of Dimensional approach stated above

6. Move the fireflies towards the Global best firefly by the following position update equation:

$$x_i^{new} = x_i^{old} + \beta_0 e^{-\gamma r_{ij}^2} (x_i^{old} - Gbestpos) + \alpha(rand - \frac{1}{2}) \qquad (13)$$

7. After the last generation, Gbest constitutes the Global optimal solution of the function and Gbestpos constitutes the coordinates at which optimal solution is found.

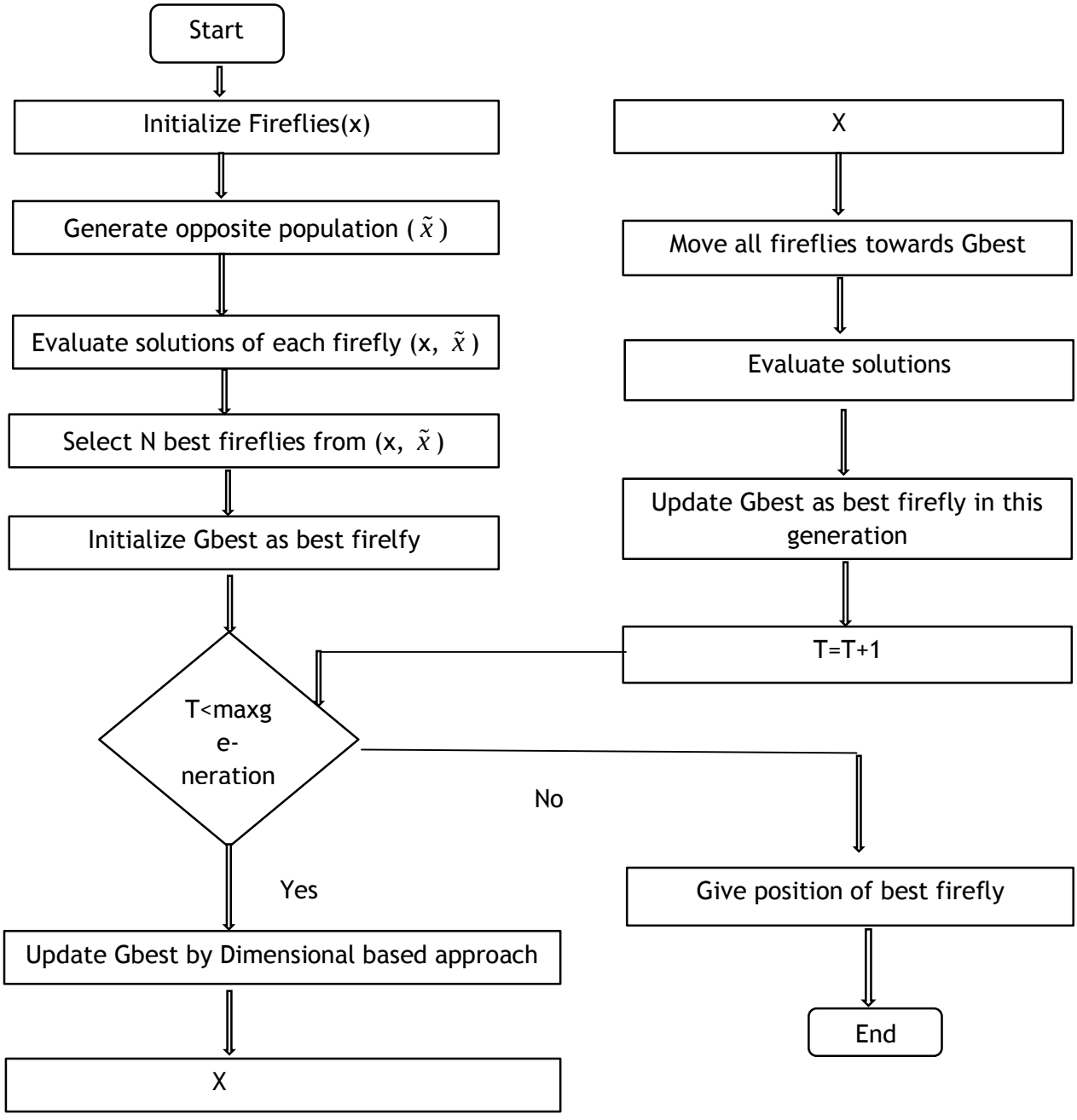


Fig. 4 Flowchart of Modified Firefly Algorithm

CHAPTER 4
EXPERIMENTAL RESULTS

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

A set of standard functions has been used for performance analysis of the proposed approach against original FA. In Appendix A, the definition of those functions with their global minimum values are enlisted. The measure used in this paper for comparing the speed of FA and ODFA is execution time. Smaller execution time means higher convergence speed. In order to compare convergence speeds, a parameter speed rate is used given by

$$K = \frac{t_{FA}}{t_{ODFA}} \qquad (14)$$

Where t_{FA} the execution is time for FA and t_{ODFA} is the execution time for ODFA. K>1 signifies ODFA is faster.

Table1 shows the value of fitness function $f(\cdot)$ obtained by modified algorithm as well as by the original FA for several standard functions with 2, 3 and 4 dimensions. Results show that the solutions obtained by the proposed approach are more accurate. The fitness value of Ackley function obtained through modified algorithm is approximately 1000 times better than the value obtained through the original FA. In the same manner, the fitness value of other functions obtained by proposed algorithm are also better than the original FA. The value of K is also mentioned in the table. The value of K is almost greater than 1 for every optimization function which proves that the convergence speed of the proposed algorithm is higher than the original FA.

4.2 Parameters Initialization: For all the experiments the parameters are initialized as follows: N=50, MaxGeneration=200, α =0.2, γ =0.001 but in Schwefel's function and Griewank's function, the search space is very large which lies in the range [-500,500] and [-600 600] respectively so number of fireflies required is more to explore the whole search space therefore, N=200 and MaxGeneration= 500 is taken for Schwefel's function. The fitness value of different functions with respect to the population size is plotted in Fig. 4 which shows that at N=50 the functions attain their global optima.

Table 1: Experimental Results

Functions	**Dimensions**								
	2-Dimension			**3-Dimension**			**4-Dimension**		
	Original $f(\cdot)$	**Proposed** $f(\cdot)$	K	**Original** $f(\cdot)$	**Proposed** $f(\cdot)$	K	**Original** $f(\cdot)$	**Proposed** $f(\cdot)$	K
Ackley	$2.08*10^{-2}$	$5.7119*10^{-5}$	1.1144	$2.125*10^{-2}$	$2.506*10^{-5}$	1.1428	$1.844*10^{-1}$	$6.421*10^{-5}$	0.9416
Rosenbrock	0	0	1.7189	0	0	2.67	0.1774	$4.156*10^{-3}$	3.175
De Jong	$2.79*10^{-6}$	$4.63*10^{-11}$	2.317	$3.307*10^{-5}$	$4.57*10^{-10}$	3.056	$1.198*10^{-4}$	$1.038*10^{-10}$	2.59
Griewank	0.1051	$4.124*10^{-11}$	1.509	0.1482	$7.7*10^{-11}$	1.125	0.1601	$1.065*10^{-10}$	1.273
Easom	-0.811	-1	3.335	NA	NA	NA	NA	NA	NA
Shubert	-186.73	-186.7309	2.928	NA	NA	NA	NA	NA	NA
Schwefel	$1.4*10^{-3}$	$2.58*10^{-5}$	1.564	0.04318	$3.818*10^{-5}$	2.86	0.056	$8.08*10^{-5}$	2.14
Rastrigin	$7.816*10^{-3}$	$6.928*10^{-9}$	1.32	$8.097*10^{-2}$	$7.414*10^{-7}$	1.51	$5.304*10^{-2}$	$4.316*10^{-6}$	1.86
Michalewicz	-1.37	-1.8013	0.946	-1.959	-2.7604	0.9029	-2.2667	-3.6571	0.7713

Table 2: Experimental results of Ackley Function

Dimensions		Original	Modified
2-D	**Position**	$(0.31,0.62)*10^{-3}$	$(0.1514,0.0511)*10^{-4}$
	Gbest Value	$2.08*10^{-2}$	$5.7119*10^{-5}$
	Time	0.20059	0.18
3-D	**Position**	0.93,0.015,0.0047	$(0.2215,0.4851,0.9)*10^{-2}$
	Gbest Value	$2.125*10^{-2}$	$2.506*10^{-5}$
	Time	0.2388	0.209
4-D	**Position**	0.0004,0.014,-0.932,0.0228	$(0.041,-0.053,3.77,1.81)*10^{-3}$
	Gbest Value	0.1844	$6.421*10^{-5}$
	Time	0.2665	0.283

Table 3: Experimental results of Rosenbrock function

Dimensions		Original	Modified
2-D	**Position**	1,1	1,1
	Gbest Value	0	0
	Time	0.1375	0.080
3-D	**Position**	1,1,1	1,1,1
	Gbest Value	0	0
	Time	0.2269	0.085
4-D	**Position**	0.9013,0.8236,0.6679,0.4408	0.9499,0.9019,0.8132,0.6692
	Gbest Value	0.1744	$4.156*10^{-3}$

	Time	0.3683	0.116

Table 4: Experimental results of De Jong function

Dimensions		**Original**	**Modified**
2-D	**Position**	0.0015,-0.0008	$(0.1514,0.0511)*10^{-4}$
	Gbest Value	$2.79*10^{-6}$	$4.63*10^{-11}$
	Time	0.38466	0.18
3-D	**Position**	-0.0055,0.0008,0.0013	$(0.2215,0.4851,0.9)*10^{-2}$
	Gbest Value	$3.3077*10^{-5}$	$4.57*10^{-10}$
	Time	0.638	0.209
4-D	**Position**	0.0068,-0.0018,0.0026,-0.0080	$(0.041,-0.053,3.77,1.81)*10^{-3}$
	Gbest Value	$1.1984*10^{-4}$	$1.038*10^{-10}$
	Time	0.73	0.283

Table 5: Experimental results of Griewank function

Dimensions		**Original**	**Modified**
2-D	**Position**	0.4621,-0.0229	$(-0.72,-0.7798)*10^{-5}$
	Gbest Value	0.1051	$4.124*10^{-11}$
	Time	0.323	0.214
3-D	**Position**	0.3408,0.4216,0.5737	$(-0.0022,-0.0433,-0.209)*10^{-4}$
	Gbest Value	0.1482	$7.7012*10^{-11}$

	Time	0.35775	0.318
4-D	**Position**	-0.536,0.0195,-0.0902,-0.4153	$(0.13,-0.053,0.05,0.03)*10^{-4}$
	Gbest Value	0.1601	$1.0655*10^{-10}$
	Time	0.4506	0.354

Table 6: Experimental results of Easom function

Dimensions		**Original**	**Modified**
2-D	**Position**	3.0525,3.211	3.416,3.1416
	Gbest Value	-0.811	-1
	Time	0.1334	0.04

Table 7: Experimental results of Shubert function

Dimensions		**Original**	**Modified**
2-D	**Position**	-0.8001,-1.4251	-0.8003,-1.4251
	Gbest Value	-186.7308	-186.7309
	Time	0.3279	0.112

Table 8: Experimental results of Schwefel function

Dimensions		**Original**	**Modified**
2-D	**Position**	421.0356,420.978	420.9702,420.9696
	Gbest Value	$1.4*10^{-3}$	$2.58*10^{-5}$
	Time	0.3722	0.238

3-D	**Position**	422.075,0.420.6,419.5737	420.9685,420.9688,420.9687
	Gbest Value	0.04318	$3.818*10^{-5}$
	Time	6.915	2.418
4-D	**Position**	418.536,420.0195,419.0902,420.4153	420.9674,421.3157,420.9673,420.2072
	Gbest Value	0.056	$8.08*10^{-5}$
	Time	7.804	3.647

Table 9: Experimental results of Rastringin function

Dimensions		**Original**	**Modified**
2-D	**Position**	$(1.261,7.896)* 10^{-2}$	$(0.057,-0.588)*10^{-5}$
	Gbest Value	$7.816*10^{-3}$	$6.9228*10^{-9}$
	Time	0.146	0.111
3-D	**Position**	0.124,0.045	$(1.068,2.6788)*10^{-3}$
	Gbest Value	$8.097*10^{-2}$	$7.414*10^{-7}$
	Time	3.243	2.418
4-D	**Position**	0.098,0.087	$(8.757,4.56)*10^{-3}$
	Gbest Value	$5.304*10^{-2}$	$4.316*10^{-6}$
	Time	6.783	3.647

Table 10: Experimental results of Michalewicz function

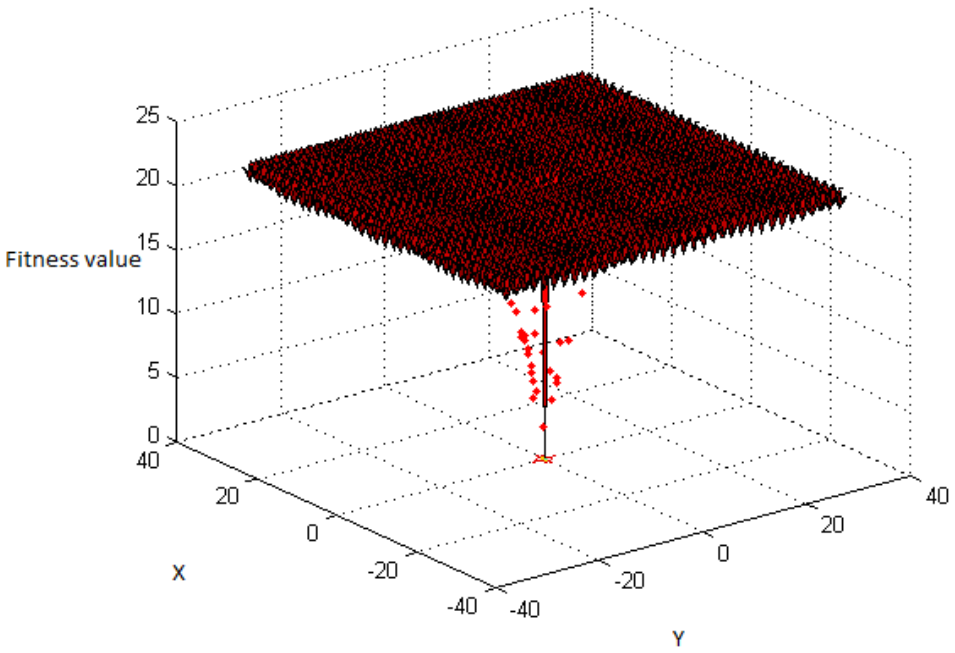
Dimensions		**Original**	**Modified**
2-D	**Position**	-0.3183, 1.5706	2.2029,1.5

	Gbest Value	-1.37	-1.8013
	Time	0.457	0.48401
3-D	**Position**	0.1514,1.5707	2.2030,1.5708,1.2850
	Gbest Value	-1.959	-2.7604
	Time	0.0498	0.055148
4-D	**Position**	1.16,-0.7196	2.2030,1.5707,1.2850
	Gbest Value	-2.2667	-3.6571
	Time	0.057	0.0739

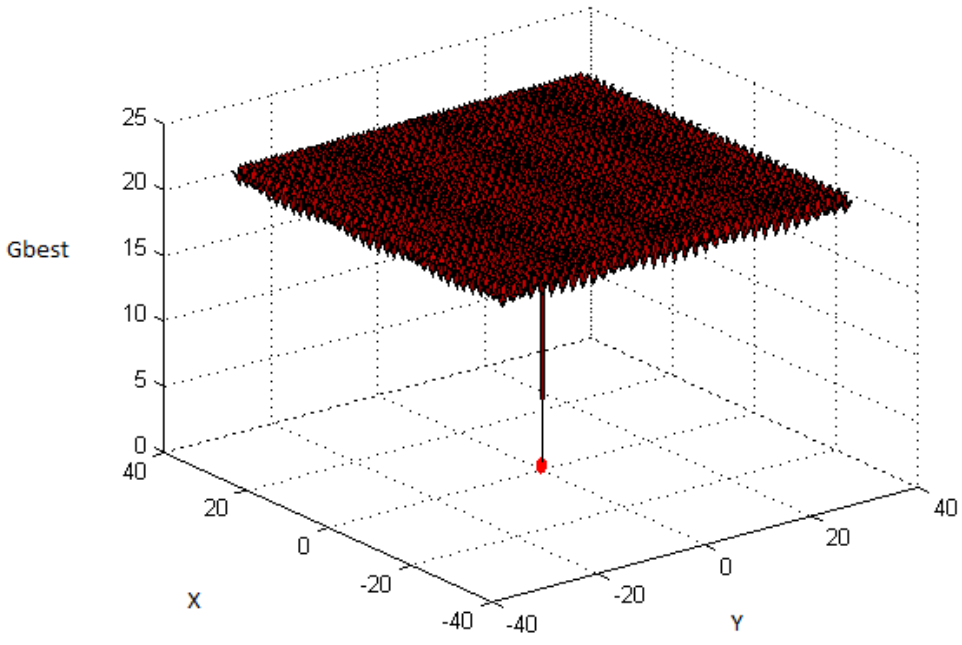
4.3 Graphs showing the global minima of functions

4.3.1 Ackley function

Fig. 5 shows the Global optimal solution of the Ackley function. (a) shows the distributed fireflies in the whole search space and their movement towards the optimal solution which is the global minimum of Ackley function. (b) shows the convergence of all fireflies to the optimal solution of the Ackley function after the last generation thus giving the optimal solution.



(a)

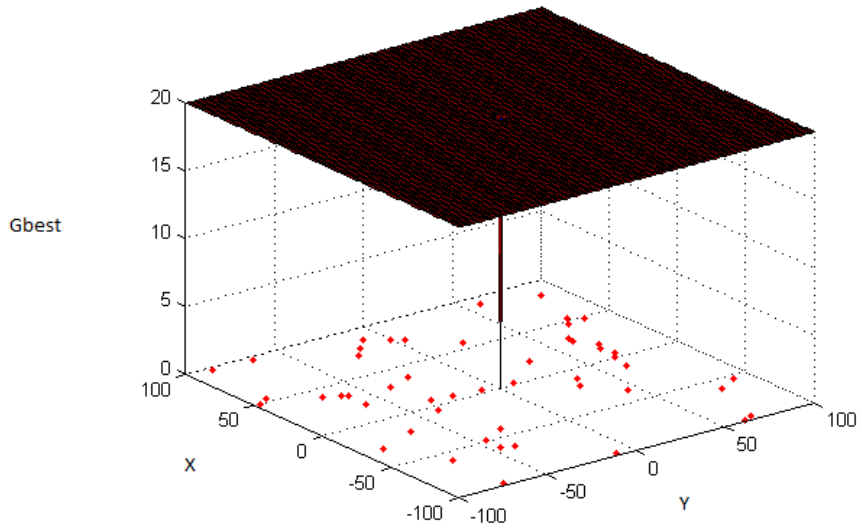


(b)

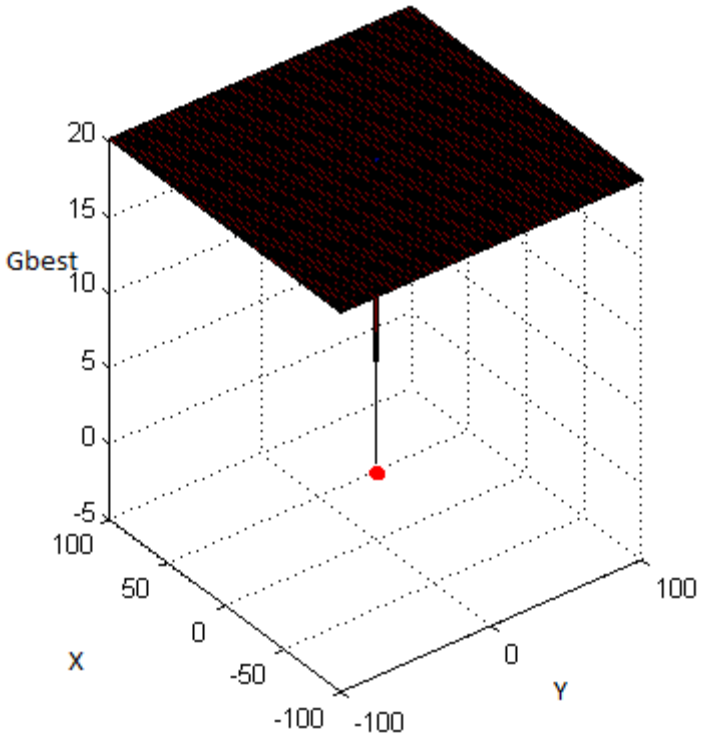
Fig. 5 Graphs showing convergence of fireflies towards optimal solution of Ackley function

4.3.2 Easom function

Fig. 6 shows the Global optimal solution of the Easom function. (a) Shows the distributed fireflies in the whole search space and their movement towards the optimal solution which is the global minimum of Easom function. (b) Shows the convergence of all fireflies to the optimal solution of the Easom function after the last generation thus giving the optimal solution.



(a)

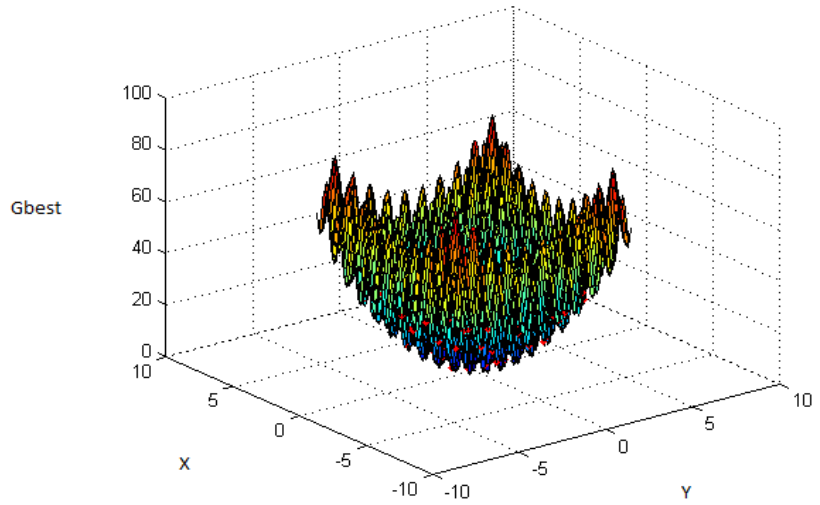


(b)

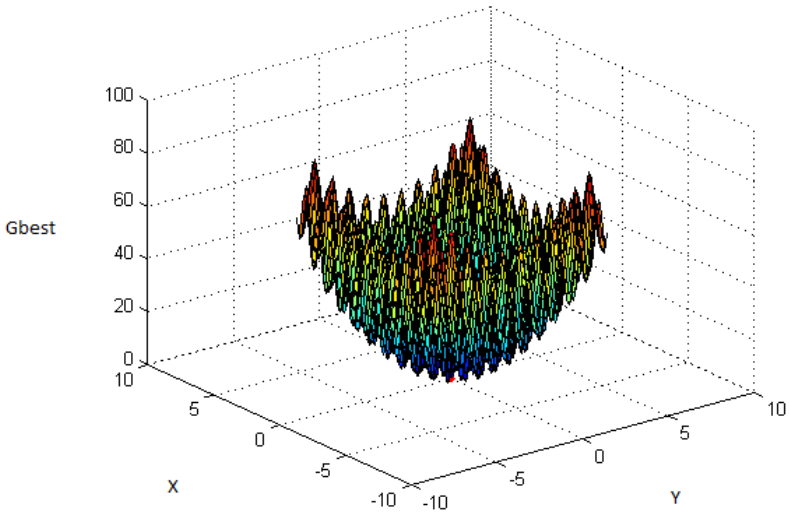
Fig. 6 Graphs showing convergence of fireflies towards optimal solution of Easom function

4.3.3 Rastringin function

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



(a)

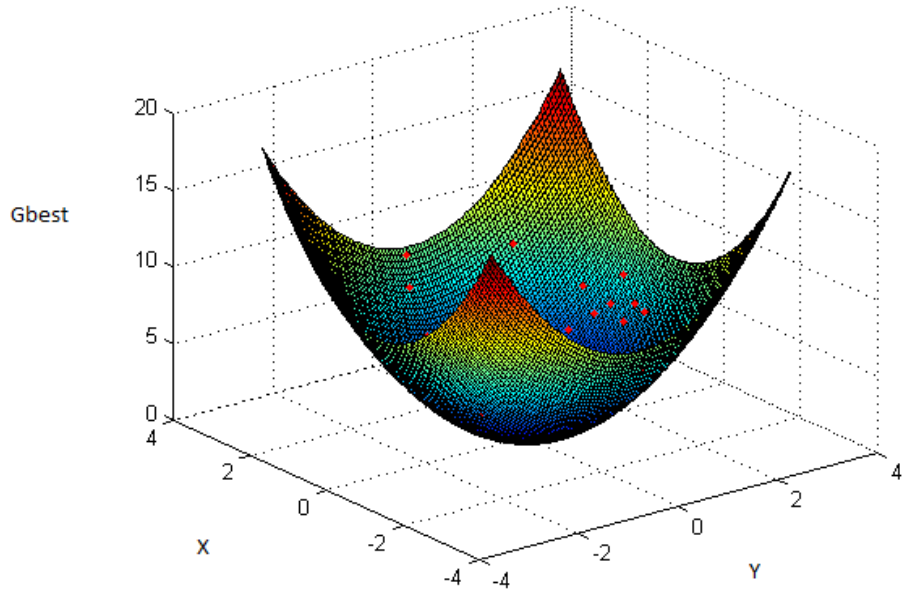


(b)

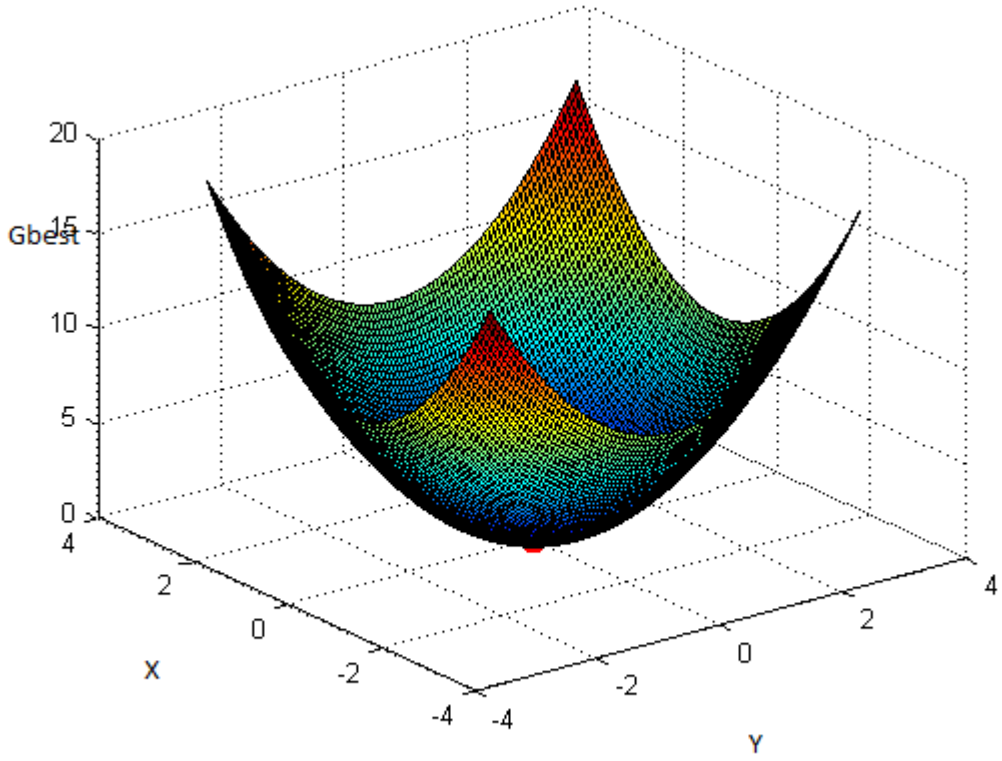
Fig. 7 Graphs showing convergence of fireflies towards optimal solution of Rastringin function

4.3.4 De Jong function

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



(a)

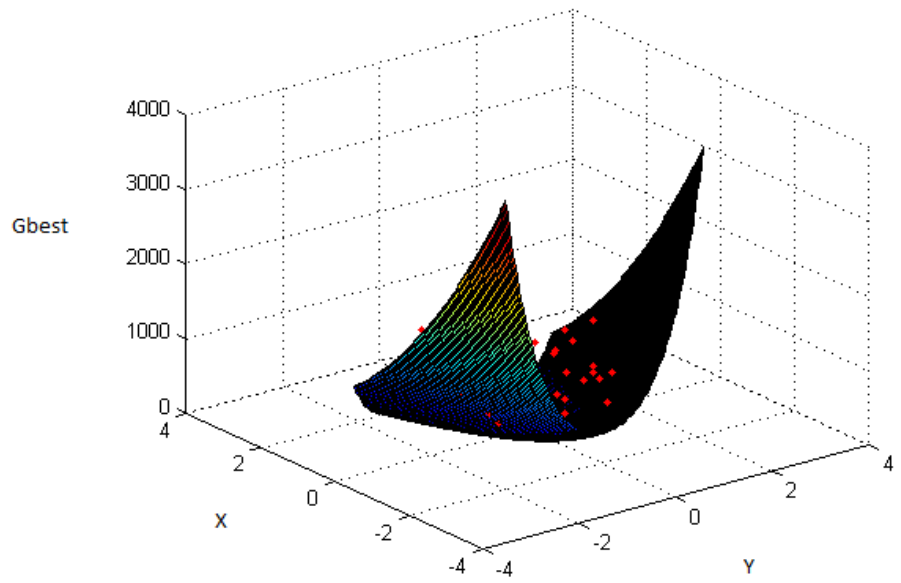


(b)

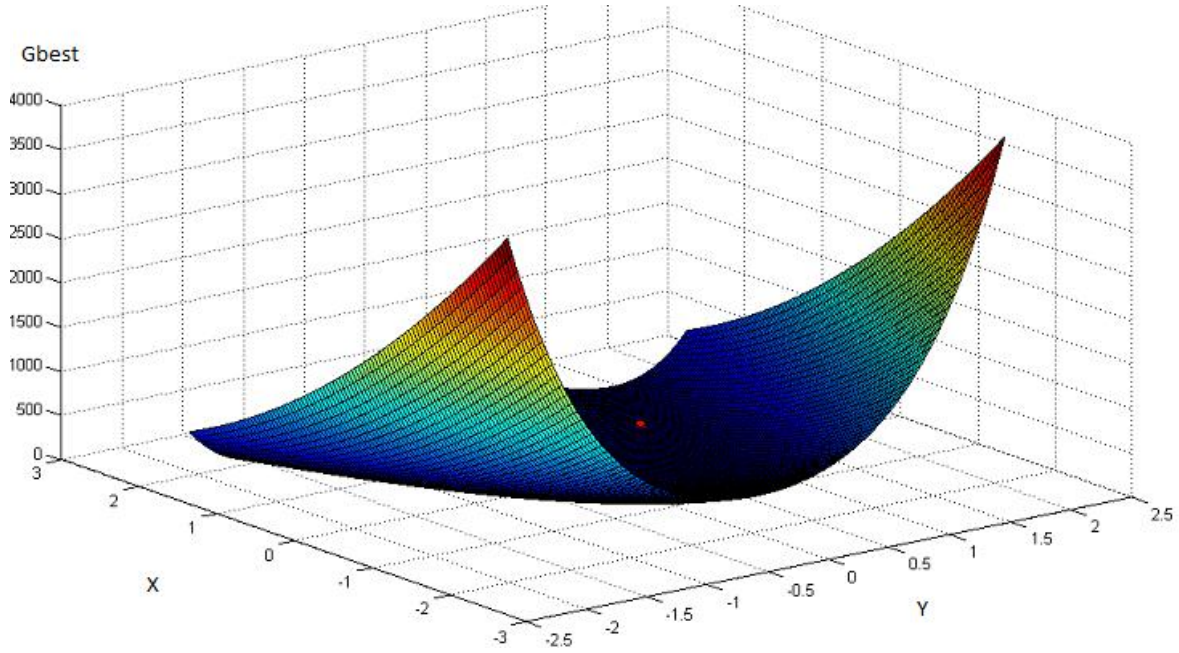
Fig. 8 Graphs showing convergence of fireflies towards optimal solution of De Jong function

4.3.5 Rosenbrock function

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



(a)

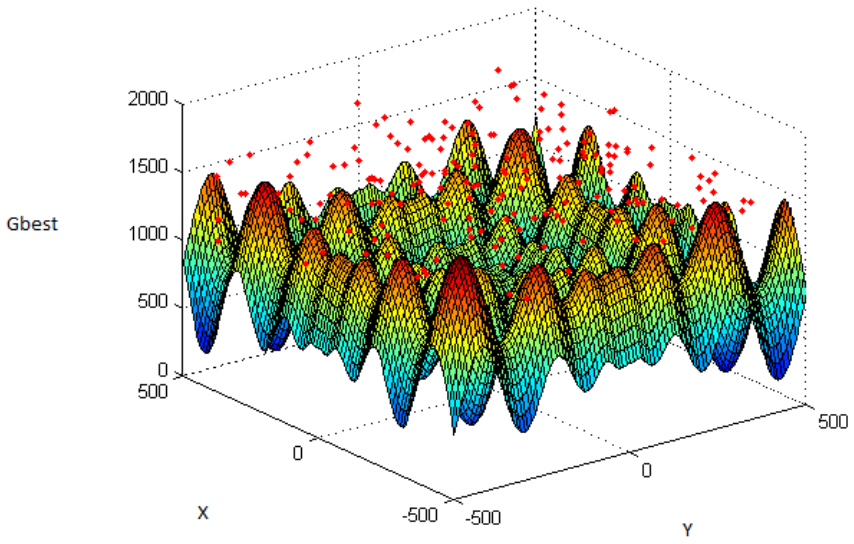


(b)

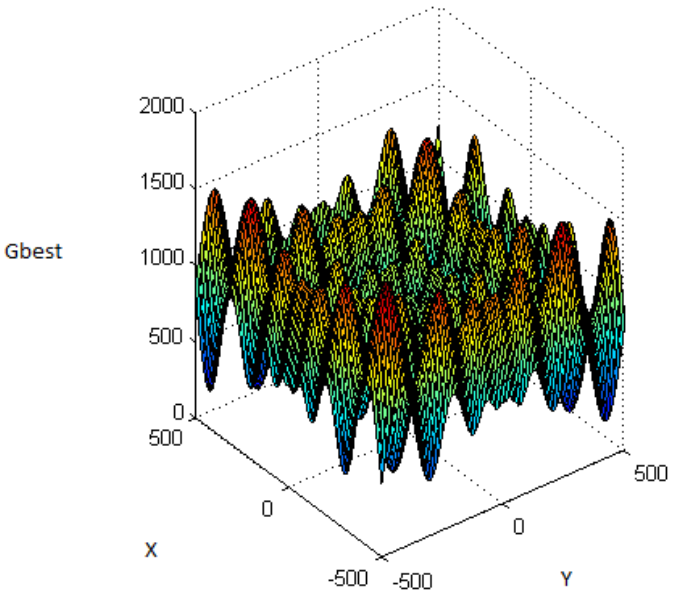
Fig. 9 Graphs showing convergence of fireflies towards optimal solution of Rosenbrock function

4.3.6 Schwefel function

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



(a)



(b)

Fig. 9 Graphs showing convergence of fireflies towards optimal solution of Rosenbrock function

CHAPTER 5
CONCLUSIONS

Conclusions

ODFA performs better than the original FA. In original FA, the position of the fireflies in D-dimensional space represents the candidate solution of the optimization problem. The initialization of these fireflies in original FA is purely random so it can be possible that all fireflies initially take their random positions in the same direction whereas the global solution lies in the other direction due to which the time taken to reach the global solution increases and there can exist a chance that fireflies stuck in the local optima. The opposition based FA helps in initialization of the fireflies more efficiently so that fireflies converge faster. It initializes the fireflies in both directions to explore the search space. Secondly, In the FA, Each update step is performed on a full D-dimensional particle. This leads to the possibility of some components in the particle having been moved closer to the solution, while others have actually been moved away from the solution. The dimensional FA helps FA to not to stuck in the local optima and hence gives the accurate global optima. Each firefly attempts to also optimize a single dimension of the solution vector instead of just as a whole item. It means that we test not only the particle as a whole item, but also the particle in each dimension. The results are more efficient and the time complexity of the modified FA is also less as compared to FA. The dimensional and the opposition based Firefly algorithm is applied on several standard optimization test functions, and it is observed that the results are more accurate as well as the time taken to get the optimal solution is much less as compared to FA.

Appendix A

List of Functions used for verification of proposed algorithm

- Ackley function

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

Where, a = 20, b = 0.2 and c = 2π, $x_i \in$[-32.768 32.768] for all i =1,……, d

Global minimum: $f(x)$=0, at x=(0,…..,0)

- Rosenbrock function

$$f(x) = \sum_{i=1}^{d-1}[100(\text{x}_{i+1} - x_i^2)^2 + (x_i - 1)^2]$$

Where, $x_i \in$[-2.048, 2.048] for all i =1,……, d

Global minimum: $f(x)$=0, at x=(1,…..,1)

- De Jong function 1

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

Where, $x_i \in$[-5.12, 5.12] for all i =1,……, d

Global minimum: $f(x)$=0, at x=(0,…..,0)

- Griewank function

$$f(x) = \sum_{i=1}^{d}\frac{x_i^2}{4000} - \prod_{i=1}^{d}\cos(\frac{x_i}{\sqrt{i}}) + 1$$

Where, $x_i \in$[-600, 600] for all i =1,……, d

Global minimum: $f(x)$=0, at x=(0,…..,0)

- Easom function

$$f(x) = -\cos(x_1)\cos(x_2)\exp(-(x_1 - \pi)^2 - (x_2 - \pi)^2)$$

Where, $x_1 \in$[-100, 100], $x_2 \in$[-100, 100]

Global minimum: $f(x)$=-1, at x=(π, π)

- Shubert function

$$f(x) = (\sum_{i=1}^{5} i\cos((i+1)x_1 + i))(\sum_{i=1}^{5} i\cos((i+1)x_2 + i))$$

Where, $x_1 \in$[-10, 10], $x_2 \in$[-10, 10]

Global minimum: $f(x)$=-186.7309

- Schwefel function

$$f(x) = 418.9829\,\mathrm{d} - \sum_{i=1}^{d} x_i \sin(\sqrt{|x_i|})$$

Where, $x_i \in$[-500,500] for all $\mathrm{i} = 1, \ldots\ldots, \mathrm{d}$

Global minimum: $f(x)$=0, at x=(1,…..,1)

- Rastrigin function

$$f(x) = 10d + \sum_{i=1}^{d} [x_i^2 - 10\cos(2\pi x_i)]$$

Where, $x_i \in$[-5.12, 5.12] for all $\mathrm{i} = 1, \ldots\ldots, \mathrm{d}$

Global minimum: $f(x)$=0, at x=(0,…..,0)

- Michalewicz function

$$f(x) = -\sum_{i=1}^{d} \sin(x_i) \sin^{2m}(\frac{ix_i^2}{\pi})$$

Where, m=10, $x_i \in [0, \pi]$

Global minimum: At d=2, $f(x)$=-1.8013, at x=(2.20,1.57)

References

[1] X. S. Yang, "Nature inspired metahuristic algorithm, Luniver press, 2008, pp. 81-95.

[2] X. S. Yang, "Firefly algorithm, stochastic test functions and design optimization", Int. J. bio-inspired computation, Vol. 2, No. 2, 2010, pp. 78-84.

[3]X. S. Yang, "Firefly algorithm, lévy Flights and global optimization", Research and Development in Intelligent Systems, Vol. 26, Springer-Verlag, 2010, pp. 209-218.

[4]X. S. Yang, "Multiobjective firefly algorithm for continuous optimization", Engineering with Computers, Vol. 29, Issue 2, Springer-Verlag, 2013, pp. 175-184.

[5] Ming-Huwi Horng, "Vector quantization using the firefly algorithm for image compression", Expert Systems with Applications,Vol. 39, Issue 1, Elsevier, 2012, pp. 1078–1091.

[6] Ming-Huwi Horng, Ren-Jean Liou, "Multilevel minimum cross entropy threshold selection based on the firefly algorithm", Expert Systems with Applications,Vol. 38, Issue 12, Elsevier, 2011, pp. 14805–14811.

[7] Amir Hossein Gandomi, X. S. Yang, Amir Hossein Alavi, "Mixed variable structural optimization using firefly algorithm", Computers & Structures, Vol. 89, Issues 23–24, Elsevier, 2011, pp. 2325–2336.

[8] J. Senthilnath, S.N. Omkar, V. Mani," Clustering using firefly algorithm: Performance study", Swarm and Evolutionary Computation, Vol. 1, Issue 3, Elsevier, 2011, pp. 164–171.

[9] Gilang Kusuma Jati, Suyanto,"Evolutionary discrete firefly algorithm for Travelling Salesman Problem", Adaptive and Intelligent Systems, Vol. 6943, Springer-Verlag, 2011, pp. 393-403.

[10] H. R. Tizhoosh, "Opposition-based learning: A new scheme for machine intelligence,"Proc. of Int. Conf. Comput. Intell. Modeling Control and Autom., Vienna, Austria, 2005, Vol. 1, pp. 695–701.

[11] H. R. Tizhoosh, "Reinforcement learning based on actions and opposite actions," in Proc. ICGST Int. Conf. Artif. Intell. Mach. Learn., Cairo, Egypt, 2005, Vol. 10 No. 4 pp. 578-585.

[12] H. R. Tizhoosh, "Opposition-based reinforcement learning," J. Advanced Comput. Intell. Intelligent Inform., Vol. 10, No. 3, 2006, pp. 578–585.

[13] Sahba, F, H. R. Tizhoosh," Application of opposition-based reinforcement learning in image segmentation", Computational Intelligence in Image and Signal Processing, CIISP 2007, pp. 246-251.

[14] Mahootchi, M., Tizhoosh, H.R. ; Ponnambalam, K.," Opposition-based reinforcement learning in the management of water resources", Approximate Dynamic Programming and Reinforcement Learning, 2007. ADPRL 2007,pp. 217-224.

[15] M. Ventresca and H. R. Tizhoosh, "Improving the convergence of backpropagation by opposite transfer functions,"in Proc. EEE World Congr. Comput. Intell., Vancouver, BC, Canada, 2006, pp. 9527–9534.

[16] S. Rahnamayan, H. R. Tizhoosh, and M. M. A Salama, "Opposition versus randomness in soft computing techniques," Applied Soft Computing, Vol. 8, Issue 2, Elsevier, 2008, pp. 906–918.

[17] 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.

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

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

[20] Swagatam Das, Arijit Biswas, Sambarta Dasgupta1, and Ajith Abraham, "Bacterial Foraging Optimization Algorithm: Theoretical Foundations, Analysis, and Applications", Vol. 203, 2009, pp. 23-55.