SYSTEMATIC STUDY OF MAYFLY ALGORITHM WITH APPLICATIONS

A DISSERTATION
SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE AWARD OF THE DEGREE
OF

MASTERS OF SCIENCE
IN
MATHEMATICS

Submitted by :

AKASH JAIN

Roll No : 2k19/MSCMAT/21

Under the Supervision of :

Prof . ANJANA GUPTA

Department of Applied Mathematics

Delhi Technological University (DTU),

New Delhi -110042

MAY, 2021

Delhi Technological University
(Formerly Delhi College of Engineering)
Bawana Road, New Delhi -110042

CANDIDATE'S DECLARATION

I , Akash Jain , Roll No : 2k19/MSCMAT/21 hereby declare that the thesis entitled " Systematic Study Of Mayfly Algorithm with Applications " which is submitted to Department of Applied Mathematics , Delhi Technological University (DTU), New Delhi in partial fulfillment for the award of degree of Masters in Mathematics is original and not copied from any source without giving proper citations. This work has not been submitted either partially or fully to any other university or college for the basis of award of degree / diploma or any fellowship.

Date: 17.05.21

Name : Akash Jain
Enrollment No : 2K19/MSCMAT/21

Department of Applied Mathematics
Delhi Technological University
(Formerly Delhi College of Engineering)
Bawana Road, New Delhi -110042

CERTIFICATE

I hereby certify that the thesis entitled “Systematic Study of Mayfly Algorithm with Applications” which is submitted to Department of Applied Mathematics, Delhi Technological University (DTU), New Delhi in partial fulfillment for the award of degree of Masters in Mathematics is carried out under my observation and supervision.

Date: 17.05.21

Prof .Anjana Gupta

Department of Applied Mathematics
Delhi Technological University (DTU)
New Delhi-110042

ACKNOWLEDGEMENT

I would like to take this opportunity to thank and express the notion of gratitude towards Department Of Applied Mathematics, Delhi Technological University for providing such a wonderful platform to inculcate the research aptitude in the students.

I express my heartiest & sincere gratitude to our Head of department Prof .S. Sivaprasad for his administration towards our academic growth.

I also express my sincere gratitude to our Coordinator cum mentor Prof . Anjana Gupta for her guidance and constant inspiration provided in successful completion of this report. Without her regular monitoring and encouragement this project wouldn't be completed.

Lastly I would like to thank almighty God , my parents , family , friends and teachers for believing in me and giving this opportunity.

AKASH JAIN

ABSTRACT

In Anthropology there is theory of Evolution by Charles Darwin based on the concept of Survival of the fittest. So as a consequence of it every living organism be it human beings , animals , insects, or even micro-organisms like Coronavirus have to adapt , mitigate and become resilient with environment if they want to survive . That means there is a constant learning with some feedback error so that the species will introduce desired changes in them. That particular thing (Learning with feedback) is the backbone of Soft Computing. In light of Bio-Inspired Computing we are dealing with the very recent algorithm which is Mayfly Algorithm (MA) developed in May -2020 itself . In this project we have done a thorough review of Mayfly Algorithms and the recent developments happened in the Mayfly Algorithm and with various future applications of it.

List of Conferences/Publications

CONFERENCE 1) International Conference on Paradigms of Communication , Computing and Data Sciences (PCCDS, 2021) .

Organisers : National Institute of Technology , Kurukshetra and Soft Computing Research Society

Title of the Paper : REVIEW ON THE RECENT DEVELOPMENTS IN THE MAYFLY ALGORITHM

Authors : Akash Jain and Anjana Gupta

Date of Full Length
Paper Submission : 25 February 2021
Date of Acceptance : 4 th April 2021
Date of Registration : 5 th April 2021
Date of Camera Ready
Paper submission : 9th April 2021
Date of Conference : 7-9 May 2021
Date of Publication : By June 2021(Expected)
Name of Publication : Springer Book : Algorithms for Intelligent systems
Publisher : Springer Nature Singapore
Link of Publication : https://www.springer.com/series/16171

CERTIFICATE OF PRESENTATION AND PARTICIPATION :


International Conference on
Paradigms of Communication, Computing
and Data Sciences (PCCDS 2021)
https://www.pccds21.scrs.in/

E-CERTIFICATE OF PARTICIPATION

AKASH JAIN

presented the paper titled
Review on the recent developments in the Mayfly Algorithm
authored by
Akash Jain and Anjana Gupta
in the International Conference on Paradigms of Communication, Computing and Data Sciences (PCCDS 2021) held during May 07-09, 2021 in India.

SCRS/PCCDS2021/162

Prof. Mayank Dave
(General Chair)

Dr. J. C. Bansal
(General Secretary, SCRS)

Organized By
National Institute of Technology, Kurukshetra, India and Soft Computing Research Society

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS AND ABBREVIATIONS

- MA - Mayfly Algorithm
- GA - Genetic Algorithm
- PSO - Particle Swarm Optimisation
- HS - Harmony Search
- OBL - Opposition Based learning
- MMA - Multiobjective Mayfly Algorithm

CHAPTER-1
Introduction

1.1 General

Generally Optimisation problems are broadly classified based on the nature of the objective functions and constraints. If the objective function and constraint both are linear in nature then the Optimisation problem is linear and can be easily solved with the help of Linear programming techniques like Simplex, Ellipsoid and Karmakar's Interior point method . But in case of Non Linearity if suppose our objective function or constraints or both of them appears to be containing a non linear term then in that scenario we go for calculus based optimization techniques but the problem is that these techniques are based on the intuition to compute derivative and hence finding critical points for to check optimality. So doing optimization with the help of calculus has two significant issues : (a) Problem of getting stuck in local optima only (b) for to find derivative we have to check certain properties of function whether this function exist or not . It's like saying that to compute a derivative we have to verify that given function is differentiable or not . So there is a need of going for alternative of derivative based optimization techniques which are also gradient free optimization techniques . These techniques are called Meta-heuristics search techniques. These are broadly classified into Evolutionary Intelligence like Genetic Algorithm , Differential Evolution and Swarm Intelligence like PSO , Firefly Algorithm etc.

1.2 Genetic Algorithm [1]

Genetic Algorithm GA (genetic algorithm) was introduced in the 1960s by Holland and further analyzed by Goldberg(1989).

It has basically three operators : Crossover , Mutation and Selection operators.

Crossover operator : Used for exploration of the search space.

Mutation operator : Used for exploitation of the search space.

Selection Operator : Used for discretization of the search space.

Crossover : Suppose Parents are abc:de and efg:hi . Then after applying crossover last parts are swapped then new offsprings are formed which are abchi and efgde respectively .

Mutation : In this we have to accept those offsprings having fitness value strictly greater than parents otherwise there would be no meaning of evolution of present generation .

1.2.1 Working Of Genetic Algorithm

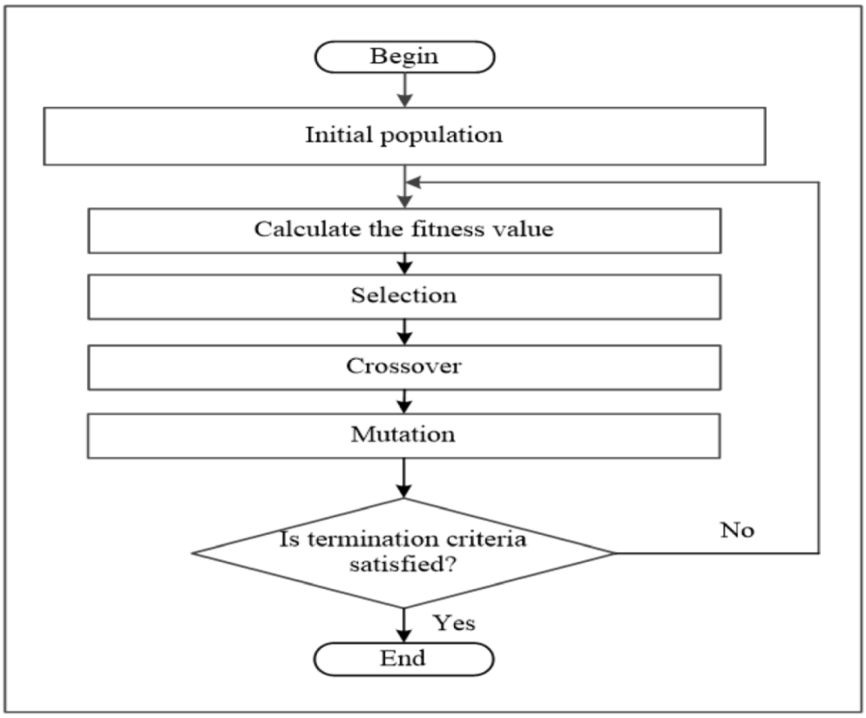


Fig 1.1 Figure depicting working of genetic algorithm

Source Image : *Albadr, Musatafa Abbas, et al. "Genetic Algorithm Based on Natural Selection Theory for Optimization Problems." Symmetry 12.11 (2020): 1758.*

NUMERICAL OF GENETIC ALGORITHM

Maximize $f(x) = x^3$ with x is in the set {0,1,2,,,,,,,,,….30}

Step 1) Generate the initial population at random . these are called chromosomes/genotypes.

Foreg: 01101 is binary code of 13
11000 is binary code of 24
01001 is binary code of 9
10101 is binary code of 21

Step 2) Calculate the fitness values with the help of fitness function which is a cubic function.
For eg : 13 corresponds to 2197
24 corresponds to 13,824
07 corresponds to 343
18 corresponds to 5,832

F_h = Fitness value for the string h in the population.

p_h = probability of string h being selected.

n = Number of individuals in the population.

$n * p_h$ = Expected Count .

This Method of selection is commonly known as Roulette Wheel Selection .

Table 1.1 SELECTION TABLE

String No	Initial Population	X value	Fitness value	p_h	Expected Count
1.	01101	13	2197	0.084	0.336
2.	11000	24	13,824	0.531	2.124
3.	01001	09	729	0.028	0.112
4.	10101	21	9,261	0.356	1.424
SUM			26,011		

(least discard with highest ie 3rd string by 2nd string)

CROSSOVER :

For eg : 100|11101 (An example of one point crossover)
101|01011

Thus concatenated offsprings are : 100 01011
101 11101

Similary one can we even go for n points if the string has at least n +1 characters.

Table 1.2 Crossover table

String No	Mating Pool	Crossover Point	Offsprings after Crossover	Integer value x	$F(x) = x^3$
1.	0110 | 1	4th	01100	12	1,728
2.	1100 | 0	4th	11001	25	15,625
2.	11|000	2nd	11101	29	24,389
4.	10| 101	2nd	10000	16	4,096
SUM					45,838

Clearly fitness value is 45,838 much greater than as compared to previous 26,011 due to only selection operator applied .

MUTATION :

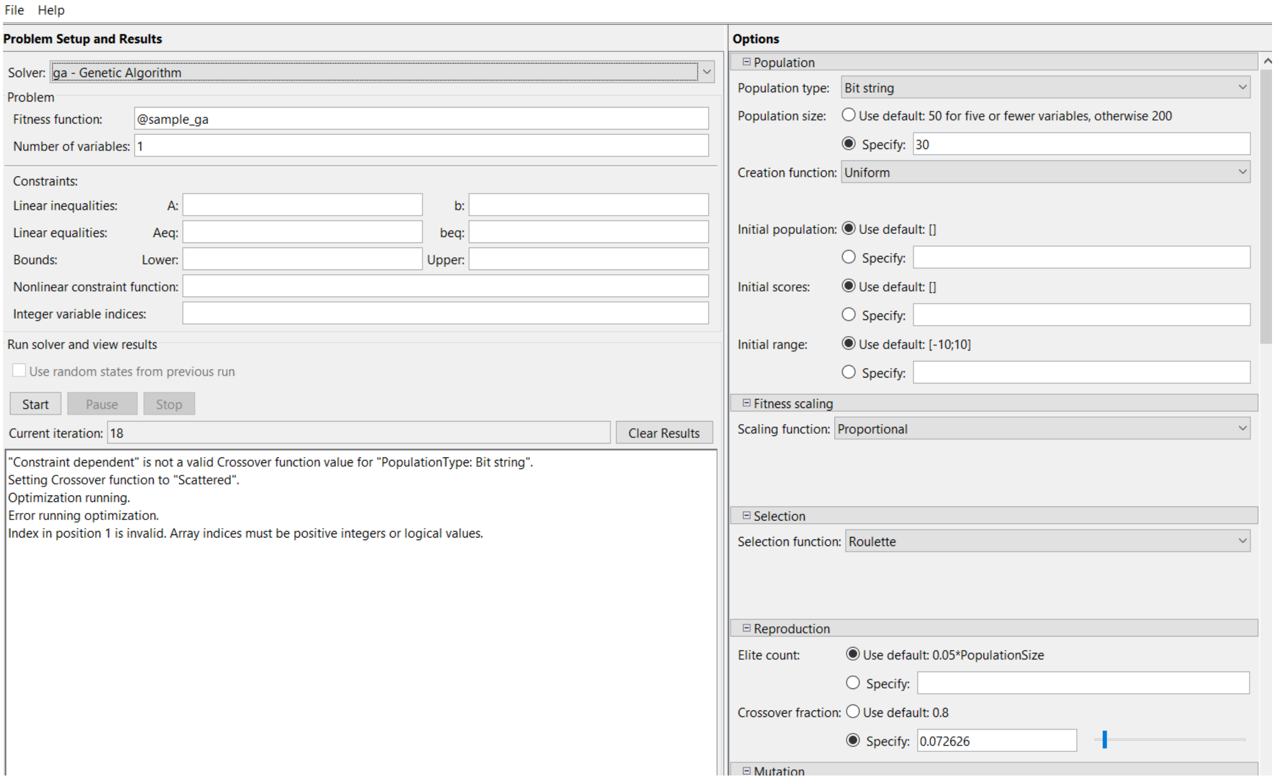
- Applied to each child individually after crossover technique.
- Bits are changed from 0 to 1 or viceversa 1 to 0 at randomly chosen places of randomly selected strings.
- Since the string number 2 and string number 2 are the highest fitness value obtained in the previous stage. So No need to alter them.

Table 1.3 Mutation Table

String Number	Offspring after crossover	Offspring after Mutation	X-value	Fitness value
1.	01100	11100	26	17,576
2.	11001	11001	25	15,625
2.	11101	11101	29	24,389
4.	10000	10100	18	5,832
Sum				63,422

1.2.2 Matlab work of Genetic Algorithm

Using genetic Toolbox



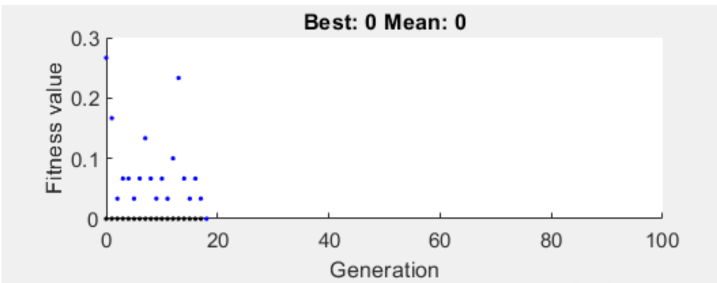


Fig1.2 Showing Fitness value vs generation

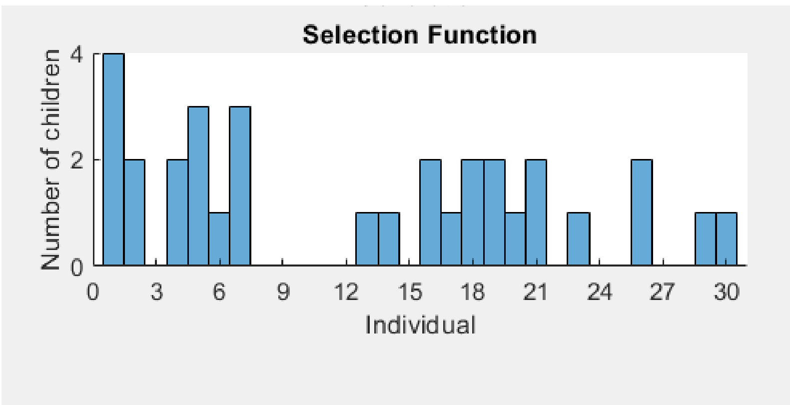


Fig1.3 Showing selection function using Roulette wheel

1.3 PSO [2]

PSO was initially proposed by Ebehart , Kennedy and Shi in 1995 . It basically optimize the function based on the behaviour of swarm or group of animals , flies , insects etc. Intution is to update the velocity and position equations stochastically based on certain random parameters.

1.3.1 Working of PSO

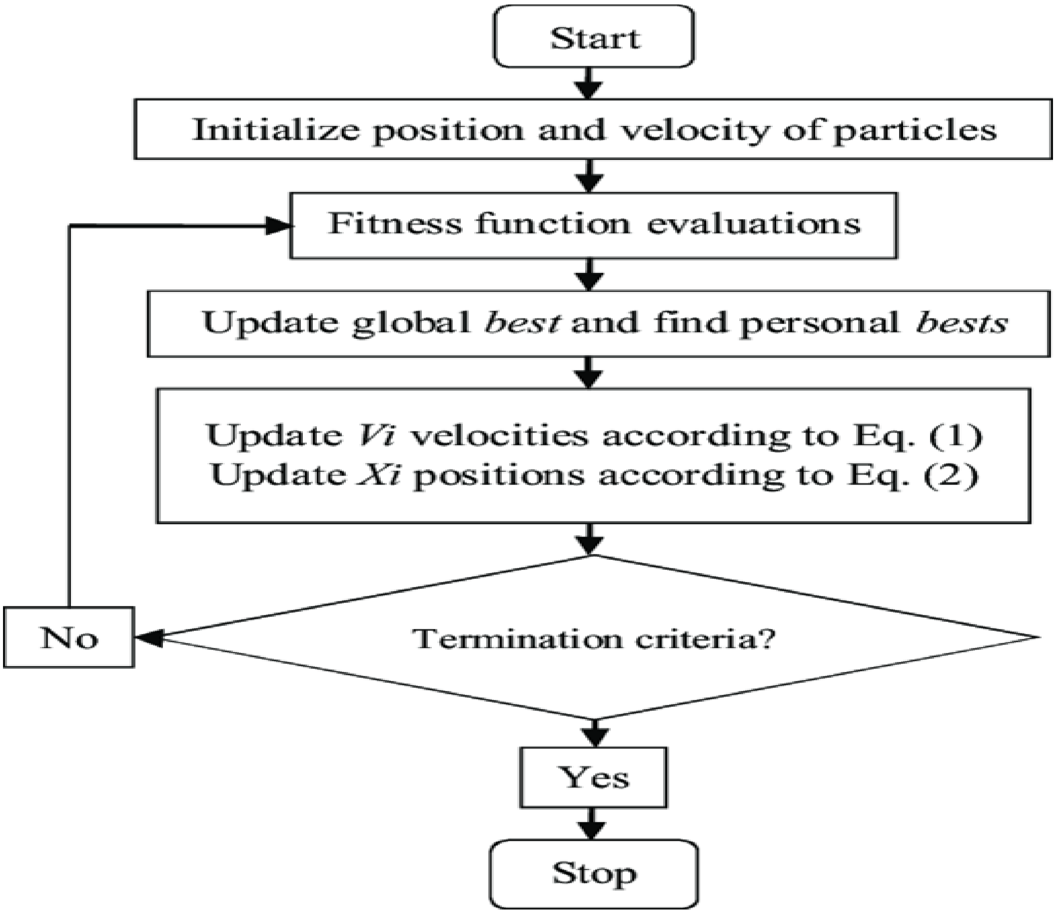


Fig. 1.4 Flowchart representing the working of PSO

Source Image : *Aydilek, Ibrahim Berkan. "A hybrid firefly and particle swarm optimization algorithm for computationally expensive numerical problems." Applied Soft Computing 66 (2018): 232-249.*

Understanding of PSO with help of Numerical

Velocity Update equation :

$$v_{t+1} = w_t v_t + c_1 r_1 (p_t^b - x_t) + c_2 r_2 (p^g - x_t) \qquad (1.1)$$

Position Update equation can be find out as then :

$$x_{t+1} = x_t + v_{t+1} \qquad (1.2)$$

Here symbols are as :

t → Iteration Number Count

r_1, r_2 → Random number between 0 and 1

p_t^b → pbest position at t iteration

v_t → Velocity at t iteration

x_t → Position at t iteration

w_t → Inertia weight

$c_{1,} c_2$ → Correction factors (parameters)

p^g → Global best position

NUMERICAL :

Maximize $f(x) = x_1^2 - x_1 * x_2 + x_2^2 + 2 * x_1 + 4 * x_2 + 3$ where $-5 \le x_1, x_2 \le 5$.

Here we are taking Population size =5

$c_1 = c_2 = 1.5$

Max Iteration=20

Dimension of Problem= 2 (Number of linearly independent vectors)

Inertia weight (w) = 0.9

- Since population size =5 . So 5 components are there $v_{11}, v_{12}, v_{13}, v_{14}$ & $v_{15.}$
- Similarly $v_{21}, v_{22,} v_{23}, v_{24}$ & $v_{25.}$

For first Iteration :

- Randomly choose velocity between 0 and 1.
- For position $x = L + rand * (U - L)$ where L and U can be considered as extreme points of domain -5 and 5 respectively.
- Calculate the fitness value f(x).
- Since there is no previous iteration present for comparison. So $p^{best} = x(position)$ itself.
- Calculate $g_{best}(p^g)$. Since our problem is of maximization type . So maximum fitness Value =48.67 corresponding to that position value will be taken.

V (Velocity)

(v1)	(v2)
0.219	0.224
0.190	0.221
0.381	0.321
0.397	0.352
0.093	0.377

x (Position)

(x1)	(x2)
3.147	-4.024
4.057	-2.215
-3.730	0.462
4.133	4.575
1.323	4.648

F(x) =

31.964
32.616
13.267
48.675
41.453

3.147	-4.024
4.057	-2.215
-3.730	0.462
4.133	4.575
1.323	4.648
(x1)	(x2)

p_i^b (Personal Best Position)

[4.133 4.575] -→→→ p^g (Global Best Position)

Iteration Number -2

For Ist Particle , Ist component we will get using PSO velocity update equation

Which is Eqn (1.1)

$$v_{t+1} = w_t v_t + c_1 r_1 (p_t^b - x_t) + c_2 r_2 (p^g - x_t)$$

v_{11} = (0.9*0.291)+(1.5*0.5949)*(3.1472-3.1472)+1.5*(0.085)*(4.138-3.1472)
= 0.3240

Therefore our position using Eqn (1.2) would become

$$x_{t+1} = x_t + v_{t+1}$$

x_{11} → 3.4712 which belongs to (-5,5)

0.324	1.323
0.181	1.071
1.353	0.817
0.357	0.319
0.447	0.330
(v1)	(v2)

V (velocity)

3.471	-2.701
4.239	-1.143
-2.372	1.886
4.491	**4.894**
1.768	4.971

X (Position)

$$F(x) = \begin{vmatrix} 27.866 \\ 31.032 \\ 13.753 \\ \mathbf{53.706} \\ 45.565 \end{vmatrix}$$

Clearly , $p^g best$ = [4.491 4.894]

- Since 53.706 > 48.675 (means fitness value of 2nd Iteration > fitness value of Ist Iteration)

- $$p_i^b(best) = \begin{vmatrix} 3.147 & -4.024 \\ 4.057 & -2.215 \\ -2.377 & 1.286 \\ 4.491 & 4.894 \\ 1.768 & 4.974 \end{vmatrix}$$

The Upper two rows in p_i^b are same as in that fitness value in 2nd iteration less than that of Ist one . So no need to change.

In last three rows since our 2nd iteration is greater than Ist one so applying greedy based approach we will change .

REMARK : How to handle boundary violation .

In position update equation if x_{ij} doesn't belongs to given domain say (-5,5) .

Then for eg : If x_{11} = 5.2131 clearly doesn't belongs to (-5,5) we choose 5.000

by default and same goes negative direction as well -5.2131 = -5.000

STOPPING CRITERION :

In these algorithms stopping criterion is said to be reached whenever the difference between two consecutive iterations become acceptable or permissible .

Similarly we can perform Iteration Number -3

V (velocity)

0.133	0.984
0.033	0.782
2.098	1.198
0.321	0.287
0.749	0.286

x (position)

3.604	-1.717
4.293	-0.3611
-0.278	2.4848
4.813	5.000
2.151	5.000

F(x) =

25.471
30.034
19.325
56.730
46.785

$p_i^b (pbest)$

3.471	-2.701
4.237	-1.143
-0.278	2.484
4.813	5.000
2.517	5.000

$p^g (gbest)$ = [4.8137 5.000]

Now we will perform Iteration No :4th

V (velocity)

0.156	0.869
0.069	0.693
2.541	1.401
0.289	0.258
0.968	0.257

x (position)

3.760	-0.848
4.342	0.332
2.263	3.886
5.000	5.000
3.488	5.000

F(x) =

25.179
30.540
34.499
58.000
49.695

$p^g(gbest)$ = [5.000 5.000]

- Since still error is not negligible . So we will perform one more iteration.

Iteration Number -5 :

V (Velocity)

0.160	0.756
0.146	1.222
2.638	1.404
0.260	0.232
1.066	0.231

x (Position)

3.920	-0.094
4.489	1.555
4.902	5.000
5.000	5.000
4.552	5.000

$$F(x) = \begin{vmatrix} 26.215 \\ 33.793 \\ 57.324 \\ \mathbf{58.000} \\ 55.067 \end{vmatrix}$$ ← **Optimal value**

$$p_i^b(pbest) = \begin{vmatrix} 3.920 & -0.091 \\ 4.489 & 1.555 \\ 4.902 & 5.000 \\ 5.000 & 5.000 \\ 4.552 & 5.000 \end{vmatrix}$$

$$p^{g(best)} = [\ 5.000 \quad 5.000]$$

Also the stopping criterion reached as

$$|\ f_{\max}^{iter(5th)} - f_{\max}^{iter(4th)}\ | = |\ 58 \ -58\ | = 0$$

Required Answer : $x_1 = 5, x_2 = 5, f(x_1, x_2) = 58$

For our unconstrained, Nonlinear , Maximization type PSO problem.

1.3.2 Matlab work of PSO

Editor - C:\Users\Perfect\Downloads\psoakash.m

psoakash.m | fun.m

```
tic
clc
clear all
close all
rng default
LB=[-5 -5]; %lower bounds of variables
UB=[5 5]; %upper bounds of variables
% pso parameters values
m=2; % number of variables
n=5; % population size
w=0.9; % inertia weight
c1=1.5; % acceleration factor
c2=1.5; % acceleration factor
% pso main program----------------------------------------------------start
maxite=20; % set maximum number of iteration
maxrun=10; % set maximum number of runs need to be
for run=1:maxrun
run
% pso initialization----------------------------------------------start
```

(a)

```
for i=1:n
for j=1:m
x0(i,j)=round(LB(j)+rand()*(UB(j)-LB(j)));
end
end
x=x0; % initial population
v=0.1*x0; % initial velocity
for i=1:n
f0(i,1)=fun(x0(i,:));
end
[fmax0,index0]=max(f0);
pbest=x0; % initial pbest
gbest=x0(index0,:); % initial gbest
% pso initialization------------------------------------------------end
% pso algorithm---------------------------------------------------start
ite=1;
```

(B)

```
while ite<=maxite && tolerance>10^-12
% pso velocity updates
for i=1:n
for j=1:m
v(i,j)=w*v(i,j)+c1*rand()*(pbest(i,j)-x(i,j))..
+c2*rand()*(gbest(1,j)-x(i,j));
end
end
% pso position update
for i=1:n
for j=1:m
x(i,j)=x(i,j)+v(i,j);
end
end
% handling boundary violations
for i=1:n
for j=1:m
if x(i,j)<LB(j)
x(i,j)=LB(j);
elseif x(i,j)>UB(j)
x(i,j)=UB(j);
end
end
end
% evaluating fitness
for i=1:n
f(i,1)=fun(x(i,:));
end
% updating pbest and fitness
```

(c)

```
if f(i,1)>f0(i,1)
    pbest(i,:)=x(i,:);
f0(i,1)=f(i,1);
end
end
[fmax,index]=max(f0); % finding out the best particle
ffmax(ite,run)=fmax; % storing best fitness
ffite(run)=ite; % storing iteration count
% updating gbest and best fitness
if fmax>fmax0
gbest=pbest(index,:);
fmax0=fmax;
end
% calculating tolerance
if ite>100;
tolerance=abs(ffmin(ite-100,run)-fmin0);
end
% displaying iterative results
if ite==1
disp(sprintf('Iteration Best particle Objective fun'));
end
fprintf('%8g %8g %8.4f\n',ite,index,fmax0);
ite=ite+1;
end
% pso algorithm-----------------------------------------------------en
gbest;
fvalue=gbest(1)^2-gbest(1)*gbest(2)+gbest(2)^2+2*gbest(1)+4*gbest(2)+3
fff(run)=fvalue;
rgbest(run,:)=gbest;
```

(d)

Plot commands

```
disp(sprintf('--------------------------------------'));
end
% pso main program------------------------------------------------------end
disp(sprintf('\n'));
fprintf('**********************************************************\n');
disp(sprintf('Final Results-----------------------------'));
[bestfun,bestrun]=min(fff)
best_variables=rgbest(bestrun,:)
disp(sprintf('**********************************************************'));
toc
% PSO convergence characteristic
plot(ffmax(1:ffite(bestrun),bestrun),'-k');
xlabel('Iteration');
ylabel('Fitness function value');
title('PSO convergence characteristic')
```

(E)

RESULTS :

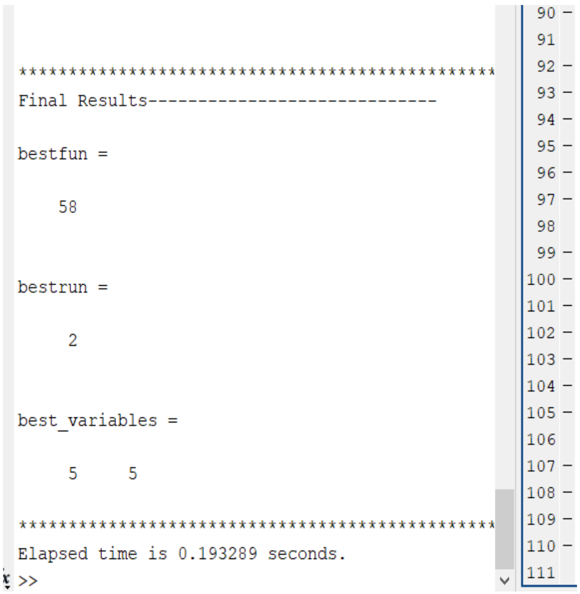


Fig 1.5 Showing Output of our Numerical

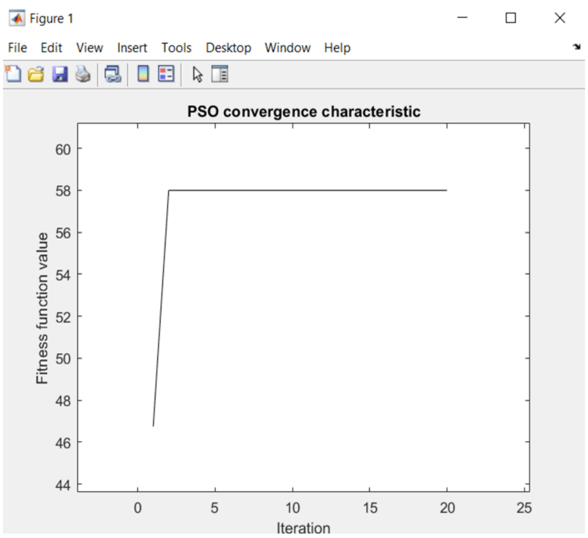


Fig 1.6 Showing Convergence graph of our numerical

1.4 Firefly Algorithm [3]

Firefly Algorithm was recently proposed by Yang in 2007. It becomes a special case of PSO by putting scaling parameter $\gamma = 0$.

1.4.1 Working of Firefly Algorithm

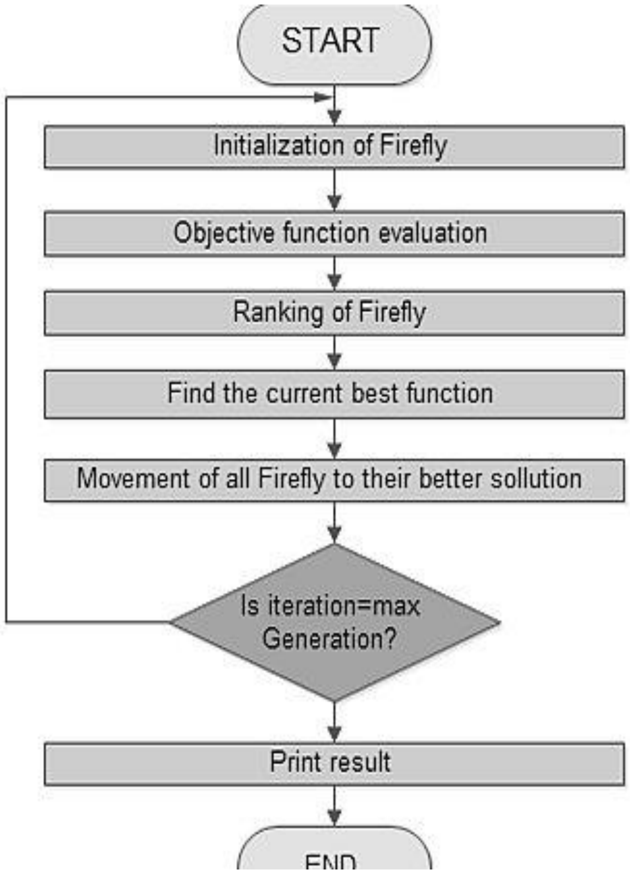


Fig 1.7 Representing the working of Firefly Algorithm

Image Source : *Kumar, Ram, et al. "Quality factor optimisation of spiral inductor using firefly algorithm and its application in amplifier." International Journal of Advanced Intelligence Paradigms 11.3-4 (2018): 299-314.*

Mathematical Equations of firefly Algorithm

Assumptions :

- All the firelies are unisex that means any firefly can be attracted to any other brighter firefly regardless of their sex.
- Brightness is determined by calculating the fitness value of objective function.
- Since we know the fact that intensity of light is inversely proportional to the square of the distance.

Therefore Variation of attractiveness β with distance r from source given by

$$\beta(r) = \beta_0 e^{-\gamma * r^2_{hj}} \quad (1.3)$$

Position Update equation in case of firefly is given by

$$X_h^{t+1} = X_h^t + \beta_0 e^{-\gamma * r^2_{hj}} (X_j^t - X_h^t) + \alpha_t \chi_h^t \quad (1.4)$$

Where α_t is the random parameter lies between (0,1)

And r_{hj} is calculated as per Eucledian distance norm .

χ_h^t is the vector of random numbers drawn from Gaussian or any statistical distribution at time t.

1.4.2 Matlab Work of FireFly

Solving the above same question with FA as we did in case of PSO we get :

Maximize $f(x) = x_1^2 - x_1 * x_2 + x_2^2 + 2 * x_1 + 4 * x_2 + 3$ where $-5 \le x_1, x_2 \le 5$

Algorithm of firefly in MATLAB :

Code of Objective function :

Editor - F:\fun.m

fun.m | firefly.m | Untitled5 | Untitled6

```
function out = fun(X)
            x1 = X(:,1);
            x2 = X(:,2);
            out =  x1.^2-x1.*x2+x2.^2+2.*x1+4.*x2+3;
end
```

Code of Firefly Algorithm :

fun.m | firefly.m | Untitled5 | Untitled6

```
clear all
clc
d=2;
n=5;                          % Population size (number of fireflies)
alpha= 0.9;                     % Randomness strength 0--1 (highly random)
beta0= 0.9;                     % Attractiveness constant
gamma=0.9;                    % Absorption coefficient
theta=0.9;                    % Randomness reduction factor theta=10^(-5/tMax)
d=2;                          % Number of dimensions
iter_max=7;                      % Maximum number of iterations
Lb=[-5 -5];        % Lower bounds/limits
Ub=[5 5];         % Upper bounds/limits
for i=1:n
    for j =1:d
   popln(i,j)= Lb(:,j)+rand.*(Ub(:,j)-Lb(:,j))        % Randomization
    end
end

 fx(i)=fun(popln(i,:));

alpha=alpha*theta;      % Reduce alpha by a factor theta
 scale=abs(Ub-Lb);       % Scale of the optimization problem
for iter = 1:iter_max
for i=1:n
    for j=1:n
      % Evaluate the objective values of current solutions
      fx(i)=fun(popln(i,:));              % Call the objective
      % Update moves
      if fx(i)>fx(j),             % Brighter/more attractive
          popln(i,:)=popln(i,:);
      elseif fx(i)<fx(j)
```

```
          Xi= popln(i,:);
          Xj= popln(j,:);
          r=sqrt(sum((Xi - Xj).^2));
           beta=beta0*exp(-gamma*r.^2);    % Attractiveness
         steps=alpha.*(rand(1,d)-0.5).*scale;
          Xnew=Xi+beta*(Xj-Xi)+steps;
          %% Checking Bounds
          for k=1:size(Xnew,2)
              if Xnew(k)>Ub(k)
                  Xnew(k)=Ub(k);
              elseif Xnew(k)<Lb(k)
                  Xnew(k)=Lb(k);
      end
    end
% greedy based approach
fnew=fun(Xnew);
if fnew >fx(i)
    fx(i)=fnew;
    popln(i,:)=Xnew;
end
      end
    end
end
% Memorizing the solution
[optimumval,opd]=max(fx(i));
Bestfnew(iter)=optimumval;
BestX(iter,:)=popln(opd,:);
disp(['Iteration' num2str(iter)...
    ': Best Cost = ' num2str(Bestfnew(iter))]);
plot(Bestfnew,'LineWidth',2)
end
```

RESULTS OF FIREFLY ALGORITHM :

```
Iteration1: Best Cost = 25.4845
Iteration2: Best Cost = 41.9805
Iteration3: Best Cost = 41.9805
Iteration4: Best Cost = 58
Iteration5: Best Cost = 58
Iteration6: Best Cost = 58
Iteration7: Best Cost = 58
>>
```

Fig 1.8 Showing values in each iteration of Firefly Algo

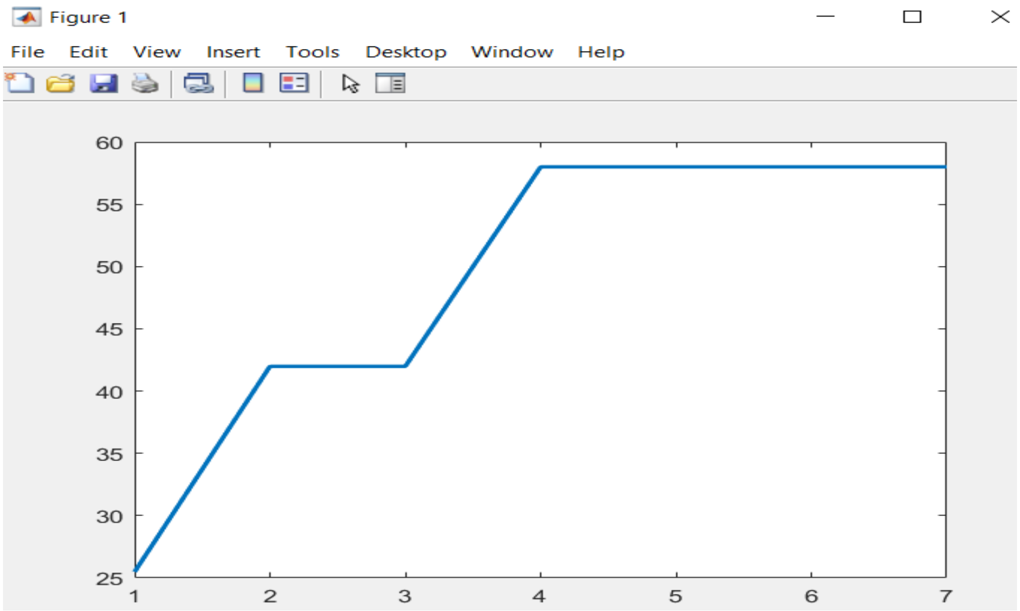


Fig .1.9 Showing convergence graph of Firefly where on x axis it number of iteration And on Y axis representing fitness function value coreeesponding.

REMARK : Since size and dimension of our problem is very small just for simplification and Illustration purpose but if we increase the size very large then Firefly algorithm will better convergence rate can be visible clearly as compare to PSO.

CHAPTER -2
MAYFLY ALGORITHM

2.1 MAYFLY ALGORITHM [4]

Mayfly Algorithm is a very novel algorithm . It is proposed by Zervoudakis and Tsafarakis in May 2020[4].

Since Mayfly Algorithm is inspired by the mating and the levy flight behaviour of Mayflies. But its Mathematical model is coming straight from the GA , PSO and FA. So it becomes very essential to -first understand the these three GA, PSO and FA to get the better understanding of MA. So that's the -reason why Chapter -1 Introduction was entirely dedicated for these algorithms . As these algorithms are serving as a building block for Mayfly Algorithm.

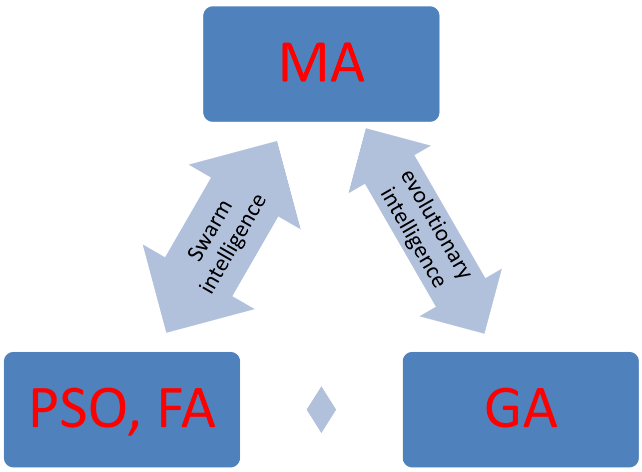


Fig 2.1 Reprsenting the components of Mayfly Algorithm

2.1.1 WORKING OF MAYFLY ALGORITHM

Position Equation of Male mayflies

$$x_l^{t+1} = x_l^t + v_l^{t+1} \qquad (2.1)$$

Here x_l^t is the present position of l^{th} mayfly at iteration t ; the position is modified with the help of introduction of a velocity v_l^{t+1} to the just previous position.

Velocity Equation of Male mayflies

$$v_{lj}^{t+1} = g * v_{lj}^t + c_1 e^{-br_p^2}(pbest_{lj} - x_{hj}^t) + c_2 e^{-br_g^2}(gbest_j - x_{lj}^t) \quad - \qquad (2.2)$$

Symbols can be interpreted as :

- v_{lj}^t is the velocity of l^{th} mayfly in dimension j =1, 2, …..n at time step t. x_{lj}^t is the position of l^{th} mayfly in dimension j at time step t.
- c_1 and c_2 are attraction constants for scaling the influence of cognitive and social component respectively.
- $pbest_l$ is the best position of mayfly l^{th} had ever visited.
- b is visibility coefficient.

- r_p is the Cartesian distance between x_l and $pbest_l$.
- Whereas r_g is the distance between x_l and $gbest$.
- Then the calculation of distances is done by Eqn (2.3)

$$\| x_l - X_l \| = \sqrt{\sum_{h=1}^{n} (x_{lj} - X_{lj})^2} \qquad (2.3)$$

(Where X_l corresponds to $pbest_l$ or gbest)

Movement of Female Mayflies :

$$y_l^{t+1} = y_l^t + v_l^{t+1} \tag{2.4}$$

Velocity of Female Mayflies :

$$v_{lj}^{t+1} = \begin{cases} g * v_{lj}^t + c_2 e^{-br_{mf}^2}(x_{lj}^t - y_{lj}^t) & \text{if} \quad f(y_l) > f(x_l) \\ g * v_{lj}^t + rl * b' & \text{if} \quad f(y_l) \le f(x_l) \end{cases} \tag{2.5}$$

Symbols can be interpreted as :

- Here rl is the random walk coefficient
- b' is random value in the range [-1, 1]
- r_{mf} is the Cartesian distance between male and female.

Mating Process : With the help of crossover operation children can be generated as :

$offspring_C$ = Q*male+(1-Q)*female (2.6)

$offspring_D$ = Q*female+(1-Q)*male

where Q is the random value within a specific range.

Male case in multi-objective optimization:.
Remains same as in the case of a single objective.

Female case in multi-objective optimization:

$$v_{lj}^{t+1} = \begin{cases} g * v_{lj}^t + c_2 e^{-br_{mf}^2}(x_{lj}^t - y_{lj}^t) & \text{, if male leads female} \\ g * v_{lj}^t + rl * b' & \text{, Otherwise} \end{cases} \tag{2.7}$$

FLOW CHART REPRESENTING OF MAYFLY ALGO

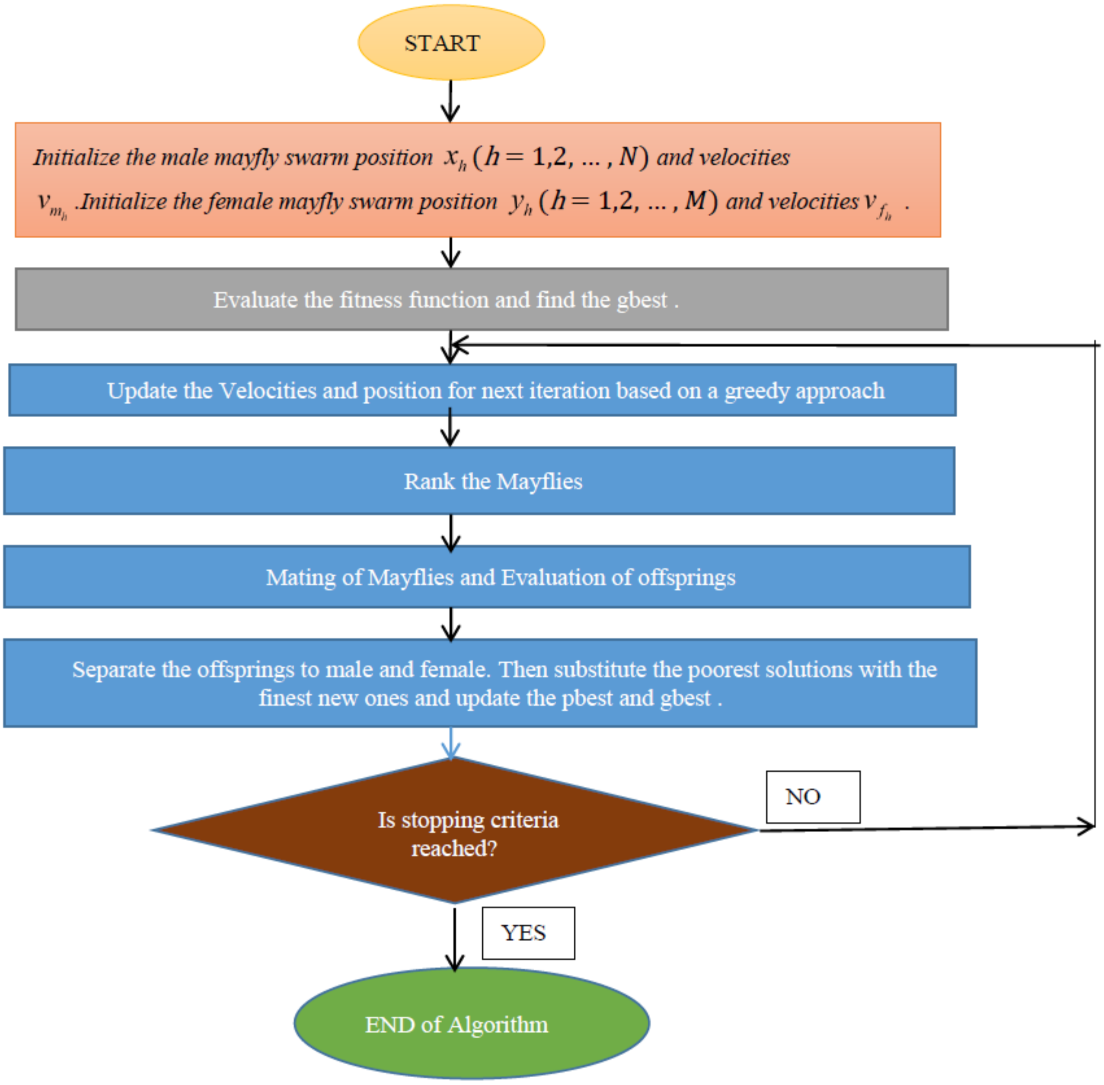


Fig.2.2 Explaining the working of MA

2.1.2 Recent developments in MA

1.Hybrid MA-Harmony Search(HS) **[5]**

- Combines MA and HS
- uses S shape transfer function to change the continuous objective into binary one.
- Can be used in Feature selection in AI and ML.

2. MA-OBL **[6]**

- Combines the MA with opposition-based rules.
- Better convergence rate with both multimodal and unimodal benchmark function as compared to MA.

3.Negative MA **[7]**

- Considers the worst position of a swarm and tries to implement the Mayfly in a negative approach.
- Better simulation result only in multimodal and non-symmetric benchmark functions.

4.Improved MA [8]

- Velocity updation of original MA.
- Useful for both uni and multimodal objective functions.

5. MA-Chebyshev map [9]

- Based on the idea to replace random coefficients with chaotic maps. Here, Chebyshev maps are used.
- Useful for unimodal functions but with less efficacy as compared to MA.

6. Regrouping MA [10]

- Based on the regrouping of a swarm of mayflies .
- Useful to avoid stagnation during the iterations of MA.

7. Multi-Start MA [11]

- Based on idea to incorporate the Multi start initialization of may-flies .
- Solves the problem of stucking to local optima to some extent.

8. Heterogeneous MA [12]

- Multiple ways to update their position in this heterogeneous type of MA .
- Increased the efficiency of the original algorithm.

CHAPTER-3
COMPARISON AND RESULTS

3.1 BENCHMARK FUNCTIONS

In order to check the consistency, robustness and stability of non gradient optimization algorithms . It becomes essential thing that they qualify their properties for certain set of universally acclaimed state-of-the-art benchmark functions.

Benchmark functions are available for Both kinds of objective optimization problem. Be it Single objective or Multiobjective.

Further benchmark functions are classified into basis of number of modality they posses. If benchmark function has single point of optimality then that benchmark function is Unimodal otherwise for multiple optimal points it falls under the category of Multimodal Benchmark functions.

3.2 Matlab Work of Benchmark functions

Unimodal Functions

- Sphere function :

$$F_1(x) = \sum_{l=1}^{\beta} x_l^2 \qquad \textbf{(3.1)}$$

Where
$-10 \le x_l \le 10$

Matlab code of sphere function

```
x = linspace(-10,10,100);
surf(sphereFN(x))
```

Surface plot of sphere function

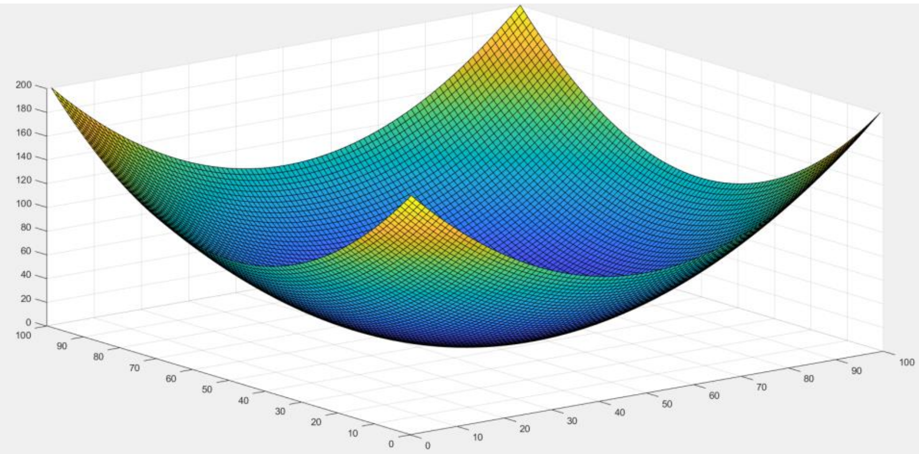


Fig3.1 Showing surface plot of sphere benchmark function

Contour Plot :

```
x = linspace(-10,10,100);
contour(sphereFN(x))
```

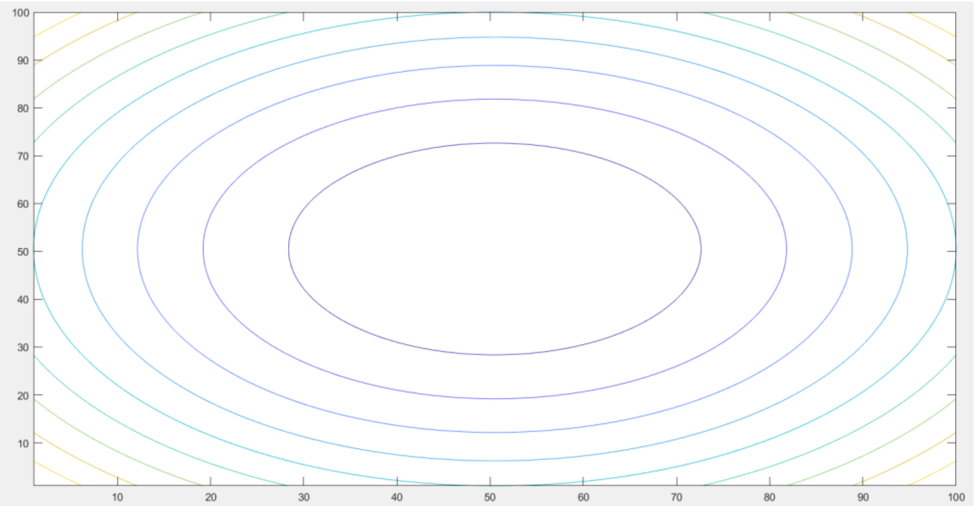


Fig.3.2 Showing the contour plot of sphere function

- RosenBrock Function :

$$F_2(x) = \sum_{l=1}^{\beta-1} [100(x_{l+1} - x_l^2)^2 + (1 - x_l)^2] \tag{3.2}$$

Matlab Code of Rosenbrock Valley function:

```
Editor - C:\Users\Perfect\Desktop\Untitled6.m
Untitled6.m
[X,Y]=meshgrid(-2:0.1:2);
Z=100*(Y-X.^2).^2+(ones(size(X))-X).^2;
surf(X,Y,Z)
```

Surface Plot of Rosenbrock Valley function :

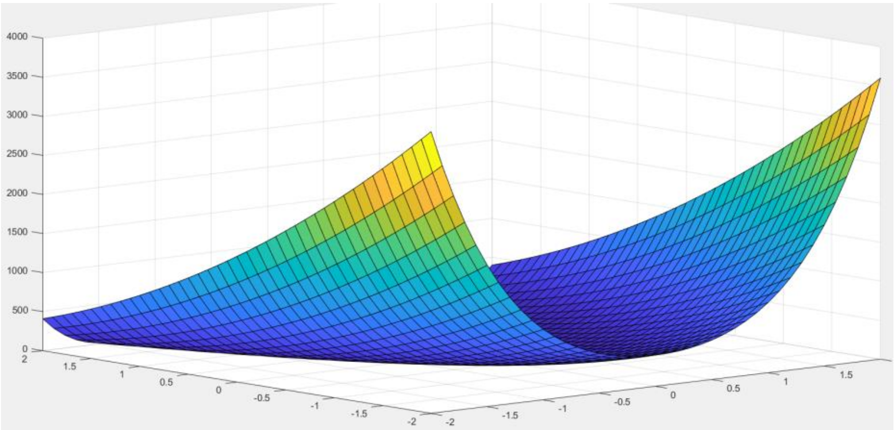


Fig 3.3 Showing surface plot of rosenbrock function

Contour Plot of Rosenbrock Function

```
Editor - C:\Users\Perfect\Desktop\contourrosenbrock.m
Untitled6.m   contourrosenbrock.m
[X,Y]=meshgrid(-2:0.1:2);
Z=100*(Y-X.^2).^2+(ones(size(X))-X).^2;
contour(X,Y,Z)
```

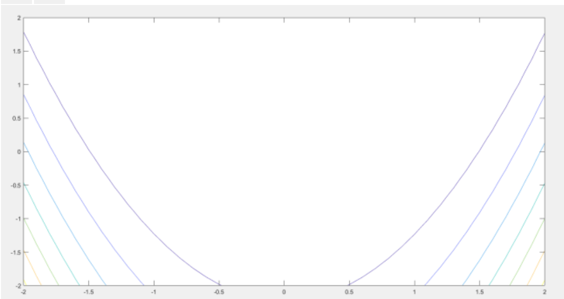


Fig.3.4 Representing the Contour curves of Rosenbrock function

Multimodal

Rastringin Function :

$$F_3(x) = 10 + \sum_{l=1}^{\beta}[x_l^2 - 10 * \cos(2\pi x_l)] \qquad \textbf{(3.3)}$$

Matlab code of rastringin

```

[X,Y]=meshgrid(-2:0.1:2);
Z=10 +(square(X) - 10*cos(2*pi*X));
surf(X,Y,Z)
```

Surface Plot of rastringin

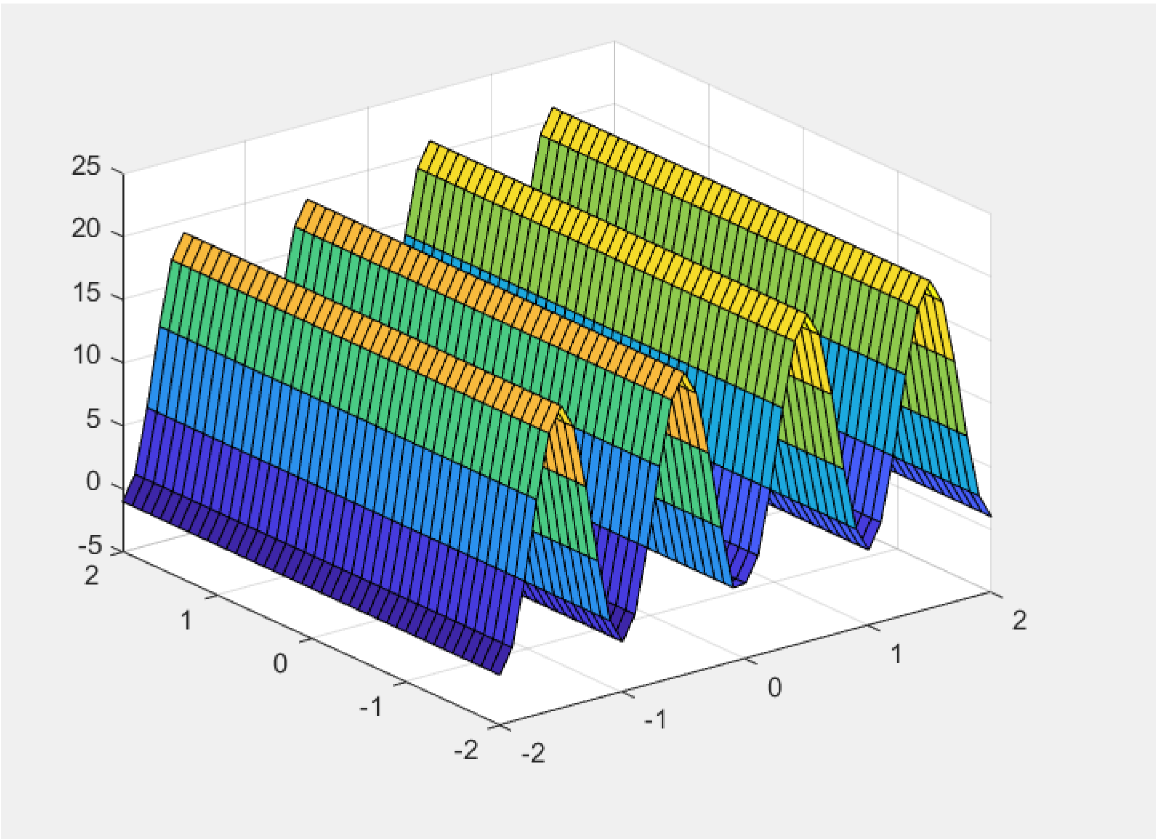


Fig.3.5 Showing Surface Plot of Rastringin function

3.3 COMPARISON OF MA with PSO, GA, and FA .

Table.3.1 MA with PSO and GA. Below we consider average run.[4]

Function ID	GA	PSO	MA
F1	1.73e-02	1.63e-07	1.17e-07
F2	1.82e+02	6.33e+01	6.77e+01
F3	2.83e+01	8.18e+01	1.19e+01

Table.3.2 MA with FA on rastringin[4]

Statistics	Firefly (FA)	Mayfly (MA)
Best run	1.34e+02	5.96e+00
Avg run	1.78e+02	1.19e+01
Worst run	2.48e+02	2.18e+01

OBSERVATIONS : which can be inferred from the above tables

- MA dominates firefly in all three scenarios of best , worst and mean.
- MA dominates in terms of superiority in case of PSO , GA on sphere and rastringin.
- But PSO showed slightly better average convergence rate in case of rosenbrock.

3.4 CONVERGENCE GRAPHS OF MAYFLY

Matlab work of Mayfly

MayflyAlgorithm.m

```
clc; clear; close all;
%% Problem Definition
% Objective Function
ANSWER=listdlg('PromptString','Choose Objective Function','SelectionMode','single', 'ListString', {'1. Sphere', '2. Rastrigin'});
if eq(ANSWER,1); ObjectiveFunction=@(x) Sphere(x); funcname='Sphere';
elseif eq(ANSWER,2); ObjectiveFunction=@(x) Rastrigin(x); funcname='Rastrigin';
else; disp('Terminated'); return
end
ProblemSize=[1 50];         % Decision Variables Size
LowerBound=-10;             % Decision Variables Lower Bound
UpperBound= 10;             % Decision Variables Upper Bound
%% Mayfly Parameters
methname='Mayfly Algorithm';
MaxIt=200;                 % Maximum Number of Iterations
nPop=20; nPopf=20;          % Population Size (males and females)
g=0.8;                      % Inertia Weight
gdamp=1;                    % Inertia Weight Damping Ratio
a1=1.0;                     % Personal Learning Coefficient
a2=1.5; a3=1.5;             % Global Learning Coefficient
beta=2;                     % Distance sight Coefficient
dance=5;                    % Nuptial Dance
fl=1;                       % Random flight
dance_damp=0.8;             % Damping Ratio
fl_damp=0.99;
% Mating Parameters
nc=20;                      % Number of Offsprings (also Parnets)
nm=round(0.05*nPop);        % Number of Mutants
mu=0.01;                    % Mutation Rate
% Velocity Limits
VelMax=0.1*(UpperBound-LowerBound); VelMin=-VelMax;
%% Initialization
empty_mayfly.Position=[];
empty_mayfly.Cost=[];
empty_mayfly.Velocity=[];
empty_mayfly.Best.Position=[];
empty_mayfly.Best.Cost=[];
Mayfly=repmat(empty_mayfly,nPop,1);    % Males
Mayflyf=repmat(empty_mayfly,nPopf,1); % Females
GlobalBest.Cost=inf;
funccount=0;
for i=1:nPop
    % Initialize Position of Males
    Mayfly(i).Position=unifrnd(LowerBound,UpperBound,ProblemSize);
    % Initialize Velocity
    Mayfly(i).Velocity=zeros(ProblemSize);
    % Evaluation
    Mayfly(i).Cost=ObjectiveFunction(Mayfly(i).Position);
    % Update Personal Best
    Mayfly(i).Best.Position=Mayfly(i).Position;
    Mayfly(i).Best.Cost=Mayfly(i).Cost;
    funccount=funccount+1;
    % Update Global Best
    if Mayfly(i).Best.Cost<GlobalBest.Cost
        GlobalBest=Mayfly(i).Best;
    end
```

```
    end
for i=1:nPopf
    % Initialize Position of Females
    Mayflyf(i).Position=unifrnd(LowerBound,UpperBound,ProblemSize);
    Mayflyf(i).Velocity=zeros(ProblemSize);
    Mayflyf(i).Cost=ObjectiveFunction(Mayflyf(i).Position);
    funccount=funccount+1;
    % Update Global Best (Uncomment if you use the PGB-IMA version)
    %if Mayflyf(i).Best.Cost<GlobalBest.Cost
    %    GlobalBest=Mayflyf(i).Best;
    %end
end
BestSolution=zeros(MaxIt,1);
%% Mayfly Main Loop
for it=1:MaxIt
    for i=1:nPopf
        % Update Females
        e=unifrnd(-1,+1,ProblemSize);
        rmf=(Mayfly(i).Position-Mayflyf(i).Position);
        if Mayflyf(i).Cost>Mayfly(i).Cost
            Mayflyf(i).Velocity = g*Mayflyf(i).Velocity ...
                +a3*exp(-beta.*rmf.^2).*(Mayfly(i).Position-Mayflyf(i).Position);
        else
            Mayflyf(i).Velocity = g*Mayflyf(i).Velocity+fl*(e);
        end
        % Apply Velocity Limits
        Mayflyf(i).Velocity = max(Mayflyf(i).Velocity,VelMin);
        Mayflyf(i).Velocity = min(Mayflyf(i).Velocity,VelMax);
        % Update Position
        Mayflyf(i).Position = Mayflyf(i).Position + Mayflyf(i).Velocity;
        % Velocity Mirror Effect
        %IsOutside=(Mayflyf(i).Position<LowerBound | Mayflyf(i).Position>UpperBound);
        %Mayflyf(i).Velocity(IsOutside)=-Mayflyf(i).Velocity(IsOutside);
        % Position Limits
        Mayflyf(i).Position = max(Mayflyf(i).Position,LowerBound);
        Mayflyf(i).Position = min(Mayflyf(i).Position,UpperBound);
        % Evaluation
        Mayflyf(i).Cost = ObjectiveFunction(Mayflyf(i).Position);
        funccount=funccount+1;
        % Update Global Best (Uncomment if you use the PGB-IMA version)
        %if Mayflyf(i).Best.Cost<GlobalBest.Cost
        %    GlobalBest=Mayflyf(i).Best;
        %end
    end
    for i=1:nPop
        % Update Males
        rpbest=(Mayfly(i).Best.Position-Mayfly(i).Position);
        rgbest=(GlobalBest.Position-Mayfly(i).Position);
        e=unifrnd(-1,+1,ProblemSize);
        % Update Velocity
        if Mayfly(i).Cost>GlobalBest.Cost
            Mayfly(i).Velocity = g*Mayfly(i).Velocity ...
                +a1*exp(-beta.*rpbest.^2).*(Mayfly(i).Best.Position-Mayfly(i).Position) .
                +a2*exp(-beta.*rgbest.^2).*(GlobalBest.Position-Mayfly(i).Position);
```

```
        else
            Mayfly(i).Velocity = g*Mayfly(i).Velocity+dance*(e);
        end
        % Apply Velocity Limits
        Mayfly(i).Velocity = max(Mayfly(i).Velocity,VelMin);
        Mayfly(i).Velocity = min(Mayfly(i).Velocity,VelMax);
        % Update Position
        Mayfly(i).Position = Mayfly(i).Position + Mayfly(i).Velocity;
        % Velocity Mirror Effect
        %IsOutside=(Mayfly(i).Position<LowerBound | Mayfly(i).Position>UpperBound);
        %Mayfly(i).Velocity(IsOutside)=-Mayfly(i).Velocity(IsOutside);
        % Position Limits
        Mayfly(i).Position = max(Mayfly(i).Position,LowerBound);
        Mayfly(i).Position = min(Mayfly(i).Position,UpperBound);
        % Evaluation
        Mayfly(i).Cost = ObjectiveFunction(Mayfly(i).Position);
        funccount=funccount+1;
        % Update Personal Best
        if Mayfly(i).Cost<Mayfly(i).Best.Cost
            Mayfly(i).Best.Position=Mayfly(i).Position;
            Mayfly(i).Best.Cost=Mayfly(i).Cost;
            % Update Global Best
            if Mayfly(i).Best.Cost<GlobalBest.Cost
                GlobalBest=Mayfly(i).Best;
            end
        end
    end
    [~, SortMayflies]=sort([Mayfly.Cost]);
    Mayfly=Mayfly(SortMayflies);
    [~, SortMayflies]=sort([Mayflyf.Cost]);
    Mayflyf=Mayflyf(SortMayflies);
    % MATE
    MayflyOffspring=repmat(empty_mayfly,nc/2,2);
    for k=1:nc/2
        % Select Parents
        i1=k;
        i2=k;
        p1=Mayfly(i1);
        p2=Mayflyf(i2);
        % Apply Crossover
        [MayflyOffspring(k,1).Position, MayflyOffspring(k,2).Position]=Crossover(p1.Position,p2.Position,LowerBound,UpperBound);
        % Evaluate Offsprings
        MayflyOffspring(k,1).Cost=ObjectiveFunction(MayflyOffspring(k,1).Position);
        if MayflyOffspring(k,1).Cost<GlobalBest.Cost
            GlobalBest=MayflyOffspring(k,1);
        end
        funccount=funccount+1;
        MayflyOffspring(k,2).Cost=ObjectiveFunction(MayflyOffspring(k,2).Position);
        if MayflyOffspring(k,2).Cost<GlobalBest.Cost
            GlobalBest=MayflyOffspring(k,2);
        end
        funccount=funccount+1;
        MayflyOffspring(k,1).Best.Position = MayflyOffspring(k,1).Position;
        MayflyOffspring(k,1).Best.Cost = MayflyOffspring(k,1).Cost;
        MayflyOffspring(k,1).Velocity= zeros(ProblemSize);
        MayflyOffspring(k,2).Best.Position = MayflyOffspring(k,2).Position;
        MayflyOffspring(k,2).Best.Cost = MayflyOffspring(k,2).Cost;
        MayflyOffspring(k,2).Velocity= zeros(ProblemSize);
    end
    MayflyOffspring=MayflyOffspring(:);
    % Mutation
    MutMayflies=repmat(empty_mayfly,nm,1);
    for k=1:nm
        % Select Parent
        i=randi([1 nPop]);
        p=MayflyOffspring(i);
        %p=Mayfly(i);
        MutMayflies(k).Position=Mutate(p.Position,mu,LowerBound,UpperBound);
        % Evaluate Mutant
        MutMayflies(k).Cost=ObjectiveFunction(MutMayflies(k).Position);
        if MutMayflies(k).Cost<GlobalBest.Cost
            GlobalBest=MutMayflies(k);
        end
        MutMayflies(k).Best.Position = MutMayflies(k).Position;
        MutMayflies(k).Best.Cost = MutMayflies(k).Cost;
        MutMayflies(k).Velocity= zeros(ProblemSize);
    end
    % Create Merged Population
    MayflyOffspring=[MayflyOffspring
        MutMayflies]; %#ok
    split=round((size(MayflyOffspring,1))/2);
```

```
        newmayflies=MayflyOffspring(1:split);
        Mayfly=[Mayfly
            newmayflies]; %#ok
        newmayflies=MayflyOffspring(split+1:size(MayflyOffspring,1));
        Mayflyf=[Mayflyf
            newmayflies]; %#ok
        [~, SortMayflies]=sort([Mayfly.Cost]);
        Mayfly=Mayfly(SortMayflies);
        Mayfly=Mayfly(1:nPop); % Keep best males
        [~, SortMayflies]=sort([Mayflyf.Cost]);
        Mayflyf=Mayflyf(SortMayflies);
        Mayflyf=Mayflyf(1:nPopf); % Keep best females
        BestSolution(it)=GlobalBest.Cost;
        disp([methname ' on the ' funcname  ' Function: Iteration = ' num2str(it)  ', ' funcname ', Evaluations = '
        g=g*gdamp;
        dance = dance*dance_damp;
        fl = fl*fl_damp;
    end
    %% Results
    figure;
    plot(BestSolution,'LineWidth',2); semilogy(BestSolution,'LineWidth',2);
    xlabel('Iterations'); ylabel('Objective function'); grid on;
    %%
function [off1, off2]=Crossover(x1,x2,LowerBound,UpperBound)
    L=unifrnd(0,1,size(x1));
    off1=L.*x1+(1-L).*x2;
    off2=L.*x2+(1-L).*x1;
    % Position Limits
    off1=max(off1,LowerBound); off1=min(off1,UpperBound);
    off2=max(off2,LowerBound); off2=min(off2,UpperBound);
    end
    %%
function y=Mutate(x,mu,LowerBound,UpperBound)
    nVar=numel(x);
    nmu=ceil(mu*nVar);
    j=randsample(nVar,nmu);
    sigma(1:nVar)=0.1*(UpperBound-LowerBound);
    y=x;
    y(j)=x(j)+sigma(j)*(randn(size(j))');
    y=max(y,LowerBound); y=min(y,UpperBound);
    end
    %%
function z=Sphere(x)
    z=sum(x.^2);
    end
function z=Rastrigin(x)
    n=numel(x);
    A=10;
    z=n*A+sum(x.^2-A*cos(2*pi*x));
    end
```

Results of Mayfly in case of Sphere (unimodal function)

For iterations = 100

```
Iteration = 1, Sphere, Evaluations = 100. Best Cost = 700.455
Iteration = 2, Sphere, Evaluations = 160. Best Cost = 437.4814
Iteration = 3, Sphere, Evaluations = 220. Best Cost = 308.7689
Iteration = 4, Sphere, Evaluations = 280. Best Cost = 212.4862
Iteration = 5, Sphere, Evaluations = 340. Best Cost = 157.2799
Iteration = 6, Sphere, Evaluations = 400. Best Cost = 92.0261
Iteration = 7, Sphere, Evaluations = 460. Best Cost = 76.4375
Iteration = 8, Sphere, Evaluations = 520. Best Cost = 49.4866
Iteration = 9, Sphere, Evaluations = 580. Best Cost = 46.8509
Iteration = 10, Sphere, Evaluations = 640. Best Cost = 41.9728
Iteration = 11, Sphere, Evaluations = 700. Best Cost = 38.188
Iteration = 12, Sphere, Evaluations = 760. Best Cost = 32.7997
Iteration = 13, Sphere, Evaluations = 820. Best Cost = 29.0739
Iteration = 14, Sphere, Evaluations = 880. Best Cost = 23.9193
Iteration = 15, Sphere, Evaluations = 940. Best Cost = 21.4393
Iteration = 16, Sphere, Evaluations = 1000. Best Cost = 18.2108
Iteration = 17, Sphere, Evaluations = 1060. Best Cost = 16.5022
Iteration = 18, Sphere, Evaluations = 1120. Best Cost = 15.7501
Iteration = 19, Sphere, Evaluations = 1180. Best Cost = 14.8442
Iteration = 20, Sphere, Evaluations = 1240. Best Cost = 14.0706
Iteration = 21, Sphere, Evaluations = 1300. Best Cost = 10.5493
Iteration = 22, Sphere, Evaluations = 1360. Best Cost = 10.2046
Iteration = 23, Sphere, Evaluations = 1420. Best Cost = 9.863
Iteration = 24, Sphere, Evaluations = 1480. Best Cost = 8.7606
Iteration = 25, Sphere, Evaluations = 1540. Best Cost = 7.5073
Iteration = 26, Sphere, Evaluations = 1600. Best Cost = 6.8966
Iteration = 27, Sphere, Evaluations = 1660. Best Cost = 6.4739
Iteration = 28, Sphere, Evaluations = 1720. Best Cost = 6.179
Iteration = 29, Sphere, Evaluations = 1780. Best Cost = 5.7568
Iteration = 30, Sphere, Evaluations = 1840. Best Cost = 5.2327
Iteration = 31, Sphere, Evaluations = 1900. Best Cost = 4.7906
Iteration = 32, Sphere, Evaluations = 1960. Best Cost = 4.581
Iteration = 33, Sphere, Evaluations = 2020. Best Cost = 4.1818
Iteration = 34, Sphere, Evaluations = 2080. Best Cost = 3.6519
Iteration = 35, Sphere, Evaluations = 2140. Best Cost = 3.4394
Iteration = 36, Sphere, Evaluations = 2200. Best Cost = 2.9931
Iteration = 37, Sphere, Evaluations = 2260. Best Cost = 2.7998
Iteration = 38, Sphere, Evaluations = 2320. Best Cost = 2.6699
Iteration = 39, Sphere, Evaluations = 2380. Best Cost = 2.5773
Iteration = 40, Sphere, Evaluations = 2440. Best Cost = 2.4714
Iteration = 41, Sphere, Evaluations = 2500. Best Cost = 2.3174
Iteration = 42, Sphere, Evaluations = 2560. Best Cost = 2.2543
Iteration = 43, Sphere, Evaluations = 2620. Best Cost = 2.1514
Iteration = 44, Sphere, Evaluations = 2680. Best Cost = 1.9949
Iteration = 45, Sphere, Evaluations = 2740. Best Cost = 1.8729
Iteration = 46, Sphere, Evaluations = 2800. Best Cost = 1.7355
Iteration = 47, Sphere, Evaluations = 2860. Best Cost = 1.692
Iteration = 48, Sphere, Evaluations = 2920. Best Cost = 1.5993
Iteration = 49, Sphere, Evaluations = 2980. Best Cost = 1.4997
Iteration = 50, Sphere, Evaluations = 3040. Best Cost = 1.4206
Iteration = 51, Sphere, Evaluations = 3100. Best Cost = 1.3936
Iteration = 52, Sphere, Evaluations = 3160. Best Cost = 1.3334
Iteration = 53, Sphere, Evaluations = 3220. Best Cost = 1.2776
Iteration = 54, Sphere, Evaluations = 3280. Best Cost = 1.2541
Iteration = 55, Sphere, Evaluations = 3340. Best Cost = 1.2249
Iteration = 56, Sphere, Evaluations = 3400. Best Cost = 1.2007
Iteration = 57, Sphere, Evaluations = 3460. Best Cost = 1.1714
Iteration = 58, Sphere, Evaluations = 3520. Best Cost = 1.056
Iteration = 59, Sphere, Evaluations = 3580. Best Cost = 0.99098
Iteration = 60, Sphere, Evaluations = 3640. Best Cost = 0.92165
Iteration = 61, Sphere, Evaluations = 3700. Best Cost = 0.88258
Iteration = 62, Sphere, Evaluations = 3760. Best Cost = 0.8249
Iteration = 63, Sphere, Evaluations = 3820. Best Cost = 0.78179
Iteration = 64, Sphere, Evaluations = 3880. Best Cost = 0.77074
Iteration = 65, Sphere, Evaluations = 3940. Best Cost = 0.70776
Iteration = 66, Sphere, Evaluations = 4000. Best Cost = 0.68343
Iteration = 67, Sphere, Evaluations = 4060. Best Cost = 0.64937
Iteration = 68, Sphere, Evaluations = 4120. Best Cost = 0.62462
Iteration = 69, Sphere, Evaluations = 4180. Best Cost = 0.62018
Iteration = 70, Sphere, Evaluations = 4240. Best Cost = 0.61246
Iteration = 71, Sphere, Evaluations = 4300. Best Cost = 0.60661
Iteration = 72, Sphere, Evaluations = 4360. Best Cost = 0.5931
Iteration = 73, Sphere, Evaluations = 4420. Best Cost = 0.58318
Iteration = 74, Sphere, Evaluations = 4480. Best Cost = 0.56818
Iteration = 75, Sphere, Evaluations = 4540. Best Cost = 0.56699
Iteration = 76, Sphere, Evaluations = 4600. Best Cost = 0.54539
Iteration = 77, Sphere, Evaluations = 4660. Best Cost = 0.51133
```

```
Iteration = 78, Sphere, Evaluations = 4720. Best Cost = 0.478
Iteration = 79, Sphere, Evaluations = 4780. Best Cost = 0.46025
Iteration = 80, Sphere, Evaluations = 4840. Best Cost = 0.43501
Iteration = 81, Sphere, Evaluations = 4900. Best Cost = 0.41383
Iteration = 82, Sphere, Evaluations = 4960. Best Cost = 0.40927
Iteration = 83, Sphere, Evaluations = 5020. Best Cost = 0.3991
Iteration = 84, Sphere, Evaluations = 5080. Best Cost = 0.38804
Iteration = 85, Sphere, Evaluations = 5140. Best Cost = 0.38156
Iteration = 86, Sphere, Evaluations = 5200. Best Cost = 0.37944
Iteration = 87, Sphere, Evaluations = 5260. Best Cost = 0.37012
Iteration = 88, Sphere, Evaluations = 5320. Best Cost = 0.36544
Iteration = 89, Sphere, Evaluations = 5380. Best Cost = 0.36328
Iteration = 90, Sphere, Evaluations = 5440. Best Cost = 0.35623
Iteration = 91, Sphere, Evaluations = 5500. Best Cost = 0.35054
Iteration = 92, Sphere, Evaluations = 5560. Best Cost = 0.3455
Iteration = 93, Sphere, Evaluations = 5620. Best Cost = 0.3383
Iteration = 94, Sphere, Evaluations = 5680. Best Cost = 0.33351
Iteration = 95, Sphere, Evaluations = 5740. Best Cost = 0.31718
Iteration = 96, Sphere, Evaluations = 5800. Best Cost = 0.3088
Iteration = 97, Sphere, Evaluations = 5860. Best Cost = 0.30237
Iteration = 98, Sphere, Evaluations = 5920. Best Cost = 0.28933
Iteration = 99, Sphere, Evaluations = 5980. Best Cost = 0.28413
Iteration = 100, Sphere, Evaluations = 6040. Best Cost = 0.27443
```

CONVERGENCE GRAPH :

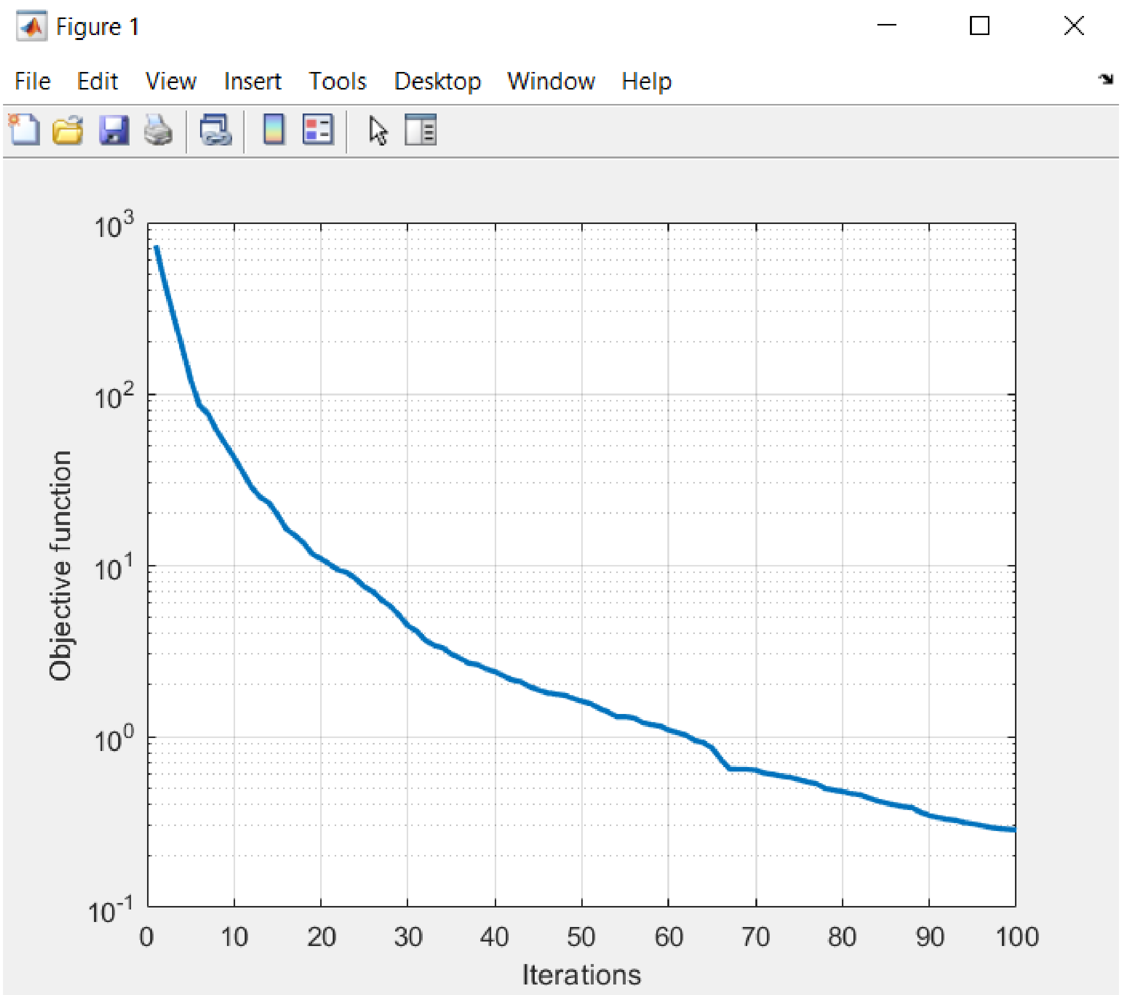


Fig.3.6 Showing convergence of MA on Sphere objective function

Results of MA in case of Rastringin (multimodal)

```
Iteration = 1, Rastrigin, Evaluations = 100. Best Cost = 1216.2473
Iteration = 2, Rastrigin, Evaluations = 160. Best Cost = 940.9884
Iteration = 3, Rastrigin, Evaluations = 220. Best Cost = 713.8422
Iteration = 4, Rastrigin, Evaluations = 280. Best Cost = 637.3465
Iteration = 5, Rastrigin, Evaluations = 340. Best Cost = 521.4825
Iteration = 6, Rastrigin, Evaluations = 400. Best Cost = 521.4825
Iteration = 7, Rastrigin, Evaluations = 460. Best Cost = 521.4825
Iteration = 8, Rastrigin, Evaluations = 520. Best Cost = 467.0568
Iteration = 9, Rastrigin, Evaluations = 580. Best Cost = 464.7963
Iteration = 10, Rastrigin, Evaluations = 640. Best Cost = 464.7963
Iteration = 11, Rastrigin, Evaluations = 700. Best Cost = 436.9671
Iteration = 12, Rastrigin, Evaluations = 760. Best Cost = 431.9787
Iteration = 13, Rastrigin, Evaluations = 820. Best Cost = 409.7375
Iteration = 14, Rastrigin, Evaluations = 880. Best Cost = 409.7375
Iteration = 15, Rastrigin, Evaluations = 940. Best Cost = 335.0713
Iteration = 16, Rastrigin, Evaluations = 1000. Best Cost = 335.0713
Iteration = 17, Rastrigin, Evaluations = 1060. Best Cost = 320.8916
Iteration = 18, Rastrigin, Evaluations = 1120. Best Cost = 300.422
Iteration = 19, Rastrigin, Evaluations = 1180. Best Cost = 287.7589
Iteration = 20, Rastrigin, Evaluations = 1240. Best Cost = 279.4518
Iteration = 21, Rastrigin, Evaluations = 1300. Best Cost = 262.481
Iteration = 22, Rastrigin, Evaluations = 1360. Best Cost = 255.7332
Iteration = 23, Rastrigin, Evaluations = 1420. Best Cost = 255.7332
Iteration = 24, Rastrigin, Evaluations = 1480. Best Cost = 242.4817
Iteration = 25, Rastrigin, Evaluations = 1540. Best Cost = 233.1273
Iteration = 26, Rastrigin, Evaluations = 1600. Best Cost = 222.8694
Iteration = 27, Rastrigin, Evaluations = 1660. Best Cost = 213.0287
Iteration = 28, Rastrigin, Evaluations = 1720. Best Cost = 208.6155
Iteration = 29, Rastrigin, Evaluations = 1780. Best Cost = 190.5666
Iteration = 30, Rastrigin, Evaluations = 1840. Best Cost = 186.2855
Iteration = 31, Rastrigin, Evaluations = 1900. Best Cost = 183.4238
Iteration = 32, Rastrigin, Evaluations = 1960. Best Cost = 181.1235
Iteration = 33, Rastrigin, Evaluations = 2020. Best Cost = 174.4768
Iteration = 34, Rastrigin, Evaluations = 2080. Best Cost = 166.4201
Iteration = 35, Rastrigin, Evaluations = 2140. Best Cost = 155.6282
Iteration = 36, Rastrigin, Evaluations = 2200. Best Cost = 147.8133
Iteration = 37, Rastrigin, Evaluations = 2260. Best Cost = 146.6992
Iteration = 38, Rastrigin, Evaluations = 2320. Best Cost = 143.8117
Iteration = 39, Rastrigin, Evaluations = 2380. Best Cost = 139.7357
Iteration = 40, Rastrigin, Evaluations = 2440. Best Cost = 136.6741
Iteration = 41, Rastrigin, Evaluations = 2500. Best Cost = 134.1876
Iteration = 42, Rastrigin, Evaluations = 2560. Best Cost = 132.1691
Iteration = 43, Rastrigin, Evaluations = 2620. Best Cost = 130.6333
Iteration = 44, Rastrigin, Evaluations = 2680. Best Cost = 129.8382
Iteration = 45, Rastrigin, Evaluations = 2740. Best Cost = 127.381
Iteration = 46, Rastrigin, Evaluations = 2800. Best Cost = 125.7237
Iteration = 47, Rastrigin, Evaluations = 2860. Best Cost = 122.7965
Iteration = 48, Rastrigin, Evaluations = 2920. Best Cost = 122.7965
Iteration = 49, Rastrigin, Evaluations = 2980. Best Cost = 121.4482
Iteration = 50, Rastrigin, Evaluations = 3040. Best Cost = 121.4482
Iteration = 51, Rastrigin, Evaluations = 3100. Best Cost = 117.1392
Iteration = 52, Rastrigin, Evaluations = 3160. Best Cost = 116.545
Iteration = 53, Rastrigin, Evaluations = 3220. Best Cost = 115.7112
Iteration = 54, Rastrigin, Evaluations = 3280. Best Cost = 114.713
Iteration = 55, Rastrigin, Evaluations = 3340. Best Cost = 111.8119
Iteration = 56, Rastrigin, Evaluations = 3400. Best Cost = 109.457
Iteration = 57, Rastrigin, Evaluations = 3460. Best Cost = 106.8769
Iteration = 58, Rastrigin, Evaluations = 3520. Best Cost = 105.0072
Iteration = 59, Rastrigin, Evaluations = 3580. Best Cost = 103.8582
Iteration = 60, Rastrigin, Evaluations = 3640. Best Cost = 100.9999
Iteration = 61, Rastrigin, Evaluations = 3700. Best Cost = 99.605
Iteration = 62, Rastrigin, Evaluations = 3760. Best Cost = 98.1288
Iteration = 63, Rastrigin, Evaluations = 3820. Best Cost = 96.6212
Iteration = 64, Rastrigin, Evaluations = 3880. Best Cost = 95.6302
Iteration = 65, Rastrigin, Evaluations = 3940. Best Cost = 94.3134
Iteration = 66, Rastrigin, Evaluations = 4000. Best Cost = 93.3896
Iteration = 67, Rastrigin, Evaluations = 4060. Best Cost = 92.5628
Iteration = 68, Rastrigin, Evaluations = 4120. Best Cost = 91.8004
Iteration = 69, Rastrigin, Evaluations = 4180. Best Cost = 90.7743
Iteration = 70, Rastrigin, Evaluations = 4240. Best Cost = 89.5841
Iteration = 71, Rastrigin, Evaluations = 4300. Best Cost = 89.175
Iteration = 72, Rastrigin, Evaluations = 4360. Best Cost = 88.8624
Iteration = 73, Rastrigin, Evaluations = 4420. Best Cost = 88.6405
Iteration = 74, Rastrigin, Evaluations = 4480. Best Cost = 87.6678
Iteration = 75, Rastrigin, Evaluations = 4540. Best Cost = 87.0274
Iteration = 76, Rastrigin, Evaluations = 4600. Best Cost = 86.7723
Iteration = 77, Rastrigin, Evaluations = 4660. Best Cost = 86.4553
Iteration = 78, Rastrigin, Evaluations = 4720. Best Cost = 86.1237
Iteration = 79, Rastrigin, Evaluations = 4780. Best Cost = 85.7057
Iteration = 80, Rastrigin, Evaluations = 4840. Best Cost = 85.308
Iteration = 81, Rastrigin, Evaluations = 4900. Best Cost = 82.378
Iteration = 82, Rastrigin, Evaluations = 4960. Best Cost = 82.378
Iteration = 83, Rastrigin, Evaluations = 5020. Best Cost = 82.378
Iteration = 84, Rastrigin, Evaluations = 5080. Best Cost = 82.378
Iteration = 85, Rastrigin, Evaluations = 5140. Best Cost = 82.378
Iteration = 86, Rastrigin, Evaluations = 5200. Best Cost = 82.378
Iteration = 87, Rastrigin, Evaluations = 5260. Best Cost = 82.378
Iteration = 88, Rastrigin, Evaluations = 5320. Best Cost = 82.378
Iteration = 89, Rastrigin, Evaluations = 5380. Best Cost = 82.378
Iteration = 90, Rastrigin, Evaluations = 5440. Best Cost = 82.378
Iteration = 91, Rastrigin, Evaluations = 5500. Best Cost = 82.378
Iteration = 92, Rastrigin, Evaluations = 5560. Best Cost = 82.378
Iteration = 93, Rastrigin, Evaluations = 5620. Best Cost = 82.378
Iteration = 94, Rastrigin, Evaluations = 5680. Best Cost = 82.378
Iteration = 95, Rastrigin, Evaluations = 5740. Best Cost = 82.378
Iteration = 96, Rastrigin, Evaluations = 5800. Best Cost = 82.378
Iteration = 97, Rastrigin, Evaluations = 5860. Best Cost = 82.378
Iteration = 98, Rastrigin, Evaluations = 5920. Best Cost = 82.378
Iteration = 99, Rastrigin, Evaluations = 5980. Best Cost = 82.378
Iteration = 100, Rastrigin, Evaluations = 6040. Best Cost = 82.378
```

Convergence Graph of MA :

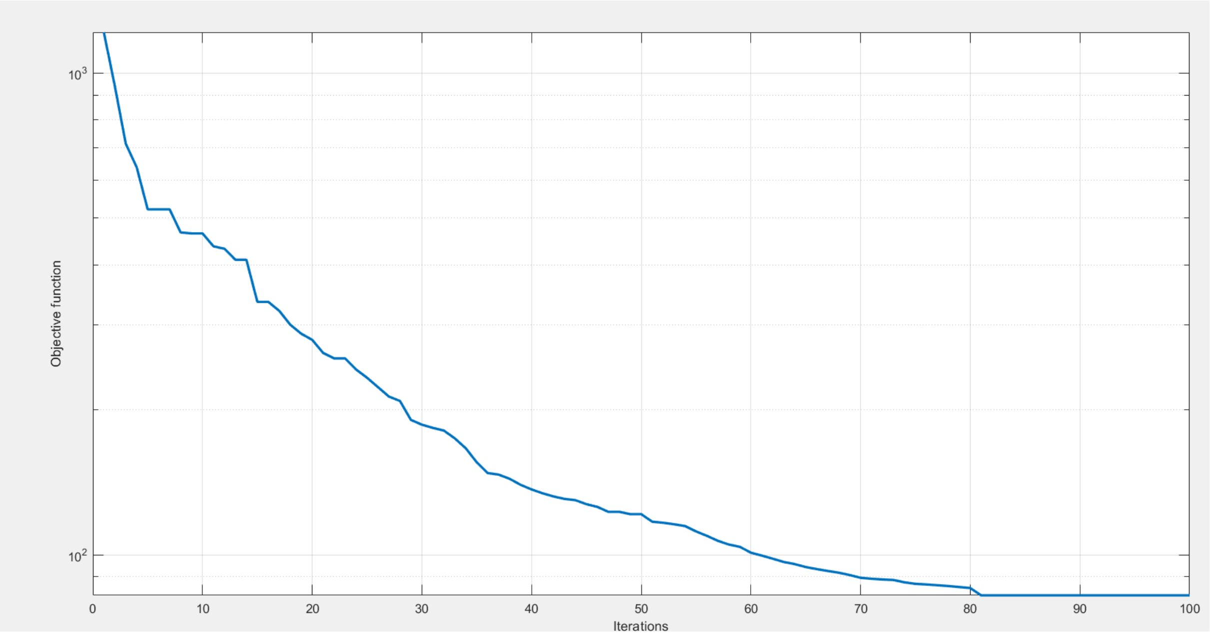


Fig.3.7 Shows convergence graph of MA in Multimodal function

CHAPTER- 4

APPLICATIONS OF MAYFLY ALGORITHM

Table4.1 Applications of MA

DOMAIN	APPLICATIONS
1.Artificial Intelligence	• In neural networks like optimization of weights of neural nets. • Can better work in tuning of parameters as compare to GA. • Hence can be used in the wide field of Robotics , Automation and Soft Computing.
2.Optimization	• In finding optimal solution of NP hard and NP complete problems like Travelling salesman problem,Knapsack problem, other routing and scheduling problems.
3.Multiobjective Optimisation based Engineering problems	• Like in various structural engineering problems,VLSI designing , mechanical and instrumentation problems.
4. Operational Research	• Can easily handle big size constrained and unconstrained optimization problems. • Like of Economics planning, multicriterion decision making, Statistical Quality control , inventory problems etc.
5.Data Mining and Machine learning	• Can be used in ML techniques to minimize the learning error in place of gradient descent or conventional evolutionary algorithms. • Can be used in data mining techniques as it can work better for multimodal functions. Also MA is good alternative to Firefly algorithm in the solving problem of reduction of dimension or size of data . FEATURE SELECTION. • Prediction , Forecasting and Projection Analysis.
6. Fuzzy Inference Systems and Clustering	• Can be used as integrated approach in fuzzy logic based inference systems to increase the efficiency of the device. • In design of Hybrid Neural-Fuzzy systems.

CHAPTER -5
CONCLUSION AND FUTURE PROSPECTS

5.1 ADVANTAGES OF MA

- Bypass the problem of getting caught in local optima due to exploration and exploitation of complete search space.[4]
- Suitable for both single objective and Multi-objective optimization problems .
- Multiobjective MA can better handle the Pareto front as compare to NSGA-II. So it can be widely deployed in those problems having two or more objective functions.
- Even though it seems difficult for other landmark algorithms like GA, PSO, etc. to locate global optima. MA located superior values on various state-of-the-art test functions including both unimodal as well as Multi-modal benchmark functions.
- With the same resources, it has better efficiency, consistency, and convergence rate as compared to previous algorithms (GA, FA, PSO, etc.).
- It can tackle both continuous and discrete optimization problems.

5.2 CHALLENGES OF MA

- Premature Convergence.
- Problem of feature selection.
- Problem of stucking to local optima still prevails.
- Velocity updation may cause stability issues due to change in existing solutions.

5.3 FUTURE PROSPECTS

- After doing a review of the MA and the recent developments that happened in MA. We are in a position to say that MA is a better algorithm as compared to the previous landmark algorithms like PSO, GA, and FA.
- By the virtue of the No Free Lunch Theorem, it becomes essential to dig further so to improve the loopholes of the present algorithm and make it more robust to wider applications.
- So the scope of further improvements and research haven’t been finished yet.
- We can also go for a hybrid of MA with other landmark group behavior algorithms like ABC (Artificial Bee Colony) [14], Social Group Optimization (SGO)[15], etc. By replacing the demerits of MA on certain parameters with merits of the latter one.

- Even one can also go into Quantum computing. For that, they can replace the PSO-based position and velocity equations of MA with Quantum-PSO [16] equations to bring more robustness, stability, and better convergence.

- Again for future perspectives Multiobjective MA can be used to solve various types of multiobjective optimization-based engineering and real-life problems[13] as this method is more robust than NSGA-II.

- Lastly, the dynamic alteration of parameters involved in the velocity and position equations with the help of fuzzy reasoning can increase the efficiency of the original algorithm.

REFERENCES

[1] Goldberg, David E., and John Henry Holland. "Genetic algorithms and machine learning." (1988).

[2] Kennedy, James, and Russell Eberhart. "Particle swarm optimization." Proceedings of ICNN'95-international conference on neural networks. Vol. 4. IEEE, 1995.

[3] Yang, Xin-She. "Firefly algorithm, stochastic test functions and design optimisation." International journal of bio-inspired computation 2.2 (2010): 78-84.

[4] Zervoudakis, Konstantinos, and Stelios Tsafarakis. "A mayfly optimization algorithm." Computers & Industrial Engineering 145 (2020): 106559.

[5]Bhattacharyya, Trinav, et al. "Mayfly in harmony: A new hybrid meta-heuristic feature selection algorithm." IEEE Access 8 (2020): 195929-195945.

[6] Gao, Zheng-Ming, et al. "The improved mayfly optimization algorithm with opposition based learning rules." Journal of Physics: Conference Series. Vol. 1693. No. 1. IOP Publishing, 2020.

[7] Zhao, Juan, and Zheng-Ming Gao. "The negative mayfly optimization algorithm." Journal of Physics: Conference Series. Vol. 1693. No. 1. IOP Publishing, 2020.

[8] Gao, Zheng-Ming, et al. "The improved mayfly optimization algorithm." Journal of Physics: Conference Series. Vol. 1684. No. 1. IOP Publishing, 2020.

[9] Zhao, Juan, and Zheng-Ming Gao. "The improved mayfly optimization algorithm with Chebyshev map." Journal of Physics: Conference Series. Vol. 1684. No. 1. IOP Publishing, 2020.

[10] Zhao, Juan, and Zheng-Ming Gao. "The regrouping mayfly optimization algorithm." 2020 7th International Forum on Electrical Engineering and Automation (IFEEA). IEEE, 2020.

[11] Zhao, Juan, and Zheng-Ming Gao. "The multi-start mayfly optimization algorithm." 2020 7th International Forum on Electrical Engineering and Automation (IFEEA). IEEE, 2020.

[12] Zheng-Ming, G. A. O., et al. "Heterogeneous mayfly optimization algorithm." 2020 2nd International Conference on Machine Learning, Big Data and Business Intelligence (MLBDBI). IEEE, 2020.

[13]Kalyanmoy Deb, Multi-Objective Optimization using Evolutionary Algorithms. Ist Edn., John Wiley & Sons, Ltd, England (2001).

[14] Karaboga, Dervis, and Bahriye Basturk. "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm." Journal of global optimization 39.3 (2007): 459-471.

[15] Satapathy, Suresh, and Anima Naik. "Social group optimization (SGO): a new population evolutionary optimization technique." Complex & Intelligent Systems 2.3 (2016): 173-203.

[16] Yang, Shuyuan, and Min Wang. "A quantum particle swarm optimization." Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No. 04TH8753). Vol. 1. IEEE, 2004.