MECHANICAL ENGINEERING DEPARTMENT

OPTIMIZATION OF MECHANICAL SYSTEMS

FINAL PROJECT QUESTION I

Student : UFUK MAMİKOĞLU

Number : 150415055

Supervisor : Prof. Dr.Aykut Kentli

INDEX

Introduction

Optimization is everywhere, and is thus an important paradigm itself with a wide range of applications. In almost all applications in engineering and industry, we are always trying to optimize something -- whether to minimize the cost and energy consumption, or to maximize the profit, output, performance and efficiency. In reality, resources, time and money are always limited; consequently, optimization is far more important in practice

The optimal use of available resources of any sort requires a paradigm shift in scientific thinking, this is because most real-world applications have far more complicated factors and parameters to affect how the system behaves.

in this project we tried to do learn how to adapt them to our 3 different questions by using different algorithms with using Black Hole Algorithm

Welded Beam Design
Spring Design
Speed Reducer Design

Black Hole Algorithm

The basic idea of a black hole is simply a region of space that has so much mass
concentrated in it that there is no way for a nearby object to escape its gravitational pull. Anything falling into a black hole, including light, is forever gone from us universe.

Terminology of Black Hole Algorithm

Black Hole: In black hole algorithm, the best candidate among all the candidates at each iteration is selected as a black hole.

Stars: All the other candidates form the normal stars. The creation of the black hole is not random and it is one of the real candidates of the population.

Movement: Then, all the candidates are moved towards the black hole based on their current location and a random number.

1. Black hole algorithm (black hole) starts with an initial population of candidate
solutions to an optimization problem and an objective function that is calculated for them.
2. At each iteration of the Black Hole, the best candidate is selected to be the black
hole and the rest form the normal stars. After the initialization process, the black hole starts pulling stars around it.
3. If a star gets too close to the black hole it will be swallowed by the black hole
and is gone forever. In such a case, a new star (candidate solution) is randomly generated and placed in the search space and starts a new search.

Advantage of Black Hole Algorithm

It has a simple structure and it is easy to implement.

It is free from tuning parameter issues like genetic algorithm local search utilizes the schemata(S) theorem of higher order O(S) (compactness) and longer defining length δ(S). In Genetic Algorithm, to improve the fine-tuning capabilities of a genetic algorithm, which is a must for high precision problem over the traditional representation of binary string of chromosomes? It was required a new mutation operator over the traditional mutation operator however, it only uses only local knowledge i.e. it stuck into local minimum optimal value. The Black Hole algorithm converges to global optimum in all the runs while the other heuristic algorithms may get trapped in local optimum solutions like genetic algorithm, Ant colony Optimization algorithm simulated Annealing algorithm.

Calculation of Fitness Value for Black Hole Algorithm

1. Initial Population:

$P(x) = \{x_1^t, x_2^t, x_3^t, \ldots, x_n^t\}$

randomly generated population of candidate solutions (the stars) are placed in the search space of some problem or function.

2. Find the total Fitness of population:

$$f_i = \sum_{i=1}^{pop\_size} eval(p(t) \qquad (1)$$

$$f_{BH} = \sum_{i=1}^{pop\_size} eval(p(t)$$

3. where fi and fBH are the fitness values of black hole and ith star in the initializedpopulation. The population is estimated and the best candidate in the population, which has the best fitness value, fi is selected to be the blackhole and the remaining form the normal stars. The black hole has the capability toabsorb the stars that surround it. After initializing the first black hole and stars, theblack hole starts absorbing the stars around it and all the stars start moving towards the black hole.

CASE PROBLEMS:

1.CASE : WELDED BEAM DESIGN

E01: Welded beam design optimization problem

The problem is to design a welded beam for minimum cost, subject to some constraints [23]. Figure 1 shows the welded beam structure which consists of a beam A and the weld required to hold it to member B. The objective is to find the minimum fabrication cost, considerating four design variables: x_1, x_2, x_3, x_4 and constraints of shear stress τ, bending stress in the beam σ, buckling load on the bar P_c, and end deflection on the beam δ. The optimization model is summarized in the next equation:

Minimize:

$$f(\vec{x}) = 1.10471 x_1^2 x_2 + 0.04811 x_3 x_4 (14.0 + x_2)$$

subject to:

$$g_1(\vec{x}) = \tau(\vec{x}) - 13,600 \le 0$$

$$g_2(\vec{x}) = \sigma(\vec{x}) - 30,000 \le 0$$

$$g_3(\vec{x}) = x_1 - x_4 \le 0$$

$$g_4(\vec{x}) = 0.10471(x_1^2) + 0.04811 x_3 x_4 (14 + x_2) - 5.0 \le 0$$

$$g_5(\vec{x}) = 0.125 - x_1 \le 0$$

$$g_6(\vec{x}) = \delta(\vec{x}) - 0.25 \le 0$$

$$g_7(\vec{x}) = 6,000 - Pc(\vec{x}) \le 0$$

with:

$$\tau(\vec{x}) = \sqrt{(\tau')^2 + (2\tau'\tau'')\frac{x_2}{2R} + (\tau'')^2}$$

$$\tau' = \frac{6,000}{\sqrt{2}x_1x_2}$$

$$\tau'' = \frac{MR}{J}$$

$$M = 6,000\left(14 + \frac{x_2}{2}\right)$$

$$R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2}$$

$$J = 2\left\{x_1x_2\sqrt{2}\left[\frac{x_2^2}{12} + \left(\frac{x_1 + x_3}{2}\right)^2\right]\right\}$$

$$\sigma(\vec{x}) = \frac{504,000}{x_4x_3^2}$$

$$\delta(\vec{x}) = \frac{65,856,000}{(30 \times 10^6)x_4x_3^3}$$

$$Pc(\vec{x}) = \frac{4.013(30 \times 10^6)\sqrt{\frac{x_3^2x_4^6}{36}}}{196}\left(1 - \frac{x_3\sqrt{\frac{30\times10^6}{4(12\times10^6)}}}{28}\right)$$

with $0.1 \le x_1, x_4 \le 2.0$, and $0.1 \le x_2, x_3 \le 10.0$.
Best solution:

$$x^* = (0.205730, 3.470489, 9.036624, 0.205729)$$

where $f(x^*) = 1.724852$.

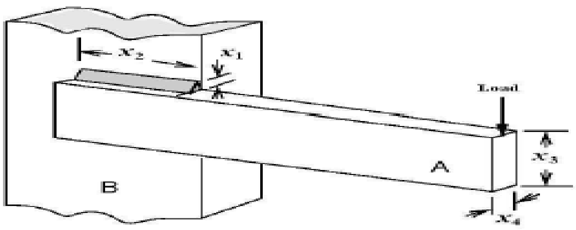


Figure 1: Weldem Beam.

MATLAB CODE ANSWER:

```
ans =

    0.1109    3.0797    9.0830    1.0300
```

Code gives an acceptable answer

2.CASE: SPEED REDUCER

E03: Speed Reducer design optimization problem

The design of the speed reducer [12] shown in Fig. 3, is considered with the face width x_1, module of teeth x_2, number of teeth on pinion x_3, length of the first shaft between bearings x_4, length of the second shaft between bearings x_5, diameter of the first shaft x_6, and diameter of the first shaft x_7 (all variables continuous except x_3 that is integer). The weight of the speed reducer is to be minimized subject to constraints on bending stress of the gear teeth, surface stress, transverse deflections of the shafts and stresses in the shaft. The problem is:

Minimize:

$$\begin{aligned} f(\vec{x}) \;=\; & 0.7854x_1x_2^2(3.3333x_3^2 + 14.9334x_3 - 43.0934) \\ & - 1.508x_1(x_6^2 + x_7^2) + 7.4777(x_6^3 + x_7^3) \\ & + 0.7854(x_4x_6^2 + x_5x_7^2) \end{aligned}$$

subject to:

$$g_1(\vec{x}) = \frac{27}{x_1x_2^2x_3} - 1 \le 0$$

$$g_2(\vec{x}) = \frac{397.5}{x_1x_2^2x_3^2} - 1 \le 0$$

$$g_3(\vec{x}) = \frac{1.93x_4^3}{x_2x_3x_6^4} - 1 \le 0$$

$$g_4(\vec{x}) = \frac{1.93x_5^3}{x_2x_3x_7^4} - 1 \le 0$$

$$g_5(\vec{x}) = \frac{1.0}{110x_6^3}\sqrt{\left(\frac{745.0x_4}{x_2x_3}\right)^2 + 16.9 \times 10^6} - 1 \le 0$$

$$g_6(\vec{x}) = \frac{1.0}{85x_7^3}\sqrt{\left(\frac{745.0x_5}{x_2x_3}\right)^2 + 157.5 \times 10^6} - 1 \le 0$$

$$g_7(\vec{x}) = \frac{x_2x_3}{40} - 1 \le 0$$

$$g_8(\vec{x}) = \frac{5x_2}{x_1} - 1 \le 0$$

$$g_9(\vec{x}) = \frac{x_1}{12x_2} - 1 \le 0$$

$$g_{10}(\vec{x}) = \frac{1.5x_6 + 1.9}{x_4} - 1 \le 0$$

$$g_{11}(\vec{x}) = \frac{1.1x_7 + 1.9}{x_5} - 1 \le 0$$

with $2.6 \le x_1 \le 3.6, 0.7 \le x_2 \le 0.8, 17 \le x_3 \le 28, 7.3 \le x_4 \le 8.3, 7.8 \le x_5 \le 8.3, 2.9 \le x_6 \le 3.9$, and $5.0 \le x_7 \le 5.5$.

Best solution:

$$x^* = (3.500000, 0.7, 17, 7.300000, 7.800000, 3.350214, 5.286683)$$

where $f(x^*) = 2,996.348165$.

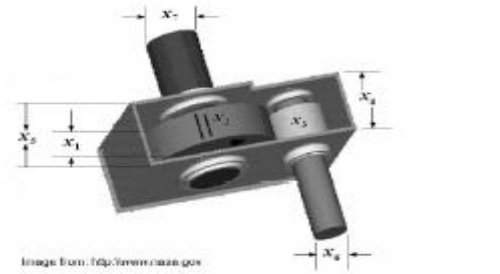


Figure 3: Speed Reducer.

MATLAB ANSWER:

```
ans =

    3.0427    0.7744   17.0683    7.9716    8.0048    3.0434    5.0719
```

Code gives an acceptable answer

3.CASE: SPRING DESIGN

5.1 E04: Tension/compression spring design optimization problem

This problem [2] [3] minimizes the weight of a tension/compression spring (Fig. 4), subject to constraints of minimum deflection, shear stress, surge frequency, and limits on outside diameter and on design variables. There are three design variables: the wire diameter x_1, the mean coil diameter x_2, and the number of active coils x_3. The mathematical formulation of this problem is:

Minimize:

$$f(\vec{x}) = (x_3 + 2)x_2x_1^2$$

subject to:

$$g_1(\vec{x}) = 1 - \frac{x_2^3 x_3}{7,178x_1^4} \leq 0$$

$$g_2(\vec{x}) = \frac{4x_2^2 - x_1x_2}{12,566(x_2x_1^3) - x_1^4} + \frac{1}{5,108x_1^2} - 1 \leq 0$$

$$g_3(\vec{x}) = 1 - \frac{140.45x_1}{x_2^2 x_3} \leq 0$$

$$g_4(\vec{x}) = \frac{x_2 + x_1}{1.5} - 1 \leq 0$$

with $0.05 \leq x_1 \leq 2.0, 0.25 \leq x_2 \leq 1.3$, a $2.0 \leq x_3 \leq 15.0$.

Best solution:

$$x^* = (0.051690, 0.356750, 11.287126)$$

where $f(x^*) = 0.012665$.

MATLAB ANSWER:

```
ans =

    0.1457    0.9660    2.0421
```

Code gives an acceptable answer

SUMMARY AND CONCLUSION

In this project, we learned that no matter how specific the optimization algorithms are, they enable us to achieve very realistic results when we run them at appropriate value ranges and constraints. Being able to design a simple spring with a program written to calculate the gravitational force created by the stars shows us why these programs have been given such importance and have developed so much in the last 20 years.

REFERENCES:

https://www.researchgate.net/publication/281786410_Black_Hole_Algorithm_and_Its_Applications

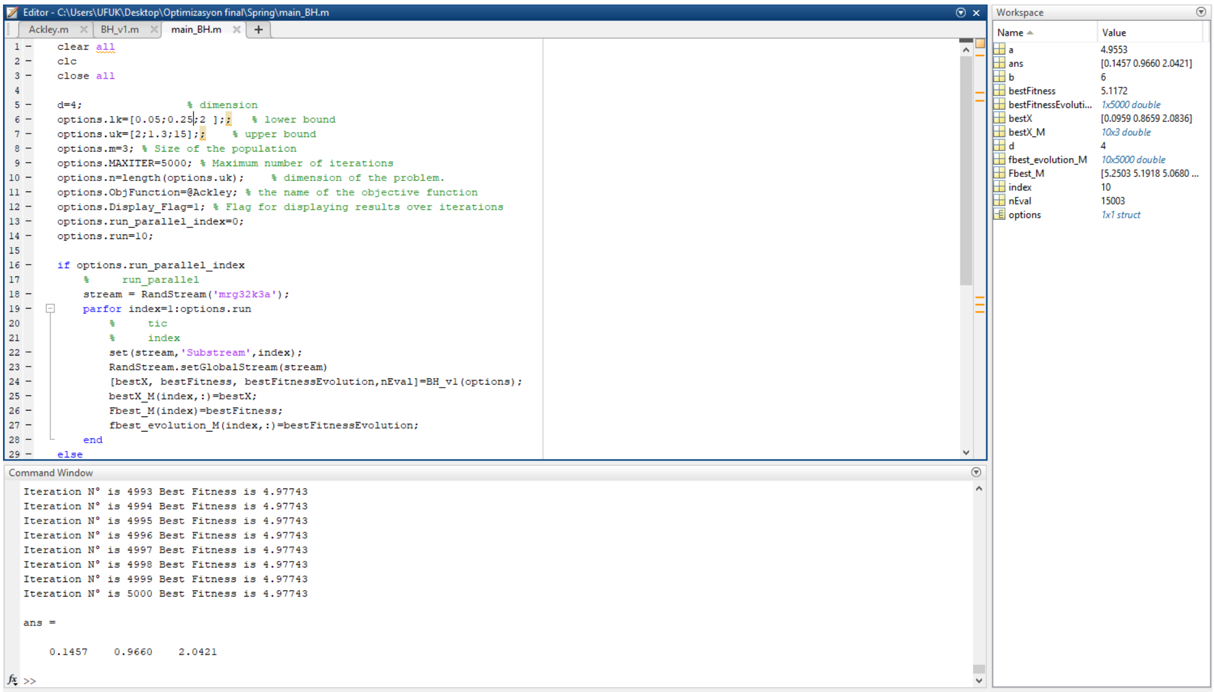
https://www.researchgate.net/publication/225448393_Optimization_Algorithms

https://www.mathworks.com/matlabcentral/fileexchange/72223-black-hole-optimization-algorithm

APENDIX :

MATLAB CODE:

a)SPRING DESIGN



Main_BH.m

```
function [bestX, bestFitness, bestFitnessEvolution,nEval]=BH_v1(options)
%------------------------------------------------------------------------
% Black Hole Algorithm
% Dr Hpussem BOUCHEKARA
% 20/07/2019
%------------------------------------------------------------------------
% 1. Bouchekara, H. R. E. H. (2013). Optimal design of electromagnetic
% devices using a black-Hole-Based optimization technique. IEEE
% Transactions on Magnetics, 49(12). doi:10.1109/TMAG.2013.2277694
```

```
%
% 2. Bouchekara, H. R. E. H. (2014). Optimal power flow using black-hole-based
% optimization approach. Applied Soft Computing, 24, 879–888.
% doi:10.1016/j.asoc.2014.08.056
%
% 3. Smail, M. K., Bouchekara, H. R. E. H., Pichon, L., Boudjefdjouf, H.,
% Amloune, A., & Lacheheb, Z. (2016). Non-destructive diagnosis of wiring
% networks using time domain reflectometry and an improved black hole
% algorithm. Nondestructive Testing and Evaluation.
% doi:10.1080/10589759.2016.1200576
%-----------------------------------------------------------------------
% Initialize a population of stars with random locations in the search space
% Loop
%   For each star, evaluate the objective function
%   Select the best star that has the best fitness value as the black hole
%   Change the location of each star according to Eq. (3)
%   If a star reaches a location witch lower cost than the black hole, exchange their
locations
%   If a star crosses the event horizon of the black hole, replace it with a new star in a
random location in the search space
%   If a termination criterion (a maximum number of iterations or a sufficiently good
fitness) is met, exit the loop
% End loop
%-----------------------------------------------------------------------

ObjFunction=options.ObjFunction; % the name of the objective function
n=options.n;    % dimension of the problem.
uk=options.uk;   % upper bound in the kth dimension.
lk=options.lk;  % lower bound in the kth dimension.
m=options.m; % m: number of sample points
MAXITER=options.MAXITER; % MAXITER: maximum number of iterations
nEval=0;
[x,xBH,iBH,ObjFunctionValue]=Initialize(options);
nEval=nEval+size(x,1);
```

```
for iteration =1:MAXITER
   %     tic
   %     Change the location of each star according to Eq. (3)
   for i = 1 : m
      if i ~= iBH
         landa=rand;
         for k = 1 : n
            if landa<0.5
               x(i,k)=x(i,k) + rand*(xBH(k)- x(i,k));
            else
               x(i,k)=x(i,k) + rand*(xBH(k)- x(i,k));
            end
         end
      end
   end
   %   If a star reaches a location with lower cost than the black
   %   hole, exchange their locations
   ObjFunctionValue=feval(ObjFunction,x);
   nEval=nEval+size(x,1);
   %     [x]=bound(x,lk,uk);
   %     [xBH,iBH]=argmin(x,ObjFunctionValue,options);
   %   If a star crosses the event horizon of the black hole, replace it
   %   with a new star in a random location in the search space
   R=ObjFunctionValue(iBH)/sum(ObjFunctionValue);
   %     R=exp(-n*ObjFunctionValue(iBH)/sum(ObjFunctionValue))
   %     pause
   for i = 1 : m
      Distance(i)=norm(xBH- x(i,:));
   end

[x,ObjFunctionValue]=NewStarGeneration(x,Distance,R,options,iBH,ObjFunctionValu
e);
   [x]=bound(x,lk,uk);
   [xBH,iBH]=argmin(x,ObjFunctionValue,options);
```

```
    %-----------------------------------------------------------------------------
    bestFitnessEvolution(iteration)=ObjFunctionValue(iBH);
    %-----------------------------------------------------------------------------


    if options.Display_Flag==1
        fprintf('Iteration N° is %g Best Fitness is %g\n',iteration,ObjFunctionValue(iBH))
    end

end
bestX=xBH;
bestFitness=ObjFunctionValue(iBH);
end

function [x,xBH,iBH,ObjFunctionValue]=Initialize(options)
ObjFunction=options.ObjFunction; % the name of the objective function.
n=options.n;    % n: dimension of the problem.
uk=options.uk;  % up: upper bound in the kth dimension.
lk=options.lk;  % lp: lower bound in the kth dimension.
m=options.m;    % m: number of sample points

for i = 1 : m
    for k = 1 : n
        landa=rand;
        x(i,k) = lk(k) + landa*(uk(k) - lk(k));
    end
end
% x(end+1,:)=x0;
ObjFunctionValue=feval(ObjFunction,x);
[index1,index2]=sort(ObjFunctionValue);
x=x(index2(1:m),:);
xBH=x(1,:);
iBH=1;
```

```
ObjFunctionValue=ObjFunctionValue(index2(1:m));
end

function [xb,ib,xw,iw]=argmin(x,f,options)
[minf,ib]=min(f);
xb=x(ib,:);
[maxf,iw]=max(f);
xw=x(iw,:);
end



function
[x,ObjFunctionValue]=NewStarGeneration(x,Distance,R,options,iBH,ObjFunctionValu
e)
ObjFunction=options.ObjFunction; % the name of the objective function.
n=options.n;    % n: dimension of the problem.
uk=options.uk;  % up: upper bound in the kth dimension.
lk=options.lk;  % lp: lower bound in the kth dimension.
index=find(Distance<R);
for i=1:length(index)
   if index(i) ~= iBH
      for k = 1 : n
         x(i,k) = lk(k) + rand*(uk(k) - lk(k));
      end
      ObjFunctionValue(i)=feval(ObjFunction,x(i,:));
   end
end
end
function [x]=bound(x,l,u)
for j = 1:size(x,1)
   for k = 1:size(x,2)
      % check upper boundary
      if x(j,k) > u(k),
         x(j,k) = u(k);
```

```
        end
        % check lower boundary
        if x(j,k) < l(k),
            x(j,k) = l(k);
        end
    end
end
end
```

Ackley.m

```
function [F, lb, ub, FGO] = Ackley(x)
% Ackley function
if (nargin==0)
    F=[];
    d=2;                % dimension
    lb=-32*ones(1,d);   % lower bound
    ub=32*ones(1,d);    % upper bound
    FGO=0;              % Global Optimum
else
    n=size(x,2);
    for ix=1:size(x,1)
        x0=x(ix,:);
        F(ix) = -20*exp(-0.2*sqrt(1/n*sum(x0.^2)))-...
                exp(1/n*sum(cos(2*pi*x0)))+20+exp(1);
    end
end
```

BH_V1.m

```
function [bestX, bestFitness, bestFitnessEvolution,nEval]=BH_v1(options)
%-----------------------------------------------------------------------
% Black Hole Algorithm
% Dr Hpussem BOUCHEKARA
% 20/07/2019
%-----------------------------------------------------------------------
% 1. Bouchekara, H. R. E. H. (2013). Optimal design of electromagnetic
% devices using a black-Hole-Based optimization technique. IEEE
% Transactions on Magnetics, 49(12). doi:10.1109/TMAG.2013.2277694
%
% 2. Bouchekara, H. R. E. H. (2014). Optimal power flow using black-hole-based
% optimization approach. Applied Soft Computing, 24, 879–888.
% doi:10.1016/j.asoc.2014.08.056
%
% 3. Smail, M. K., Bouchekara, H. R. E. H., Pichon, L., Boudjefdjouf, H.,
% Amloune, A., & Lacheheb, Z. (2016). Non-destructive diagnosis of wiring
% networks using time domain reflectometry and an improved black hole
% algorithm. Nondestructive Testing and Evaluation.
% doi:10.1080/10589759.2016.1200576
%-----------------------------------------------------------------------
% Initialize a population of stars with random locations in the search space
% Loop
%   For each star, evaluate the objective function
%   Select the best star that has the best fitness value as the black hole
%   Change the location of each star according to Eq. (3)
%   If a star reaches a location witch lower cost than the black hole, exchange their
locations
%   If a star crosses the event horizon of the black hole, replace it with a new star in a
random location in the search space
%   If a termination criterion (a maximum number of iterations or a sufficiently good
fitness) is met, exit the loop
% End loop
%-----------------------------------------------------------------------
```

```
ObjFunction=options.ObjFunction; % the name of the objective function
n=options.n;    % dimension of the problem.
uk=options.uk;   % upper bound in the kth dimension.
lk=options.lk;  % lower bound in the kth dimension.
m=options.m; % m: number of sample points
MAXITER=options.MAXITER; % MAXITER: maximum number of iterations
nEval=0;
[x,xBH,iBH,ObjFunctionValue]=Initialize(options);
nEval=nEval+size(x,1);
for iteration =1:MAXITER
    %     tic
    %     Change the location of each star according to Eq. (3)
    for i = 1 : m
        if i ~= iBH
            landa=rand;
            for k = 1 : n
                if landa<0.5
                    x(i,k)=x(i,k) + rand*(xBH(k)- x(i,k));
                else
                    x(i,k)=x(i,k) + rand*(xBH(k)- x(i,k));
                end
            end
        end
    end
    %   If a star reaches a location with lower cost than the black
    %   hole, exchange their locations
    ObjFunctionValue=feval(ObjFunction,x);
    nEval=nEval+size(x,1);
    %     [x]=bound(x,lk,uk);
    %     [xBH,iBH]=argmin(x,ObjFunctionValue,options);
    %   If a star crosses the event horizon of the black hole, replace it
    %   with a new star in a random location in the search space
    R=ObjFunctionValue(iBH)/sum(ObjFunctionValue);
```

```
    %     R=exp(-n*ObjFunctionValue(iBH)/sum(ObjFunctionValue))
    %     pause
    for i = 1 : m
        Distance(i)=norm(xBH- x(i,:));
    end

[x,ObjFunctionValue]=NewStarGeneration(x,Distance,R,options,iBH,ObjFunctionValu
e);
    [x]=bound(x,lk,uk);
    [xBH,iBH]=argmin(x,ObjFunctionValue,options);

    %---------------------------------------------------------------------------
    bestFitnessEvolution(iteration)=ObjFunctionValue(iBH);
    %---------------------------------------------------------------------------



    if options.Display_Flag==1
        fprintf('Iteration N° is %g Best Fitness is %g\n',iteration,ObjFunctionValue(iBH))
    end

end
bestX=xBH;
bestFitness=ObjFunctionValue(iBH);
end

function [x,xBH,iBH,ObjFunctionValue]=Initialize(options)
ObjFunction=options.ObjFunction; % the name of the objective function.
n=options.n;    % n: dimension of the problem.
uk=options.uk;  % up: upper bound in the kth dimension.
lk=options.lk;  % lp: lower bound in the kth dimension.
m=options.m;    % m: number of sample points

for i = 1 : m
    for k = 1 : n
```

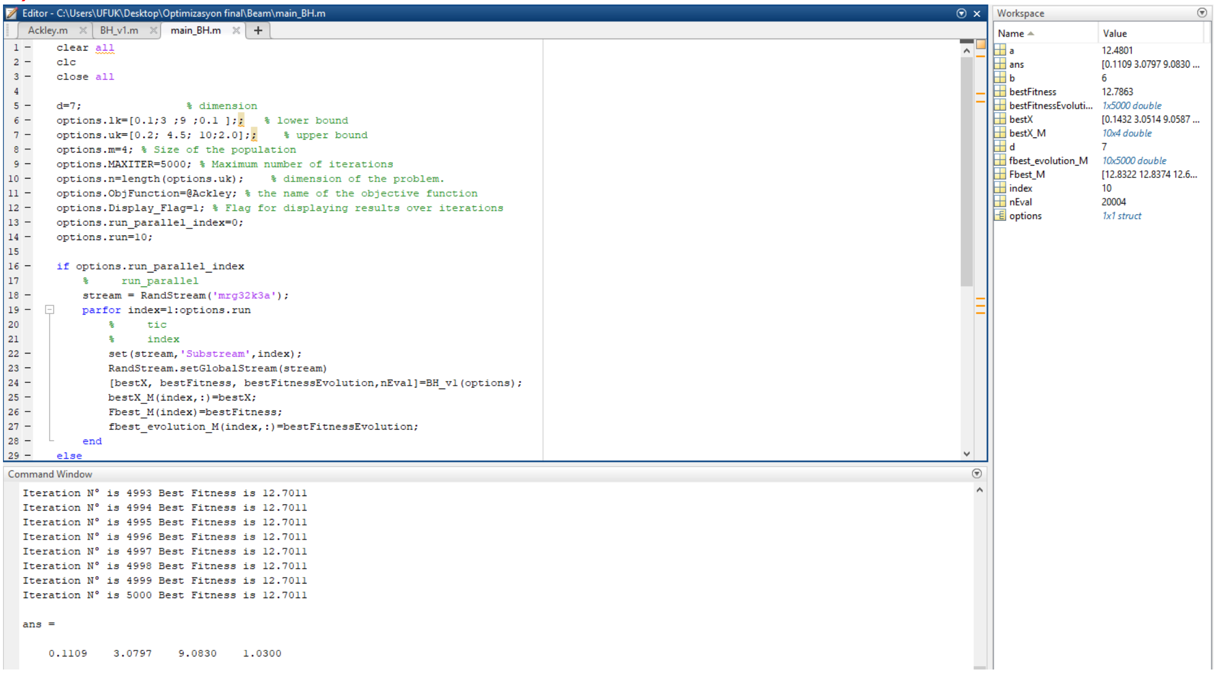
```
        landa=rand;
        x(i,k) = lk(k) + landa*(uk(k) - lk(k));
    end
end
% x(end+1,:)=x0;
ObjFunctionValue=feval(ObjFunction,x);
[index1,index2]=sort(ObjFunctionValue);
x=x(index2(1:m),:);
xBH=x(1,:);
iBH=1;
ObjFunctionValue=ObjFunctionValue(index2(1:m));
end

function [xb,ib,xw,iw]=argmin(x,f,options)
[minf,ib]=min(f);
xb=x(ib,:);
[maxf,iw]=max(f);
xw=x(iw,:);
end


function
[x,ObjFunctionValue]=NewStarGeneration(x,Distance,R,options,iBH,ObjFunctionValu
e)
ObjFunction=options.ObjFunction; % the name of the objective function.
n=options.n;    % n: dimension of the problem.
uk=options.uk;  % up: upper bound in the kth dimension.
lk=options.lk;  % lp: lower bound in the kth dimension.
index=find(Distance<R);
for i=1:length(index)
    if index(i) ~= iBH
        for k = 1 : n
            x(i,k) = lk(k) + rand*(uk(k) - lk(k));
        end
```

```
        ObjFunctionValue(i)=feval(ObjFunction,x(i,:));
    end
end
end
function [x]=bound(x,l,u)
for j = 1:size(x,1)
    for k = 1:size(x,2)
        % check upper boundary
        if x(j,k) > u(k),
            x(j,k) = u(k);
        end
        % check lower boundary
        if x(j,k) < l(k),
            x(j,k) = l(k);
        end
    end
end
end
```

b)Welded Beam



Main_BH.m

```
function [bestX, bestFitness, bestFitnessEvolution,nEval]=BH_v1(options)
%-----------------------------------------------------------------------
% Black Hole Algorithm
% Dr Hpussem BOUCHEKARA
% 20/07/2019
%-----------------------------------------------------------------------
% 1. Bouchekara, H. R. E. H. (2013). Optimal design of electromagnetic
% devices using a black-Hole-Based optimization technique. IEEE
% Transactions on Magnetics, 49(12). doi:10.1109/TMAG.2013.2277694
%
% 2. Bouchekara, H. R. E. H. (2014). Optimal power flow using black-hole-based
% optimization approach. Applied Soft Computing, 24, 879–888.
% doi:10.1016/j.asoc.2014.08.056
%
% 3. Smail, M. K., Bouchekara, H. R. E. H., Pichon, L., Boudjefdjouf, H.,
% Amloune, A., & Lacheheb, Z. (2016). Non-destructive diagnosis of wiring
% networks using time domain reflectometry and an improved black hole
% algorithm. Nondestructive Testing and Evaluation.
```

```
% doi:10.1080/10589759.2016.1200576
%-------------------------------------------------------------------------
% Initialize a population of stars with random locations in the search space
% Loop
%   For each star, evaluate the objective function
%   Select the best star that has the best fitness value as the black hole
%   Change the location of each star according to Eq. (3)
%   If a star reaches a location witch lower cost than the black hole, exchange their locations
%   If a star crosses the event horizon of the black hole, replace it with a new star in a random location in the search space
%   If a termination criterion (a maximum number of iterations or a sufficiently good fitness) is met, exit the loop
% End loop
%-------------------------------------------------------------------------

ObjFunction=options.ObjFunction; % the name of the objective function
n=options.n;    % dimension of the problem.
uk=options.uk;   % upper bound in the kth dimension.
lk=options.lk;  % lower bound in the kth dimension.
m=options.m; % m: number of sample points
MAXITER=options.MAXITER; % MAXITER: maximum number of iterations
nEval=0;
[x,xBH,iBH,ObjFunctionValue]=Initialize(options);
nEval=nEval+size(x,1);
for iteration =1:MAXITER
    %     tic
    %     Change the location of each star according to Eq. (3)
    for i = 1 : m
        if i ~= iBH
            landa=rand;
            for k = 1 : n
                if landa<0.5
                    x(i,k)=x(i,k) + rand*(xBH(k)- x(i,k));
```

```
            else
               x(i,k)=x(i,k) + rand*(xBH(k)- x(i,k));
            end
         end
      end
   end
   %   If a star reaches a location with lower cost than the black
   %   hole, exchange their locations
   ObjFunctionValue=feval(ObjFunction,x);
   nEval=nEval+size(x,1);
   %     [x]=bound(x,lk,uk);
   %     [xBH,iBH]=argmin(x,ObjFunctionValue,options);
   %   If a star crosses the event horizon of the black hole, replace it
   %   with a new star in a random location in the search space
   R=ObjFunctionValue(iBH)/sum(ObjFunctionValue);
   %     R=exp(-n*ObjFunctionValue(iBH)/sum(ObjFunctionValue))
   %     pause
   for i = 1 : m
      Distance(i)=norm(xBH- x(i,:));
   end

[x,ObjFunctionValue]=NewStarGeneration(x,Distance,R,options,iBH,ObjFunctionValu
e);
   [x]=bound(x,lk,uk);
   [xBH,iBH]=argmin(x,ObjFunctionValue,options);

   %----------------------------------------------------------------------------
   bestFitnessEvolution(iteration)=ObjFunctionValue(iBH);
   %----------------------------------------------------------------------------



   if options.Display_Flag==1
      fprintf('Iteration N° is %g Best Fitness is %g\n',iteration,ObjFunctionValue(iBH))
   end
```

```
end
bestX=xBH;
bestFitness=ObjFunctionValue(iBH);
end

function [x,xBH,iBH,ObjFunctionValue]=Initialize(options)
ObjFunction=options.ObjFunction; % the name of the objective function.
n=options.n;    % n: dimension of the problem.
uk=options.uk;  % up: upper bound in the kth dimension.
lk=options.lk;  % lp: lower bound in the kth dimension.
m=options.m;    % m: number of sample points

for i = 1 : m
    for k = 1 : n
        landa=rand;
        x(i,k) = lk(k) + landa*(uk(k) - lk(k));
    end
end
% x(end+1,:)=x0;
ObjFunctionValue=feval(ObjFunction,x);
[index1,index2]=sort(ObjFunctionValue);
x=x(index2(1:m),:);
xBH=x(1,:);
iBH=1;
ObjFunctionValue=ObjFunctionValue(index2(1:m));
end

function [xb,ib,xw,iw]=argmin(x,f,options)
[minf,ib]=min(f);
xb=x(ib,:);
[maxf,iw]=max(f);
xw=x(iw,:);
end
```

```
function
[x,ObjFunctionValue]=NewStarGeneration(x,Distance,R,options,iBH,ObjFunctionValu
e)
ObjFunction=options.ObjFunction; % the name of the objective function.
n=options.n;    % n: dimension of the problem.
uk=options.uk;  % up: upper bound in the kth dimension.
lk=options.lk;  % lp: lower bound in the kth dimension.
index=find(Distance<R);
for i=1:length(index)
   if index(i) ~= iBH
      for k = 1 : n
         x(i,k) = lk(k) + rand*(uk(k) - lk(k));
      end
      ObjFunctionValue(i)=feval(ObjFunction,x(i,:));
   end
end
end
function [x]=bound(x,l,u)
for j = 1:size(x,1)
   for k = 1:size(x,2)
      % check upper boundary
      if x(j,k) > u(k),
         x(j,k) = u(k);
      end
      % check lower boundary
      if x(j,k) < l(k),
         x(j,k) = l(k);
      end
   end
end
end
```

Ackley.m

```
function [F, lb, ub, FGO] = Ackley(x)
% Ackley function
if (nargin==0)
    F=[];
    d=2;                % dimension
    lb=-32*ones(1,d);   % lower bound
    ub=32*ones(1,d);    % upper bound
    FGO=0;              % Global Optimum
else
    n=size(x,2);
    for ix=1:size(x,1)
        x0=x(ix,:);
        F(ix) = -20*exp(-0.2*sqrt(1/n*sum(x0.^2)))-...
                exp(1/n*sum(cos(2*pi*x0)))+20+exp(1);
    end
end
```

BH_v1.m

```
function [bestX, bestFitness, bestFitnessEvolution,nEval]=BH_v1(options)
%-----------------------------------------------------------------------
% Black Hole Algorithm
% Dr Hpussem BOUCHEKARA
% 20/07/2019
%-----------------------------------------------------------------------
% 1. Bouchekara, H. R. E. H. (2013). Optimal design of electromagnetic
% devices using a black-Hole-Based optimization technique. IEEE
```

```
% Transactions on Magnetics, 49(12). doi:10.1109/TMAG.2013.2277694
%
% 2. Bouchekara, H. R. E. H. (2014). Optimal power flow using black-hole-based
% optimization approach. Applied Soft Computing, 24, 879–888.
% doi:10.1016/j.asoc.2014.08.056
%
% 3. Smail, M. K., Bouchekara, H. R. E. H., Pichon, L., Boudjefdjouf, H.,
% Amloune, A., & Lacheheb, Z. (2016). Non-destructive diagnosis of wiring
% networks using time domain reflectometry and an improved black hole
% algorithm. Nondestructive Testing and Evaluation.
% doi:10.1080/10589759.2016.1200576
%-----------------------------------------------------------------------
% Initialize a population of stars with random locations in the search space
% Loop
%   For each star, evaluate the objective function
%   Select the best star that has the best fitness value as the black hole
%   Change the location of each star according to Eq. (3)
%   If a star reaches a location witch lower cost than the black hole, exchange their
locations
%   If a star crosses the event horizon of the black hole, replace it with a new star in a
random location in the search space
%   If a termination criterion (a maximum number of iterations or a sufficiently good
fitness) is met, exit the loop
% End loop
%-----------------------------------------------------------------------

ObjFunction=options.ObjFunction; % the name of the objective function
n=options.n;    % dimension of the problem.
uk=options.uk;   % upper bound in the kth dimension.
lk=options.lk;  % lower bound in the kth dimension.
m=options.m; % m: number of sample points
MAXITER=options.MAXITER; % MAXITER: maximum number of iterations
nEval=0;
[x,xBH,iBH,ObjFunctionValue]=Initialize(options);
```

```
nEval=nEval+size(x,1);
for iteration =1:MAXITER
    %     tic
    %     Change the location of each star according to Eq. (3)
    for i = 1 : m
        if i ~= iBH
            landa=rand;
            for k = 1 : n
                if landa<0.5
                    x(i,k)=x(i,k) + rand*(xBH(k)- x(i,k));
                else
                    x(i,k)=x(i,k) + rand*(xBH(k)- x(i,k));
                end
            end
        end
    end
    %   If a star reaches a location with lower cost than the black
    %   hole, exchange their locations
    ObjFunctionValue=feval(ObjFunction,x);
    nEval=nEval+size(x,1);
    %     [x]=bound(x,lk,uk);
    %     [xBH,iBH]=argmin(x,ObjFunctionValue,options);
    %   If a star crosses the event horizon of the black hole, replace it
    %   with a new star in a random location in the search space
    R=ObjFunctionValue(iBH)/sum(ObjFunctionValue);
    %     R=exp(-n*ObjFunctionValue(iBH)/sum(ObjFunctionValue))
    %     pause
    for i = 1 : m
        Distance(i)=norm(xBH- x(i,:));
    end

[x,ObjFunctionValue]=NewStarGeneration(x,Distance,R,options,iBH,ObjFunctionValu
e);
    [x]=bound(x,lk,uk);
```

```
    [xBH,iBH]=argmin(x,ObjFunctionValue,options);

    %-----------------------------------------------------------------------------
    bestFitnessEvolution(iteration)=ObjFunctionValue(iBH);
    %-----------------------------------------------------------------------------


    if options.Display_Flag==1
        fprintf('Iteration N° is %g Best Fitness is %g\n',iteration,ObjFunctionValue(iBH))
    end

end
bestX=xBH;
bestFitness=ObjFunctionValue(iBH);
end

function [x,xBH,iBH,ObjFunctionValue]=Initialize(options)
ObjFunction=options.ObjFunction; % the name of the objective function.
n=options.n;    % n: dimension of the problem.
uk=options.uk;  % up: upper bound in the kth dimension.
lk=options.lk;  % lp: lower bound in the kth dimension.
m=options.m;    % m: number of sample points

for i = 1 : m
    for k = 1 : n
        landa=rand;
        x(i,k) = lk(k) + landa*(uk(k) - lk(k));
    end
end
% x(end+1,:)=x0;
ObjFunctionValue=feval(ObjFunction,x);
[index1,index2]=sort(ObjFunctionValue);
x=x(index2(1:m),:);
xBH=x(1,:);
```

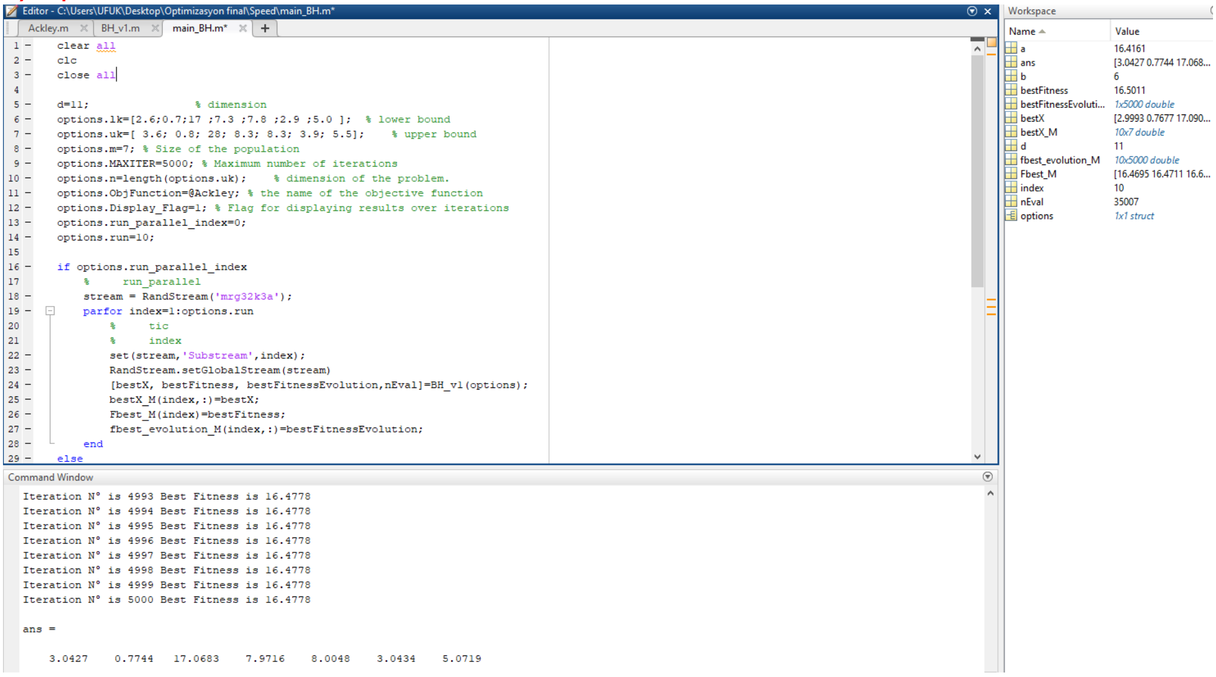
```
iBH=1;
ObjFunctionValue=ObjFunctionValue(index2(1:m));
end

function [xb,ib,xw,iw]=argmin(x,f,options)
[minf,ib]=min(f);
xb=x(ib,:);
[maxf,iw]=max(f);
xw=x(iw,:);
end


function
[x,ObjFunctionValue]=NewStarGeneration(x,Distance,R,options,iBH,ObjFunctionValu
e)
ObjFunction=options.ObjFunction; % the name of the objective function.
n=options.n;    % n: dimension of the problem.
uk=options.uk;  % up: upper bound in the kth dimension.
lk=options.lk;  % lp: lower bound in the kth dimension.
index=find(Distance<R);
for i=1:length(index)
   if index(i) ~= iBH
      for k = 1 : n
         x(i,k) = lk(k) + rand*(uk(k) - lk(k));
      end
      ObjFunctionValue(i)=feval(ObjFunction,x(i,:));
   end
end
end
function [x]=bound(x,l,u)
for j = 1:size(x,1)
   for k = 1:size(x,2)
      % check upper boundary
      if x(j,k) > u(k),
```

```
            x(j,k) = u(k);
        end
        % check lower boundary
        if x(j,k) < l(k),
            x(j,k) = l(k);
        end
    end
end
end
```

c) Speed Reducer



Main_BH.m

```
clear all

clc

close all


d=11;              % dimension

options.lk=[2.6;0.7;17 ;7.3 ;7.8 ;2.9 ;5.0 ];;   % lower bound

options.uk=[ 3.6; 0.8; 28; 8.3; 8.3; 3.9; 5.5];;    % upper bound

options.m=7; % Size of the population

options.MAXITER=5000; % Maximum number of iterations

options.n=length(options.uk);    % dimension of the problem.

options.ObjFunction=@Ackley; % the name of the objective function

options.Display_Flag=1; % Flag for displaying results over iterations

options.run_parallel_index=0;

options.run=10;


if options.run_parallel_index

   %     run_parallel

   stream = RandStream('mrg32k3a');

   parfor index=1:options.run
```

```
        %     tic
        %     index
        set(stream,'Substream',index);
        RandStream.setGlobalStream(stream)
        [bestX, bestFitness, bestFitnessEvolution,nEval]=BH_v1(options);
        bestX_M(index,:)=bestX;
        Fbest_M(index)=bestFitness;
        fbest_evolution_M(index,:)=bestFitnessEvolution;
    end
else
    rng('default')
    for index=1:options.run
        [bestX, bestFitness, bestFitnessEvolution,nEval]=BH_v1(options);
        bestX_M(index,:)=bestX;
        Fbest_M(index)=bestFitness;
        fbest_evolution_M(index,:)=bestFitnessEvolution;
    end
end


[a,b]=min(Fbest_M);
figure
plot(1:options.MAXITER,fbest_evolution_M(b,:))
xlabel('Iterations')
ylabel('Fitness')

fprintf(' MIN=%g  MEAN=%g  MEDIAN=%g MAX=%g  SD=%g \n',...
    min(Fbest_M),mean(Fbest_M),median(Fbest_M),max(Fbest_M),std(Fbest_M))
```

Ackley.m

```
function [F, lb, ub, FGO] = Ackley(x)
% Ackley function
if (nargin==0)
    F=[];
    d=2;                % dimension
    lb=-32*ones(1,d);   % lower bound
    ub=32*ones(1,d);    % upper bound
    FGO=0;              % Global Optimum
else
    n=size(x,2);
    for ix=1:size(x,1)
        x0=x(ix,:);
        F(ix) = -20*exp(-0.2*sqrt(1/n*sum(x0.^2)))-...
                exp(1/n*sum(cos(2*pi*x0)))+20+exp(1);
    end
end
```

BH_v1.m

```
function [bestX, bestFitness, bestFitnessEvolution,nEval]=BH_v1(options)
%------------------------------------------------------------------------
% Black Hole Algorithm
% Dr Hpussem BOUCHEKARA
% 20/07/2019
%------------------------------------------------------------------------
% 1. Bouchekara, H. R. E. H. (2013). Optimal design of electromagnetic
% devices using a black-Hole-Based optimization technique. IEEE
```

```
% Transactions on Magnetics, 49(12). doi:10.1109/TMAG.2013.2277694
%
% 2. Bouchekara, H. R. E. H. (2014). Optimal power flow using black-hole-based
% optimization approach. Applied Soft Computing, 24, 879–888.
% doi:10.1016/j.asoc.2014.08.056
%
% 3. Smail, M. K., Bouchekara, H. R. E. H., Pichon, L., Boudjefdjouf, H.,
% Amloune, A., & Lacheheb, Z. (2016). Non-destructive diagnosis of wiring
% networks using time domain reflectometry and an improved black hole
% algorithm. Nondestructive Testing and Evaluation.
% doi:10.1080/10589759.2016.1200576
%-----------------------------------------------------------------------
% Initialize a population of stars with random locations in the search space
% Loop
%   For each star, evaluate the objective function
%   Select the best star that has the best fitness value as the black hole
%   Change the location of each star according to Eq. (3)
%   If a star reaches a location witch lower cost than the black hole, exchange their
locations
%   If a star crosses the event horizon of the black hole, replace it with a new star in a
random location in the search space
%   If a termination criterion (a maximum number of iterations or a sufficiently good
fitness) is met, exit the loop
% End loop
%-----------------------------------------------------------------------

ObjFunction=options.ObjFunction; % the name of the objective function
n=options.n;    % dimension of the problem.
uk=options.uk;   % upper bound in the kth dimension.
lk=options.lk;  % lower bound in the kth dimension.
m=options.m; % m: number of sample points
MAXITER=options.MAXITER; % MAXITER: maximum number of iterations
nEval=0;
[x,xBH,iBH,ObjFunctionValue]=Initialize(options);
```

```
nEval=nEval+size(x,1);
for iteration =1:MAXITER
    %     tic
    %     Change the location of each star according to Eq. (3)
    for i = 1 : m
        if i ~= iBH
            landa=rand;
            for k = 1 : n
                if landa<0.5
                    x(i,k)=x(i,k) + rand*(xBH(k)- x(i,k));
                else
                    x(i,k)=x(i,k) + rand*(xBH(k)- x(i,k));
                end
            end
        end
    end
    %   If a star reaches a location with lower cost than the black
    %   hole, exchange their locations
    ObjFunctionValue=feval(ObjFunction,x);
    nEval=nEval+size(x,1);
    %     [x]=bound(x,lk,uk);
    %     [xBH,iBH]=argmin(x,ObjFunctionValue,options);
    %   If a star crosses the event horizon of the black hole, replace it
    %   with a new star in a random location in the search space
    R=ObjFunctionValue(iBH)/sum(ObjFunctionValue);
    %     R=exp(-n*ObjFunctionValue(iBH)/sum(ObjFunctionValue))
    %     pause
    for i = 1 : m
        Distance(i)=norm(xBH- x(i,:));
    end

[x,ObjFunctionValue]=NewStarGeneration(x,Distance,R,options,iBH,ObjFunctionValu
e);
    [x]=bound(x,lk,uk);
```

```
    [xBH,iBH]=argmin(x,ObjFunctionValue,options);

    %---------------------------------------------------------------------------
    bestFitnessEvolution(iteration)=ObjFunctionValue(iBH);
    %---------------------------------------------------------------------------



    if options.Display_Flag==1
        fprintf('Iteration N° is %g Best Fitness is %g\n',iteration,ObjFunctionValue(iBH))
    end

end
bestX=xBH;
bestFitness=ObjFunctionValue(iBH);
end

function [x,xBH,iBH,ObjFunctionValue]=Initialize(options)
ObjFunction=options.ObjFunction; % the name of the objective function.
n=options.n;    % n: dimension of the problem.
uk=options.uk;  % up: upper bound in the kth dimension.
lk=options.lk;  % lp: lower bound in the kth dimension.
m=options.m;    % m: number of sample points

for i = 1 : m
    for k = 1 : n
        landa=rand;
        x(i,k) = lk(k) + landa*(uk(k) - lk(k));
    end
end
% x(end+1,:)=x0;
ObjFunctionValue=feval(ObjFunction,x);
[index1,index2]=sort(ObjFunctionValue);
x=x(index2(1:m),:);
xBH=x(1,:);
```

```
iBH=1;
ObjFunctionValue=ObjFunctionValue(index2(1:m));
end

function [xb,ib,xw,iw]=argmin(x,f,options)
[minf,ib]=min(f);
xb=x(ib,:);
[maxf,iw]=max(f);
xw=x(iw,:);
end


function
[x,ObjFunctionValue]=NewStarGeneration(x,Distance,R,options,iBH,ObjFunctionValu
e)
ObjFunction=options.ObjFunction; % the name of the objective function.
n=options.n;    % n: dimension of the problem.
uk=options.uk;  % up: upper bound in the kth dimension.
lk=options.lk;  % lp: lower bound in the kth dimension.
index=find(Distance<R);
for i=1:length(index)
   if index(i) ~= iBH
      for k = 1 : n
         x(i,k) = lk(k) + rand*(uk(k) - lk(k));
      end
      ObjFunctionValue(i)=feval(ObjFunction,x(i,:));
   end
end
end
function [x]=bound(x,l,u)
for j = 1:size(x,1)
   for k = 1:size(x,2)
      % check upper boundary
      if x(j,k) > u(k),
```

```
            x(j,k) = u(k);
        end
        % check lower boundary
        if x(j,k) < l(k),
            x(j,k) = l(k);
        end
    end
end
end
```