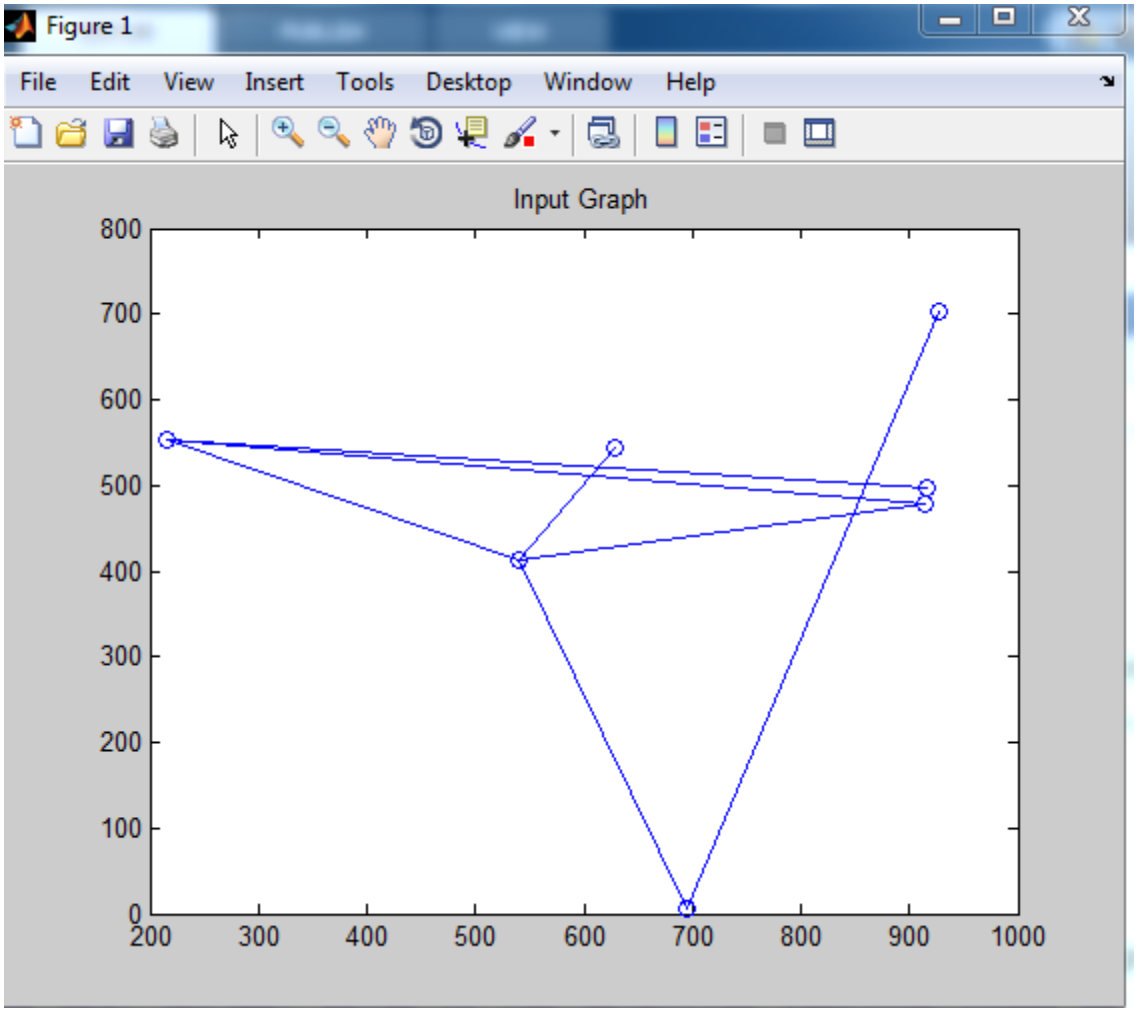
حل مسئله رنگ آمیزی گراف با الگوریتم ملخ GOA در متلب

صورت مسئله:

می خواهیم رئوس گراف زیر را به گونه ای با سه رنگ سبز، مشکی، آبی رنگ کنیم که هیچ دو یال مجاوری همرنگ نباشند.



شرح کد:

این برنامه از ۷ فایل ایجاد شده که شامل:

Main.m, S_func.m, initialization.m, GOA.m, Get_Functions_details.m, distance.m, CreateModel.m

و هر کدام مسیول انجام کاری هستند که مورد به مورد بررسی می کنیم

CreateModel.m: ایجاد یک مدل از گراف به صورت رندم یا سفارشی و محاسبه تعداد نودها، تعداد رنگ ها، گراف، و تعداد یال ها می باشد.

```
function model=CreateModel()
                                                    گراف ورودی
        N_nodes=input('Enter Number of Nodes:');
                                                تعداد نود ها را وارد کن
        N_Colors=input('Enter Number of Colors:');
                                                تعداد رنگ ها را وارد کن
        G=zeros(N_nodes);
        type=input('Graph Type: Custom(1) / Random(2)?');
    اگر می خواهی گراف سفارشی خودت را بسازی 1 و اگر می خواهی
                                                رندم ساخته شود 2 را وارد کن
        if type==1
                                                گراف سفارشی

            for i=1:N_nodes
                for j=i:N_nodes
                    if i~=j
                        connectivity=input(['Is there a
link between nodes ',num2str(i),' and ',num2str(j), ' ?
(0/1) : ']);
  برای هر راس با رئوس دیگر می پرسد که آیا می خواهی یالی بین
                                                        آن ها باشد یا خیر
                        if connectivity==1
                            G(i,j)=1;
                            G(j,i)=1;
                        end
                    end
                end
            end
        else
                                                    گراف رندم
            for i=1:N_nodes
                for j=i:N_nodes
                    if i~=j
                        if rand>0.5
                            G(i,j)=1;
                            G(j,i)=1;
                        end
```

```
                    end
                end
            end
        end
```

کشیدن گراف

```
        position=randsrc(2,N_nodes,1:1000);
        figure(1),
        plot(position(1,:),position(2,:),'o');
        title('Input Graph');
        hold on
        tedadyal=0;
        for i=1:N_nodes
            for j=1:N_nodes
                if G(i,j)==1

line([position(1,i),position(1,j)],[position(2,i),position(
2,j)]);
                    tedadyal=tedadyal+1;
                end
            end
        end
        hold off
```

تعداد نود ها و تعداد رنگ ها و خود گراف و تعداد یال ها را به تابع اصلی می فرستیم، در این جا متغیر های مسیله تعریف شد.

```
    model.N_nodes=N_nodes;
    model.N_Colors=N_Colors;
    model.G=G;
    model.position=position;
    model.tedadyal=tedadyal/2;
end
```

Get_Functions_details.m: این تابع برای محاسبه فیتنس است، فیتنس را تعداد یال های مجاور غیر همرنگ در نظر گرفتیم.

```
function fobj = Get_Functions_details(F)
switch F
    case 'F1'
```

اگر ورودی F1 است تابع F1 را اجرا کن

```
        fobj = @F1;
 end
end

% F1
function f = F1(malakh,G,ub)
```

```
ورودی این تابع هر ملخ که یک برداری به طول ابعاد مسیله است
هست به همراه گرافش و حد بالا مسله که حد اکثر تعداد رنگ هاست.
F فیتنس و y تعداد کل یال ها است.
    f=0;
    y=0;
    for i=1:ub
        if sum(i==malakh)==0
                        اگر هیچ راسی رنگ نشده بود فیتنس 0 است.
            f=0;
            return;
        end
    end
    for i=1:size(G,1)
        for j=1:size(G,2)
            if G(i,j)==1
                            اگر بین i وj یالی هست به yاضافه کن
                y=y+1;
                if malakh(i)==malakh(j)
                 اگر یال های مجاور همرنگ هستند به f اضافه کن
                    f=f+1;
                end
            end
        end
    end
  تعداد کل یال ها می شود y/2 و تعداد کل یال های مجاور می شود
 f/2 چون در دو حلقه تو در توی بالا دو بار می شمارد از i به jو
    از j به i. که نهایتا از هم کم می کنیم تا خروجی حاصل شود.
   f=(y/2)-(f/2);
end
```

main.m: این فایل برای اجرا ی برنامه است و دو فایل قبلی در این فایل فراخوانی می شوند و متغیر های مسیله و پارامترهای الگوریتم ملخ مقداردهی شده و مقادیر را به تابع الگوریتم ملخ می فرستیم و سپس نتایج الگوریتم برمی گردد و آن را نمایش می دهیم.

dim = تعداد متغیرهای شما

حداکثر_یتر = حداکثر تعداد نسل ها

SearchAgents_no = تعداد ملخ ها

lb = [lb1، lb2، ...، lbn] که در آن lbn مرز پایینی متغیر n است

ub = [ub1؛ ub2؛ ...؛ ubn] که ubn مرز فوقاری متغیر n است

اگر همه متغیرها دارای حد پایین تر باشند ، می توانید فقط عمل کنید

lb و ub را به عنوان دو عدد تک عددی تعریف کنید

برای اجرای الگوریتم ملخ:

GOA: [Best_score ،Best_pos ،GOA_cg_curve] = GOA (SearchAgents_no ،Max_iteration ،lb ،ub ، dim ،fobj)

```
clear all
clc

SearchAgents_no=8; تعداد ملخ ها

model=CreateModel();

N_nodes=model.N_nodes;
N_Colors=model.N_Colors;
G=model.G;
Position=model.position;
tedadyal=model.tedadyal;

lb=1;
ub=N_Colors;
dim=N_nodes;

Function_name='F1'          شماره تابع فیتنس

Max_iteration=25;        ماکزیمم تعداد تکرار

                                                تابع فیتنس
fobj=Get_Functions_details(Function_name);

                                                تابع الگوریتم ملخ
[Target_score,Target_pos,GOA_cg_curve,
Trajectories,fitness_history,
position_history]=GOA(SearchAgents_no,Max_iteration,lb,ub,d
im,fobj,G);
                                                ایجاد جدول رنگ
```

```
    w=9;
    Color={[0 0 0],[0 0 1],[0 1 0],[0 1 1],[1 0 0],[1 0
1],[1 1 0],[1 1 1]};
    stepsize=0.1;
    for i=0.1:stepsize:0.9
        for j=0.1:stepsize:0.9
            for k=0.1:stepsize:0.9
                Color{w}=[i j k];
                w=w+1;
            end
        end
    end
```

کشیدن گراف رنگ آمیزی

```
    figure(2),
    for i=1:dim
        for j=1:dim
            if G(i,j)==1

line([Position(1,i),Position(1,j)],[Position(2,i),Position(
2,j)]);
                 hold on
            end
        end
    end
for i=1:N_nodes

plot(Position(1,i),Position(2,i),'o','MarkerFaceColor',Colo
r{Target_pos(i)},'MarkerEdgeColor',Color{Target_pos(i)},'Ma
rkerSize',11);

text(Position(1,i),Position(2,i),num2str(Target_pos(i)),'co
lor','r')
    hold on
end
title('Colored Graph');
```

تعداد یال های گراف

```
display(['The tedadyal grapg is : ', num2str(tedadyal)]);
```

ملخ با بیشترین فیتنس

```
display(['The best solution obtained by GOA is : ',
num2str(Target_pos)]);
```

فیتنس ملخ با بیشترین فیتنس

```
display(['The best optimal value of the objective funciton
found by GOA is : ', num2str(Target_score)]);
```

initialization.m: این تابع به صورت رندم مکان ملخ ها را در فضای جست و جو مقداردهی اولیه می کند

```
function [X]=initialization(N,dim,up,down)
if size(up,1)==1
    X=rand(N,dim).*(up-down)+down;
end
if size(up,1)>1
    for i=1:dim
        high=up(i);low=down(i);
        X(:,i)=rand(1,N).*(high-low)+low;
    end
end
```

GOA.m: الگوریتم بهینه سازی ملخ

```
function
[TargetFitness,TargetPosition,Convergence_curve,Trajectorie
s,fitness_history, position_history]=GOA(N, Max_iter,
lb,ub, dim, fobj,G)

tic
disp('GOA is now estimating the global optimum for your
problem....')

flag=0;
if size(ub,1)==1
    ub=ones(dim,1)*ub;
    lb=ones(dim,1)*lb;
end

if (rem(dim,2)~=0)
این الگوریتم بهتر است با تعداد زوج اجرا شود.این خط عدد فرد
را کنترل می کند
    ub = [ub; 100];
    lb = [lb; -100];
    flag=1;
```

```
end

                                        مقداردهی اولیه جمعیت ملخ ها
ghp=initialization(N,dim,ub,lb);
GrassHopperPositions=round(ghp);
GrassHopperFitness = zeros(1,N);

fitness_history=zeros(N,Max_iter);
position_history=zeros(N,Max_iter,dim);
Convergence_curve=zeros(1,Max_iter);
Trajectories=zeros(N,Max_iter);

cMax=1;
cMin=0.00004;
                                        محاسبه فیتنس ملخ های اولیه

for i=1:size(GrassHopperPositions,1)
    if flag == 1

GrassHopperFitness(1,i)=fobj(GrassHopperPositions(i,1:end-
1),G,ub);
    else

GrassHopperFitness(1,i)=fobj(GrassHopperPositions(i,:),G,ub
);
    end
    fitness_history(i,1)=GrassHopperFitness(1,i);
    position_history(i,1,:)=GrassHopperPositions(i,:);
    Trajectories(:,1)=GrassHopperPositions(:,1);
end

[sorted_fitness,sorted_indexes]=sort(GrassHopperFitness,'de
scend');

                                    بهترین ملخ هدف را در جمعیت اولیه پیدا کن.

for newindex=1:N

Sorted_grasshopper(newindex,:)=GrassHopperPositions(sorted_
indexes(newindex),:);
end

TargetPosition=Sorted_grasshopper(1,:);
TargetFitness=sorted_fitness(1);
```

```
                                                    حلقه اصلی
l=2;
   از تکرار دوم شروع کن چون در تکرار اول تصمیم گیری ها انجام
                                                    شد و فیتنس محاسبه شد.
while l<Max_iter+1

c=cMax-l*((cMax-cMin)/Max_iter); آپدیت پارامتر c


      for i=1:size(GrassHopperPositions,1)
        temp= GrassHopperPositions';
       % for k=1:2:dim
            S_i=zeros(dim,1);
            for j=1:N
                if i~=j
                      Dist=distance(temp(:,j), temp(:,i)); %
                                                محاسبه فاصله بین دو ملخ

                    r_ij_vec=(temp(:,j)-
temp(:,i))/(Dist+eps); % xj-xi/dij in Eq. (2.7)
                    xj_xi=2+rem(Dist,2); % |xjd - xid|

                    s_ij=((ub -
lb)*c/2)*S_func(xj_xi).*r_ij_vec;  اولین بخش درون براکت بزرگ
در معادله ملخ
                    S_i=S_i+s_ij;
                end
            end
            S_i_total = S_i;

      %  end

        X_new = c * S_i_total'+ (TargetPosition);  معادله
مقاله
        GrassHopperPositions_temp(i,:)=X_new';
      end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    % GrassHopperPositions
    GrassHopperPositions=round(GrassHopperPositions_temp);
```

```
    for i=1:size(GrassHopperPositions,1)
                ملخ را به بیرون از فضای سرچ جابه جا کن

Tp=GrassHopperPositions(i,:)>ub';Tm=GrassHopperPositions(i,
:)<lb';GrassHopperPositions(i,:)=(GrassHopperPositions(i,:)
.*(~(Tp+Tm)))+ub'.*Tp+lb'.*Tm;

                                    محاسبه فیتنس همه ملخ ها
        if flag == 1

GrassHopperFitness(1,i)=fobj(GrassHopperPositions(i,1:end-
1),G,ub);
        else

GrassHopperFitness(1,i)=fobj(GrassHopperPositions(i,:),G,ub
);
        end
        fitness_history(i,l)=GrassHopperFitness(1,i);
        position_history(i,l,:)=GrassHopperPositions(i,:);

        Trajectories(:,l)=GrassHopperPositions(:,1);

                                                آپدیت هدف
        if GrassHopperFitness(1,i)>TargetFitness
            TargetPosition=GrassHopperPositions(i,:);
            TargetFitness=GrassHopperFitness(1,i);
        end
    end

    Convergence_curve(l)=TargetFitness;
    disp(['In iteration #', num2str(l), ' , target''s
objective = ', num2str(TargetFitness)])

    l = l + 1;
end


if (flag==1)
    TargetPosition = TargetPosition(1:dim-1);
end

time=toc
```

distance.m: فاصله اقلیدسی بین ملخ a و b

```
function d = distance(a,b)

if (nargin ~= 2)
   error('Not enough input arguments');
end

if (size(a,1) ~= size(b,1))
   error('A and B should be of same dimensionality');
end

aa=sum(a.*a,1); bb=sum(b.*b,1); ab=a'*b;
d = sqrt(abs(repmat(aa',[1 size(bb,2)]) +
repmat(bb,[size(aa,2) 1]) - 2*ab));
```

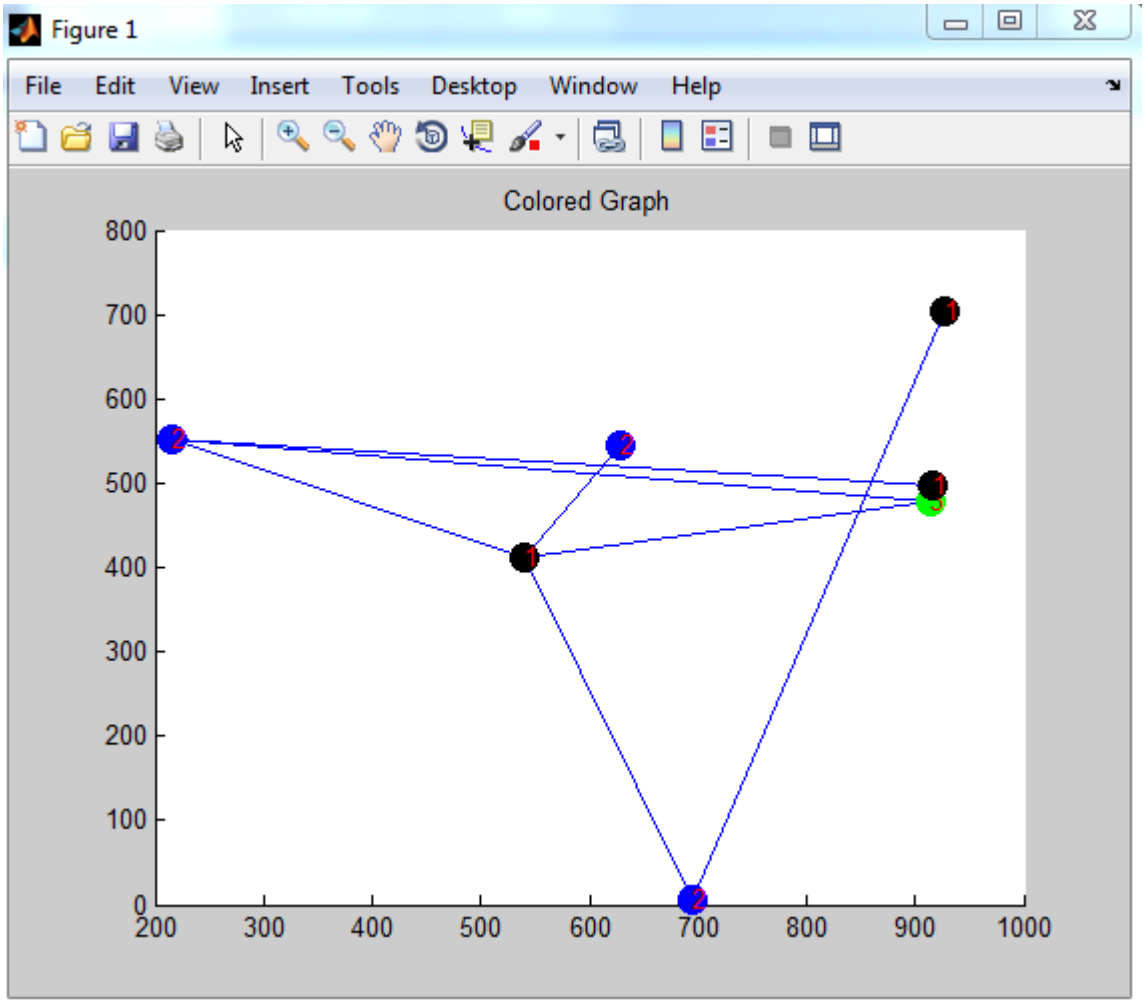
S_func.m: جاذبه دافعه کل ملخ ها نسبت به یک علف

f شدت جاذبه وl طول جاذبه است.

```
function o=S_func(r)
f=0.5;
l=1.5;
o=f*exp(-r/l)-exp(-r);  معادله جاذبه دافعه
end
```

در ادامه نتیجه اجرا برنامه با ۷ راس و ۳ رنگ و گراف رندم به این صورت است:

نتایج اجرا main.m



همانطور که مشاهده می کنیم الگوریتم ملخ توانست مسیله رنگ امیزی گراف را حل کند

```
Enter Number of Nodes:7
Enter Number of Colors:3
Graph Type: Custom(1) / Random(2)?2
GOA is now estimating the global optimum for your
problem....
In iteration #2 , target's objective = 4
In iteration #3 , target's objective = 4
```

```
In iteration #4 , target's objective = 4
In iteration #5 , target's objective = 4
In iteration #6 , target's objective = 4
In iteration #7 , target's objective = 4
In iteration #8 , target's objective = 4
In iteration #9 , target's objective = 4
In iteration #10 , target's objective = 4
In iteration #11 , target's objective = 4
In iteration #12 , target's objective = 4
In iteration #13 , target's objective = 5
In iteration #14 , target's objective = 5
In iteration #15 , target's objective = 5
In iteration #16 , target's objective = 5
In iteration #17 , target's objective = 6
In iteration #18 , target's objective = 6
In iteration #19 , target's objective = 6
In iteration #20 , target's objective = 6
In iteration #21 , target's objective = 6
In iteration #22 , target's objective = 6
In iteration #23 , target's objective = 6
In iteration #24 , target's objective = 6
In iteration #25 , target's objective = 7

time =

    0.1019

The tedadyal grapg is : 7
The best solution obtained by GOA is : 3  1  1  2  1  2
2
The best optimal value of the objective funciton found
by GOA is : 7
>>
```