حل مسئله طراحی چیدمان با الگوریتم ازدحام ذرات PSO در متلب

مسئله چیدمان عبارت است از ایجاد یک نظم و ترتیب در هر آنچه که برای تولید کالا یا ایجاد یک سرویس نیاز است. یک تسهیل عبارت است از آن چه که به انجام یک وظیفه (Task) کمک می کند. این تسهیل ممکن است یک ماشین، یک مرکز کار، یک سلول تولیدی، یک کارگاه از ماشین آلات، یک دپارتمان، یک انبار و غیره باشد

صورت مسئله:

۸ تا جعبه مستطیلی با ابعاد مشخص و در موقعیت مشخص داریم که می خواهیم آن ها را در یک فضای مشخص به گونه ای بچینیم که ضایعات ناشی از برش با استفاده از بهینه چینی قطعات کاهش یابد.

```
w=[23 24 12 24 20 11 14 18]     عرض جعبه ها
h=[25 25 12 25 25 17 22 12]     طول جعبه ها
```

شرح کد:

این سورس کد شامل ۹ فایل می باشد که عبارتند از:

CreateModel.m: برای ایجاد مدل(همان جعبه ها) و تعین موقعیت آن ها و مقدار دهی اولیه پارامتر های مدل مسیله از آن استفاده می شود.

```
function model=CreateModel()

                                                    سایز بلوک ها
    w=[23 24 12 24 20 11 14 18];     عرض ها
    h=[25 25 12 25 25 17 22 12];     طول ها

    delta=[2 1 4 1 3 1 4 1];     شکاف

    rin  = [0.17 0.70 0.73 0.27 0.04 0.09 0.42 0.69];

                                                    محل ورودی Gate

    rout = [0.31 0.95 0.03 0.43 0.38 0.76 0.79 0.18];
                                                    محل خروجی Gate


    n=numel(w);
```

```
xin=zeros(1,n);
yin=zeros(1,n);

xout=zeros(1,n);
yout=zeros(1,n);

for i=1:n
    if rin(i)>=0 && rin(i)<=0.25
        xin(i)=(4*rin(i)-0.5)*w(i);
        yin(i)=-h(i)/2;

    elseif rin(i)>0.25 && rin(i)<=0.5
        xin(i)=w(i)/2;
        yin(i)=(4*rin(i)-1.5)*h(i);

    elseif rin(i)>0.5 && rin(i)<=0.75
        xin(i)=(2.5-4*rin(i))*w(i);
        yin(i)=h(i)/2;

    else
        xin(i)=-w(i)/2;
        yin(i)=(3.5-4*rin(i))*h(i);

    end

    if rout(i)>=0 && rout(i)<=0.25
        xout(i)=(4*rout(i)-0.5)*w(i);
        yout(i)=-h(i)/2;

    elseif rout(i)>0.25 && rout(i)<=0.5
        xout(i)=w(i)/2;
        yout(i)=(4*rout(i)-1.5)*h(i);

    elseif rout(i)>0.5 && rout(i)<=0.75
        xout(i)=(2.5-4*rout(i))*w(i);
        yout(i)=h(i)/2;

    else
        xout(i)=-w(i)/2;
        yout(i)=(3.5-4*rout(i))*h(i);

    end

end
```

```
    a=[ 0  50  45  20   0  19  46  15
       28   0  13  15  24  27  25  48
       13  28   0   0  31  12   0  49
        0  14  20   0  26  47  41  33
       47  49  42  33   0  48  25  12
       16  10  27  32  19   0  19   0
       43  41  47  15  15  30   0  24
       32   0  17  44  17  23  13   0];

    W=100;
    H=80;

    phi=50000;

    model.n=n;
    model.w=w;
    model.h=h;
    model.delta=delta;
    model.rin=rin;
    model.xin=xin;
    model.yin=yin;
    model.rout=rout;
    model.xout=xout;
    model.yout=yout;
    model.a=a;
    model.W=W;
    model.H=H;
    model.phi=phi;

end
```

CreateRandomSolution.m: ایجاد راه حل رندم

```
function sol1=CreateRandomSolution(model)

    n=model.n;
```

برداری شامل یک سطر و n ستون از اعداد رندم قرار میدهد که n را از مدل گرفته است.

```
    sol1.xhat=rand(1,n);
    sol1.yhat=rand(1,n);
    sol1.rhat=rand(1,n);

end
```

ImproveSolution.m:بهبود راه حل، حرکت جعبه ها به بالا و پایین و چپ و راست و بالا چپ و پایین چپ و بالا راست و پایین راست و چرخش جعبه ها در این مرحله انجام می شود.

```
function sol2=ImproveSolution(sol1,model,Vars)

    n=model.n;

    A=randperm(n);

    for i=A
        sol1=MoveMachine(i,sol1,model,Vars);
    end

    sol2=sol1;

end


function [sol2, z2]=MoveMachine(i,sol1,model,Vars)

    dmax=0.5;

                                                    صفر
    [newsol(1), z(1)]=RotateMachine(i,sol1,model);

                                                حرکت به بالا
    newsol(2)=sol1;
    dy=unifrnd(0,dmax);
    newsol(2).yhat(i)=sol1.yhat(i)+dy;
    newsol(2).yhat(i)=max(newsol(2).yhat(i),Vars.yhat.Min);
    newsol(2).yhat(i)=min(newsol(2).yhat(i),Vars.yhat.Max);
    [newsol(2), z(2)]=RotateMachine(i,newsol(2),model);

                                                حرکت به پایین
    newsol(3)=sol1;
    dy=unifrnd(0,dmax);
    newsol(3).yhat(i)=sol1.yhat(i)-dy;
    newsol(3).yhat(i)=max(newsol(3).yhat(i),Vars.yhat.Min);
    newsol(3).yhat(i)=min(newsol(3).yhat(i),Vars.yhat.Max);
    [newsol(3), z(3)]=RotateMachine(i,newsol(3),model);

                                                حرکت به راست
    newsol(4)=sol1;
    dx=unifrnd(0,dmax);
```

```
newsol(4).xhat(i)=sol1.xhat(i)+dx;
newsol(4).xhat(i)=max(newsol(4).xhat(i),Vars.xhat.Min);
newsol(4).xhat(i)=min(newsol(4).xhat(i),Vars.xhat.Max);
[newsol(4), z(4)]=RotateMachine(i,newsol(4),model);
```

حرکت به چپ

```
newsol(5)=sol1;
dx=unifrnd(0,dmax);
newsol(5).xhat(i)=sol1.xhat(i)-dx;
newsol(5).xhat(i)=max(newsol(5).xhat(i),Vars.xhat.Min);
newsol(5).xhat(i)=min(newsol(5).xhat(i),Vars.xhat.Max);
[newsol(5), z(5)]=RotateMachine(i,newsol(5),model);
```

حرکت به بالا و راست

```
newsol(6)=sol1;
dx=unifrnd(0,dmax);
newsol(6).xhat(i)=sol1.xhat(i)+dx;
newsol(6).xhat(i)=max(newsol(6).xhat(i),Vars.xhat.Min);
newsol(6).xhat(i)=min(newsol(6).xhat(i),Vars.xhat.Max);
dy=unifrnd(0,dmax);
newsol(6).yhat(i)=sol1.yhat(i)+dy;
newsol(6).yhat(i)=max(newsol(6).yhat(i),Vars.yhat.Min);
newsol(6).yhat(i)=min(newsol(6).yhat(i),Vars.yhat.Max);
[newsol(6), z(6)]=RotateMachine(i,newsol(6),model);
```

حرکت به پایین و چپ

```
newsol(7)=sol1;
dx=unifrnd(0,dmax);
newsol(7).xhat(i)=sol1.xhat(i)-dx;
newsol(7).xhat(i)=max(newsol(7).xhat(i),Vars.xhat.Min);
newsol(7).xhat(i)=min(newsol(7).xhat(i),Vars.xhat.Max);
dy=unifrnd(0,dmax);
newsol(7).yhat(i)=sol1.yhat(i)+dy;
newsol(7).yhat(i)=max(newsol(7).yhat(i),Vars.yhat.Min);
newsol(7).yhat(i)=min(newsol(7).yhat(i),Vars.yhat.Max);
[newsol(7), z(7)]=RotateMachine(i,newsol(7),model);
```

حرکت به راست و پایین

```
newsol(8)=sol1;
dx=unifrnd(0,dmax);
newsol(8).xhat(i)=sol1.xhat(i)+dx;
newsol(8).xhat(i)=max(newsol(8).xhat(i),Vars.xhat.Min);
newsol(8).xhat(i)=min(newsol(8).xhat(i),Vars.xhat.Max);
dy=unifrnd(0,dmax);
```

```
    newsol(8).yhat(i)=sol1.yhat(i)-dy;
    newsol(8).yhat(i)=max(newsol(8).yhat(i),Vars.yhat.Min);
    newsol(8).yhat(i)=min(newsol(8).yhat(i),Vars.yhat.Max);
    [newsol(8), z(8)]=RotateMachine(i,newsol(8),model);

                                                حرکت به چپ و پایین
    newsol(9)=sol1;
    dx=unifrnd(0,dmax);
    newsol(9).xhat(i)=sol1.xhat(i)-dx;
    newsol(9).xhat(i)=max(newsol(9).xhat(i),Vars.xhat.Min);
    newsol(9).xhat(i)=min(newsol(9).xhat(i),Vars.xhat.Max);
    dy=unifrnd(0,dmax);
    newsol(9).yhat(i)=sol1.yhat(i)-dy;
    newsol(9).yhat(i)=max(newsol(9).yhat(i),Vars.yhat.Min);
    newsol(9).yhat(i)=min(newsol(9).yhat(i),Vars.yhat.Max);
    [newsol(9), z(9)]=RotateMachine(i,newsol(9),model);

    [z2, ind]=min(z);

    sol2=newsol(ind);

end

function [sol2, z2]=RotateMachine(i,sol1,model)

    newsol(1)=sol1;
    newsol(1).rhat(i)=0.1;
    z(1)=MyCost(newsol(1),model);

    newsol(2)=sol1;
    newsol(2).rhat(i)=0.35;
    z(2)=MyCost(newsol(2),model);

    newsol(3)=sol1;
    newsol(3).rhat(i)=0.6;
    z(3)=MyCost(newsol(3),model);

    newsol(4)=sol1;
    newsol(4).rhat(i)=0.85;
    z(4)=MyCost(newsol(4),model);

    [z2, ind]=min(z);

    sol2=newsol(ind);
```

```
end
```

Mutate.m: مرحله جهش

```
function newsol=Mutate(sol,Vars)

    newsol=sol;

    sigma=0.1*(Vars.xhat.Max-Vars.xhat.Min);
    j=randi([1 Vars.xhat.Count]);
    dxhat=sigma*randn;
    newsol.xhat(j)=sol.xhat(j)+dxhat;
    newsol.xhat=max(newsol.xhat,Vars.xhat.Min);
    newsol.xhat=min(newsol.xhat,Vars.xhat.Max);

    sigma=0.1*(Vars.yhat.Max-Vars.yhat.Min);
    j=randi([1 Vars.yhat.Count]);
    dyhat=sigma*randn;
    newsol.yhat(j)=sol.yhat(j)+dyhat;
    newsol.yhat=max(newsol.yhat,Vars.yhat.Min);
    newsol.yhat=min(newsol.yhat,Vars.yhat.Max);

end
```

MyCost.m: محاسبه cost

```
function [z, sol]=MyCost(sol1,model)

    sol=ParseSolution(sol1,model);

    z=sol.z;

end
```

ParseSolution.m: پارس کردن راه حل

```
function sol2=ParseSolution(sol1,model)

    n=model.n;
    delta=model.delta;
    W=model.W;
    H=model.H;
    a=model.a;

    rhat=sol1.rhat;
    r=min(floor(4*rhat),3);
    theta=r*pi/2;
```

```
    w=abs(cos(theta)).*model.w+abs(sin(theta)).*model.h;
    h=abs(cos(theta)).*model.h+abs(sin(theta)).*model.w;

    xhat=sol1.xhat;
    xmin=w/2+delta;
    xmax=W-xmin;
    x=xmin+(xmax-xmin).*xhat;

    yhat=sol1.yhat;
    ymin=h/2+delta;
    ymay=H-ymin;
    y=ymin+(ymay-ymin).*yhat;

    xl=x-w/2;
    yl=y-h/2;
    xu=x+w/2;
    yu=y+h/2;

    xin=x+cos(theta).*model.xin-sin(theta).*model.yin;
    yin=y+cos(theta).*model.yin+sin(theta).*model.xin;
    xout=x+cos(theta).*model.xout-sin(theta).*model.yout;
    yout=y+cos(theta).*model.yout+sin(theta).*model.xout;

    d=zeros(n,n);
    V=zeros(n,n);
    for i=1:n
        for j=i+1:n
            d(i,j)=norm([xout(i)-xin(j) yout(i)-yin(j)],1);
            d(j,i)=norm([xout(j)-xin(i) yout(j)-yin(i)],1);

            DELTA=max(delta(i),delta(j));

            XVij=max(0,1-abs(x(i)-
x(j))/((w(i)+w(j))/2+DELTA));
            YVij=max(0,1-abs(y(i)-
y(j))/((h(i)+h(j))/2+DELTA));

            V(i,j)=min(XVij,YVij);
            V(j,i)=V(i,j);
        end
    end

    v=mean(V(:));
```

```
ad=a.*d;

SumAD=sum(ad(:));

XMIN=min(xl);
YMIN=min(yl);
XMAX=max(xu);
YMAX=max(yu);

ContainerArea=(XMAX-XMIN)*(YMAX-YMIN);
MachinesArea=sum(w.*h);
UnusedArea=max(ContainerArea-MachinesArea,0);
UnusedAreaRatio=UnusedArea/ContainerArea;
UnusedAreaCost=model.phi*UnusedAreaRatio;

%beta=100;
%z=SumAD*(1+beta*v);

alpha=1e12;
z=SumAD+UnusedAreaCost+alpha*v;

sol2.r=r;
sol2.theta=theta;
sol2.x=x;
sol2.y=y;
sol2.xl=xl;
sol2.yl=yl;
sol2.xu=xu;
sol2.yu=yu;
sol2.xin=xin;
sol2.yin=yin;
sol2.xout=xout;
sol2.yout=yout;
sol2.d=d;
sol2.ad=ad;
sol2.SumAD=SumAD;
sol2.XMIN=XMIN;
sol2.YMIN=YMIN;
sol2.XMAX=XMAX;
sol2.YMAX=YMAX;
sol2.ContainerArea=ContainerArea;
sol2.MachinesArea=MachinesArea;
sol2.UnusedArea=UnusedArea;
sol2.UnusedAreaRatio=UnusedAreaRatio;
```

```
    sol2.UnusedAreaCost=UnusedAreaCost;
    sol2.V=V;
    sol2.v=v;
    sol2.IsFeasible=(v==0);
    sol2.z=z;

end
```

PlotSolution.m: برای رسم راه حل از این تابع استفاده می شود.

```
Function PlotSolution(sol,model)

    n=model.n;

    x=sol.x;
    y=sol.y;
    xl=sol.xl;
    yl=sol.yl;
    xu=sol.xu;
    yu=sol.yu;

    xin=sol.xin;
    yin=sol.yin;
    xout=sol.xout;
    yout=sol.yout;

    XMIN=sol.XMIN;
    YMIN=sol.YMIN;
    XMAX=sol.XMAX;
    YMAX=sol.YMAX;

    Colors=hsv(n);

    for i=1:n
                                ایجاد رنگ های مختلف برای اشیا
        Color=Colors(i,:);
        White=[1 1 1];

        Color=0.4*Color+0.6*White;

        X=[xl(i) xu(i) xu(i) xl(i)];
        Y=[yl(i) yl(i) yu(i) yu(i)];
```

```
        fill(X,Y,Color);
        hold on;

        text(x(i),y(i),num2str(i),...
            'FontSize',20,...
            'FontWeight','bold',...
            'HorizontalAlignment','center',...
            'VerticalAlignment','middle');
    end

    plot(xin,yin,'ko','MarkerSize',8);

plot(xout,yout,'ko','MarkerSize',8,'MarkerFaceColor','black
');

    plot([XMIN XMAX XMAX XMIN XMIN],[YMIN YMIN YMAX YMAX
YMIN],'r-.');

    axis equal;
    grid on;

    xlim([0 model.W]);
    ylim([0 model.H]);

    hold off;

end
```

Pso.m:خود الگوریتم ازدحام ذرات که برای حل مسیله به کار می رود.

```
clc;
clear;
close all;

                                                تعریف مسیله
model=CreateModel();                            ایجاد مدل
CostFunction=@(sol1) MyCost(sol1,model);        تابع کاست

Vars.xhat.Min=0;
Vars.xhat.Max=1;
Vars.xhat.Size=[1 model.n];
Vars.xhat.Count=prod(Vars.xhat.Size);
Vars.xhat.VelMax=0.1*(Vars.xhat.Max-Vars.xhat.Min);
Vars.xhat.VelMin=-Vars.xhat.VelMax;

Vars.yhat.Min=0;
```

```
Vars.yhat.Max=1;
Vars.yhat.Size=[1 model.n];
Vars.yhat.Count=prod(Vars.yhat.Size);
Vars.yhat.VelMax=0.1*(Vars.yhat.Max-Vars.yhat.Min);
Vars.yhat.VelMin=-Vars.yhat.VelMax;

Vars.rhat.Min=0;
Vars.rhat.Max=1;
Vars.rhat.Size=[1 model.n];
Vars.rhat.Count=prod(Vars.rhat.Size);
Vars.rhat.VelMax=0.1*(Vars.rhat.Max-Vars.rhat.Min);
Vars.rhat.VelMin=-Vars.rhat.VelMax;
```

پارامترهای PSO

```
MaxIt=100;      حداکثر تعداد تکرار

nPop=50;        جمعیت (تعداد انبوه)

w=1.0;          وزن اینرسی
wdamp=0.99;     نسبت میرایی وزن اینرسی
c1=0.7;         ضریب یادگیری شخصی
c2=1.5;         ضریب یادگیریجمعی
```

مقدار دهی اولیه

```
empty_particle.Position=[];
empty_particle.Cost=[];
empty_particle.Sol=[];
empty_particle.Velocity=[];
empty_particle.Best.Position=[];
empty_particle.Best.Cost=[];
empty_particle.Best.Sol=[];

particle=repmat(empty_particle,nPop,1);

GlobalBest.Cost=inf;

for i=1:nPop
```

مکان اولیه

```
    particle(i).Position=CreateRandomSolution(model);
```

سرعت اولیه

```
    particle(i).Velocity.xhat=zeros(Vars.xhat.Size);
    particle(i).Velocity.yhat=zeros(Vars.yhat.Size);
    particle(i).Velocity.rhat=zeros(Vars.rhat.Size);

                                                    ارزیابی
    [particle(i).Cost,
particle(i).Sol]=CostFunction(particle(i).Position);

                                                    آپدیت بهترین شخصی
    particle(i).Best.Position=particle(i).Position;
    particle(i).Best.Cost=particle(i).Cost;
    particle(i).Best.Sol=particle(i).Sol;

                                                    آپدیت بهترین جمعی
    if particle(i).Best.Cost<GlobalBest.Cost
        GlobalBest=particle(i).Best;
    end
end

BestFitness=zeros(MaxIt,1);


                                                    حلقه اصلی pso

for it=1:MaxIt

    for i=1:nPop

                                                    حرکت در xhat

                                                    آپدیت سرعت
        particle(i).Velocity.xhat =
w*particle(i).Velocity.xhat ...

+c1*rand(Vars.xhat.Size).*(particle(i).Best.Position.xhat-
particle(i).Position.xhat) ...

+c2*rand(Vars.xhat.Size).*(GlobalBest.Position.xhat-
particle(i).Position.xhat);

                                                    محدودیت های سرعت را اعمال کنید
        particle(i).Velocity.xhat =
max(particle(i).Velocity.xhat,Vars.xhat.VelMin);
```

```
        particle(i).Velocity.xhat =
min(particle(i).Velocity.xhat,Vars.xhat.VelMax);
```

آپدیت مکان

```
        particle(i).Position.xhat =
particle(i).Position.xhat + particle(i).Velocity.xhat;
```

تاثیر سرعت آینه

```
        IsOutside=(particle(i).Position.xhat<Vars.xhat.Min
| particle(i).Position.xhat>Vars.xhat.Max);
        particle(i).Velocity.xhat(IsOutside)=-
particle(i).Velocity.xhat(IsOutside);
```

اعمال محدودیت های موقعیت

```
        particle(i).Position.xhat =
max(particle(i).Position.xhat,Vars.xhat.Min);
        particle(i).Position.xhat =
min(particle(i).Position.xhat,Vars.xhat.Max);
```

حرکت در yhat

آپدیت سرعت

```
        particle(i).Velocity.yhat =
w*particle(i).Velocity.yhat ...

+c1*rand(Vars.yhat.Size).*(particle(i).Best.Position.yhat-
particle(i).Position.yhat) ...

+c2*rand(Vars.yhat.Size).*(GlobalBest.Position.yhat-
particle(i).Position.yhat);
```

محدودیت های سرعت را اعمال کنید

```
particle(i).Velocity.yhat =
max(particle(i).Velocity.yhat,Vars.yhat.VelMin);
        particle(i).Velocity.yhat =
min(particle(i).Velocity.yhat,Vars.yhat.VelMax);
```

آپدیت مکان

```
        particle(i).Position.yhat =
particle(i).Position.yhat + particle(i).Velocity.yhat;
```

تاثیر سرعت آینه

```
        IsOutside=(particle(i).Position.yhat<Vars.yhat.Min
| particle(i).Position.yhat>Vars.yhat.Max);
        particle(i).Velocity.yhat(IsOutside)=-
particle(i).Velocity.yhat(IsOutside);
```

اعمال محدودیت های موقعیت

```
particle(i).Position.yhat =
max(particle(i).Position.yhat,Vars.yhat.Min);
        particle(i).Position.yhat =
min(particle(i).Position.yhat,Vars.yhat.Max);
```

حرکت روی rhat
آپدیت سرعت

```
 particle(i).Velocity.rhat = w*particle(i).Velocity.rhat
...

+c1*rand(Vars.rhat.Size).*(particle(i).Best.Position.rhat-
particle(i).Position.rhat) ...

+c2*rand(Vars.rhat.Size).*(GlobalBest.Position.rhat-
particle(i).Position.rhat);
```

افزودن حدود سرعت

```
        particle(i).Velocity.rhat =
max(particle(i).Velocity.rhat,Vars.rhat.VelMin);
        particle(i).Velocity.rhat =
min(particle(i).Velocity.rhat,Vars.rhat.VelMax);
```

آپدیت موقعیت

```
        particle(i).Position.rhat =
particle(i).Position.rhat + particle(i).Velocity.rhat;
```

تاثیر آینه ای سرعت

```
        IsOutside=(particle(i).Position.rhat<Vars.rhat.Min
| particle(i).Position.rhat>Vars.rhat.Max);
        particle(i).Velocity.rhat(IsOutside)=-
particle(i).Velocity.rhat(IsOutside);
```

اعمال محدودیت های موقعیت

```
        particle(i).Position.rhat =
max(particle(i).Position.rhat,Vars.rhat.Min);
```

```
        particle(i).Position.rhat =
min(particle(i).Position.rhat,Vars.rhat.Max);

                                                    ارزیابی
        [particle(i).Cost, particle(i).Sol] =
CostFunction(particle(i).Position);

                                                افزودن جهش
        NewParticle=particle(i);
        NewParticle.Position = Mutate(particle(i).Position,
Vars);
        [NewParticle.Cost,
NewParticle.Sol]=CostFunction(NewParticle.Position);
        if NewParticle.Cost<=particle(i).Cost || rand < 0.2
            particle(i)=NewParticle;
        end

                                                آپدیت بهترین شخصی
        if particle(i).Cost<particle(i).Best.Cost

            particle(i).Best.Position=particle(i).Position;
            particle(i).Best.Cost=particle(i).Cost;
            particle(i).Best.Sol=particle(i).Sol;

                                                آپدیت بهترین جمعی
            if particle(i).Best.Cost<GlobalBest.Cost
                GlobalBest=particle(i).Best;
            end
        end

    end

  جستجوی محلی (بهبود) را در بهترین موارد جهانی اعمال می کند.
    NewParticle=GlobalBest;

NewParticle.Position=ImproveSolution(GlobalBest.Position,mo
del,Vars);
    [NewParticle.Cost,
NewParticle.Sol]=CostFunction(NewParticle.Position);
    if NewParticle.Cost<=GlobalBest.Cost
        GlobalBest=NewParticle;
    end

    BestFitness(it)=1/(1+GlobalBest.Cost);محاسبه فیتنس
```

```
        if GlobalBest.Sol.IsFeasible
            FLAG=' (Feasible)';
        else
            FLAG='';
        end
        disp(['Iteration ' num2str(it) ': Best Fitness = '
num2str(BestFitness (it)) FLAG]);

        w=w*wdamp;

                                                    رسم راه حل
        figure(1);
        PlotSolution(GlobalBest.Sol,model);
        pause(0.01);

end

BestSol = GlobalBest;

                                                    نتایج

figure;
plot(BestFitness,'LineWidth',2);
xlabel('Iteration');
ylabel('Best Fitness');
grid on;
```

نتایج:

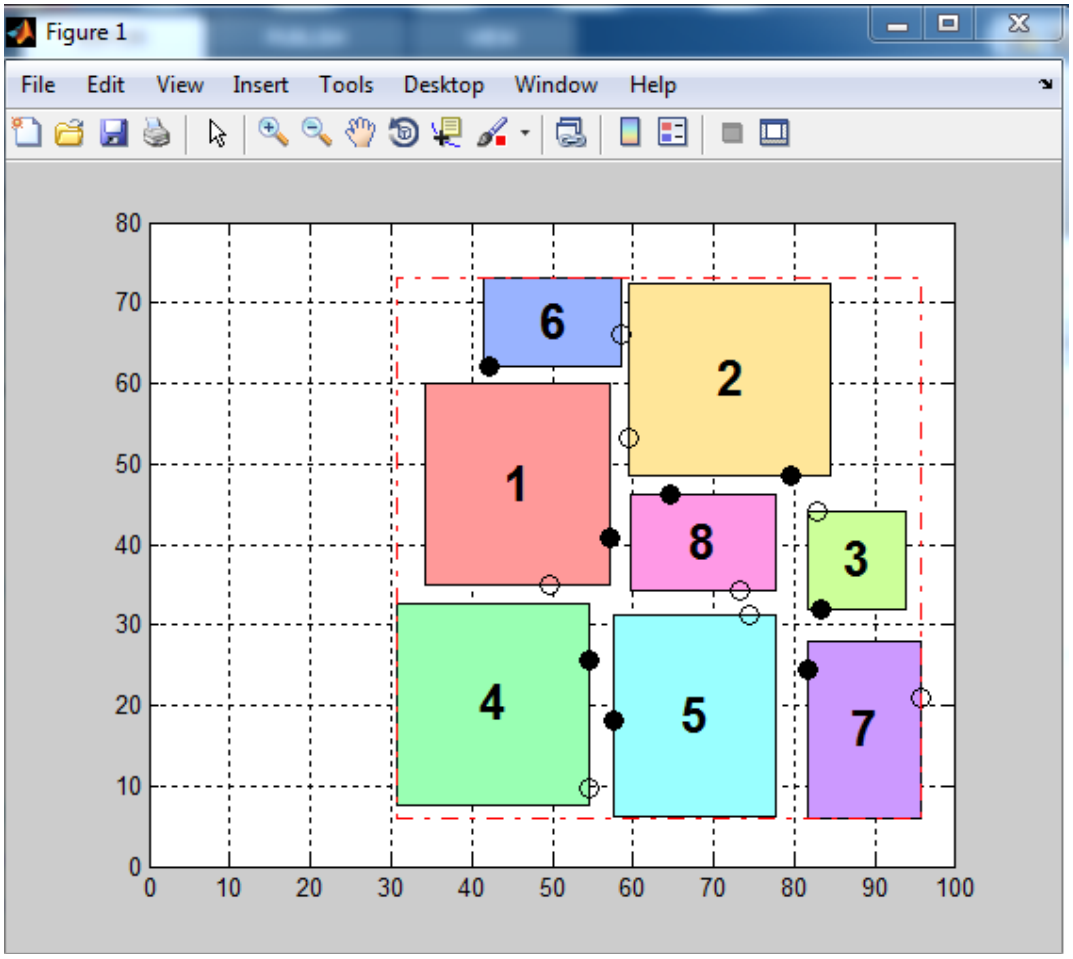
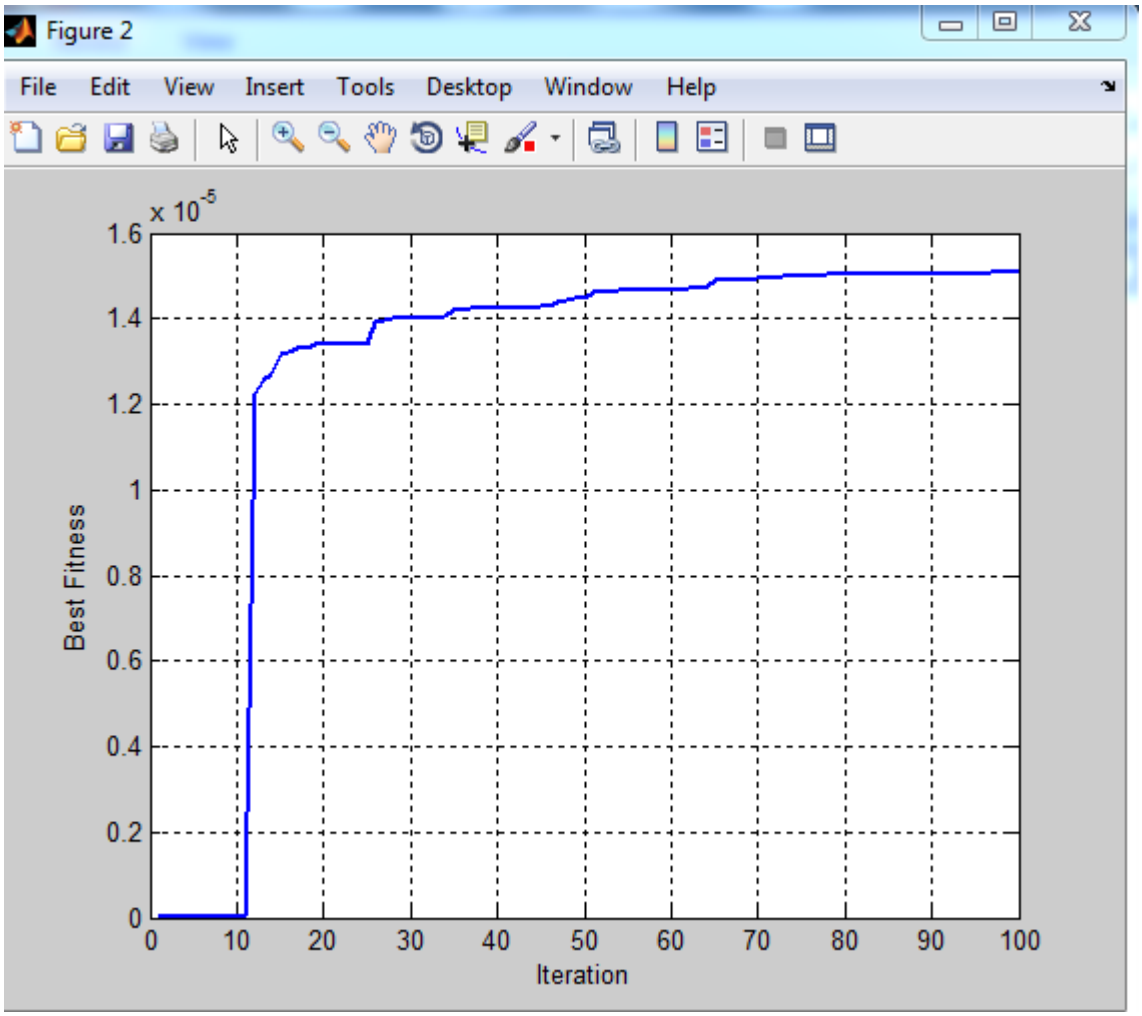


Figure 1
File Edit View Insert Tools Desktop Window Help
6
2
1
8
3
4
5
7
0 10 20 30 40 50 60 70 80 90 100
0 10 20 30 40 50 60 70 80




```
Iteration 1: Best Fitness = 1.6538e-10
Iteration 2: Best Fitness = 1.7018e-10
Iteration 3: Best Fitness = 1.9106e-10
Iteration 4: Best Fitness = 1.915e-10
Iteration 5: Best Fitness = 1.915e-10
Iteration 6: Best Fitness = 1.915e-10
Iteration 7: Best Fitness = 1.915e-10
Iteration 8: Best Fitness = 2.0319e-10
Iteration 9: Best Fitness = 2.0319e-10
Iteration 10: Best Fitness = 1.2634e-09
Iteration 11: Best Fitness = 1.2634e-09
Iteration 12: Best Fitness = 1.2232e-05 (Feasible)
Iteration 13: Best Fitness = 1.2551e-05 (Feasible)
Iteration 14: Best Fitness = 1.2718e-05 (Feasible)
Iteration 15: Best Fitness = 1.3177e-05 (Feasible)
```

```
Iteration 16: Best Fitness = 1.322e-05 (Feasible)
Iteration 17: Best Fitness = 1.3311e-05 (Feasible)
Iteration 18: Best Fitness = 1.3311e-05 (Feasible)
Iteration 19: Best Fitness = 1.3409e-05 (Feasible)
Iteration 20: Best Fitness = 1.3412e-05 (Feasible)
Iteration 21: Best Fitness = 1.3412e-05 (Feasible)
Iteration 22: Best Fitness = 1.3423e-05 (Feasible)
Iteration 23: Best Fitness = 1.3423e-05 (Feasible)
Iteration 24: Best Fitness = 1.3423e-05 (Feasible)
Iteration 25: Best Fitness = 1.3423e-05 (Feasible)
Iteration 26: Best Fitness = 1.3911e-05 (Feasible)
Iteration 27: Best Fitness = 1.3944e-05 (Feasible)
Iteration 28: Best Fitness = 1.4031e-05 (Feasible)
Iteration 29: Best Fitness = 1.4031e-05 (Feasible)
Iteration 30: Best Fitness = 1.4031e-05 (Feasible)
Iteration 31: Best Fitness = 1.4034e-05 (Feasible)
Iteration 32: Best Fitness = 1.4034e-05 (Feasible)
Iteration 33: Best Fitness = 1.4034e-05 (Feasible)
Iteration 34: Best Fitness = 1.4073e-05 (Feasible)
Iteration 35: Best Fitness = 1.4221e-05 (Feasible)
Iteration 36: Best Fitness = 1.4221e-05 (Feasible)
Iteration 37: Best Fitness = 1.4248e-05 (Feasible)
Iteration 38: Best Fitness = 1.4248e-05 (Feasible)
Iteration 39: Best Fitness = 1.4248e-05 (Feasible)
Iteration 40: Best Fitness = 1.4248e-05 (Feasible)
Iteration 41: Best Fitness = 1.4248e-05 (Feasible)
Iteration 42: Best Fitness = 1.4248e-05 (Feasible)
Iteration 43: Best Fitness = 1.4248e-05 (Feasible)
Iteration 44: Best Fitness = 1.4248e-05 (Feasible)
Iteration 45: Best Fitness = 1.4296e-05 (Feasible)
Iteration 46: Best Fitness = 1.4296e-05 (Feasible)
Iteration 47: Best Fitness = 1.4392e-05 (Feasible)
Iteration 48: Best Fitness = 1.4438e-05 (Feasible)
Iteration 49: Best Fitness = 1.4496e-05 (Feasible)
Iteration 50: Best Fitness = 1.4497e-05 (Feasible)
Iteration 51: Best Fitness = 1.4611e-05 (Feasible)
Iteration 52: Best Fitness = 1.4611e-05 (Feasible)
Iteration 53: Best Fitness = 1.4614e-05 (Feasible)
Iteration 54: Best Fitness = 1.4654e-05 (Feasible)
Iteration 55: Best Fitness = 1.4654e-05 (Feasible)
```

```
Iteration 56: Best Fitness = 1.4654e-05 (Feasible)
Iteration 57: Best Fitness = 1.4654e-05 (Feasible)
Iteration 58: Best Fitness = 1.4654e-05 (Feasible)
Iteration 59: Best Fitness = 1.4682e-05 (Feasible)
Iteration 60: Best Fitness = 1.4682e-05 (Feasible)
Iteration 61: Best Fitness = 1.4683e-05 (Feasible)
Iteration 62: Best Fitness = 1.4699e-05 (Feasible)
Iteration 63: Best Fitness = 1.4708e-05 (Feasible)
Iteration 64: Best Fitness = 1.4727e-05 (Feasible)
Iteration 65: Best Fitness = 1.492e-05 (Feasible)
Iteration 66: Best Fitness = 1.492e-05 (Feasible)
Iteration 67: Best Fitness = 1.492e-05 (Feasible)
Iteration 68: Best Fitness = 1.492e-05 (Feasible)
Iteration 69: Best Fitness = 1.492e-05 (Feasible)
Iteration 70: Best Fitness = 1.493e-05 (Feasible)
Iteration 71: Best Fitness = 1.4949e-05 (Feasible)
Iteration 72: Best Fitness = 1.4958e-05 (Feasible)
Iteration 73: Best Fitness = 1.4999e-05 (Feasible)
Iteration 74: Best Fitness = 1.4999e-05 (Feasible)
Iteration 75: Best Fitness = 1.4999e-05 (Feasible)
Iteration 76: Best Fitness = 1.4999e-05 (Feasible)
Iteration 77: Best Fitness = 1.4999e-05 (Feasible)
Iteration 78: Best Fitness = 1.5023e-05 (Feasible)
Iteration 79: Best Fitness = 1.5023e-05 (Feasible)
Iteration 80: Best Fitness = 1.5023e-05 (Feasible)
Iteration 81: Best Fitness = 1.5023e-05 (Feasible)
Iteration 82: Best Fitness = 1.5023e-05 (Feasible)
Iteration 83: Best Fitness = 1.5023e-05 (Feasible)
Iteration 84: Best Fitness = 1.5023e-05 (Feasible)
Iteration 85: Best Fitness = 1.5023e-05 (Feasible)
Iteration 86: Best Fitness = 1.5031e-05 (Feasible)
Iteration 87: Best Fitness = 1.5031e-05 (Feasible)
Iteration 88: Best Fitness = 1.5032e-05 (Feasible)
Iteration 89: Best Fitness = 1.5034e-05 (Feasible)
Iteration 90: Best Fitness = 1.5035e-05 (Feasible)
Iteration 91: Best Fitness = 1.5038e-05 (Feasible)
Iteration 92: Best Fitness = 1.5038e-05 (Feasible)
Iteration 93: Best Fitness = 1.5039e-05 (Feasible)
Iteration 94: Best Fitness = 1.5039e-05 (Feasible)
Iteration 95: Best Fitness = 1.5039e-05 (Feasible)
```

```
Iteration 96: Best Fitness = 1.5056e-05 (Feasible)
Iteration 97: Best Fitness = 1.5065e-05 (Feasible)
Iteration 98: Best Fitness = 1.507e-05 (Feasible)
Iteration 99: Best Fitness = 1.5074e-05 (Feasible)
Iteration 100: Best Fitness = 1.5079e-05 (Feasible)
>>
```