حل مسئله فروشنده دوره گرد TSP با الگوریتم گرگ خاکستری GWO در متلب

صورت مسئله:

یک فروشنده دوره گرد تمامی شهرها (اینجا ۲۳ شهر و هزینه پیمایش بین هر جفت از آنها مشخص است) را به گونه ای ملاقات کند که هر یک از این شهرها را فقط یک بارملاقات کرده باشد و دوباره به شهر آغازین برگردد با این شرط که با کمترین هزینه پیمایش این کار را انجام دهد.

به طور کلی هدف پیدا کردن کم هزینه ترین تور برای ملاقات همه شهرها و بازگشت به شهر آغازین حرکت است.

شرح کد:

این سورس کد شامل ۵ فایل می باشد که عبارتند از:

CreateModel.m: برای ایجاد فاصله شهرها و مختصات هر یک از شهرها از این تابع استفاده می شود.

```
function model=CreateModel()

    x=[70 10 12 66 3 85 94 94 63 9 28 55 96 97 15 98 96 49
80 12 80 55];

    y=[96 49 80 12 80 15 42 92 80 96 66 3 85 94 68 76 75 39
66 17 78 80 45];


    n=numel(x);

    d=zeros(n,n);

    for i=1:n-1
        for j=i+1:n
            d(i,j)=sqrt((x(i)-x(j))^2+(y(i)-y(j))^2);
            d(j,i)=d(i,j);
        end
    end

    xmin=0;
    xmax=100;
```

```
    ymin=0;
    ymax=100;

    model.n=n;
    model.x=x;
    model.y=y;
    model.d=d;
    model.xmin=xmin;
    model.xmax=xmax;
    model.ymin=ymin;
    model.ymax=ymax;

end
```

PlotSolution.m: برای رسم مسیر های بین شهرها از این تابع استفاده می شود.

```
function PlotSolution(sol,model)

    xmin=model.xmin;
    xmax=model.xmax;
    ymin=model.ymin;
    ymax=model.ymax;

    tour=sol;

    tour=[tour tour(1)];

    plot(model.x(tour),model.y(tour),'k-o',...
        'MarkerSize',14,...
        'MarkerFaceColor','r',...
        'LineWidth',1.5);


    xlabel('x');
    ylabel('y');

    axis equal;
    grid on;

    alpha = 0.1;

    dx = xmax - xmin;
    xmin = floor((xmin - alpha*dx)/10)*10;
```

```
    xmax = ceil((xmax + alpha*dx)/10)*10;
    xlim([xmin xmax]);

    dy = ymax - ymin;
    ymin = floor((ymin - alpha*dy)/10)*10;
    ymax = ceil((ymax + alpha*dy)/10)*10;
    ylim([ymin ymax]);

end
```

TourLength.m: این تابع برای محاسبه طول مسیرهای یک تور یا یک پیمایش کامل شهرها بکار می رود.

```
function L=TourLength(tour,model)

    n=numel(tour);

    tour=[tour tour(1)];

    L=0;

    for k=1:n

        i=tour(k);
        j=tour(k+1);

        L=L+model.d(i,j);

    end

end
```

initialization.m: این تابع برای محاسبه مقدار اولیه جواب های ممکن برای الگوریتم بکار می رود.

```
                                    این تابع جمعیت اولیه گرگ ها را آغاز می کند
function
Positions=initialization(SearchAgents_no,dim,ub,lb)

Boundary_no= size(ub,2);
                                                    تعداد مرزها


                    اگر مرزهای همه متغیرها مساوی هستند کاربر یک عدد برای حد بالا و پایین وارد کند
if Boundary_no==1
```

```
    Positions=rand(SearchAgents_no,dim).*(ub-lb)+lb;
end

                                اگر هر متغیر lb و ub متفاوت داشته باشد
if Boundary_no>1
    for i=1:dim
        ub_i=ub(i);
        lb_i=lb(i);
        Positions(:,i)=rand(SearchAgents_no,1).*(ub_i-
lb_i)+lb_i;
    end
end
```

TSPGWO.m: فایل اصلی برنامه است و فراخوانی دیگر توابع و مقادیر پارامتر ها و الگوریتم GWO در داخل این فایل قرار دارد.

برای ۱۰۰ تکرار الگوریتم را اجرا می کنیم.

```
clc;
clear;
close all;

                                                تنظیمات مسیله

model=CreateModel();
                                                ایجاد مدل مسیله

CostFunction=@(tour) TourLength(tour,model);


nVar=model.n;
                                        تعداد متغیرهای ناشناخته (تصمیم)
VarSize=[1 nVar];

                                        اندازه ماتریس متغیرهای تصمیم گیری
SearchAgents_no=50;
Max_iter=100;
lb=-5 ;
                                                کران پایین
ub=5 ;
                                                کران بالا
dim=nVar;
```

مقداردهی اولیه گرگ آلفا ، بتا و delta_pos را انجام دهید

```
Alpha_pos=zeros(1,dim);
Alpha_score=inf;
```

این را برای به حداکثر رساندن به -inf تغییر دهید

```
Beta_pos=zeros(1,dim);
Beta_score=inf;
```

برای مسایل حداکثر این را به -inf تغییر دهید

```
Delta_pos=zeros(1,dim);
Delta_score=inf;
```

برای مسایل حداکثر این را به -inf تغییر دهید

موقعیت های اولیه گرگ ها را شروع کنید

```
Positions=initialization(SearchAgents_no,dim,ub,lb);

Convergence_curve=zeros(1,Max_iter);

l=0;
```

شمارنده حلقه

```
while l<Max_iter
    for i=1:size(Positions,1)
```

گرگ ها را که فراتر از مرزهای فضای جستجو هستند برگردانید

```
        Flag4ub=Positions(i,:)>ub;
        Flag4lb=Positions(i,:)<lb;

Positions(i,:)=(Positions(i,:).*(~(Flag4ub+Flag4lb)))+ub.*F
lag4ub+lb.*Flag4lb;
        [~,sol]=sort(Positions,2);
```

عملکرد هدف را برای هر گرگ محاسبه کنید

```
        fitness=CostFunction(sol(i,:));
```

گرگ آلفا و بتا و دلتا را آپدیت کن

```
        if fitness<Alpha_score
```

گرگ آلفا آپدیت کن

```
            Alpha_score=fitness;
            Alpha_pos=Positions(i,:);
        end
```

```
        if fitness>Alpha_score && fitness<Beta_score
                                                گرگ بتا را آپدیت کن

            Beta_score=fitness;
            Beta_pos=Positions(i,:);
        end

        if fitness>Alpha_score && fitness>Beta_score &&
fitness<Delta_score
                                                گرگ دلتا را آپدیت کن

            Delta_score=fitness;
            Delta_pos=Positions(i,:);
        end
    end

                                    a به صورت خطی از 2 به 0 کاهش می یابد


    a=2-l*((2)/Max_iter);
                                        موقعیت گرگ امگا را به روز کنید
    for i=1:size(Positions,1)
        for j=1:size(Positions,2)

            r1=rand();
                                        r1 یک عدد تصادفی در [0،1] است
            r2=rand();
                                        r2 یک عدد تصادفی در [0،1] است


            A1=2*a*r1-a;
                                                        معادله (3.3)
            C1=2*r2;
                                                        معادله (3.4)

            D_alpha=abs(C1*Alpha_pos(j)-Positions(i,j));
                                                    معادله (3.5)بخش ا
            X1=Alpha_pos(j)-A1*D_alpha;
                                                معادله (3.6)بخش 1
            r1=rand();
            r2=rand();

            A2=2*a*r1-a;
                                                        معادله (3.3)
```

```
            C2=2*r2;
```
معادله (3.4)

```
            D_beta=abs(C2*Beta_pos(j)-Positions(i,j));
```
معادله (3.5) بخش 2

```
            X2=Beta_pos(j)-A2*D_beta;
```
معادله (3.6) بخش 2

```
            r1=rand();
            r2=rand();

            A3=2*a*r1-a;
```
معادله (3.3)

```
            C3=2*r2;
```
معادله (3.4)

```
            D_delta=abs(C3*Delta_pos(j)-Positions(i,j));
```
معادله (3.5) بخش 3

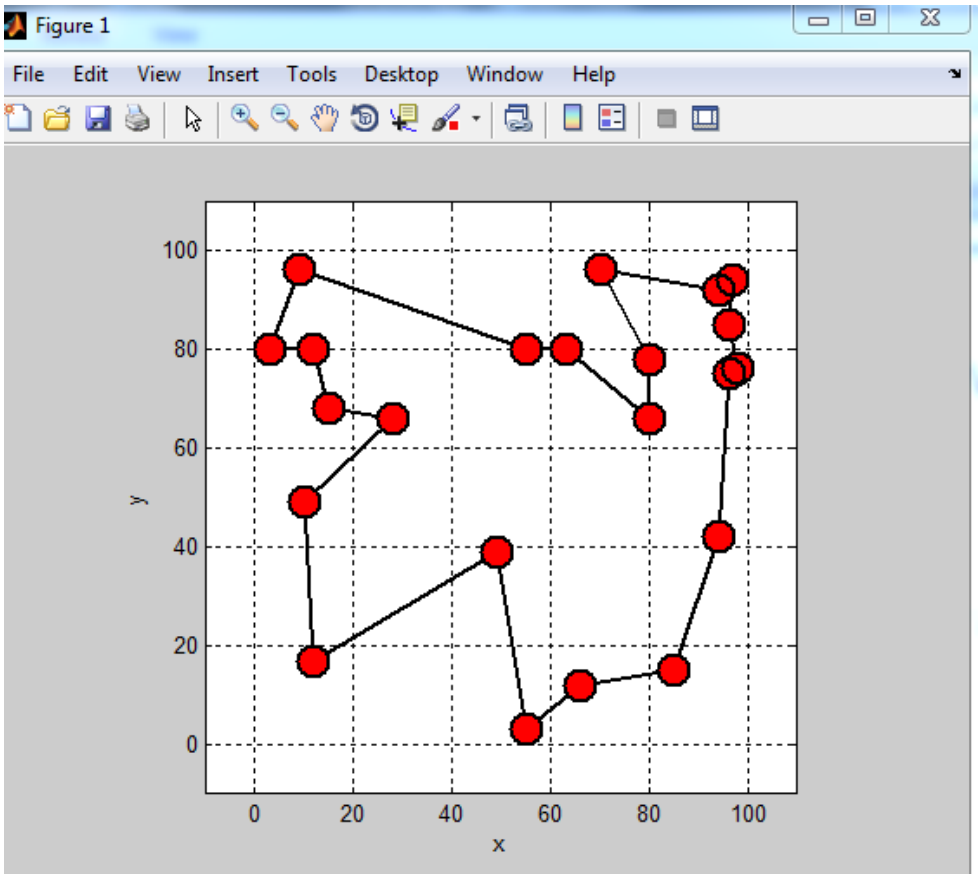
```
            X3=Delta_pos(j)-A3*D_delta;
```
معادله (3.5) بخش 3

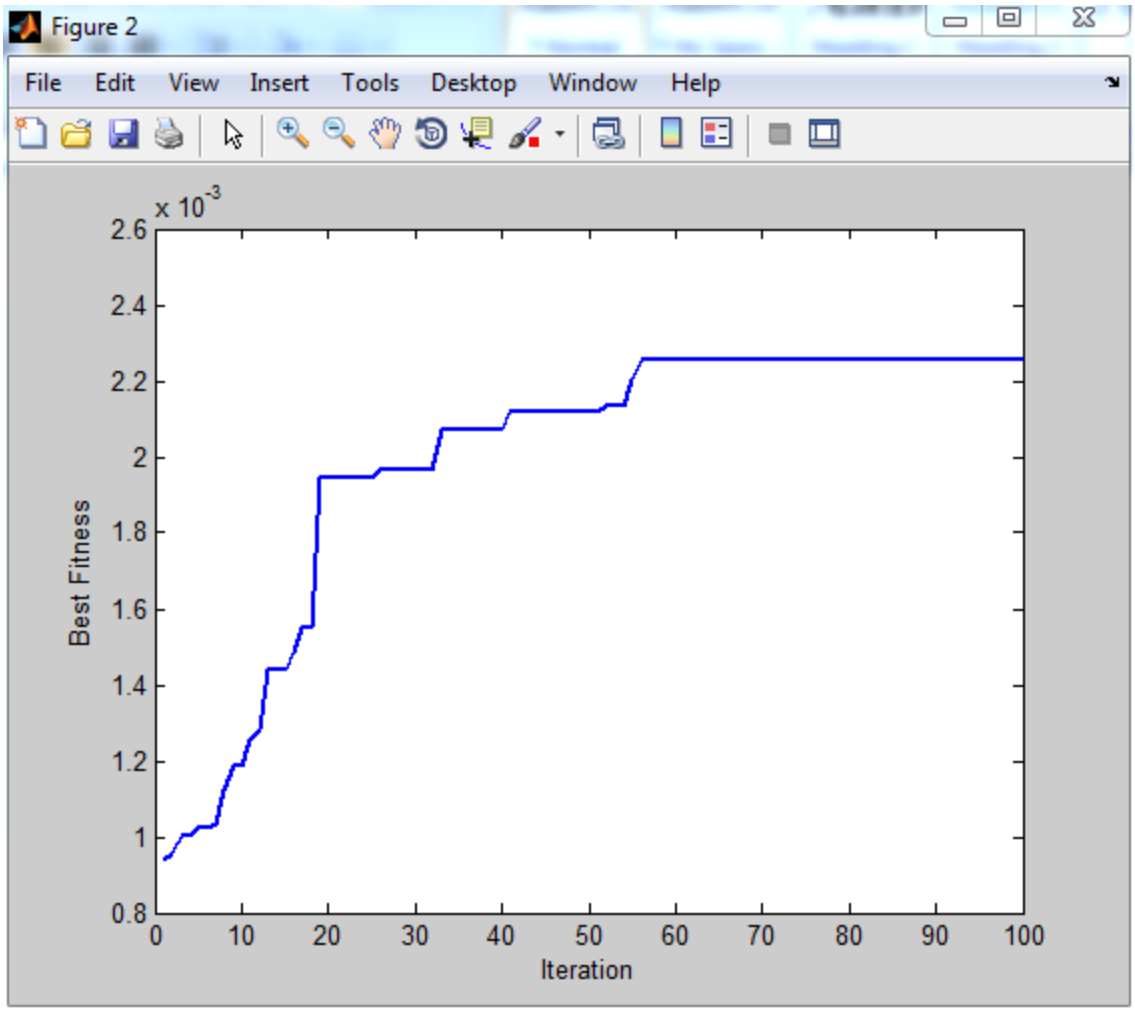
```
            Positions(i,j)=(X1+X2+X3)/3;
```
معادله (3.7)

```
        end
    end
    l=l+1;
    Convergence_curve(l)= 1/(Alpha_score+1);
    disp(['Iteration ' num2str(l) ': Best Fitness= '
num2str(Alpha_score)]);

    [~,Bsol]=sort(Alpha_pos);
    figure(1);
    PlotSolution(Bsol,model);
    pause(0.01);
end

figure(2);
plot(Convergence_curve,'LineWidth',2);
```

```
xlabel('Iteration');
ylabel('Best Fitness');
```

نتایج کد:





بعد از ۱۰۰ دور تکرار نتایج حاصل می شود.

Iteration 1: Best Fitness = 0.00093969

Iteration 2: Best Fitness = 0.00095483

Iteration 3: Best Fitness = 0.0010006

Iteration 4: Best Fitness = 0.0010006

Iteration 5: Best Fitness = 0.0010236

Iteration 6: Best Fitness = 0.0010236

Iteration 7: Best Fitness = 0.0010348

Iteration 8: Best Fitness = 0.001123

Iteration 9: Best Fitness = 0.0011862

Iteration 10: Best Fitness = 0.0011862

Iteration 11: Best Fitness = 0.001255

Iteration 12: Best Fitness = 0.0012833

Iteration 13: Best Fitness = 0.0014369

Iteration 14: Best Fitness = 0.0014369

Iteration 15: Best Fitness = 0.0014369

Iteration 16: Best Fitness = 0.001489

Iteration 17: Best Fitness = 0.0015491

Iteration 18: Best Fitness = 0.0015491

Iteration 19: Best Fitness = 0.0019433

Iteration 20: Best Fitness = 0.0019433

Iteration 21: Best Fitness = 0.0019433

Iteration 22: Best Fitness = 0.0019433

Iteration 23: Best Fitness = 0.0019433

Iteration 24: Best Fitness = 0.0019433

Iteration 25: Best Fitness = 0.0019433

Iteration 26: Best Fitness = 0.0019683

Iteration 27: Best Fitness = 0.0019683

Iteration 28: Best Fitness = 0.0019683

Iteration 29: Best Fitness = 0.0019683

Iteration 30: Best Fitness = 0.0019683

Iteration 31: Best Fitness = 0.0019683

Iteration 32: Best Fitness = 0.0019683

Iteration 33: Best Fitness = 0.0020697

Iteration 34: Best Fitness = 0.0020697

Iteration 35: Best Fitness = 0.0020697

Iteration 36: Best Fitness = 0.0020697

Iteration 37: Best Fitness = 0.0020697

Iteration 38: Best Fitness = 0.0020697

Iteration 39: Best Fitness = 0.0020697

Iteration 40: Best Fitness = 0.0020697

Iteration 41: Best Fitness = 0.002117

Iteration 42: Best Fitness = 0.002117

Iteration 43: Best Fitness = 0.002117

Iteration 44: Best Fitness = 0.002117

Iteration 45: Best Fitness = 0.002117

Iteration 46: Best Fitness = 0.002117

Iteration 47: Best Fitness = 0.002117

Iteration 48: Best Fitness = 0.002117

Iteration 49: Best Fitness = 0.002117

Iteration 50: Best Fitness = 0.002117

Iteration 51: Best Fitness = 0.002117

Iteration 52: Best Fitness = 0.0021341

Iteration 53: Best Fitness = 0.0021341

Iteration 54: Best Fitness = 0.0021341

Iteration 55: Best Fitness = 0.0022013

Iteration 56: Best Fitness = 0.0022537

Iteration 57: Best Fitness = 0.0022537

Iteration 58: Best Fitness = 0.0022537

Iteration 59: Best Fitness = 0.0022537

Iteration 60: Best Fitness = 0.0022537

Iteration 61: Best Fitness = 0.0022537

Iteration 62: Best Fitness = 0.0022537

Iteration 63: Best Fitness = 0.0022537

Iteration 64: Best Fitness = 0.0022537

Iteration 65: Best Fitness = 0.0022537

Iteration 66: Best Fitness = 0.0022537

Iteration 67: Best Fitness = 0.0022537

Iteration 68: Best Fitness = 0.0022537

Iteration 69: Best Fitness = 0.0022537

Iteration 70: Best Fitness = 0.0022537

Iteration 71: Best Fitness = 0.0022537

Iteration 72: Best Fitness = 0.0022537

Iteration 73: Best Fitness = 0.0022537

Iteration 74: Best Fitness = 0.0022537

Iteration 75: Best Fitness = 0.0022537

Iteration 76: Best Fitness = 0.0022537

Iteration 77: Best Fitness = 0.0022537

Iteration 78: Best Fitness = 0.0022537

Iteration 79: Best Fitness = 0.0022537

Iteration 80: Best Fitness = 0.0022537

Iteration 81: Best Fitness = 0.0022537

Iteration 82: Best Fitness = 0.0022537

Iteration 83: Best Fitness = 0.0022537

Iteration 84: Best Fitness = 0.0022537

Iteration 85: Best Fitness = 0.0022537

Iteration 86: Best Fitness = 0.0022537

Iteration 87: Best Fitness = 0.0022537

Iteration 88: Best Fitness = 0.0022537

Iteration 89: Best Fitness = 0.0022537

Iteration 90: Best Fitness = 0.0022537

Iteration 91: Best Fitness = 0.0022537

Iteration 92: Best Fitness = 0.0022537

Iteration 93: Best Fitness = 0.0022537

Iteration 94: Best Fitness = 0.0022537

Iteration 95: Best Fitness = 0.0022537

Iteration 96: Best Fitness = 0.0022537

Iteration 97: Best Fitness = 0.0022537

Iteration 98: Best Fitness = 0.0022537

Iteration 99: Best Fitness = 0.0022537

Iteration 100: Best Fitness = 0.0022537