به نام خدا

گزارش برنامه فروشنده دوره گرد توسط شبکه عصبی هاپفیلد:

مسئله به صورت بهینه سازی با تابع انرژی و مشتق آن به صورت زیر می باشد:

In this paper E is of the form

$$E = E_1 + E_2 \tag{5}$$

where

$$E_1 = \frac{A}{2}\sum_{x=1}^{n}\sum_{i=1}^{n}\sum_{\substack{j=1\\ j\neq i}}^{n} v_{xi}v_{xj} + \frac{B}{2}\sum_{i=1}^{n}\sum_{x=1}^{n}\sum_{\substack{y=1\\ y\neq x}}^{n} v_{xi}v_{yi} + \frac{C}{2}\Big(\sum_{x=1}^{n}\sum_{i=1}^{n} v_{xi} - (n+\sigma)\Big)^2 \tag{5.1}$$

and

$$E_2 = \frac{D}{2}\sum_{x=1}^{n}\sum_{\substack{y=1\\ y\neq x}}^{n}\sum_{i=1}^{n} d_{xy}v_{xi}(v_{y,i+1} + v_{y,i-1}) \tag{5.2}$$

$$\frac{du_{xi}}{dt} = -\frac{u_{xi}}{\tau} - A\sum_{\substack{j=1\\ j\neq i}}^{n} v_{xj} - B\sum_{\substack{y=1\\ y\neq x}}^{n} v_{yi} - C\Big(\sum_{x=1}^{n}\sum_{j=1}^{n} v_{xj} - (n+\sigma)\Big) - D\sum_{y=1}^{n} d_{xy}(v_{y,i+1} + v_{y,i-1}) \tag{6}$$

with $\tau = 1$.

```
%TSP Solving by Hopfield Neural Network
function TSP_hopfield()
clear ;
close all;
rand('state',1)
```

تعیین وزن مربوط به ترم های مختلف و میزان به روز رسانی ها

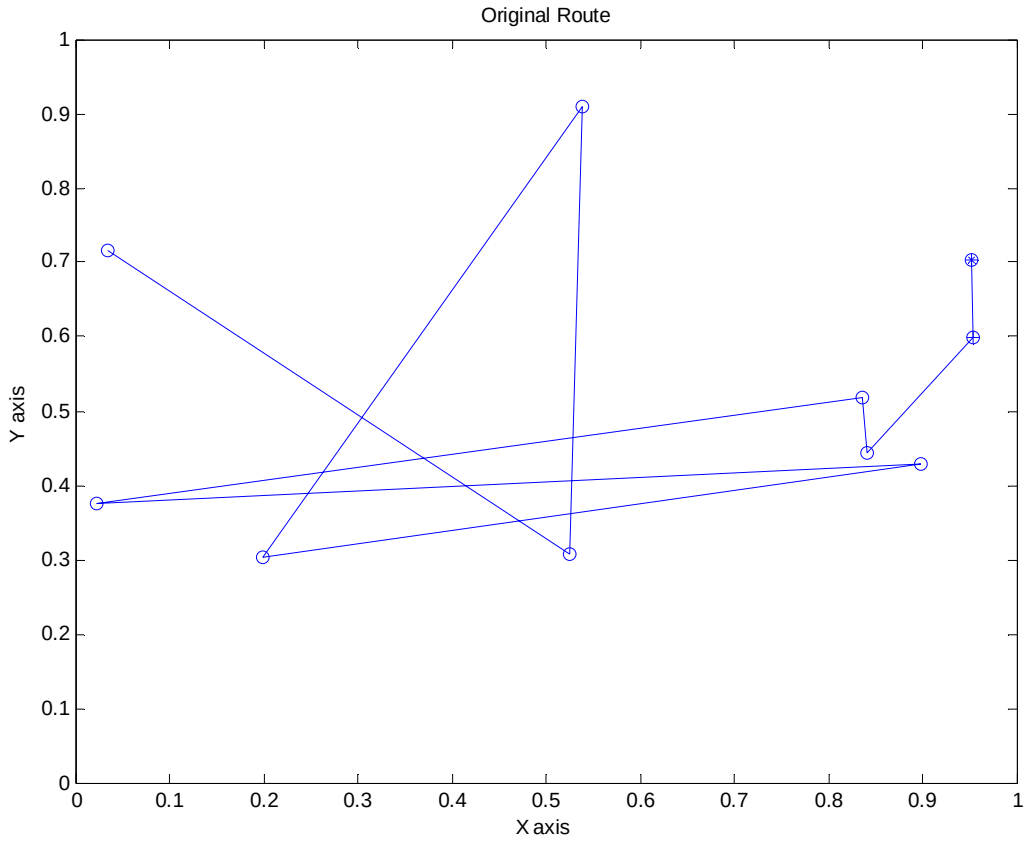
```
A=1.5;
D=1;
u0=0.01;
Step=0.01;
```

تعیین تعداد شهر ها و تولید نقشه آنها به صورت تصادفی

```
N=10;
citys=rand(2,N);
```

ترسیم نقشه شهرها و ترتیب حرکت در جواب اولیه

```
figure(1)
plot( citys(1,1), citys(2,1),'*' );   %First city
hold on;
plot( citys(1,2), citys(2,2),'+' );   %Second city
hold on;
plot( citys(1,:), citys(2,:),'o-' )
xlabel('X axis')
ylabel('Y axis')
title('Original Route');
axis([0,1,0,1]);
```



محاسبه فاصله شهر ها از یکدیگر

```
DistanceCity=dist(citys',citys);
```

محاسبه v و u اولیه

$$v(u) = 1/2\,[1 + \tanh(\alpha u)]$$

```
u=2*rand(N,N)-1;
```

```
U=0.5*u0*log(N-1)+u;
V=(1+tanh(U/u0))/2;
```

تعیین تعداد حلقه های مورد نیاز برای همگرایی

```
for k=1:1:10000
    times(k)=k;
```

محاسبه dU در هر مرحله و افزودن آن به U مرحله قبل توسط تابع DeltaU

```
 dU=DeltaU(V,DistanceCity,A,D);
    U=U+dU*Step;

    V=(1+tanh(U/u0))/2;
    %     V=double(U>0);
```

محاسبه انرژی در هر مرحله توسط تابع Energy

```
    E=Energy(V,DistanceCity,A,D);
    Ep(k)=E;

    if rem(k,10)==0
```

تابع RouteCheck چک میکند که آیا مسیر از تمامی شهر ها عبور کرده است یا خیر؟

```
 [V1,CheckR]=RouteCheck(V);
Final_Length(k)=Final_RouteLength(V1,citys);
figure(2)
citys=[citys citys(:,1)];

[xxx,order]=max(V);
New=citys(:,order);
New=[New New(:,1)];
```

ترسیم جواب محاسبه شده پس از هر 10 مرحله

```
figure(2)
plot( citys(1,:), citys(2,:),'o' ),
hold on
plot( New(1,1), New(2,1),'*' );   %First city
plot( New(1,2), New(2,2),'+' );   %Second city
plot(New(1,:),New(2,:),'o-');
xlabel('X axis');ylabel('Y axis');
axis([0,1,0,1]);
axis on
hold off
title(['TSP solution in iteration:',num2str(k)]);
        pause(0.001)
    end
end
figure(4)
plot(Final_Length)
```

ترسیم انرژی محاسبه شده در هر مرحله

```
figure(3);
semilogy(times,Ep,'r');
title('Energy Function Change');
xlabel('k');ylabel('E');

%%%%%%% Energy
function E=Energy(V,d,A,D)
```

```
[n,n]=size(V);
```
محاسبه انرژی چند ترم دارد. که باید بر روی تابع V که همان مسر می باشد عمل نماید. یک بار بر روی سطر ها و یکبار بر روی ستون ها عمل کرده و میزان اندازه بردار را محاسبه نماید. همچنین یک بار بر روی چرخش (جایگشت) مسیرها و ضرایب آنها کار نماید

این تابع در محاسبات تاثیری ندارد.

```
t1=sumsqr(sum(V,2)-1);
t2=sumsqr(sum(V,1)-1);
PermitV=V(:,2:n);
PermitV=[PermitV,V(:,1)];
temp=d*PermitV;
t3=sum(sum(V.*temp));
[V1,CheckR]=RouteCheck(V);

E=0.5*(A*t1+A*t2+D*t3+10*CheckR);
```

%%%%%%du

تابع DeltaU عملا مشتق تابع انرژی می باشد و تفسیر ترم های آن به صورت بیان شده در تابع energy می باشد.

تابع اصلی محاسبات این تابع می باشد.

```
function du=DeltaU(V,d,A,D)
[n,n]=size(V);
t1=repmat(sum(V,2)-1,1,n);
t2=repmat(sum(V,1)-1,n,1);
PermitV1=V(:,2:n);
PermitV1=[PermitV1, V(:,1)];
t3=d*PermitV1;
PermitV2=V(:,1:n-1);
PermitV2=[ V(:,n) PermitV2];
t4=d*PermitV2;
D1=0;
du=-1*(A*t1+A*t2+D*t3+D1*t4);
```

تابع RouteCheck چک میکند که آیا مسیر از تمامی شهر ها عبور کرده است یا خیر؟

برای این منظور ابتدا بیشترین مقدار به دست آمده در V را برای هر سطر محاسبه کرده وو معادل با یک قرار داده و سپس بقیه نقاط آن سطر را معادل با صفر قرار می دهد. این عملیات یک بار برای سطر و یکبار برای ستون انجام می پذیرد. هر چه فاصله اقلیدسی بین حاصل عملیات سطری و ستونی کمتر باشد یعنی از شهرهای بیشتری عبور کرده است. در جواب بهینه بایستی مقدار خروجی این تابع معادل صفر شود.

این تابع در محاسبات تاثیری ندارد.

```
function [V1,CheckR]=RouteCheck(V)
[rows,cols]=size(V);
V1=zeros(rows,cols);
[XC,Order]=max(V);
for j=1:cols
    V1(Order(j),j)=1;
end
```

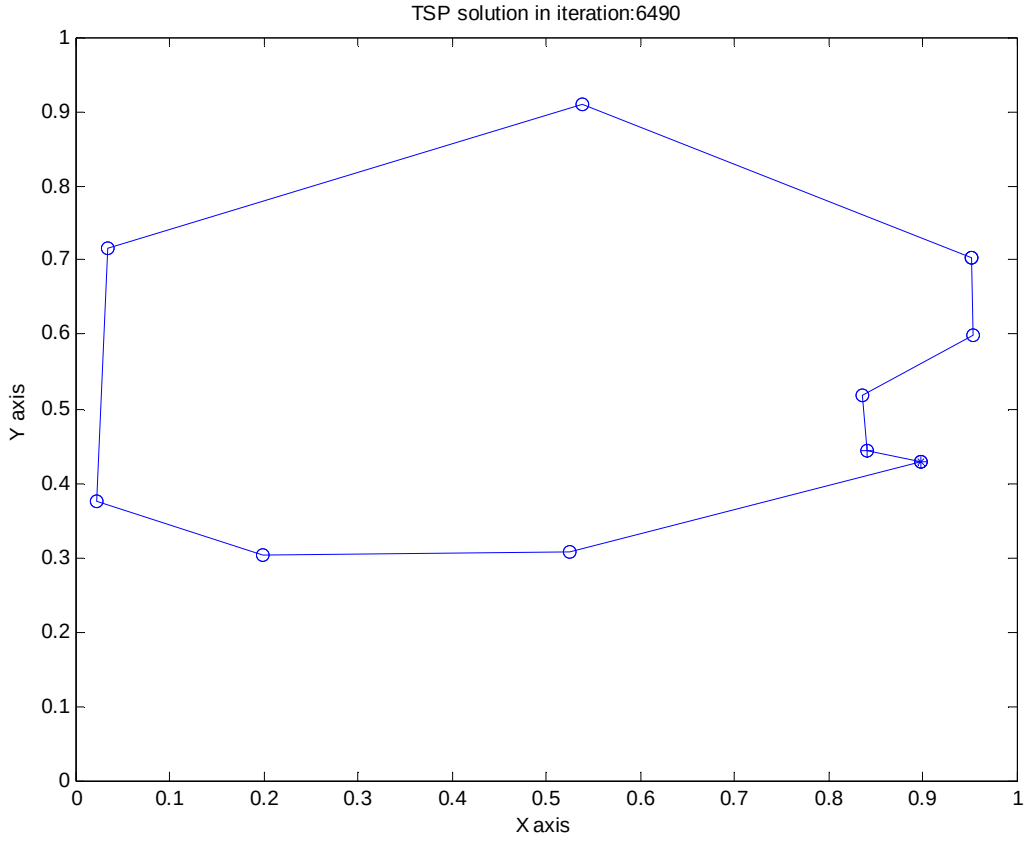
```
C=sum(V1);
R=sum(V1');
CheckR=sumsqr(C-R);
```

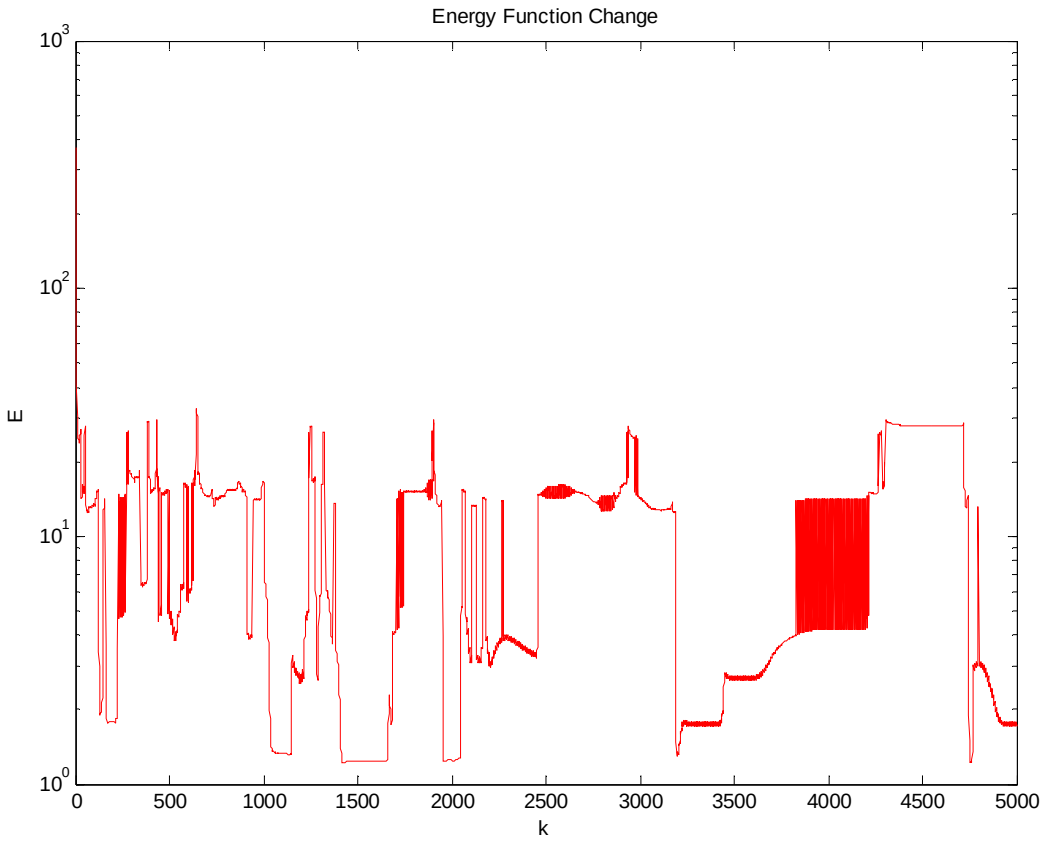
این تابع نیز تاثیری در محاسبات نداشته و فقط طول مسیرها را اندازه گیری می کند.

```
function L=Final_RouteLength(V,citys)
[xxx,order]=max(V);
New=citys(:,order);
New=[New New(:,1)];
[rows,cs]=size(New);

L=0;
for i=2:cs
    L=L+dist(New(:,i-1)',New(:,i));
end
```

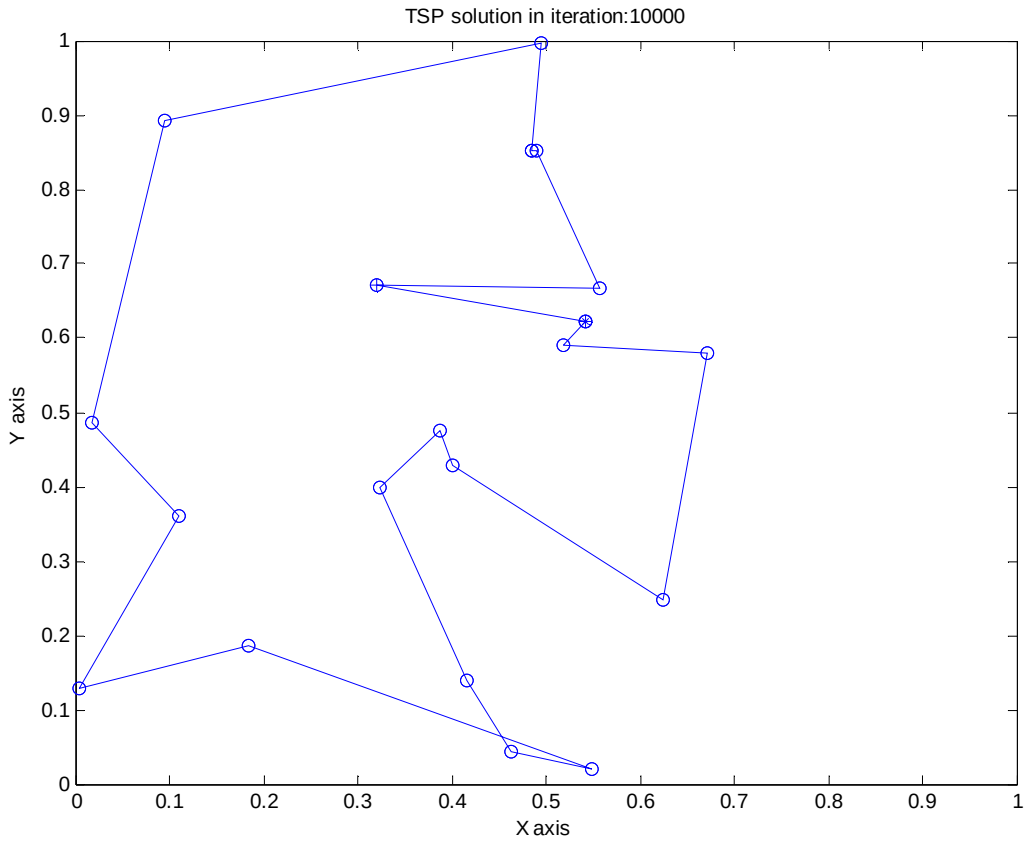
جواب نهایی برای 10 شهر به صورت زیر می باشد:

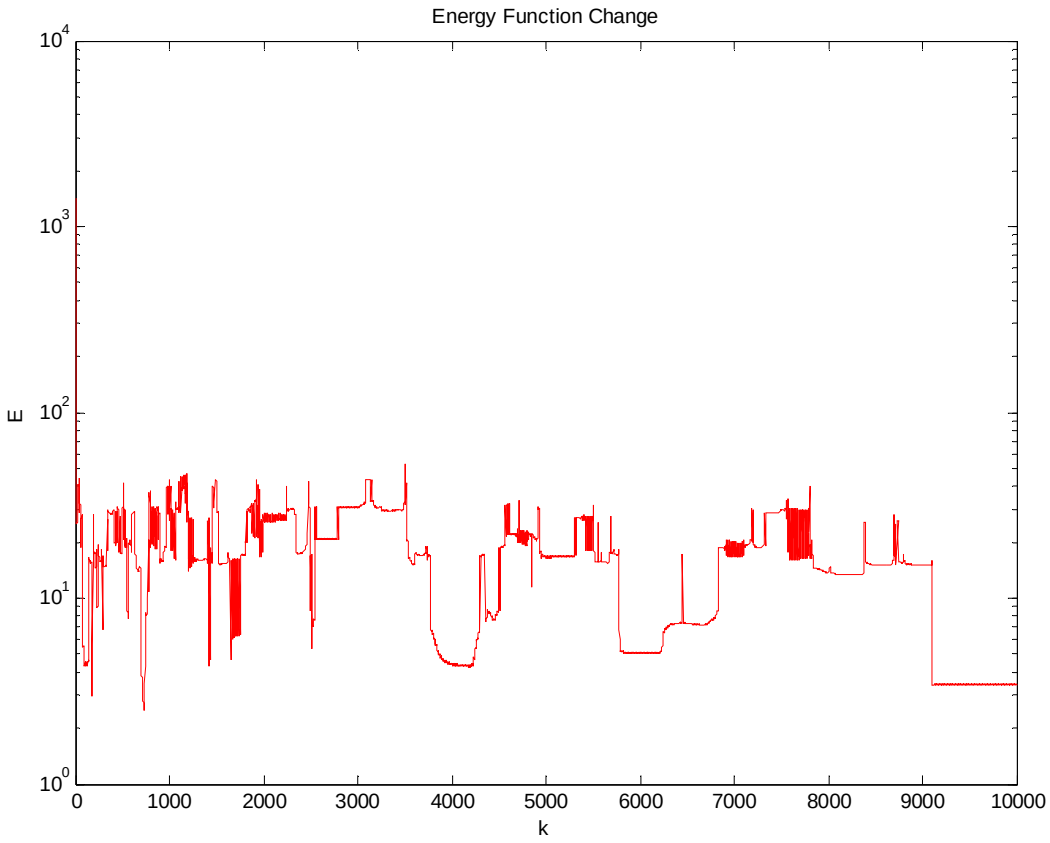




میزان انرژی در جواب های به دست آمده

اگر تعداد شهرها را به 20 افزایش دهیم نیز جواب منطقی پس از تعداد مراحل بیشتری محاسبه می شود:


TSP solution in iteration:10000
Y axis
X axis
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1



میزان تابع انرژی در بالا ترسیم شده است.