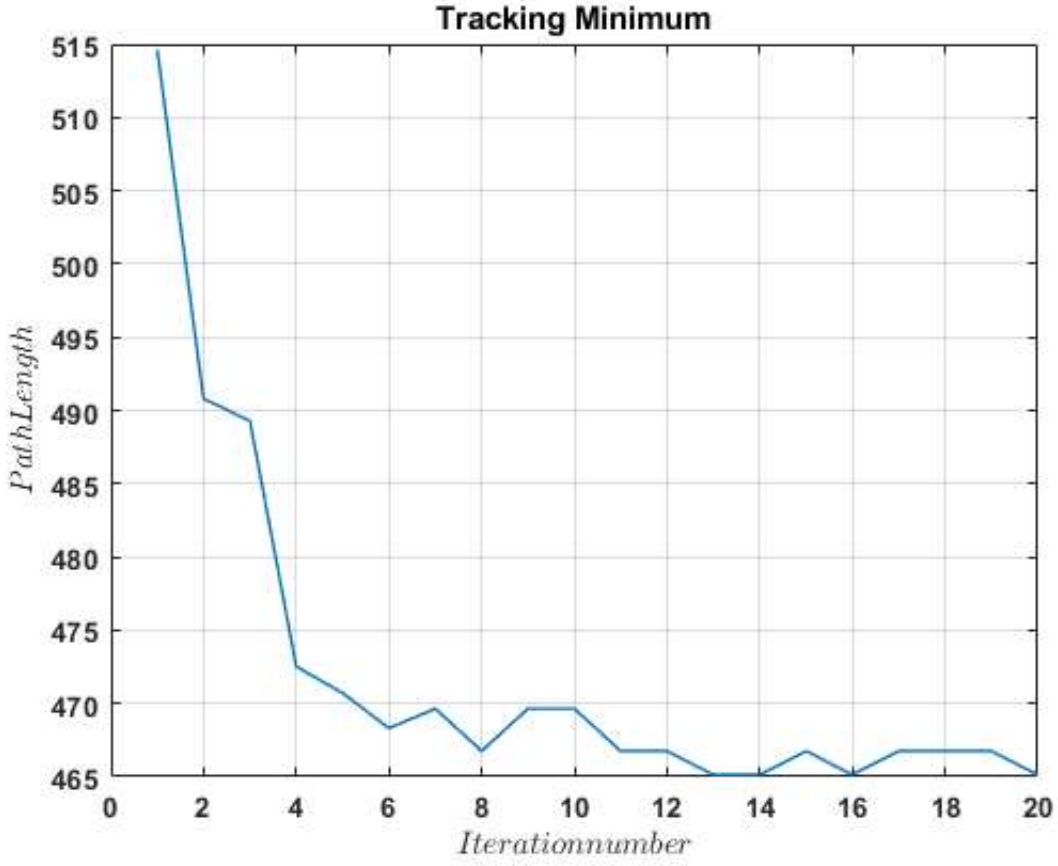
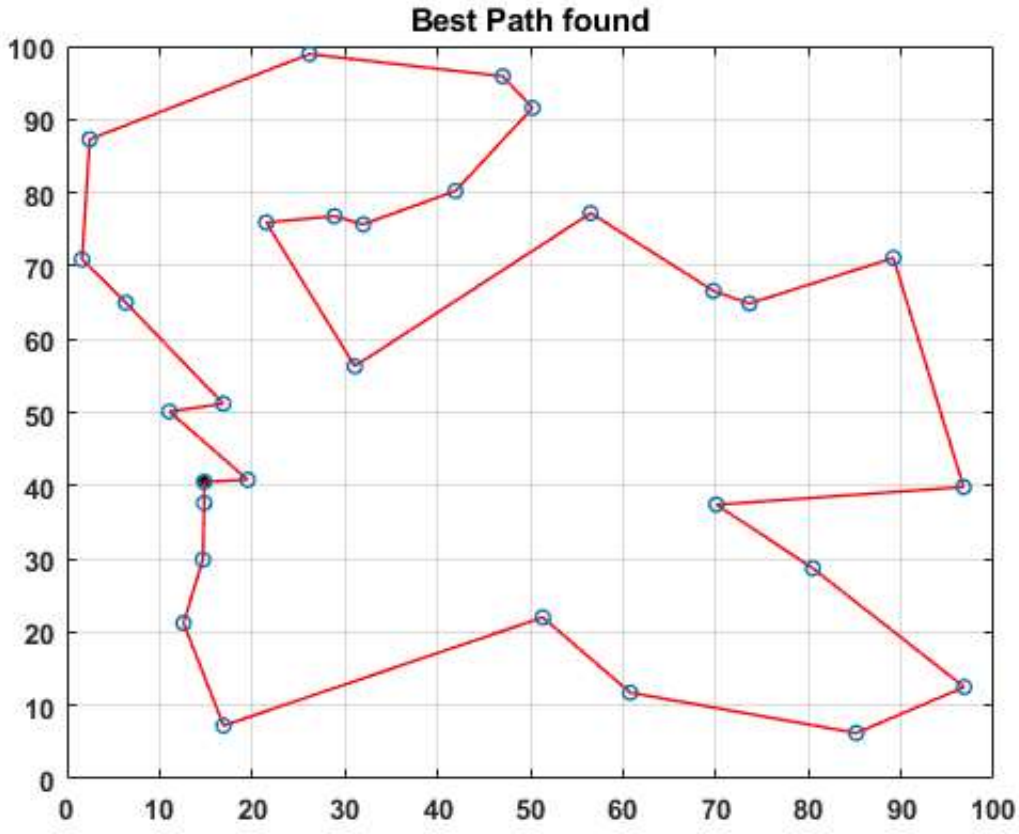
```
%%Ant System algorithm--- Shashank Iyengar Johann Koshy
clc
clear all
close all

% prompt="Enter number of cities: ";
% N=input(prompt);
N=30;

%%Create Cities

x(1:N)=rand(1,N)*100;
y(1:N)=rand(1,N)*100;

city=[1:N];
city;


 %%Variables
 alpha=1;
 beta=5;
 Q=100;
 tow0=1e-6;
 e=5;
 evap_fact=0.5;



 %Initialising Pheromone
 for i=1:N
     for j=1:N
         tow(i,j)=tow0;
     end
 end

 %Initialising Distance matrix
 for i=1:N
     for j=1:N
        X=[x(i),y(i);x(j),y(j)];
        distance(i,j)=pdist(X,'euclidean');
        eta(i,j)=1/distance(i,j);
     end
 end

%%Building tour for each ant
tmax=20;
m=N;
global_min=100000000;
route_t=[];
figure(1)
for t=1:tmax
    route_m=[];
    delta_tow=zeros(N,N);
    delta_towe=zeros(N,N);
```

```
    for ant=1:m
        tabu=city(1:N);
        st=ceil(N*rand(1));    %% Deciding start city
%         st=1;
        start_city=st; %% Deciding start city
        tabu(tabu==start_city)=[];
        route_distm(ant)=0;
        route_m(ant,:)=[st];
        for i=1:N-1
            cityi=st;
            [cityj tabu]=prob(st,tabu,tow,eta);
            route_m(ant,i+1)=[cityj];
            route_distm(ant)=route_distm(ant)+distance(cityi,cityj);
            st=cityj;
        end
        route_m(ant,N+1)=start_city;
        route_distm(ant)=route_distm(ant)+distance(cityj,start_city);
        for c=1:N
            delta_tow(route_m(ant,c),route_m(ant,c+1))=delta_tow(route_m(ant,c),route_m(ant,c
+1))+(Q/route_distm(ant));
        end
    end
    delta_tow;
    [min_antvalue min_antid]=min(route_distm);
    min_antval(t)=min_antvalue;
    route_t(t,:)=route_m(min_antid,:);
    if min_antval(t)<global_min
        global_min=min_antval(t);
        route(1,1:N+1)=route_t(t,:);
    end
    for f=1:5
        [abc min_antid]=min(route_distm);
        for c=1:N
            delta_towe(route_m(min_antid,c),route_m(min_antid,c+1))=delta_towe(route_m(min_an
tid,c),route_m(min_antid,c+1))+e*(Q/global_min);
        end
        route_distm(min_antid)=10000;
    end
    delta_towe;
    %%Pheromone update
    for i=1:N
        for j=1:N
            tow(i,j)=(1-evap_fact)*tow(i,j)+delta_tow(i,j)+delta_towe(i,j);
        end
    end
    tow;
    %Plotting at every step
    for i=1:N+1
        x1(i)=x(route_t(t,i));
        y1(i)=y(route_t(t,i));
    end
    figure(1)
    plot(x(1),y(1),'*k')
    hold on
    plot(x1,y1,'-r')
    hold on
    plot(x,y,'o')
```

```
    txt=['Iteration: ', num2str(t)];
    title(txt)
    grid on
    pause(2);
    clf


end

%%Ploting global minimum
figure
plot(1:t,min_antval)
xlabel(['$ Iteration number $'],'interpreter','latex')
ylabel(['$ Path Length $'],'interpreter','latex')
txt=['Tracking Minimum'];
title(txt)
grid on

for i=1:N+1
        x1(i)=x(route(1,i));
        y1(i)=y(route(1,i));
end
figure
plot(x(1),y(1),'*k')
hold on
plot(x1,y1,'-r')
hold on
plot(x,y,'o')
pause(2);
txt=['Best Path found'];
title(txt)
grid on



function [cityj tabulist]=prob(a,tabulist,tow,eta)
denom_P=0;
alpha=1;
beta=5;
for fi=1:length(tabulist)
    denom_P=denom_P+(tow(a,tabulist(fi))^alpha*eta(a,tabulist(fi))^beta);
end
for fi=1:length(tabulist)
    P(a,tabulist(fi))=(tow(a,tabulist(fi))^alpha*eta(a,tabulist(fi))^beta)/denom_P;
%     P(a,tabulist(fi))=P(a,tabulist(fi))*(1/denom_P)
end
P_select=rand(1);
total_p=0;
for fi=1:length(tabulist)
    total_p=total_p+P(a,tabulist(fi));
    if P_select<total_p
        cityj=tabulist(fi);
        tabulist(fi)=[];
        break;
    end
end
end
```


Tracking Minimum
515
510
505
500
495
490
485
480
475
470
465
PathLength
0
2
4
6
8
10
12
14
16
18
20
Iterationnumber



Published with MATLAB® R2018b