Experiment 3: Ant Colony Optimisation

Table of Contents

Name: Ventrapragada Sai Shravani PRN:17070123120 Batch: G-5 (E&TC)

This program is developed to find shortest path (minimum cost)between

some cities.

There are 4 parts in this program: 1.Main program of ACO (myaco.m) 2.Function to generate solution (ant_tour.m) 3.Function to calculate the cost (distance) (calculate_cost.m) 4.Function to update the traces (update_the_trace.m)

```
% Theory: Ant Colony Optimization (ACO) are a set of probabilistic
% metaheuristics and an intelligent optimization algorithms, inspired
 by
% social behavior of ants. ACO algorithms are also categorized as
 Swarm
% Intelligence methods, because of implementation of this paradigm,
 via
% simulation of ants behavior in the structure of these algorithms.
```

Code:

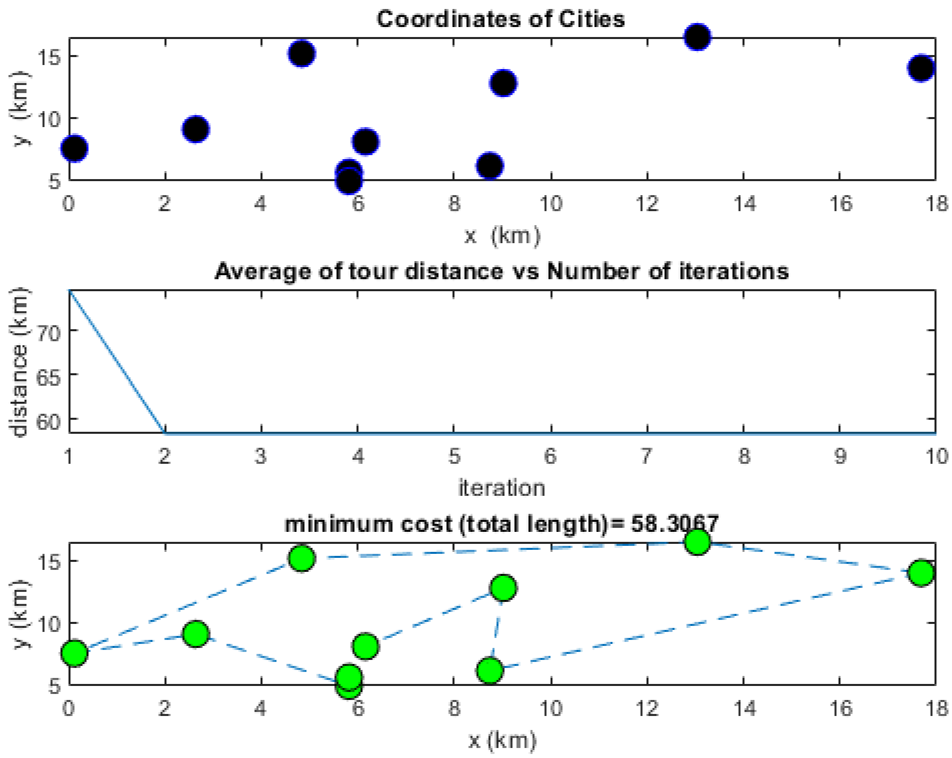
```
function myaco()
% inputs
miter=10;
m=10;
n=10;
% parameters
e=.15;              % evaporation coefficient.
alpha=1;            % effect of ants' sight.
beta=4;             % trace's effect.
t=0.0001*ones(n); % primary tracing.
el=.97;             % common cost elimination.
%
 -------------------------------------------------------------------------
% Generate coordinates of cities and plot
for i=1:n
    x(i)=rand*20;
    y(i)=rand*20;
end
subplot(3,1,1);
```

```
plot(x,y,'o','MarkerFaceColor','k','MarkerEdgeColor','b','MarkerSize',10);
title('Coordinates of Cities');
xlabel('x  (km)');
ylabel('y  (km)');

% generating distace between cities matrix.
for i=1:n
    for j=1:n
        d(i,j)=sqrt((x(i)-x(j))^2+(y(i)-y(j))^2);
    end
end

% generating sight matrix.
for i=1:n
    for j=1:n
        if d(i,j)==0
            h(i,j)=0;
        else
            h(i,j)=1/d(i,j);
        end
    end
end
h=h
%
 -----------------------------------------------------------------------
%             Main Algorithm: ACO Meta heuristic procedure
% a.  Probabilistic solution construction biased by
%     pheromone trails, without forward pheromone
%     updating
% b.  Deterministic backward path with loop elimination
%     and with pheromone updating--> update_the_trace
% c.  Evaluation of the quality of the solutions
%     generated and use of the solution quality in
%     determining the quantity of pheromone to deposit--
>calculate_cost
%
 ------------------------------------------------------------------------
for i=1:miter
% Step 1: Forward ants and solution construction

% Generate places for each ant
for j=1:m
    start_places(j,1)=fix(1+rand*(n-1));
end
% Step 2:probabilistic solution contruction
    [tour]=ant_tour(start_places,m,n,h,t,alpha,beta);
    tour=horzcat(tour,tour(:,1));

% Step 3: Calculate the cost --> total distace
    [cost,f]=calculate_cost(m,n,d,tour,el);
    [t]=update_the_trace(m,n,t,tour,f,e);
    average_cost(i)=mean(cost);

% Step 4: Determine the best route
```

```
    [min_cost(i),best_index]=min(cost);
    besttour(i,:)=tour(best_index,:);
    iteration(i)=i;
end
%
 --------------------------------------------------------------------------

% Plot Average of tour distance vs Number of Iterations
subplot(3,1,2);plot(iteration,average_cost);
title('Average of tour distance vs Number of iterations');
xlabel('iteration');
ylabel('distance (km)');

% Plot the best route
[k,l]=min(min_cost);
for i=1:n+1
    X(i)=x(besttour(l,i));
    Y(i)=y(besttour(l,i));
end
subplot(3,1,3);plot(X,Y,'--o',...
                'MarkerEdgeColor','k',...
                'MarkerFaceColor','g',...
                'MarkerSize',10)
xlabel('x (km)');ylabel('y (km)');
title(['minimum cost (total length)= ',num2str(k)]);
end
```

```
h =

  Columns 1 through 7

         0    0.1352    0.1820    0.1803    0.1143    0.2075    0.1261
    0.1352         0    0.0780    0.2730    0.0631    0.1533    0.2113
    0.1820    0.0780         0    0.0915    0.1892    0.1203    0.0760
    0.1803    0.2730    0.0915         0    0.0771    0.1375    0.3965
    0.1143    0.0631    0.1892    0.0771         0    0.0775    0.0686
    0.2075    0.1533    0.1203    0.1375    0.0775         0    0.1031
    0.1261    0.2113    0.0760    0.3965    0.0686    0.1031         0
    0.0965    0.3374    0.0634    0.1649    0.0534    0.1107    0.1661
    0.1488    0.1474    0.0884    0.3088    0.0837    0.1009    0.3368
    0.1167    0.1897    0.0728    0.3110    0.0667    0.0962    1.4317

  Columns 8 through 10

    0.0965    0.1488    0.1167
    0.3374    0.1474    0.1897
    0.0634    0.0884    0.0728
    0.1649    0.3088    0.3110
    0.0534    0.0837    0.0667
    0.1107    0.1009    0.0962
    0.1661    0.3368    1.4317
         0    0.1145    0.1593
    0.1145         0    0.3153
```

```
    0.1593    0.3153         0
```



Conclusion:

This code implementation of Ant Colony Optimization (ACO) to solve traveling #salesman problem (TSP). Given a list of cities and their pairwise distances, the task is to find a shortest #possible tour that visits each city exactly once. #And hence through the output graphs plotted we can observe the desired output.

Published with MATLAB® R2020a