Delhi Technological University

Department of Applied Mathematics

M.Sc Mathematics

TSP : Ant Colony Optimization

Submitted By :
Vidipt Vashist
2K20/MSCMAT/33
Bharat Varshney
2K20/MSCMAT/07

Submitted To :
Prof. Dhirendra KUMAR

TSP : Ant Colony Optimization

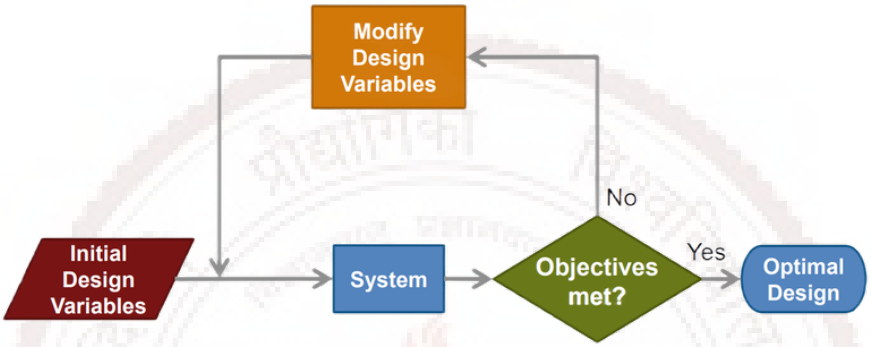
Vidipt Vashist (2K20/MSCMAT/33)
Bharat Varshney (2K20/MSCMAT/07)

November 16, 2021

Contents

1 INTRODUCTION

An optimization problem consists of maximizing or minimizing a real function by systematically choosing input values from within an allowed set and computing the value of the function



The algorithmic world is beautiful with multifarious strategies and tools being developed round the clock to render to the need for high-performance computing. Different optimization techniques have thus evolved based on such evolutionary algorithms and thereby opened up the domain of metaheuristics. Metaheuristic has been derived from two Greek words, namely, Meta meaning one level above and heuriskein meaning to find. Algorithms such as the Particle Swarm Optimization (PSO) and Ant Colony Optimization (ACO) are examples of swarm intelligence and metaheuristics. The goal of swarm intelligence is to design intelligent multi-agent systems by taking inspiration from the collective behaviour of social insects such as ants, termites, bees, wasps, and other animal societies such as flocks of birds or schools of fish.

Objective of Optimization Problem

1. Finding better (optimal) designs and decisions
2. Faster design and decision evaluations
3. Automate routine decisions

4. Useful for trade-off analysis
5. Non-intuitive designs may be found

2 TRAVELLING SALESMAN PROBLEM

The idea of the traveling salesman problem (TSP) is to find a tour of a given number of cities, visiting each city exactly once and returning to the starting city where the length of this tour is minimized.

2.1 Problem Statement - TSP

Traveling salesman problem for integer linear programming as follows:

- Generate all possible trips, meaning all distinct pairs of stops.
- Calculate the distance for each trip.
- The cost function to minimize is the sum of the trip distances for each trip in the tour.
- The decision variables are binary, and associated with each trip, where each 1 represents a trip that exists on the tour, and each 0 represents a trip that is not on the tour.
- To ensure that the tour includes every stop, include the linear constraint that each stop is on exactly two trips. This means one arrival and one departure from the stop.

2.2 Practical Applications

These situations arise mostly in various routing and scheduling problems.

1. **School bus routing problem:**
   (Angel et al., 1972) investigated the problem of scheduling buses

as a variation of the mTSP with some side constraints. The objective of the scheduling is to obtain a bus loading pattern such that the number of routes is minimized, the total distance travelled by all buses is kept at minimum, no bus is overloaded and the time required to traverse any route does not exceed a maximum allowed policy.

2. **Crew scheduling problem:**
   An application for deposit carrying between different branch banks is reported by (Svestka Huckfeldt, 1973). Here, deposits need to be picked up at branch banks and returned to the central office by a crew of messengers. The problem is to determine the routes having a total minimum cost. Two similar applications are described by (Lenstra Rinnooy Kan , 1975 and Zhang et al., 1999). Papers can be referred for delaited analysis.

3. **The order-picking problem in warehouses**
   This problem is associated with material handling in a warehouse (Ratliff Rosenthal, 1983). Assume that at a warehouse an order arrives for a certain subset of the items stored in the warehouse. Some vehicle has to collect all items of this order to ship them to the customer. The relation to the TSP is immediately seen. The storage locations of the items correspond to the nodes of the graph. The distance between two nodes is given by the time needed to move the vehicle from one location to the other. The problem of finding a shortest route for the vehicle with minimum pickup time can now be solved as a TSP. In special cases this problem can be solved easily, see (van Dal, 1992) for an extensive discussion and for references.

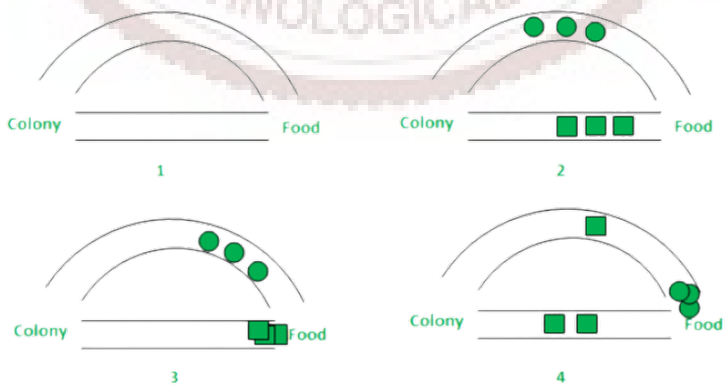
3 ACO : Ant Colony Optimization

Ant Colony Optimization technique is purely inspired from the foraging behaviour of ant colonies, first introduced by Marco Dorigo in the 1990s. Ants are eusocial insects that prefer community survival and

sustaining rather than as individual species. They communicate with each other using sound, touch and pheromone. Pheromones are organic chemical compounds secreted by the ants that trigger a social response in members of same species. These are chemicals capable of acting like hormones outside the body of the secreting individual, to impact the behaviour of the receiving individuals. Since most ants live on the ground, they use the soil surface to leave pheromone trails that may be followed (smelled) by other ants.

3.1 Principle Of Ant Colony Optimization

Ants live in community nests and the underlying principle of ACO is to observe the movement of the ants from their nests in order to search for food in the shortest possible path. Initially, ants start to move randomly in search of food around their nests. This randomized search opens up multiple routes from the nest to the food source. Now, based on the quality and quantity of the food, ants carry a portion of the food back with necessary pheromone concentration on its return path. Depending on these pheromone trials, the probability of selection of a specific path by the following ants would be a guiding factor to the food source. Evidently, this probability is based on the concentration as well as the rate of evaporation of pheromone. It can also be observed that since the evaporation rate of pheromone is also a deciding factor, the length of each path can easily be accounted for.



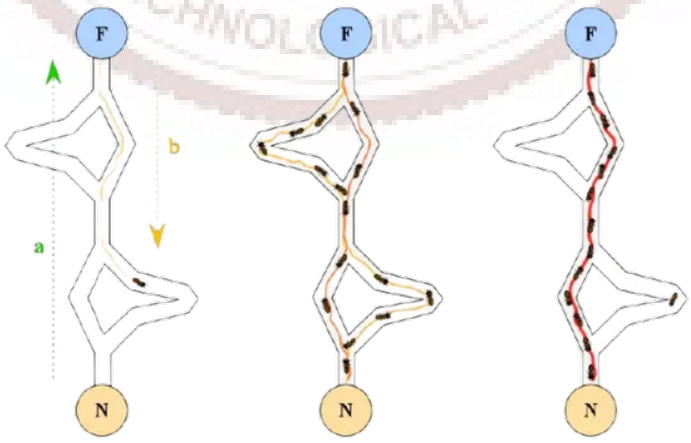
The stages can be analyzed as follows:

1. **Stage 1:** All ants are in their nest. There is no pheromone content in the environment. (For algorithmic design, residual pheromone amount can be considered without interfering with the probability)

2. **Stage 2:** Ants begin their search with equal (0.5 each) probability along each path. Clearly, the curved path is the longer and hence the time taken by ants to reach food source is greater than the other.

3. **Stage 3:** The ants through the shorter path reaches food source earlier. Now, evidently they face with a similar selection dilemma, but this time due to pheromone trail along the shorter path already available, probability of selection is higher.

4. **Stage 4:** More ants return via the shorter path and subsequently the pheromone concentrations also increase. Moreover, due to evaporation, the pheromone concentration in the longer path reduces, decreasing the probability of selection of this path in further stages. Therefore, the whole colony gradually uses the shorter path in higher probabilities. So, path optimization is attained.



3.2 Mathematics Behind : ACO

For simplicity, a single food source and single ant colony have been considered with just two paths of possible traversal. The whole scenario can be realized through weighted graphs where the ant colony and the food source act as vertices (or nodes); the paths serve as the edges and the pheromone values are the weights associated with the edges. Let the graph be G = (V, E) where V, E are the edges and the vertices of the graph. The vertices according to our consideration are Vs (Source vertex – ant colony) and Vd (Destination vertex – Food source), The two edges are E1 and E2 with lengths L1 and L2 assigned to each. Now, the associated pheromone values (indicative of their strength) can be assumed to be R1 and R2 for vertices E1 and E2 respectively. Thus for each ant, the starting probability of selection of path (between E1 and E2) can be expressed as follows:

$$P_i = \frac{R_i}{R_1 + R_2}$$

Evidently, if R1¿R2, the probability of choosing E1 is higher and vice-versa. Now, while returning through this shortest path say Ei, the pheromone value is updated for the corresponding path. The updation is done based on the length of the paths as well as the evaporation rate of pheromone. So, the update can be step-wise realized as follows:

1. In accordance to path length –

$$R_i \leftarrow R_i \frac{K}{L_i}$$

2. In accordance to evaporation rate of pheromone –

$$R_i \leftarrow (1 - v)\ XR_i$$

At each iteration, all ants are placed at source vertex Vs (ant colony). Subsequently, ants move from Vs to Vd (food source) following step 1. Next, all ants conduct their return trip and reinforce their chosen path based on step 2

3.3 Concept for TSP solution

The above behaviour of real ants has inspired ant system, an algorithm in which a set of artificial ants cooperate to the solution of a problem by exchanging information via pheromone deposited on graph edges. Ant system has been applied to combinatorial optimization problems such as the traveling salesman problem. This algorithm is used to produce near-optimal solutions to the TSP. The traveling salesman problem (TSP) is the problem of finding a shortest closed tour which visits all the cities in a given set. Here we will restrict attention to TSPs in which cities are on a plane and a path (edge) exists between each pair of cities (i.e., the TSP graph is completely connected)

4 SOLVING USING PYTHON : TSP

Example 4.1

Detail Code is available at : `https://colab.research.google.com/drive/1x7AR5r7kuuFhzpN7wvvPoOV3vWSjS-8p?usp=sharing`.

4.1 Code

```
import numpy as np
from numpy import inf

#given values for the problems

d = np.array([[0,10,12,11,14]
,[10,0,13,15,8]
,[12,13,0,9,14]
,[11,15,9,0,16]
,[14,8,14,16,0]])


iteration = 100
n_ants = 5
n_citys = 5
In [62]:
```

```
# intialization part

m = n_ants
n = n_citys
e = .5          #evaporation rate
alpha = 1      #pheromone factor
beta = 2        #visibility factor
In [ ]:
#calculating the visibility of the next city visibility(i,j)=1/d(i
 ,j)

visibility = 1/d
visibility[visibility == inf ] = 0
In [64]:
#intializing pheromne present at the paths to the cities

pheromne = .1*np.ones((m,n))

#intializing the rute of the ants with size rute(n_ants,n_citys+1)
#note adding 1 because we want to come back to the source city

rute = np.ones((m,n+1))
In [77]:
for ite in range(iteration):

rute[:,0] = 1           #initial starting and ending positon of
 every ants '1' i.e city '1'

for i in range(m):

temp_visibility = np.array(visibility)          #creating a copy of
  visibility

for j in range(n-1):
#print(rute)

combine_feature = np.zeros(5)      #intializing combine_feature
 array to zero
cum_prob = np.zeros(5)             #intializing cummulative
 probability array to zeros

cur_loc = int(rute[i,j]-1)         #current city of the ant

temp_visibility[:,cur_loc] = 0      #making visibility of the
 current city as zero

p_feature = np.power(pheromne[cur_loc,:],beta)          #
 calculating pheromne feature
```

```
58  v_feature = np.power(temp_visibility[cur_loc,:],alpha)  #
     calculating visibility feature
59
60  p_feature = p_feature[:,np.newaxis]                      #adding
     axis to make a size[5,1]
61  v_feature = v_feature[:,np.newaxis]                      #adding
     axis to make a size[5,1]
62
63  combine_feature = np.multiply(p_feature,v_feature)     #
     calculating the combine feature
64
65  total = np.sum(combine_feature)                         #sum of all
      the feature
66
67  probs = combine_feature/total   #finding probability of element
     probs(i) = comine_feature(i)/total
68
69  cum_prob = np.cumsum(probs)     #calculating cummulative sum
70  #print(cum_prob)
71  r = np.random.random_sample()   #randon no in [0,1)
72  #print(r)
73  city = np.nonzero(cum_prob>r)[0][0]+1       #finding the next city
      having probability higher then random(r)
74  #print(city)
75
76  rute[i,j+1] = city              #adding city to route
77
78  left = list(set([i for i in range(1,n+1)])-set(rute[i,:-2]))[0]
       #finding the last untraversed city to route
79
80  rute[i,-2] = left                    #adding untraversed city to
     route
81
82  rute_opt = np.array(rute)                #intializing optimal route
83
84  dist_cost = np.zeros((m,1))              #intializing
     total_distance_of_tour with zero
85
86  for i in range(m):
87
88  s = 0
89  for j in range(n-1):
90
91  s = s + d[int(rute_opt[i,j])-1,int(rute_opt[i,j+1])-1]   #
     calcualting total tour distance
92
93  dist_cost[i]=s                       #storing distance of tour for
     'i'th ant at location 'i'
94
```

```
95  dist_min_loc = np.argmin(dist_cost)                #finding location
     of minimum of dist_cost
96  dist_min_cost = dist_cost[dist_min_loc]            #finging min of
     dist_cost
97
98  best_route = rute[dist_min_loc,:]                  #intializing
     current traversed as best route
99  pheromne = (1-e)*pheromne                          #evaporation of
     pheromne with (1-e)
100
101 for i in range(m):
102 for j in range(n-1):
103 dt = 1/dist_cost[i]
104 pheromne[int(rute_opt[i,j])-1,int(rute_opt[i,j+1])-1] = pheromne[
     int(rute_opt[i,j])-1,int(rute_opt[i,j+1])-1] + dt
105 #updating the pheromne with delta_distance
106 #delta_distance will be more with min_dist i.e adding more weight
     to that route  peromne
107
108 print('route of all the ants at the end :')
109 print(rute_opt)
110 print()
111 print('best path :',best_route)
112 print('cost of the best path',int(dist_min_cost[0]) + d[int(
     best_route[-2])-1,0])
113 route of all the ants at the end :
```

4.2 Output

```
[[1. 4. 3. 2. 5. 1.]
 [1. 4. 3. 2. 5. 1.]
 [1. 4. 3. 2. 5. 1.]
 [1. 4. 3. 2. 5. 1.]
 [1. 4. 3. 2. 5. 1.]]

best path : [1. 4. 3. 2. 5. 1.]
cost of the best path 55
```

5 LIMITATION

5.1 Efficient Route vs Shortest Route

- The shortest distance between the locations you set. This may not be the quickest route, especially if the shortest route is through a town or city.
- Efficient route which takes the least time and the most fuel-efficient route for your trip.

For instance the Efficient Route might have a lot of left turns and the shortest route may drive through heavy traffic. In both these route options, your fuel consumption will go up significantly. Also, what if the fastest route is three minutes faster, but several miles longer? Is it still an efficient one? What if the shortest path allows you to drive 5 miles less, but makes you go through two tolls? Is it an efficient one?

5.2 New Approach (to overcome limitation)

We introduce a new parameter called as **Carrying Capacity of route** i.e., amount of object (namely cars , ants..etc) per unit of route.

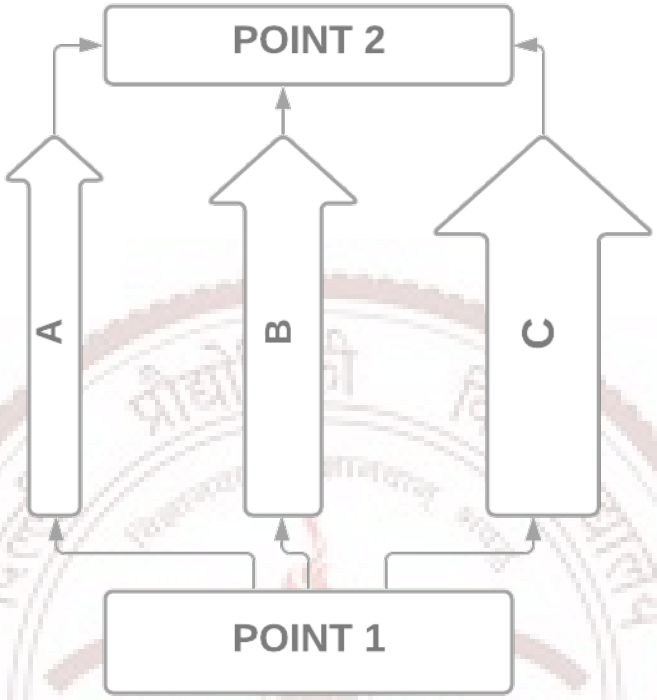
5.2.1 Experiment

let consider 3 route A , B and C, such that Carry Capacity of route A less then Carry Capacity of route B less then Carry Capacity of route C.

$$CC(A) \leq CC(B) \leq CC(C)$$

Also the distance of Route A less then distance of Route B less then distance of Route C

$$D(A) \leq D(B) \leq D(C)$$



Now if ants start moving from point 1 to point 2

1. **ITERATION-Start**

   - at time = 0 , three ants start from point 1 each going into 3 different routes (A , B and C) labelling them Ant_A , Ant_B and Ant_C respectively.
   - we see that Ant_A reaches point 2 first then Ant_B and then Ant_C (because of distance already mentioned)
   - at time = t0 , Ant_A has already reached again at point 1 and other ants are still on the way , so the concentration of pheromone at route A is more then concentration of pheromone at route B is more then concentration of pheromone at route C.

2. **ITERATION-Middle**

- as the concentration of pheromone is more in route A so more number of ants moves into route A then route B and then least in route C.
- this increase in number of ants causes more congestion of routes A, B and C resp.
- now last step of previous iteration repeat itself causing more concentration of pheromone.

3. **ITERATION-Final**

- hence after n - iteration the route A becomes very much block as the number of ants exceeds its carrying capacity thus now it takes more time for ants in route A compare to route B
- route B having average pheromone concentration along with average distance between points and better carrying capacity then A route becomes the most efficient routes among all the routs.

5.2.2 Conclusion

The shortest distance route is not always the best route as we see in experiment that concept of carrying capacity of routes plays a very important role in selecting route as route having less carrying capapcity will become more congested thus the objects will not me able to move through it thus causing delay in time.

6 REFERENCES

1. "A Review of Modern Optimization Techniques", International Journal of Emerging Technologies and Innovative Research (www.jetir.org), ISSN:2349-5162, Vol.4, Issue 7, page no.135-137, July-2017,

2. Sangwan, Shabnam. (2018). Literature Review on Travelling Salesman Problem. International Journal of Research. 5. 1152.

3. Tahri, Houda. (2015). Mathematical Optimization Methods: Application in Project Portfolio Management. Procedia - Social and Behavioral Sciences. 210. 339-347. 10.1016/j.sbspro.2015.11.374.

4. https://in.mathworks.com/help/optim/ug/traveling-salesman-problem-based.html

5. Modeling and Optimization Approaches in Design and Management of Biomass-Based Production Chains ,Balaman, Şebnem Yılmaz

6. An improved ant colony optimization algorithm based on context for tourism route planningLiang, Shengbin Jiao, Tongtong Du, Wencai Qu, Shenming