1

Ant Colony Optimization

Pushpendra Singh
Department of Electrical Engineering
Govt. Women Engineering College, Ajmer, India

Nand K. Meena
School of Engineering and Applied Science
Aston University, Birmingham, United Kingdom

Jin Yang
School of Engineering and Applied Science
Aston University, Birmingham, United Kingdom

Adam Slowik
Department of Electronics and Computer Science
Koszalin University of Technology, Koszalin, Poland

CONTENTS

1.1 Introduction

Ants are known as social insects (as are bees and termites) because they live and work together in well-structured ant colonies or communities of the same species. The ants can construct small to large sized colonies where population may vary from a few dozens to millions of individuals from different castes of sterility. These colonies are described as superorganisms because of their

social behavior. Ants are almost blind insects but effectively search for the food source. It is very interesting to know how this creature, being a simple individual, understands the importance of togetherness in finding the shortest route between their nest and food source [3].

> "Many entomologists prefer to view ant colonies and the societies of other social insects as more like superorganisms than communities of individualized organisms"
>
> Robert L. O'Connell

Ant colony optimization (ACO) was introduced in the early nineties. The social and optimal behavior of the ant colony was implemented through artificial ants. In this artificially developed population, an ant is a simple, independent, and asynchronous agent that collaborate to find an optimal solution for complex real-life optimization problems. ACO is a population based meta-heuristic optimization technique biologically inspired from the foraging behavior of an insects horde, i.e., ant colony. It is broadly defined as the class of model based search (MBS) techniques [4]. The MBS methods are quite effective to find the optimal solution for combinatorial optimization problems. On the basis of probabilistic modes, the MBS methods are bifurcated into the following classes:

a) algorithms which employ a specified probabilistic model without reforming the model configuration during the run.

b) algorithms which reform the probabilistic model in alternating phases.

The ant colony based techniques fall into the first class. The ACO technique, during the run, updates the parameter values of the probabilistic model in such a manner that it creates a rich probability to produce high quality results over time. This chapter presents the basic fundamentals of ant behaviors and the ACO technique followed by its application.

1.2 Ants' behavior

The ants are social creatures which prefer to live in groups. Ants use sensing clues for seeking the shortest route from their nest to a food source. These insects are capable to adopt the changes in the surrounding environment. In these ant colonies, each ant executes very simple group actions without

knowing what other ants are doing; however everybody knows that the outcome is highly social and structured [1, 2]. They can easily find the next shortest route of their trailing aim even if the present one gets corrupted or is obstructed by an obstacle. This phenomenon is presented in Fig. 1.1. From the figure, it can be visualized that, a) the ants are traveling in a straight line to follow the shortest route; b) an obstacle breaks their journey and splits the path; c) ants are again seeking the next possible shortest path; and d) finally, ants are able to ensure the shortest path.

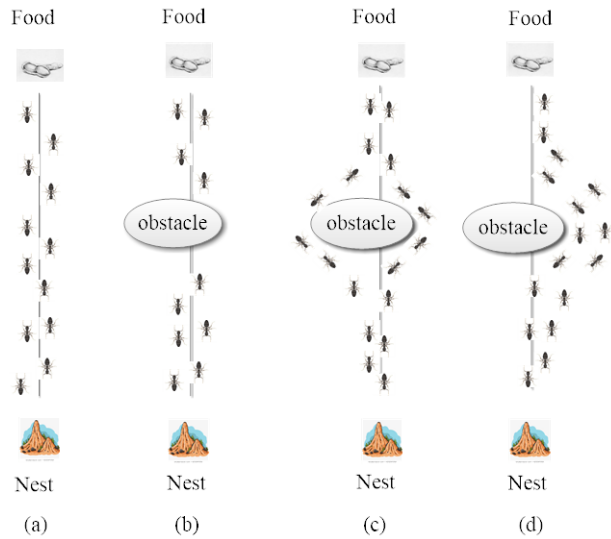

FIGURE 1.1
Traveling behavior of ant colony, a) ants following shortest path, b) an obstacle in ants' path, c) ants again seeking the shortest path, and d) ants finally locating the shortest path.

Ethnologists have found that this skill of ants is due to phenomena called *pheromone trails*. These help ants communicate with their fellows. This information also helps the ant colony to walk on the same path or to make a decision regarding a change of foraging path. Initially, ants randomly explore the area surrounding the nest to search for a food source. After seeking the food origin, ants return to their nest by carrying some food. While returning with food, they deposit an organic compound on the ground known as a *pheromone*. Usually, they deposit more pheromone when returning with food and produce less when searching for a food source. This deposited chemical on the ground guides their colleagues to reach a food source. Ants in the neighborhood presumably prefer to trace the path which has the highest pheromone concentration level. This continuous and indirect form of communication between the fellow ants by the pheromone trail helps the ant colony in

seeking the shortest route between their nest and food source. As presented in Fig. 1.1, ants are traveling in a straight line as it clearly provides the shortest possible path between two points, i.e., nest and food source. The foraging and adjacent ants with food are using the same path. It has been assumed that all ants are traveling at normal speed; thus most of the ants will appear at the shortest path in an average time. Therefore, the pheromone concentration will increase at the shortest path as most of the ants follow this path. After some time, the difference in the amount of pheromone levels on these paths will be identified easily by new ants entering into the system. These new ants will prefer to travel on the shortest path which has a rich pheromone level, that is a straight line between nest and food source. In Fig. 1.1 b), the shortest path pheromone trail has been interrupted by an obstacle. The ants are unable to follow the shortest path in this situation therefore seeking a new path, as can be observed from Fig. 1.1 c). They randomly turn left and right seeking the path to reach their destination as they have no clue to follow. After some time, it has been observed from Fig. 1.1 d) that some ants found the shortest routes around the obstacle, and will rapidly accumulate the interrupted pheromone trail as compared to the ants who have taken a longer path. The pheromone concentration level will continuously increase as more ants follow the new shortest path. This increased pheromone concentration will help the new ants to follow the next searched shortest path as explained earlier. However, this chemical has an evaporation tendency. This constructive auto-catalytic process helps the ant colony seek the shortest path to their destination very soon.

1.3 Ant colony algorithm

The ACO technique has been inspired by the foraging behavior of an ant colony, discussed in previous sections. The ant's behavior is mathematical, modeled into some set of equations to find an optimal solution for combinatorial optimization problems. Since its induction, various real-life engineering optimization problems have been solved by using this method. Among these methods, *Ant System* (AS) is the most popular, productive, and oldest method. It was developed by M. Dorigo and is the most preferred operation used in ACO techniques [5, 6, 7, 8]. One of the important characteristics of this process is that at every iteration, the concentration of pheromone is updated for the ants who have participated in constructing the solution in iteration itself. The AS is described as follows: a) all ants are randomly initialized in the starting stage, and b) these m ants are set on the vertex of the build frame and a constant amount of pheromone trail intensity, i.e. $\tau(i,j) = 1 \ \forall \ (i,j) \in Allowed$, is alloted at all the edges, where, *Allowed* represents the set of feasible neighbors of the ant.

Each artificial ant is an agent with following characteristics [5]:

1) it prefers to walk on the highest pheromone probability frame,
2) the already visited edges are prohibited until the circuit is complete,
3) when journey ends, the substance known as the trail is updated at each visited edge.

At every solution building level or step, each ant incrementally adds its part of the substance to the partial building solution frame. Let us assume that kth ant of ith edge, at the tth building step, performs a random walk from ith edge to the next edge of the building j. At each edge, stochastic decisions are taken by ants to choose the next node or edge. These decisions are taken according to the transition probability of one edge to another with respect to the present edge of the ant. The transition probability of the kth ant located at the ith edge, to travel towards the jth edge, can be determined by using random proportional state transition law [5], defined as

$$P_k(i,j) = \begin{cases} \frac{[\tau(i,j)]^{\alpha}.[\eta(i,j)^{\beta}]}{\sum_{c_{il} \in N(s^p)} [\tau(i,l)]^{\alpha}.[\eta(i,l)]^{\beta}} & \text{if } c_{ij} \in N(s^p) \\ 0 & \text{otherwise} \end{cases} \tag{1.1}$$

where, $\tau(i,j)$ denotes the pheromone intensity at path (i,j), which is a connection between the edge i and j. Similarly, $\eta(i,j)$ is a heuristic value also known as *visibility* of path (i,j). Its value is generally set as the inverse of connection cost or distance. l is an edge not yet visited by ant k. Suppose $d(i,j)$ is representing the length of path (i,j) then the value of $\eta(i,j)$ can be calculated as $1/d(i,j)$. Furthermore, $N(s^p)$ is a set of feasible components or expedient neighbors of the kth ant on edge i. α and β are the controlling parameters to determine the relative importance of pheromone versus heuristic value where, $\alpha \in (0,1]$ and $\beta \in (0,1]$.

When all the ants of the colony have achieved their solution, the pheromone trail is updated for all the edges by the pheromone global updating rule, as given below.

$$\tau(i,j) \leftarrow \zeta.\tau(i,j) + \sum_{k=1}^{m} \Delta\tau_k(i,j) \tag{1.2}$$

Here, the coefficient $\zeta \in (0,1]$ is presented in such as way that $(1-\zeta)$ would represent the evaporation of the pheromone trail between two steps/levels (the time taken to complete a cycle). The value of ζ is set to be less than unity to prohibit unlimited pheromone accumulation in the path. Similarly, the $\Delta\tau_k(i,j)$ is known as the quantity of pheromone laid by the kth ant, measured in per unit length of the path (i,j), which is expressed below

$$\Delta\tau_k(i,j) = \begin{cases} \frac{Q}{L_k} & \text{if ant } k \text{ travels through the path } (i,j) \\ 0 & \text{otherwise} \end{cases} \tag{1.3}$$

where, Q and L_k are representing constant (usually, $Q = 1$) and the total travel length of the kth ant respectively. In this technique, the heuristic value presented in (1.1), $\eta(i, j)$ is usually adopted to favor the cost effective edges of the frame with a large amount of pheromone level, whereas in (1.2) the first term models the evaporated substance. This indicates the lower trail level due to pheromone decaying factors. It has to be noted that the trail for unflattering edges will be evaporated by time. The second term of (1.2) models the reinforcement of the pheromone trail. This will help newly joined ants in the ant colony system to effectively and easily sense the path of higher pheromone concentration. The amount of deposited substance on the path highly depends on the quality of solution achieved so far. The pheromone trail updating rule helps in simulating the change in intensity due to deposition of additional pheromone by new ants and the evaporated amount from the path.

For better understanding, a simple mathematical model is given in Fig. 1.2. It presents the traveling paths of two ants between the four points represented by bold and dashed lines. The starting and ending points of the ants

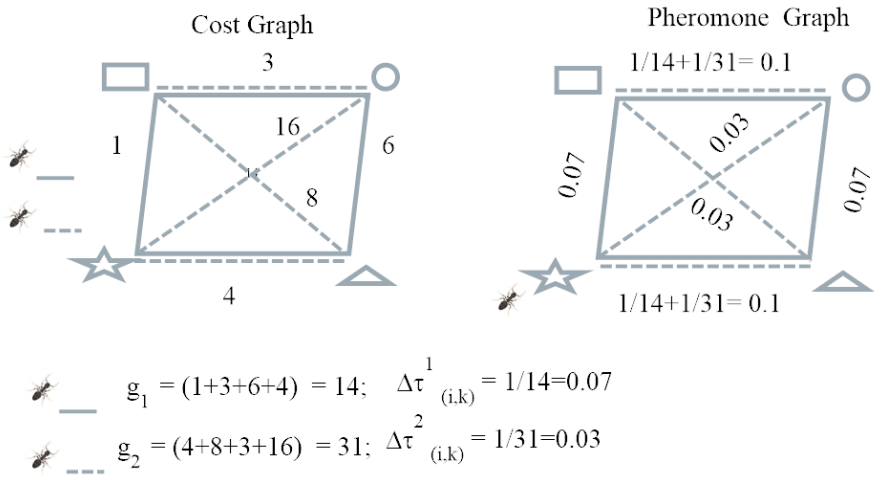

FIGURE 1.2
Demonstration of artificial ant following the shortest path with high pheromone intensity.

are considered at edges represented by *stars*. The cost graph represents the cost involved in traveling between two edges whereas, the pheromone graph presents the calculated pheromone value between two edges. Now, the probabilities of paths are calculated for selecting the shortest path by a new ant which starts its journey from the edge *star*. As per the calculation obtain from the figure, it can be analyzed that a new ant which has to start its journey from the edge *star* will move towards the edge represented by the *triangle* because this path has the highest concentration of the pheromone, i.e., 0.1, as compared to two other paths towards the nodes represented by the *square*,

and *circle* with a pheromone concentration of 0.07 and 0.03 respectively. After reaching the *triangle* node, the ant will choose the path towards the *circle* as it has the highest substance concentration, i.e., 0.07. Similarly, it will move towards the *square* edge and then back to *star*. The flowchart of ACO optimization is presented in Fig. 1.3.

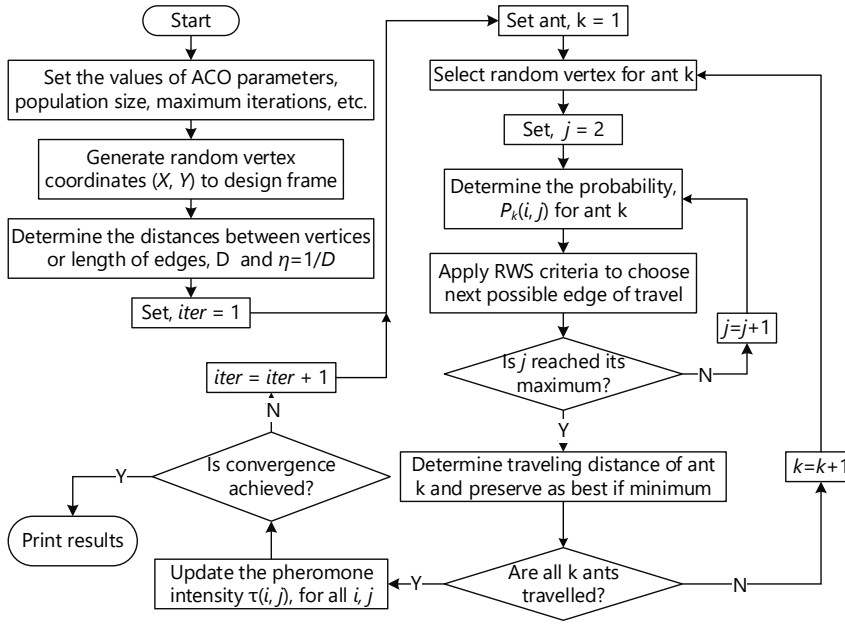

FIGURE 1.3
Flow chart of Ant Colony Optimization.

1.4 Source-code of ACO algorithm in Matlab

The Matlab source codes for fitness function, roulette wheel selection approach, and ACO algorithm are presented in Listings 1.1, 1.2 and 1.3 respectively. Here, the $OF(.)$ is representing the address of objective function.

```
% Distance calculation for TSP
function [Dist] = FUN_TSP(ant_path,nE,Distance)
% Calculation of distance traveled by ant 'k'
Dist=0;
for t=1:nE
Dist = Dist + Distance(ant_path(t),ant_path(t+1));
end
```

Listing 1.1
Fitness function for traveling salesman problem.

```
% Roulette wheel selection criteria
function loc = roulette(fitness)
partsum=0; j=1;
sumfit = sum(fitness); rn = rand*sumfit;
while j<=length(fitness)
partsum=partsum + fitness(j);
if partsum>=rn          break;     end
j=j+1;
end
if j>length(fitness)    j=length(fitness); end
loc=j;
```

Listing 1.2
Roulette wheel selection (RWS) function.

```
% ACO algorithm
clc; clear;
Max_Iteration = 450;     % Max. no. of iterations
ant_pop = 50;            % Ant population size
Q = 1;                   % Constant
%=========================================================
% Frame design...
% Coordinations of frame edges are generated randomly....
X = [1,4,10,30,11,23,28,30,15,11];  % X coordinates
Y = [42,36,10,46,30,44,8,38,19,4];  % Y coordinates
%---------------------------------------------------------
% Calculation of coordination matrix and visibility heuristic
%---------------------------------------------------------
nE = length(X);            % Number of cities or points...
Distance = zeros(nE,nE);
for i = 1:nE-1
for j = i+1:nE
% Euclidean distance between two points/coordinates
Distance(i,j) = sqrt((X(i)-X(j))^2 + (Y(i)-Y(j))^2);
Distance(j,i) = Distance(i,j);
end
end
ETA = (1./Distance);          % heuristic matrix
ETA(1:1+size(ETA,1):end) = 0;% remove inf and set values to 0
%---------------------------------------------------------
% Initialization
%---------------------------------------------------------
% ACO algorithm controlling parameters
alpha = 1; beta = 0.01; rho = 0.02;
TAU = 0.01.*rand(nE,nE);          % Initial values of pheromone
ant = zeros(ant_pop,length(X));  % Initial population of ants
fitness = inf.*ones(ant_pop,1);  % Initial fitness of ants.
best_fit = max(fitness);
Iteration = 0;                   % Initialize the iteration
while Iteration < Max_Iteration  % Iterations loop
Iteration = Iteration + 1;       % Iterations count
%---------------------------------------------------------
% Start ant journey from one point to other
%---------------------------------------------------------
for k = 1:ant_pop              % Ant k
ant0(1) = randi([1 nE],1,1);% Random start of ant k...
for l = 2:nE
i = ant0(end);     % Start the journey from chosen edge i
% numerator of transition probability
NUM = TAU(:,i).^alpha.*ETA(:,i).^beta;
DEN = sum(NUM);  % Denominator of transition probability
Prob = NUM/DEN;  % Transition probability
Prob(ant0)=0; %Set 0 Prob for path already visited by ant k
ant0(l) = roulette(Prob);  % Roulette wheel selection for roote
end
% Fitness calculation of ant k...
```

```
fitness(k) = FUN_TSP([ant0 ant0(1)],nE,Distance);
if fitness(k) < best_fit     % Checking for best solution
best_ant = ant0;
best_fit = fitness(k);
end
ant(k,:) = ant0;             % Store ant0 to ant k
ant0 = [];                   % Restore ant0
end
%--------------------------------------------------------------------
% Update Phromone intensity at all the corners/paths
%--------------------------------------------------------------------
for k=1:ant_pop    % ant k
ANT0 = [ant(k,:) ant(k,1)];
for i=1:nE
% Amount of pheromone added at path(i,l) by ant k
TAU(ANT0(i),ANT0(i+1)) = TAU(ANT0(i),ANT0(i+1)) + Q/fitness(k);
end
end
% Pheromone evaporation with the time or iterations...
TAU=(1-rho)*TAU;
end
disp(best_ant);
disp(best_fit);
```

Listing 1.3
Source-code of ACO in Matlab.

1.5 Source-code of ACO algorithm in C++

Listing 1.4 presents the the source-code of ACO algorithm in C++.

```
#include <iostream>
#include <math.h>
#include <ctime>
#include <limits>
#include <algorithm>
using namespace std;
double r() {return (double)(rand()%RAND_MAX)/RAND_MAX;}
double FUN_TSP(int ant_path[], int ant0, int nE, double *Distance)
{   int ant[nE+1];
for(int i=0; i<nE; i++)
{ant[i]=ant_path[i];}
ant[nE]=ant0;
double Dist=0;
for(int t=0; t<nE; t++)
{Dist=Dist+Distance[ant[t+1]*nE+ant[t]];}
return Dist;
}
int roulette(double fitness[], int nE)
{   double partsum=0, sumfit=0; int j=0;
for(int i=0; i<nE; i++)
{sumfit=sumfit+fitness[i];}
double rn=r()*sumfit;
while (j<nE)
{partsum=partsum+fitness[j];
if (partsum>=rn)
{return j;}
j++;}
if (j>nE-1) {j=nE-1;}
return j;
}
```

```
31 int main()
32 {    srand(time(NULL));
33 int Max_Iteration = 500;
34 int ant_pop=50;
35 double Q=1;
36 int nE=10;
37 int X[nE]={1,4,10,30,11,23,28,30,15,11};
38 int Y[nE]={42,36,10,46,30,44,8,38,19,4};
39 double Distance[nE][nE]; double ETA[nE][nE]; double TAU[nE][nE];
40 int ant[ant_pop][nE]; double fitness[ant_pop]; double best_fit;
41 double alpha=1, beta=0.01, rho=0.02;
42 double NUM[nE], DEN, Prob[nE];
43 int ant0[nE], best_ant[nE], ANT0[nE+1], END, ix;
44 int Iteration=0;
45 for(int i=0; i<nE-1; i++)
46 {
47 for(int j=0; j<nE; j++)
48 {
49 Distance[i][j]=sqrt(pow(X[i]-X[j],2)+pow(Y[i]-Y[j],2));
50 ETA[i][j]=1/Distance[i][j];
51 if (i==j) {ETA[i][j]=0;}
52 TAU[i][j]=0.01*r();
53 }
54 }
55 for(int i=0; i<ant_pop; i++)
56 {fitness[i]=std::numeric_limits<double>::max();}
57 best_fit=*min_element(fitness,fitness+nE);
58 //Initialization
59 while (Iteration<Max_Iteration)
60 {
61 Iteration=Iteration+1;
62 for(int k=0; k<ant_pop; k++)
63 {    for(int s=0; s<nE; s++){ant0[s]=-1;}
64 END=0;
65 ant0[END]=rand()%nE;
66 for(int l=1; l<nE; l++)
67 {
68 for(int s=0; s<nE; s++){if (ant0[s]>-1){END=s;}}
69 ix=ant0[END];
70 double DEN=0;
71 for(int s=0; s<nE; s++)
72 {
73 NUM[s]=TAU[s][ix]*alpha*ETA[s][ix]*beta;
74 DEN=DEN+NUM[s];
75 }
76 for(int s=0; s<nE; s++)
77 {Prob[s]=NUM[s]/DEN;}
78 for(int s=0; s<=END; s++)
79 {
80 if (ant0[s]>-1)
81 {Prob[ant0[s]]=0;}
82 }
83 ant0[l]=roulette(Prob,nE);
84 }
85 fitness[k]=FUN_TSP(ant0, ant0[0], nE, *Distance);
86 if (fitness[k]<best_fit)
87 {
88 best_fit=fitness[k];
89 for(int s=0; s<nE; s++)
90 {best_ant[s]=ant0[s];}
91 }
92 for(int s=0; s<nE; s++)
93 {
94 ant[k][s]=ant0[s];
95 ant0[s]=-1;
96 }
97 }
```

```
//Update Pheromone
for(int k=0; k<ant_pop; k++)
{
for(int s=0; s<nE; s++)
{ANT0[s]=ant[k][s];}
ANT0[nE]=ant[k][0];
for(int i=0; i<nE; i++)
{TAU[ANT0[i]][ANT0[i+1]]=TAU[ANT0[i]][ANT0[i+1]]+(Q/fitness[k]);}
}
for(int i=0; i<nE; i++)
{
for(int j=0; j<nE; j++)
{TAU[i][j]=(1-rho)*TAU[i][j];}
}
cout<<best_fit<<endl;
}
//Printing best solution
cout<<"Best ant: ["<<best_ant[0];
for(int i=1; i<nE; i++)
{cout<<", "<<best_ant[i];}
cout<<"]"<<endl;
cout<<"Best fitness: "<<best_fit<<endl;
getchar();
return 0;
}
```

Listing 1.4
Source-code of ACO algorithm in C++.

1.6 Step-by-step numerical example of ACO algorithm

Example 1 *Solve the traveling salesman problem (TSP) for five cities.*

Solution: In order to solve the TSP problem for five cities, the coordinates or distances between the cities are needed. We assume the following coordinates for these cities:

$X = [79, 87, 13, 67, 63]$
$Y = [31, 90, 23, 82, 17]$

So, the dimension or number of variables $N = 5$ for this problem.

In the first step, we define the ACO algorithm parameters. For example, population size = 4, maximum iteration = 1, $\alpha = 1.0$, $\beta = 1.0$, $\tau_0 = 0.01$, and $\rho = 0.02$.

In the second step, the euclidean distances between these cities are determined. The distance between city '1' (79, 31) to city '2' (87, 90) is calculated as:

$$D(1,2) = D(2,1) = \sqrt{\big(X(1) - X(2)\big)^2 + \big(Y(1) - Y(2)\big)^2}$$

$$= \sqrt{\big(79 - 87\big)^2 + \big(31 - 90\big)^2} = 59.54$$

Similarly, others are also calculated and presented as:

$$D = \begin{bmatrix} 0.00 & 59.54 & 66.48 & 52.39 & 21.26 \\ 59.54 & 0.00 & 99.82 & 21.54 & 76.84 \\ 66.48 & 99.82 & 0.00 & 79.98 & 50.36 \\ 52.39 & 21.54 & 79.98 & 0.00 & 65.12 \\ 21.26 & 76.84 & 50.36 & 65.12 & 0.00 \end{bmatrix}$$

In the third step, heuristic η, also known as visibility of the path, is determined as:

$$\eta = 1/D = \begin{bmatrix} \infty & 0.0168 & 0.0150 & 0.0191 & 0.0470 \\ 0.0168 & \infty & 0.0100 & 0.0464 & 0.0130 \\ 0.0150 & 0.0100 & \infty & 0.0125 & 0.0199 \\ 0.0191 & 0.0464 & 0.0125 & \infty & 0.0154 \\ 0.0470 & 0.0130 & 0.0199 & 0.0154 & \infty \end{bmatrix} \quad \% \text{ element-wise inversion}$$

In the fourth step, the initial pheromone intensity, τ is randomly allocated to edges, as suggested below.

$$\tau = \tau_0 \times rand(N, N) = \begin{bmatrix} 0.0059 & 0.0021 & 0.0077 & 0.0023 & 0.0013 \\ 0.0034 & 0.0086 & 0.0017 & 0.0062 & 0.0078 \\ 0.0034 & 0.0036 & 0.0054 & 0.0063 & 0.0008 \\ 0.0053 & 0.0085 & 0.0099 & 0.0079 & 0.0002 \\ 0.0038 & 0.0098 & 0.0038 & 0.0010 & 0.0079 \end{bmatrix}$$

In the fifth step, we start the iteration loop of the ACO algorithm. It is checked whether the maximum number of iterations is attained. If yes, we jump to the eighteenth step; otherwise move to the sixth step.

In the sixth step, a random vertex/node/city is chosen to start the journey of ant-1. Suppose it is city '3' then ant-1 will start its journey from node/city 3.

In the seventh step, we start the ants updating the loop. If the paths of all ants are covered then we jump to the fifteenth step, otherwise move to the eighth step.

In the eighth step, we determine the probabilities for ant-1 to take paths originating from city '3' to other cities, by using (1.1). The probability of ant-1 to visit city '1' is calculated as:

$NUM = \tau(:,3)^{\alpha} \star \eta(:,3)^{\beta}$ % numerator of probability function, component-wise multiplication

$$NUM = \begin{bmatrix} 0.0077 \\ 0.0017 \\ 0.0054 \\ 0.0099 \\ 0.0038 \end{bmatrix}^{\alpha=1} \star \begin{bmatrix} 0.0150 \\ 0.0100 \\ 0 \\ 0.0125 \\ 0.0199 \end{bmatrix}^{\beta=1} = \begin{bmatrix} 0.000116 \\ 0.000017 \\ 0.000000 \\ 0.000124 \\ 0.000075 \end{bmatrix}$$

$DEM = \sum \tau(:,3)^{\alpha} \times \eta(:,3)^{\beta} == 0.000116 + 0.000017 + 0.000000 + 0.000124 + 0.000075 = 0.000332$ % denominator of probability function

Now, we determine the probability for ant-1 to visit other cities as follows:

$P_3^r = NUM/DEM = \begin{bmatrix}0.3488 & 0.0513 & 0 & 0.3727 & 0.2272\end{bmatrix}^T$ % component wise division.

In the ninth step, we apply the roulette wheel selection (RWS) criteria on P_3^r to select the most probable city to be visited by ant-1. Suppose city '4' is selected because it has the highest probability then the root of ant-1 is updated as:

$ant(1,:) = [3,4]$

In the tenth step, the eighth step is repeated to determine the probability of ant-1 to visit other cities from current city '4', as calculated below:

$P_4^r = \begin{bmatrix}0.1031 & 0.6759 & 0.1850 & 0 & 0.0361\end{bmatrix}^T$

In the eleventh step, the probability of already visited cities is set to zero, as suggested below. This will avoid the revisits of this ant to already visited cities.

$P_4^r = \begin{bmatrix}0.1031 & 0.6759 & 0 & 0 & 0.0361\end{bmatrix}^T$

Suppose ant-1 is visiting city '2' (assumed from RWS); therefore the completed path of this ant can be updated as:

$ant(1,:) = [3,4,2]$

Similarly other paths are also determined and ant-1 returns to city '3' after visiting all cities, as given below:

$ant(1,:) = [3,4,2,5,1,3]$

In the twelfth step, we calculate the fitness or distance traveled by this ant, as shown below.

$D(3,4) = 79.98$; $D(4,2) = 21.54$; $D(2,5) = 76.84$; $D(5,1) = 21.26$; $D(1,3) = 66.48$;

Therefore the fitness of ant-1 or distance traveled is calculated as:
$Fit(1) = 79.98 + 21.54 + 76.84 + 21.26 + 66.48 = 266.10$

By repeating the steps eighth to twelfth, the path traveled by other ants and respective distances are also calculated, as shown below.

$ant(1,:) = [3,4,2,5,1,3]$ $\quad Fit(1) = 266.10$
$ant(2,:) = [2,4,1,5,3,2]$ $\quad Fit(2) = 245.38$
$ant(3,:) = [5,3,4,2,1,5]$ $\quad Fit(3) = 232.68$
$ant(4,:) = [2,5,1,4,3,2]$ $\quad Fit(4) = 330.30$

In the thirteenth step, we compare the fitness of each ant with the current best ant and preserve the best ant that traveled through the shortest path, e.g., $best\_ant = ant(3,:) = [5,3,4,2,1,5]$ and $best\_fit = 232.68$.

In the fourteenth step, we return to the seventh step.

In the fifteenth step, we update the pheromone intensity τ on the paths followed by these ants. The substance laid by ant k on a path connecting cities i and j is updated as:

$\tau(i,j) = \tau(i,j) + 1/Fit(k)$

The pheromone laid by ant-1 is updated as:

$$\tau(3,4) = \tau(3,4) + 1/Fit(1) = 0.0063 + 1/(266.10) = 0.0063 + 0.0038 = 0.0101$$
$$\tau(4,2) = \tau(4,2) + 1/Fit(1) = 0.0085 + 1/(266.10) = 0.0085 + 0.0038 = 0.0123$$
$$\tau(2,5) = \tau(2,5) + 1/Fit(1) = 0.0078 + 1/(266.10) = 0.0078 + 0.0038 = 0.0116$$
$$\tau(5,1) = \tau(5,1) + 1/Fit(1) = 0.0038 + 1/(266.10) = 0.0038 + 0.0038 = 0.0076$$
$$\tau(1,3) = \tau(1,3) + 1/Fit(1) = 0.0077 + 1/(266.10) = 0.0077 + 0.0038 = 0.0115$$

Similarly, the pheromone laid by other ants is also updated on these paths, as presented below:

$$\tau = \begin{bmatrix} 0.0059 & 0.0021 & 0.0115 & 0.0053 & 0.0097 \\ 0.0077 & 0.0086 & 0.0017 & 0.0103 & 0.0146 \\ 0.0034 & 0.0107 & 0.0054 & 0.0144 & 0.0008 \\ 0.0094 & 0.0166 & 0.0129 & 0.0079 & 0.0002 \\ 0.0106 & 0.0098 & 0.0122 & 0.0010 & 0.0079 \end{bmatrix}$$

In the sixteenth step, we evaporate the ρ amount of pheromone. The remaining pheromone intensity is expressed as:

$$\tau = (1-\rho) * \tau = \begin{bmatrix} 0.0058 & 0.0021 & 0.0112 & 0.0052 & 0.0095 \\ 0.0075 & 0.0084 & 0.0017 & 0.0101 & 0.0143 \\ 0.0033 & 0.0105 & 0.0053 & 0.0141 & 0.0008 \\ 0.0092 & 0.0162 & 0.0127 & 0.0077 & 0.0002 \\ 0.0104 & 0.0096 & 0.0119 & 0.0010 & 0.0077 \end{bmatrix}$$

In the seventeenth step, we return to the fifth step.

In the eighteenth step, we print the best ant, i.e. $best\_ant$, and its fitness, $best\_fit$.

1.7 Conclusion

This chapter presents the standard version of the ACO algorithm. The optimal characteristics of ant colony are mathematically modeled into some set of equations. The artificial ant follows the shortest route between nest and food source. Matlab and C++ source codes are also presented in this chapter for hands-on practice. Furthermore, the step-by-step example is presented for beginners to understand the concept of basic ACO.

Acknowledgment

This work was supported by Marie Sklodowska-Curie Fellowship (COFUND-Multiply) received from the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie Grant Agreement no. 713694.

References

1. Colorni A, Dorigo M, Maniezzo V. "Distributed optimization by ant colonies" in Proceedings of the First European Conference on Artificial Life 1992 Dec (vol. 142, pp. 134-142).
2. Dorigo M, Birattari M, Sultzle T. "Ant colony optimization: Artificial ants as a computational intelligence technique", *IEEE Computational Intelligence Magazine*, vol 1, no. 4, 2006, pp. 28-39
3. Dorigo M, Di Caro G. "Ant colony optimization: a new meta-heuristic" in *Proceedings of IEEE Congress on Evolutionary Computation, (CEC-99)*, the IEEE; vol. 2. 6-9 July 1999. pp. 1470-1477
4. Blum C, Dorigo M. "The hyper-cube framework for ant colony optimization". *IEEE Trans Systems Man Cybernetics* B 2004; 34(2), pp. 1161-1172
5. Dorigo M, Maniezzo V, Colorni A. "Ant system: optimization by a colony of cooperating agents". *IEEE Trans Syst Man Cybern B Cybern*, 1996, doi:10.1109/3477.484436
6. Dorigo M. "Optimization, learning and natural algorithms". Ph.D. dissertation, Dipartimento di Elettronica, Politecnico di Milano, Italy, 1992.
7. Dorigo M, Gambardella LM. "Ant colony system: A cooperative learning approach to the traveling salesman problem". *IEEE Trans Evol Computat*, 1997; 1(1), pp. 53-66.
8. Lee K.Y., El-Sharkawi M.A. *Modern Heuristic Optimization Techniques: Theory and Applications to Power Systems*, John Wiley & Sons, vol. 39, 2008.