Ant Colony Optimization: A New Meta-heuristic

Authors:
Marco Dorigo Gianni A. Di Caro

Presented by:

Md. Zawad Abdullah (1605002)
Bishwajit Bhattacharjee (1605003)
Md. Mahir Shahriyar (1605024)

Department of CSE,
Bangladesh University of Engineering and Technology

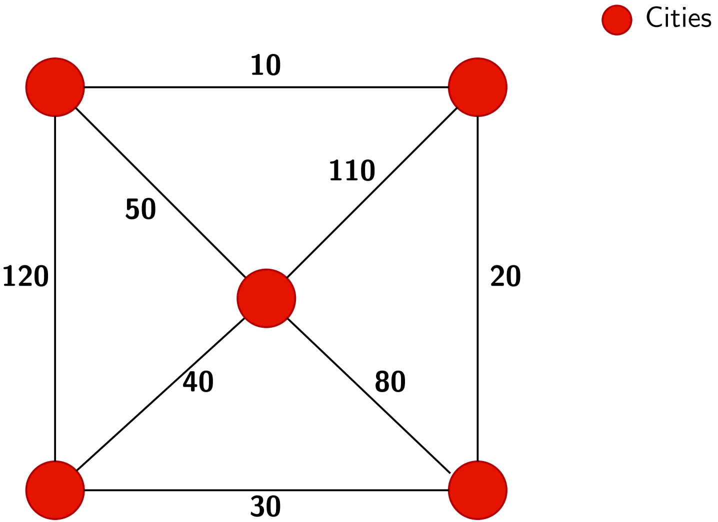
August 29, 2019

Problem Definition

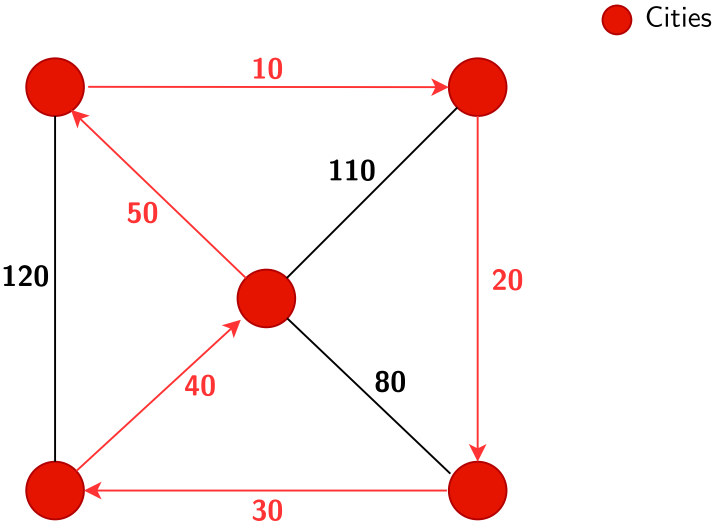
The Traveling Salesman Problem

A salesman needs to visit a number of customers located in different cities and return to the starting city using the shortest route.

Input:



Output:



Known Methods

- Backtracking

Known Methods

- Backtracking
  Issue - Complexity is exponential.

Known Methods

- Backtracking
  Issue - Complexity is exponential.
  Not Good Enough!

Known Methods

- Backtracking
  Issue - Complexity is exponential.
  Not Good Enough!
- Bitmask DP

Known Methods

- Backtracking
  Issue - Complexity is exponential.
  Not Good Enough!
- Bitmask DP
  Issue - Works efficiently when input is small.

Known Methods

- Backtracking
  Issue - Complexity is exponential.
  Not Good Enough!
- Bitmask DP
  Issue - Works efficiently when input is small.
  Not Good Enough!

Motivation

We will use Ant Colony Optimization (ACO) to solve TSP more efficiently.

Ants collecting food-1

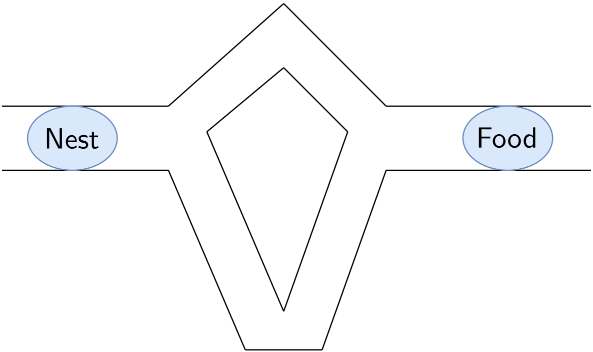


Figure: Paths From Food to Ants' Nest

Ants collecting food-2

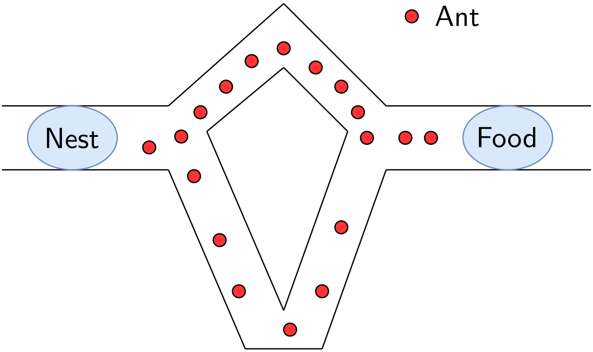


Figure: Ants Searching for Food

Ants collecting food-3

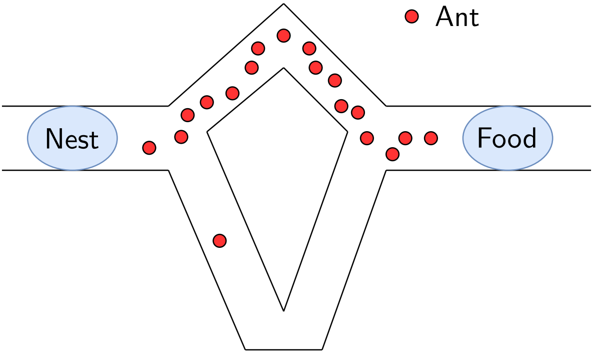


Figure: Ants Following An Optimal Path

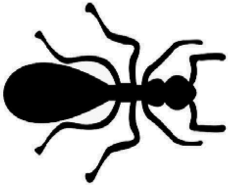
So how do they communicate??

So how do they communicate??

Pheromone

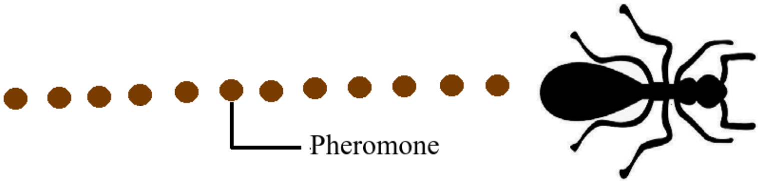
So how do they communicate??

Pheromone



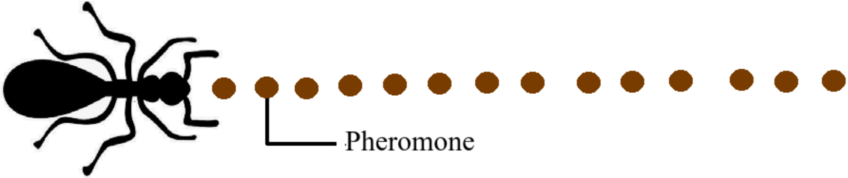
So how do they communicate??

Pheromone



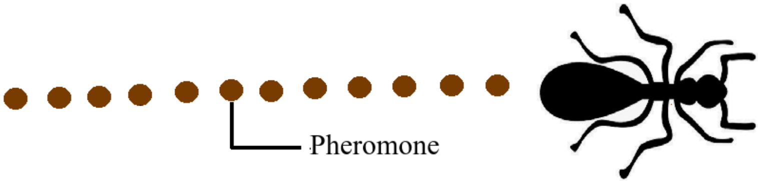
So how do they communicate??

Pheromone



So how do they communicate??

Pheromone



Previous Works

- In the year 1991, Marco Dorigo proposed an algorithm called "Ant System".
- AS was first applied to the Traveling Salesman Problem.

Figure: Marco Dorigo

Results

The ACO meta-heuristic is the result of an effort to define a common framework for all the versions of AS.

Notations

- $C = \{c_1, c_2, ..., c_{N_C}\}$ is a finite set of *components*.
- $L = \{l_{c_i c_j} \mid (c_i, c_j) \in \tilde{C}\}, \ |L| \leq N_C^2$ is a finite set of possible *connections/transitions* among the elements of $\tilde{C}$, where $\tilde{C}$ is a subset of the Cartesian product $C \times C$.
- $J_{c_i c_j} \equiv J(l_{c_i c_j}, t)$ is a *connection cost* function associated to each $l_{c_i c_j} \in L$, possibly parameterized by some time measure t.
- ψ is a *solution* of the problem.
- $J_\psi(L, t)$ is a *cost* associated to each solution ψ. $J_\psi(L, t)$ is a function of all the costs $J(c_i, c_j)$ of all the connections belonging to the solution ψ .

Adjacency Matrix

$$\begin{pmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & 0 & \times \\ a & b & c & d \end{pmatrix}$$

Here $A(i,j)$ means row i and col j

Ant Properties

Ants of the colony have the following properties :

- An ant searches for minimum cost feasible solutions $\hat{J}_\psi = min_\psi \ \hat{J}_\psi(L,t)$

Ant Properties

Ants of the colony have the following properties :

- An ant searches for minimum cost feasible solutions $\hat{J}_\psi = min_\psi \ \hat{J}_\psi(L, t)$
- An ant k has a memory $\mathcal{M}^k$ that it can use to store information on the path it followed so far. Memory can be used to build feasible solutions, to evaluate the solution found, and to retrace the path backward.

Ant Properties

Ants of the colony have the following properties :

- An ant searches for minimum cost feasible solutions $\hat{J}_\psi = min_\psi \ \hat{J}_\psi(L,t)$
- An ant k has a memory $\mathcal{M}^k$ that it can use to store information on the path it followed so far. Memory can be used to build feasible solutions, to evaluate the solution found, and to retrace the path backward.
- An ant k located on node i can move to a node j chosen in $\mathcal{N}_i^k$. The move is selected applying a probabilistic decision rule.

Ant properties continued...

- The ants' probabilistic decision rule is a function of (i) the values stored in a node local data structure $\mathcal{A}_i = [a_{ij}]$ called ant-routing table, obtained by a functional composition of node locally available pheromone trails and heuristic values, (ii) the ant's private memory storing its past history, and (iii) the problem constraints.

Ant properties continued...

- The ants' probabilistic decision rule is a function of (i) the values stored in a node local data structure $\mathcal{A}_i = [a_{ij}]$ called ant-routing table, obtained by a functional composition of node locally available pheromone trails and heuristic values, (ii) the ant's private memory storing its past history, and (iii) the problem constraints.
- When moving from node i to neighbor node j the ant can update the pheromone trail τ_{ij} on the arc (i, j) This is called *online step-by-step pheromone update*.

Ant properties continued...

- Once built a solution, the ant can retrace the same path backward and update the pheromone trails on the traversed arcs. This is called *online delayed pheromone update*.

Ant properties continued...

- Once built a solution, the ant can retrace the same path backward and update the pheromone trails on the traversed arcs. This is called *online delayed pheromone update*.
- Once it has built a solution, and, if the case, after it has retraced the path back to the source node, the ant dies, freeing all the allocated resources.

How The Method Works

- A colony of ants concurrently move through neighbor nodes of G using information stored in the node-local ant-routing tables.

How The Method Works

- A colony of ants concurrently move through neighbor nodes of G using information stored in the node-local ant-routing tables.
- By moving, they incrementally build solutions to the problem.

How The Method Works

- A colony of ants concurrently move through neighbor nodes of G using information stored in the node-local ant-routing tables.
- By moving, they incrementally build solutions to the problem.
- Once an ant has built a solution, it deposits information about the quality of the pheromone trails it used.

How The Method Works continued...

Besides ants' activities an ACO algorithm might include too more procedures:

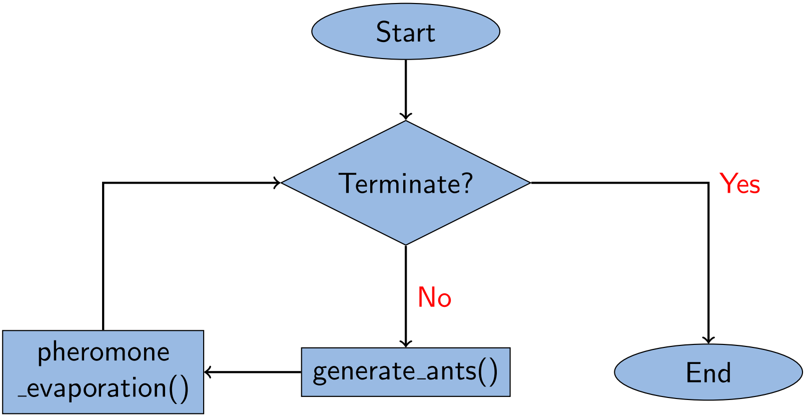
Pheromone Trail Evaporation

The process by means of which the pheromone trail intensity on the connections automatically decreases over time.

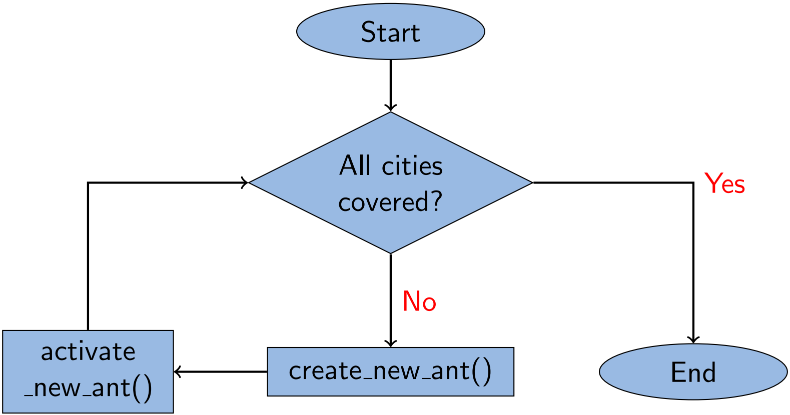
Daemon Actions

Used to implement centralized actions which cannot be performed by single ants.

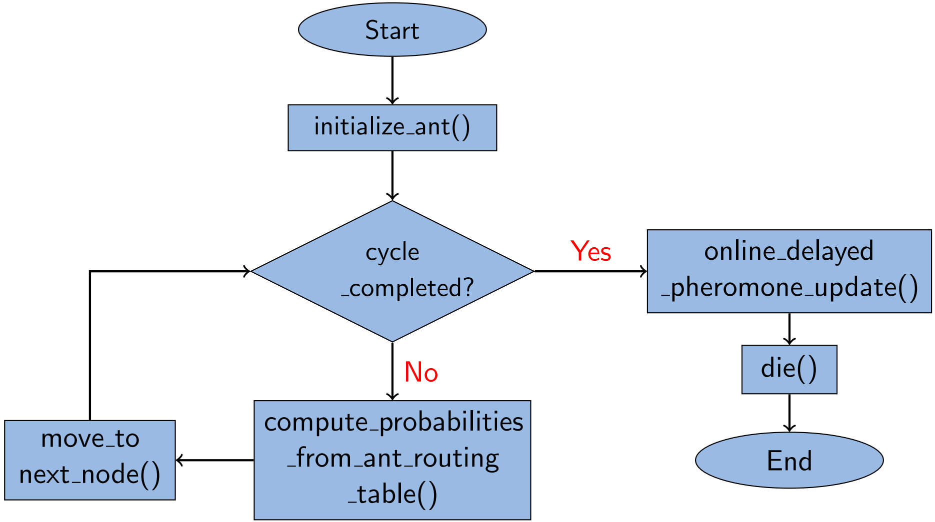
procedure ACO_meta-heuristic()



procedure generate_ants()



procedure activate_new_ant() {Ant lifecycle}



Traveling Salesman Problem

Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city and returns to the origin city?

ACO for the Traveling Salesman Problem

- We construct a graph $G = (C, L)$ where C is the set of components representing cities and L is the set of connections connecting the cities.
- $J_{c_i c_j}$ is the cost of the connection between the nodes c_i and c_j , the distance between cities i and j .
- A solution to this problem is a Hamiltonian circuit with minimal cost.

ACO for the Traveling Salesman Problem Continued

- A number m of ants are positioned randomly in parallel on m cities. Then, they enter a cycle which lasts N_C iterations.

ACO for the Traveling Salesman Problem Continued

- A number m of ants are positioned randomly in parallel on m cities. Then, they enter a cycle which lasts N_C iterations.
- During each step an ant located on node i, reads the entries of the ant-routing table. Then it computes the transition probabilities and chooses an adjacent node to move updating it's memory.

ACO for the Traveling Salesman Problem Continued

- A number m of ants are positioned randomly in parallel on m cities. Then, they enter a cycle which lasts N_C iterations.
- During each step an ant located on node i, reads the entries of the ant-routing table. Then it computes the transition probabilities and chooses an adjacent node to move updating it's memory.
- Once an ant has completed a tour, it uses its memory to evaluate the built solution. Then it retraces the same tour backward and updates the pheromone trails of the used edges.

ACO for the Traveling Salesman Problem Continued

- A number m of ants are positioned randomly in parallel on m cities. Then, they enter a cycle which lasts N_C iterations.
- During each step an ant located on node i, reads the entries of the ant-routing table. Then it computes the transition probabilities and chooses an adjacent node to move updating it's memory.
- Once an ant has completed a tour, it uses its memory to evaluate the built solution. Then it retraces the same tour backward and updates the pheromone trails of the used edges.
- It uses its memory to avoid visiting a city twice.

ACO for the Traveling Salesman Problem Continued

The formula for updating the ant-routing table is:

$$a_{ij} = \frac{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{l \in \mathcal{N}_i} [\tau_{il}(t)]^{\alpha}[\eta_{il}]^{\beta}} \qquad \forall j \in \mathcal{N}_i$$

- τ_{ij} is the intensity of pheromone trail of the edge l_{ij}
- η_{ij} is the heuristic value of the edge between i and j.

$$\eta_{ij} = \frac{1}{J_{c_i c_j}}$$

- α and β are two parameters that control the relative weight of pheromone trail and heuristic value.

ACO for the Traveling Salesman Problem Continued

The probability $p_{ij}^k(t)$ with which an ant k located in city i chooses the city $j \in \mathcal{N}_i^k$ to move to at the t-th iteration is:

$$p_{ij}^k(t) = \frac{a_{ij}(t)}{\displaystyle\sum_{l \in \mathcal{N}_i^k} a_{il}(t)}$$

where $\mathcal{N}_i^k \subseteq \mathcal{N}_i$ is the feasible neighborhood of node i for ant k.

ACO for the Traveling Salesman Problem Continued

After pheromone updating has been performed by the ants, pheromone evaporation is triggered: the following rule is applied to all the edges l_{ij} of the graph G

$$\tau_{ij}(t) \leftarrow (1-\rho)\tau_{ij}(t)$$

where $\rho \in (0,1]$ is the pheromone trail decay coefficient.

Conclusions

In this paper we briefly described ACO and its basic applications. We mainly focused on the use in Traveling Salesman Problem.