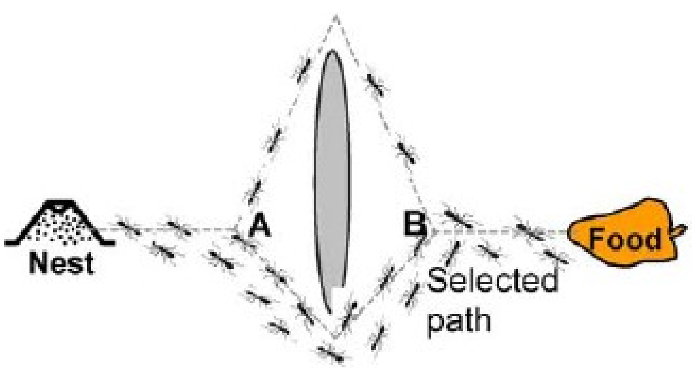
Ant Colony Optimization (ACO)



Submitted by:
Subvesh Raichand
MTech, 3rd Semester
07204007
Electronics and Communication

Optimization

General optimization problem:

given $f: X \rightarrow \mathbb{R}$,

find $x \in X$ such that $f(x)$ is minimum

- Given a graph with two specified vertices A and B, find a shortest path from A to B. → shortest path problem, polynomial
- Given a set of cities and pairwise distances, find a shortest tour.

  → traveling salesperson problem,
- Given a sequence of amino acids of a protein, find the structure of the protein. → protein structure prediction problem,
- 'Where is my manuscript for the talk, I put it on this pile of papers...'

  → needle in a haystack, hopeless

Ant Colony Optimization (ACO)

- In the real world, ants (initially) wander randomly, and upon finding food return to their colony while laying down pheromone trails. If other ants find such a path, they are likely not to keep traveling at random, but instead follow the trail laid by earlier ants, returning and reinforcing it if they eventually find food
- Over time, however, the pheromone trail starts to evaporate, thus reducing its attractive strength. The more time it takes for an ant to travel down the path and back again, the more time the pheromones have to evaporate.
- A short path, by comparison, gets marched over faster, and thus the pheromone density remains high
- Pheromone evaporation has also the advantage of avoiding the convergence to a locally optimal solution. If there were no evaporation at all, the paths chosen by the first ants would tend to be excessively attractive to the following ones. In that case, the exploration of the solution space would be constrained.

ACO

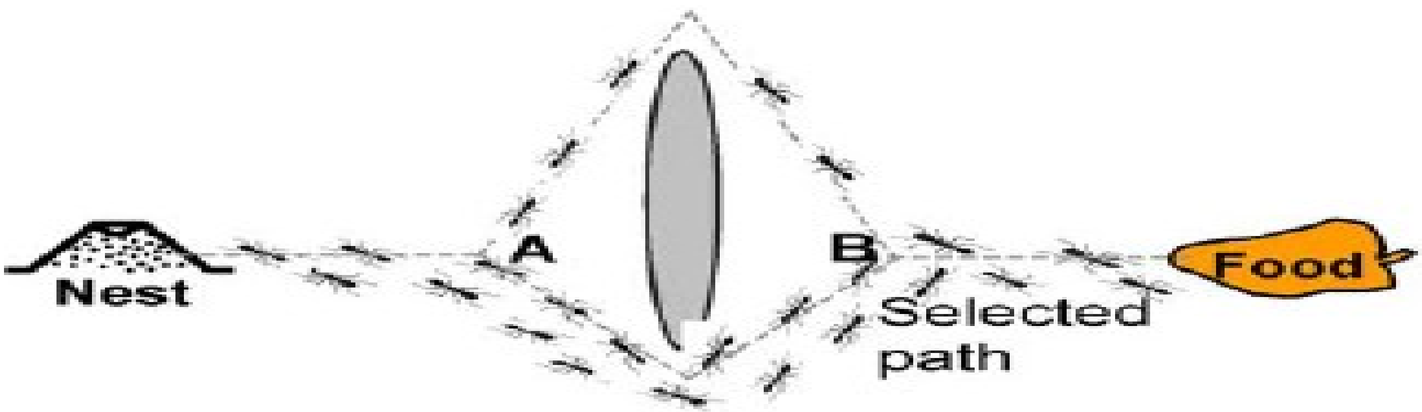
- Thus, when one ant finds a good (short) path from the colony to a food source, other ants are more likely to follow that path, and such positive feedback eventually leaves all the ants following a single path.
- The idea of the ant colony algorithm is to mimic this behavior with "simulated ants" walking around the search space representing the problem to be solved.
- Ant colony optimization algorithms have been used to produce near-optimal solutions to the traveling salesman problem.
- They have an advantage over simulated annealing and genetic algorithm approaches when the graph may change dynamically. The ant colony algorithm can be run continuously and can adapt to changes in real time.
- This is of interest in network routing and urban transportation systems.

ACO Defined

- A heuristic optimization method for shortest path and other optimization problems which borrows ideas from biological ants.
- Based on the fact that ants are able to find shortest route between their nest and source of food.

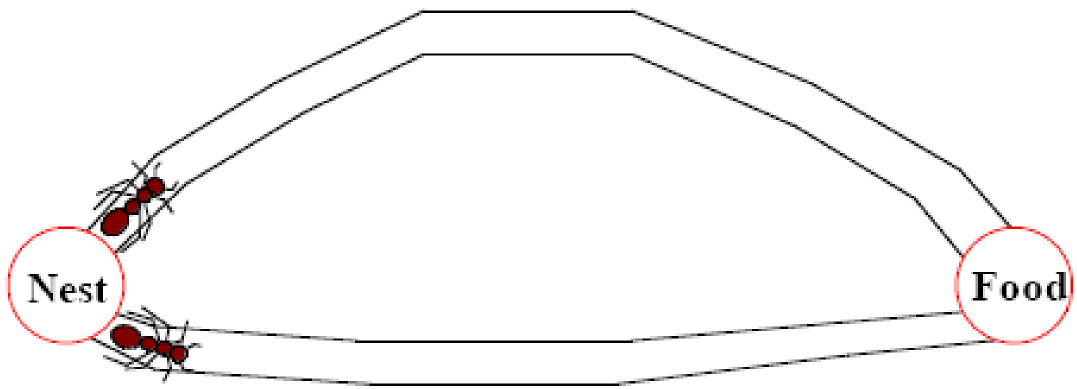
Shortest Route

- Shortest route is found using pheromone trails which ants deposit whenever they travel, as a form of indirect communication



Ant Colony Optimization

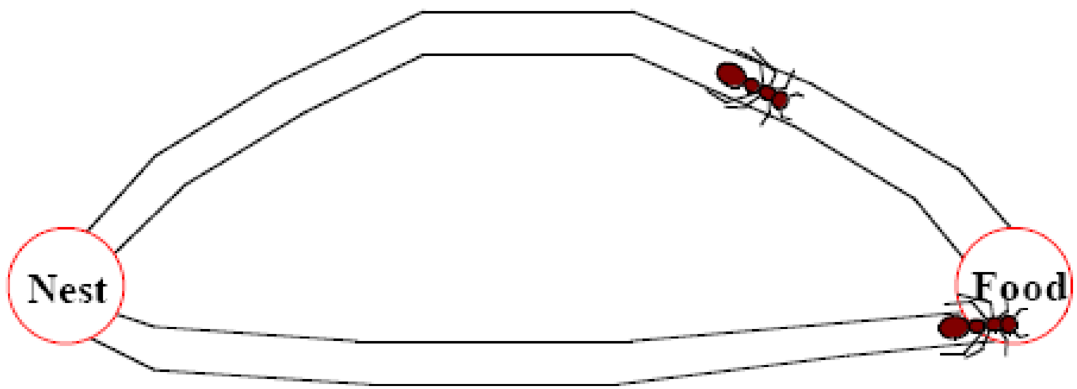
- Ant foraging – Co-operative search by pheromone trails



When ants leave their nest to search for a food source, they randomly rotate around an obstacle

Ant Colony Optimization

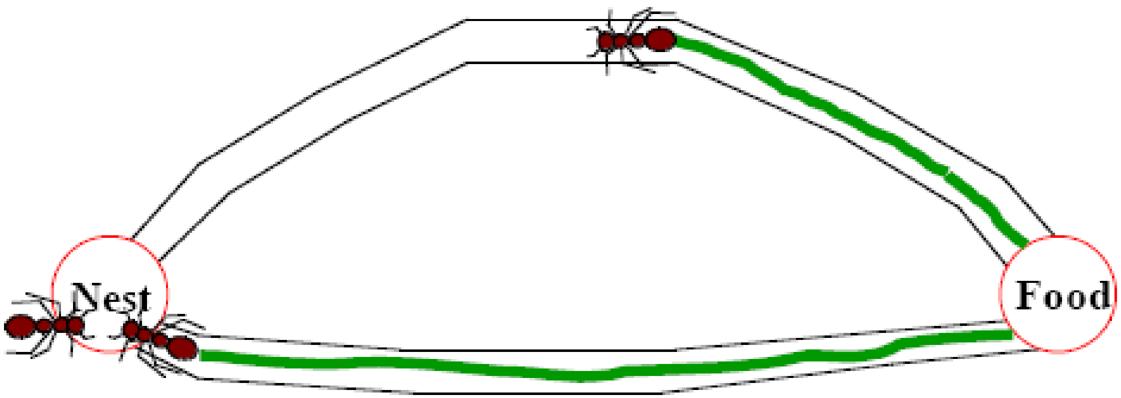
- Ant foraging – Co-operative search by pheromone trails



initially the pheromone deposits will be the same for the right and left directions

Ant Colony Optimization

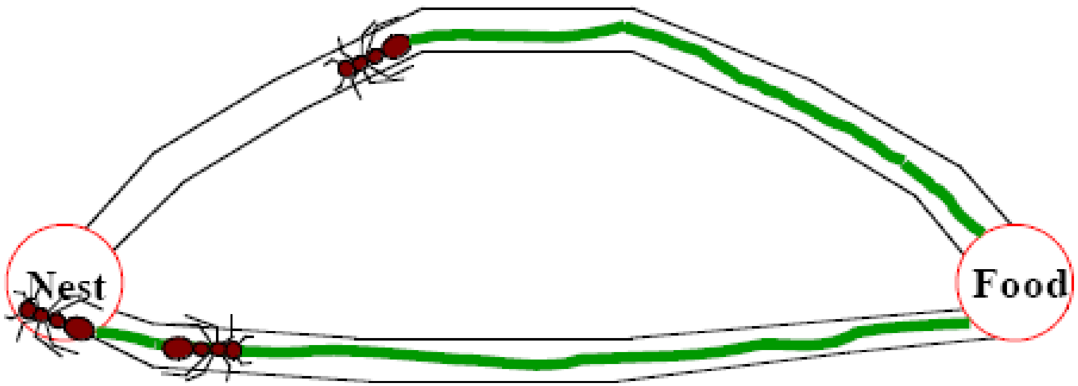
- Ant foraging – Co-operative search by pheromone trails



When the ants in the shorter direction find a food source, they carry the food and start returning back, following their pheromone trails, and still depositing more pheromone.

Ant Colony Optimization

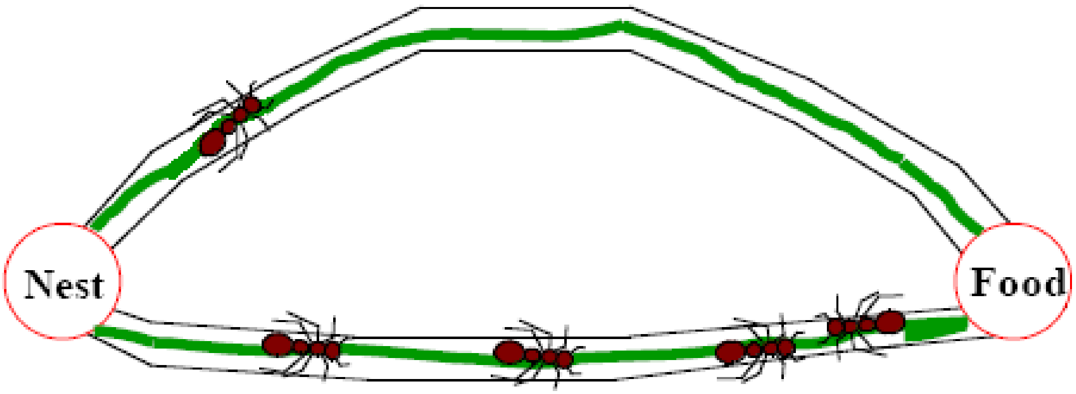
- Ant foraging – Co-operative search by pheromone trails



An ant will most likely choose the shortest path when returning back to the nest with food as this path will have the most deposited pheromone

Ant Colony Optimization

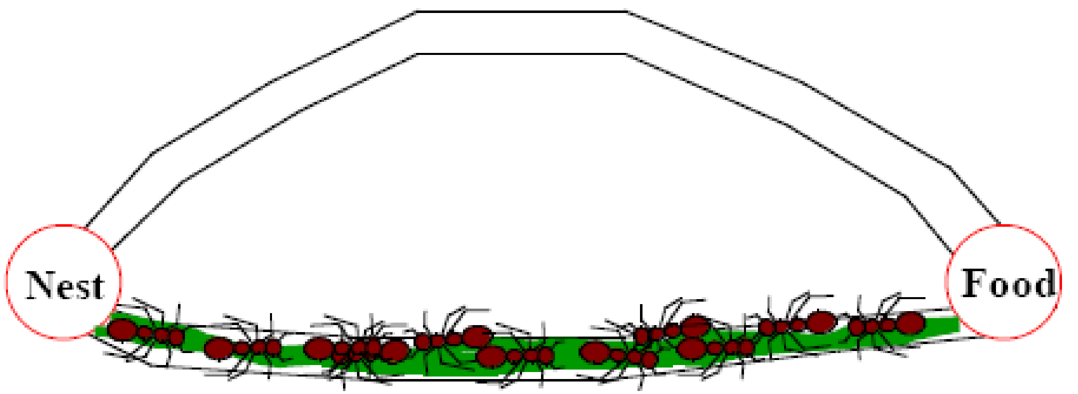
- Ant foraging – Co-operative search by pheromone trails



For the same reason, new ants that later starts out from the nest to find food will also choose the shortest path.

Ant Colony Optimization

- Ant foraging – Co-operative search by pheromone trails



Over time, this positive feedback (autocatalytic) process prompts all ants to choose the shorter path

ACO algorithm

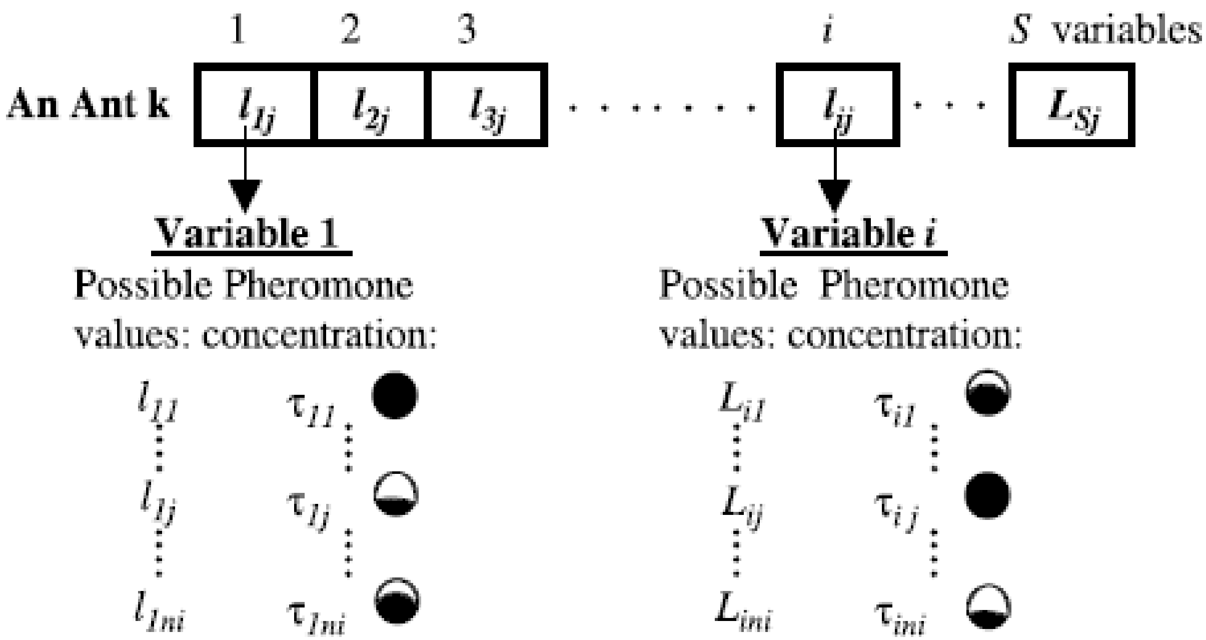
- The process starts by generating m random ants (solution).
- An ant $k \ (k=1,\ 2,\ldots,\ m)$ represents a solution string, with a selected value for each variable.
- An ant is evaluated according to an objective function.
- Accordingly, pheromone concentration associated with each possible route(variable value) is changed in a way to reinforce good solutions as follows:

ACO algorithm

$$\tau_{ij}(t) = \rho\tau_{ij}(t-1) + \Delta\tau_{ij}; \quad t = 1, 2, \ldots, T$$

where T is the number of iterations (generation cycles); $\tau_{ij}(t)$ is the revised concentration of pheromone associated with option l_{ij} at iteration t, $\tau_{ij}(t-1)$ is the concentration of pheromone at the previous iteration $(t-1)$; $\Delta\tau_{ij}$ = change in pheromone concentration; and ρ = pheromone evaporation rate (0–1). The reason for allowing pheromone evaporation is to avoid too strong influence of the old pheromone to avoid premature solution stagnation

ACO algorithm Implementation



Implementing the ACO for a certain problem requires a representation of S variables for each ant, with each variable i has a set of n_i options with their values l_{ij}, and their associated pheromone concentrations $\{\tau_{ij}\}$; where $i = 1, 2, \ldots, S$, and $j = 1, 2, \ldots, n_i$. As such, an ant is consisted of S values that describe the path chosen by the ant as shown above.

ACO Pseudo Code

Pseudo code for ACO is shown below:

```
Begin;
    Initialize the pheromone trails and parameters;
        Generate population of m solutions (ants);
        For each individual ant k∈m: calculate fitness (k);
        For each ant determine its best position;
        Determine the best global ant;
        Update the pheromone trail;
    Check if termination=true;
End;
```

Pheromone Concentration Calculations

pheromone concentration $\Delta\tau_{ij}$ is calculated as

$$\Delta\tau_{ij} = \sum_{k=1}^{m} \begin{cases} R/\text{fitness}_k & \text{if option } l_{ij} \text{ is chosen by ant } k \\ 0 & \text{otherwise} \end{cases}$$

where R is a constant called the pheromone reward factor; and fitness_k is the value of the objective function (solution performance) calculated for ant k. It is noted that the amount of pheromone gets higher as the solution improves.

Pheromone Concentration Calculation

Once the pheromone is updated after an iteration, the next iteration starts by changing the ants' paths (i.e. associated variable values) in a manner that respects pheromone concentration and also some heuristic preference. As such, an ant k at iteration t will change the value for each variable according to the following probability

$$P_{ij}(k,t) = \frac{[\tau_{ij}(t)]^{\alpha} \times [\eta_{ij}]^{\beta}}{\sum_{l_{ij}} [\tau_{ij}(t)]^{\alpha} \times [\eta_{ij}]^{\beta}}$$

where $P_{ij}(k, t)$ = probability that option l_{ij} is chosen by ant k for variable i at iteration t; $\tau_{ij}(t)$ = pheromone concentration associated with option l_{ij} at iteration t; η_{ij} = heuristic factor for preferring among available options and is an indicator of how good it is for ant k to select option l_{ij} (this heuristic factor is generated by some problem characteristics and its value is fixed for each option l_{ij}); and α and β are exponent parameters that control the relative importance of pheromone concentration versus the heuristic factor

Main parameters at glance

the main parameters involved in ACO are: number of ants m; number of iterations t; exponents α and β; pheromone evaporation rate ρ; and pheromone reward factor R.

ACO Characteristics

- Exploit a positive feedback mechanism
- Demonstrate a distributed computational architecture
- Exploit a global data structure that changes dynamically as each ant transverses the route
- Has an element of distributed computation to it involving the population of ants
- Involves probabilistic transitions among states or rather between nodes

References:

- Emad Elbeltagi, Tarek Hegazy, Donald Grierson, Comparison among five evolutionary-based optimized algorithm, 19 January 2005,Advanced Engineering informatics 19 (2005) 43-53
- www.google.com
- www.sciencedirect.com

Thank You!