ANTI COLONY OPTIMAZTION (ACO) ALGORITHMS: A SURVEY

Mr. Tushar Mehrotra, Janvi Jain , Tanisha Jain

Abstract— During most engineering projects, optimization is required due to limited resources and equipment. Hence, this paper introduces a new probabilistic method to resolve computational problems that boils down to get the best optimum paths via graphs called Ant colony Optimization algorithm (ACO) Many optimization tasks involvs graphs, such as vehicle routing and internet routing, are best solved by combining local search algorithims and artifical ants. Similarly, this study presents an Ant Colony Optimization (ACO) framework that solves a dynamic traveling salesman problem. By maintaining diversity, pheromone trails transfer information from previous environments.

Introduction

The optimization algorithm involves comparing the various solutions until one is found to be optimal or satisfactory iteratively Optimization is an integral component of computer-aided design due to the advent of computers. . Optimization algorithms fall into two categories:a

1.**Deterministic algorithm**:- A specific set of rules is used to move from one solution to another. Many engineering design problems have been successfully solved using the seal algorithms.

2.**Stochastic algorithm**:- Stochastic algorithms have probabilistic translation rules by nature. The popularity of these algorithms is due to certain properties that deterministic algorithms lack.

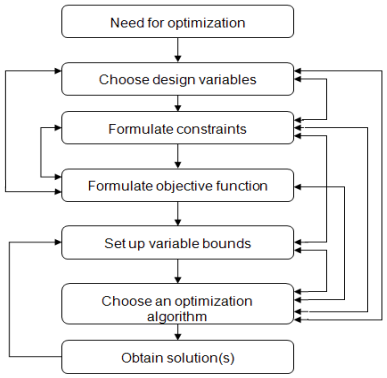


Fig.1 A flowchart of the optimal design procedure.

As opposed to AntiColony Optimization, which is a metaheuristic probabilistic search algorithm. There are several main advantages to metaheuristic searches:
1.**Broad applicability**: They can be used to tackle any problems whose solution can be formulated as a function optimization problem.
2.**Hybridization**: Optimization techniques can be combined with these techniques.

History :

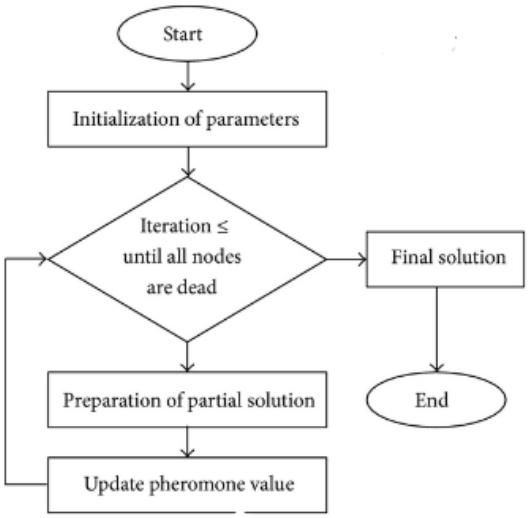
The first ant system was proposed in 1991 by Marco Dorigo in his doctoral thesis. An ant colony system, known as ant-q, was introduced in 1996 by Gambardella and Dorigo. An ACO conference was held for the first time in 1998 by Dorigo. A short leap in 2000 led Hoos and Stützle to invent the max-min ant system. **Company (Eurobios and Antoptima) uses the COA algorithm in 2001, and the first application is peptide sequence design, made in 2016.** ACO had successfully integrated the PROMETHEE method in 2017. As a result of these searches, the ant system-based approaches are often among the most efficient methods of determining the optimum meta-heuristics. In ACO algorithms, the shortest route is determined by laying pheromones and selecting them.

In swarm intelligence methods, this algorithm belongs to the ant colony algorithm family. Based on an idea developed by studying ants, several new numerical problems have developed in light of it. ACO is similar to an estimation of distribution algorithm from the perspective that it performs a model-based search. Schedules of nursing shifts, trains,

timetables, traveling salesman problems, vehicle routing scenarios, and group shops are real-world examples.

Ant Colony Optimization:-

ACO is a probabilistic problem. The famous TSP can also be solved using a similar method. Ant colonies are known to travel across terrain for food, and this algorithm mimics the ant's behavior since they communicate with a natural organic compound called pheromones. Ants use a pheromone to communicate. Once it finds food, the ant deposits pheromones on the path. When the ant returns from checking out the food, new pheromone deposits in their way. Ants can detect pheromones. If there are more pheromones in the area, there is a high chance of finding the ideal path for food.



Variants of AOC:-

A. ANT SYSTEM:

An early ACO algorithm was known as ANT SYSTEM. Several ACO algorithms are derived from it. The formal description of the Ant System for solving TSP is as follows: A TSP is constructed by AS by following the steps below:

1. *When there are n edges, place k*n ants at random positions on them.*
2. *Each ant completes its tour (Hamiltonian cycle) in the following way:*
   - *Given the working memory of ant i to be* M′k *to keep track of the vertices already traveled by the ant.*

Where M′k=∅ *at initialization for all i. If an ant visits vertex* v *v, then* v *is added to the memory.*

An ant in city x *visits city* y *among the cities not yet* visited *by computing its probability if* y *is not yet visited or 0 otherwise:* $\tau(x,y)\alpha.\gamma(x,y)\beta/\Sigma j=1\tau(x,z)\alpha.\ \tau(x,z)\beta$

1. *After iteration, according to the following set of rules the pheromone gets updated :*
   $\tau(x,y)\leftarrow(1-\rho).\tau(x,y)+\Sigma i=1\Delta\tau l(x,y)$

Where p (0,1) is the evaporation rate, I iterate between 1 and k; the number of ants and $\Delta\tau'(x, y)$ is the amount of pheromone deposited on the edge (x, y) by ant I which is computed by if (x, y) is not part of the edges visited by ant me or 0 otherwise:

$\Delta\tau l(x,y)=C/Li(3)$
Li = length of tour of ant i and
C is a constant

1. *Iterate until you receive the maximum.*

5. Return the length of the best tour.

It becomes impossible to implement a solution when many cities are greater than 30. With this algorithm, the TSP problem with multiple cities doesn't seem to be so reliable.

B. ANT COLONY SYSTEM:

The algorithm is one of the best and has been explained as follows:

A local pheromone update rule for ACS is as follows:

$$\tau(x,y)\leftarrow(1-\rho).\tau(x,y)+\rho\tau0$$

The ACS local pheromone update rules aim to help ants explore different edges to produce different results. There is no guarantee that an ant will find a similar solution in a single iteration.

By using the following formula, the ant constructs the shortest tour ever since the algorithm began or at the end of iteration:

$$\tau(x,y)\leftarrow(1-\rho).\tau(x,y)+\rho.Lost-1.$$

C. MAX-MIN ANT SYSTEM:

It uses search history to increase pheromones by only allowing the optimum solutions to be used during the updation of pheromone trail. The MMAS algorithm has also been enhanced to learn the optimum solutions and use them during an efficient search to overcome early stagnation.

Pheromone Update, let's say it evaluates to ☐:

$$☐ \leftarrow \tau(x,y) \leftarrow (1-\rho) .\tau(x,y)+\Delta\tau best(x,y)$$

Two variables determine the bounds of the pheromone: τmax (upper bound) and Tmin (lower bound). ☐ evaluates to any three values: τmax if ☐>τmax,τmin if ☐<τmin and ☐ otherwise. –When (x,y) is a part of the best ant's tour, best(x,y) evaluates to a value derived from (7) – Otherwise it evaluates to 0.

$$\Delta\tau best(x,y) \leftarrow 1/Lbest$$

Lbest is the best ants length, which can either be the global best length or the best after iteration. The algorithm implementer determined τmax and τmin based on observations and experiments.

D. RANK-BASED ANT SYSTEM

In the Rank-Based ants, are ordered by the length of their respective tours that completes. Track upgrades based on the rank of each ant (μ). After this, only a few elite ants come into consideration.
Pheromone update of the ASRank algorithm is given by;

$$\tau(x,y) \leftarrow (1-\rho). \tau(x,y)+\Sigma\mu=1\Delta\tau\mu(x,y)+\Delta\tau best(x,y)$$

$$\Delta\tau\mu(x,y) \leftarrow (\sigma-\mu)Q/L\mu$$

$$\Delta\tau best(x,y) \leftarrow \sigma(Q/Lbest)$$

Where σ is the number of elitist ants, μ = ranking index.Δτμ(x,y) = increase of trail level on edge (i, j) used by the μ-th best ant in its tour, the value becomes zero if the μ-th best ant does not pass through the edge (x, y).
Δτbest(x,y) = quantity of pheromone laid on node (i, j) used by the elitist ants in their tour, evaluates to zero if the edge (i, j) is not part of the optimum solution found.
Lμ = tour length covered by μth best ant ,
Lbest = tour length covered by the "best" ant.

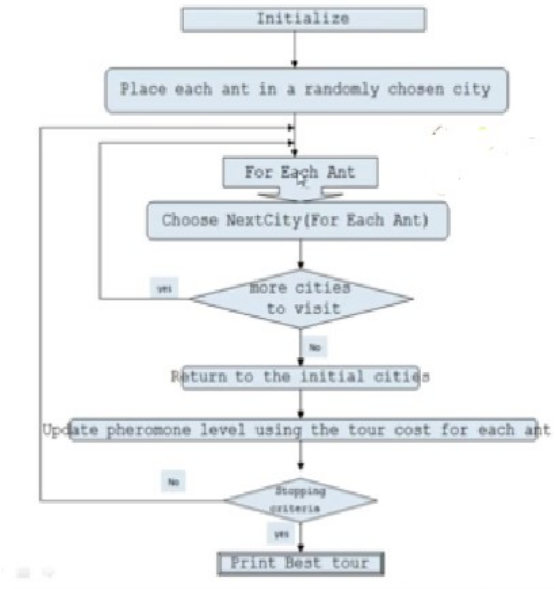
Travelling Salesperson Problem(TSP):-

It is a theoretical and combinatorial problem. Traveling salesman problem (TSP) attempts to determine the shortest route between two points. The points in the problem statement refer to the cities that a salesperson might visit. The salesman must provide a bunch of cities and the separation between each of them. To reach his destination, he should visit one city and take the most optimal path back home.

5.Proposed Approach:-

Several cities (nodes) should assign to each group of candidates, along with their distances. To find the Hamiltonian tour, each group should only visit each city once. Estimate the minimal distance (costs, time, money, energy, etc).

Nodes will be labeled after agents are placed in randomly chosen cities. As each node selects the next node to move to, the agents will know what to do next. Once the agents reach the parents starting node, the pheromone value is updated using the tour costs. Once all groups have completed the tour, the amount of pheromone produced by each group will determine the best tour.



When the number of cities increases, the complexity of the tour cycle increases. The algorithm divides and conquers used to solve the problem. Hence, the process (p) got broken up into two subprocesses (p1, p2). Ant colonies will optimize all sub-processes. Results of both sub-processes (p1 and p2)

combined with a nearby city. Thus, the problem has been resolved.

6.Conclusion:-

An algorithm based on the most popular ACO has been used to tackle the Travel salesman problem . This algorithm uses the divide-and-conquer method to optimize colonies. As a result of the analysis, a rich pheromone edge can solve the traveling salesman's problems. This study analysis how ACO can handle such variations with efficiency. ACO will also be studied further to develop an understanding of its theoretical properties in the future. This paper also discusses variants of ACO algorithms.