**Travelling Salesman Problem using ACO**

A Project Report

Submitted in the completion of the course

Mathematical Programming -2

In

Department of Computer Science and Engineering

By

2010030071 – JYOTHIN MOVVA

2010030151 – SATYAVARSAN KS

2010030040 – DEVARAJ ACHARYA

Under the supervision of

# Mr. G.Madhukar Rao

# ASSISTANT PROFESSOR



Department of Computer Science and Engineering

K L University Hyderabad,

Aziz Nagar, Moinabad Road, Hyderabad – 500 075, Telangana, India.

April, 2022

**DECLARATION**

The Project Report entitled “Simulated Annealing Algorithm for Travelling Salesman Problem” is a record of bonafide work of N.Hemanth Srivathsav (2010030113), Peri Vishwanadha Sastry (2010030470), M.Abhiram Sharma (2010030523), P.Vinay Kumar (2010030469), submitted in partial fulfillment for the award of B.Tech in the Department of Computer Science and Engineering to the K L University, Hyderabad. The results embodied in this report have not been copied from any other Departments/University/Institute.

2010030071 – JYOTHIN MOVVA

2010030151 – SATYAVARSAN KS

2010030040 – DEVARAJ ACHARYA

**CERTIFICATE**

* This is to certify that the Project Report entitled “ANT COLONY OPTIMIZATION” is being submitted by Jyothin Movva (2010030071), K S Satyavarsan(2010030151), Devaraj Acharya (2010030040) submitted in partial fulfillment for the award of B.Tech in B.Tech in CSE to the K L University, Hyderabad is a record of bonafide work carried out under our guidance and supervision.

The results embodied in this report have not been copied from any other departments/universities/institutes.

## Signature of the supervisor

Mr. G.Madhukar Rao

Assistant Professor

## Signature of the HOD Signature of the External Examiner

**ACKNOWLEDGEMENT**

First and foremost, we thank the lord almighty for all his grace & mercy showered upon us, for completing this project successfully.

We take a grateful opportunity to thank our beloved **Founder and Chairman** who has given constant encouragement during our course and motivated us to do this project. We are grateful to our principal

**Dr. L. Koteswara Rao** has been constantly bearing the torch for all the curricular activities undertaken by us.

We pay our grateful acknowledgment & sincere thanks to our Head of the Department **Dr. Chiranjeevi Manike** for his exemplary guidance, monitoring, and constant encouragement throughout the course of the project. We thank **Mr. G.Madhukar Rao** of our department who has supported us throughout this project holding the position of supervisor.

We wholeheartedly thank all the teaching and non-teaching staff of our department without whom we won’t have made this project a reality. We would like to extend our sincere thanks, especially to our parents, our family members, and friends who have supported us to make this project a grand success.

**ABSTRACT**

The ant colony optimization algorithm (ACO) is a probabilistic method for solving computing problems that may be simplified to finding good paths through graphs. And is used in computer science and operations research. Multi-agent approaches inspired by the behavior of actual ants are referred to as artificial ants. Biological ants utilize pheromones to communicate.   Artificial ants and local search algorithms have been the go-to method for a variety of graph optimization tasks.

Ant colony optimization is a type of optimization technique inspired by the behavior of ants. Artificial ants (for example, simulation agents) search for optimal solutions by traversing a parameter space that represents all possible outcomes. While exploring their surroundings, real ants leave pheromones that direct each other to resources. The simulated 'ants' also keep track of their positions and the quality of their solutions, so that more ants find better solutions in subsequent simulation iterations.

**TABLE OF CONTENT**

|  |  |  |
| --- | --- | --- |
| **S.no** | **Topics** | **Page Number** |
| **1** | Introduction | **7** |
| **2** | Literature Survey | **8** |
| **3** | Model & techniques | **9** |
| **4** | Hardware and Software Requirements | **10** |
| **5** | Implementation | **11-12** |
| **6** | Conclusion | **13** |
| **7** | Bibliography | **14** |

1. **INTRODUCTION**

Ant colony optimization algorithm (ACO) is a probabilistic technique for solving computational problems which can be reduced to finding good paths through graphs. Algorithms such as the Particle Swarm Optimization (PSO) and Ant Colony Optimization (ACO) are examples of swarm intelligence and metaheuristics.

Optimization techniques have evolved based on evolutionary algorithms and thereby opened the domain of metaheuristic. ACO technique is purely inspired from the foraging behavior of ant colonies.

Ants 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.

Ants live in community nests and the underlying principle of ACO is to observe the movement of the ants from their nests to search for food in the shortest possible path. This randomized search opens multiple routes from the nest to the food source.

**2. LITERATURE SURVEY**

**Title:**

Ant colony optimization

**Author(s):**

Marco Dorigo; Mauro Birattari; Thomas Stutz

**Publishing:**

2011 IEEE Congress of Evolutionary Computation (CEC)

**Title:**

Ant colony optimization: Introduction and recent trends

**Author(s):**

Slimane RS

**Publishing:**

2011 IEEE Congress of Evolutionary Computation (CEC)

**Title:**

Comparative Study on Ant Colony Optimization (ACO) and K-Means Clustering Approaches for Jobs Scheduling and Energy Optimization Model in Internet of Things (IoT)

**Author(s):**

Sumit Kumar1, Vijender Kumar Solanki, Saket Kumar Choudhary , Ali Selamat Rubén González Crespo

**Publishing:**

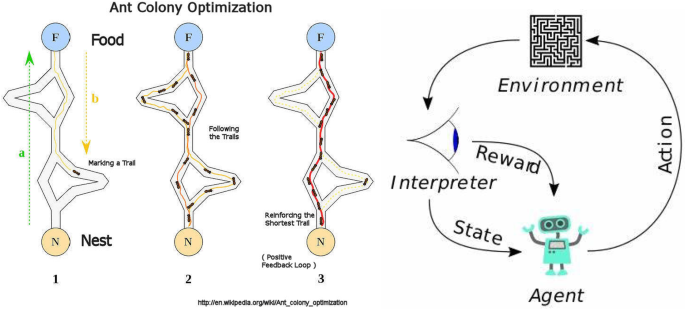
Accepted 22 January 2020 | Published 27 January 2020

1. **MODEL & TECHNIQUES**

**Ant colony optimization**

Marco Dorigo introduced ant colony optimization (ACO) in his Ph.D. thesis in the 1990s. This method is based on an ant's foraging activity when looking for a path between their colony and a source of food. It was first used to tackle the famous travelling salesman problem. It is then used to solve many difficult optimization problems.

Ants live in colonies. They dwell in groups called colonies. The ants' behaviour is governed by their desire to find food. Ants moving about their habitats while hunting An ant hops from one location to another in search of food. It leaves an organic molecule called pheromone on the ground as it moves. Pheromone trails are used by ants to communicate with one another. When an ant comes across some food, it carries as much as it can. It deposits pheromone on the pathways according on the quantity and quality of the food when it returns. Pheromone can be detected by ants. As a result, other ants can smell it and follow the trail. The higher the pheromone level, the more likely that path will be chosen.



1. **HARDWARE AND SOFTWARE REQUIREMENTS**

**HARDWARE REQUIREMENTS:**

**Operating System:** Any operating system

**Supporting System:** 64-Bit Operating System, x64-based processor

**Processor:** Intel® Core i5 7th Gen 2.50GHz

**RAM:** 8 GB

**SOFTWARE REQUIREMENTS:**

**Software:**  Pycharm Community Edition

**Programming Language:** Python 3.7

**Important Packages Requirements:** Matplotlib

1. **IMPLEMENTATION**

import random as rn  
import numpy as np  
from numpy.random import choice as np\_choice  
  
class AntColony(object):  
  
 def \_\_init\_\_(self, distances, n\_ants, n\_best, n\_iterations, decay, alpha=1, beta=1):  
   
 self.distances = distances  
 self.pheromone = np.ones(self.distances.shape) / len(distances)  
 self.all\_inds = range(len(distances))  
 self.n\_ants = n\_ants  
 self.n\_best = n\_best  
 self.n\_iterations = n\_iterations  
 self.decay = decay  
 self.alpha = alpha  
 self.beta = beta  
  
 def run(self):  
 shortest\_path = None  
 all\_time\_shortest\_path = ("placeholder", np.inf)  
 for i in range(self.n\_iterations):  
 all\_paths = self.gen\_all\_paths()  
 self.spread\_pheronome(all\_paths, self.n\_best, shortest\_path=shortest\_path)  
 shortest\_path = min(all\_paths, key=lambda x: x[1])  
 print (shortest\_path)  
 if shortest\_path[1] < all\_time\_shortest\_path[1]:  
 all\_time\_shortest\_path = shortest\_path   
 self.pheromone = self.pheromone \* self.decay   
 return all\_time\_shortest\_path  
  
 def spread\_pheronome(self, all\_paths, n\_best, shortest\_path):  
 sorted\_paths = sorted(all\_paths, key=lambda x: x[1])  
 for path, dist in sorted\_paths[:n\_best]:  
 for move in path:  
 self.pheromone[move] += 1.0 / self.distances[move]  
  
 def gen\_path\_dist(self, path):  
 total\_dist = 0  
 for ele in path:  
 total\_dist += self.distances[ele]  
 return total\_dist  
  
 def gen\_all\_paths(self):  
 all\_paths = []  
 for i in range(self.n\_ants):  
 path = self.gen\_path(0)  
 all\_paths.append((path, self.gen\_path\_dist(path)))  
 return all\_paths  
  
 def gen\_path(self, start):  
 path = []  
 visited = set()  
 visited.add(start)  
 prev = start  
 for i in range(len(self.distances) - 1):  
 move = self.pick\_move(self.pheromone[prev], self.distances[prev], visited)  
 path.append((prev, move))  
 prev = move  
 visited.add(move)  
 path.append((prev, start)) # going back to where we started   
 return path  
  
 def pick\_move(self, pheromone, dist, visited):  
 pheromone = np.copy(pheromone)  
 pheromone[list(visited)] = 0  
  
 row = pheromone \*\* self.alpha \* (( 1.0 / dist) \*\* self.beta)  
  
 norm\_row = row / row.sum()  
 move = np\_choice(self.all\_inds, 1, p=norm\_row)[0]  
 return move  
  
distances = np.array([[np.inf, 2, 2, 5, 7],  
 [2, np.inf, 4, 8, 2],  
 [2, 4, np.inf, 1, 3],  
 [5, 8, 1, np.inf, 2],  
 [7, 2, 3, 2, np.inf]])  
  
ant\_colony = AntColony(distances, 1, 1, 100, 0.95, alpha=1, beta=1)  
shortest\_path = ant\_colony.run()  
  
print ("shorted\_path: {}".format(shortest\_path))

**Result**

Finding the optimal Path

A computer screen capture

Description automatically generated with medium confidence

1. **CONCLUSION**

we have presented a method for solving the multiple traveling salesman problem based on the improved ant colony optimization. The improved ACO has used the relationship between MTSP and 1-MST, and the simplified pheromone diffusion. The new pheromone update rules helped a lot to achieve a better solution. The experiment results have shown that the new method has quick convergence speed and can be well applied to find best solutions. In future, we decide to combine multi-objective TSP with parallel processing Parallel processing can improve algorithm speed and reduce algorithm execution time. This method may be greatly improve the speed of finding best solutions. Furthermore, we will implement an application that cannot only help the logistics distribution industry but also make the transportation in earthquake relief work or military operation easier.Further it can be improvised and solve

1. **BIBLIOGRAPHY**

<https://en.wikipedia.org/wiki/Ant_colony_optimization_algorithms>

<https://www.geeksforgeeks.org/introduction-to-ant-colony-optimization/>

<https://ieeexplore.ieee.org/document/4129846>

<https://www.sciencedirect.com/science/article/abs/pii/S1571064505000333>

<https://www.ijimai.org/journal/bibcite/reference/2750>