“Minimizing the total length of salesperson”

MP

Report

Submitted in the partial fulfillment of the requirements for the award of the degree of

# Bachelor of Technology in

Department of Computer Science and Engineering

By

B Avinash (2010030358)

CSaiSantosh(2010030033)

,ChShravan(2010030536)

,M Akhil (2010030513)

under the supervision of

<“Mr Anal paul”>



Department of Computer Science and Engineering

K L University Hyderabad,

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

March, 2022

**Declaration**

The MP Report entitled “Minimizing the total length of salesperson” is a

record of bonafide work of

B Avinash (2010030358)CSaiSantosh(2010030033),ChShravan(2010030536)

,M Akhil (2010030513),,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.

**Certificate**

This is to certify that the Mp Report entitled

Minimizing the total length of salesperson” is being submitted by B Avinash (2010030358)CSaiSantosh(2010030033),ChShravan(2010030536),M Akhil (2010030513)

, submitted in partial fulfillment for the award of 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/ University/Institute.

## Signature of the Supervisor

Name and Designation

## 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 Social Internship successfully.

We take grateful opportunity to thank our beloved Founder and Chairman who has given constant encouragement during our course and motivated us to do this Social Internship. We are grateful to our Principal **Dr. L. Koteswara Rao** who has been constantly bearing the torch for all the curricular activities undertaken by us.

We pay our grateful acknowledgement & sincere thanks to our Head of the Department **Dr. Chiranjeevi Manike** for her exemplary guidance, monitoring and constant encouragement throughout the course of the Social Internship. We thank **Mr anal paul**of our department who has supported throughout this Social Internship holding a position of supervisor.

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

**ABSTRACT**

Now a days we have delivery system in food groceries and other things.

But it is a problem to find a shortest path to cover all required cites or nodes and return back to the initial city. This is the case of a multiple travelling salesman problem (MTSP).for this we can use ACO(ANT COLONY OPTIMIZATION) technique to solve and obtain the shortest path.

ACO is nature inspired algorithm based on the behavior of ants.

T The optimization of logistics distribution can be defined as the multiple traveling salesman problem (MTSP). The purpose of existing heuristic algorithms, such as Genetic Algorithm (GA), Ant Colony Algorithm (ACO), etc., is to find the optimal path in a short time. However, two important factors of logistics distribution optimization, including work time window and the carrying capacity of the vehicle in distribution system, have been ignored. In this paper, we consider the influences of time limitation of modern commercial logistics and carrying capacity of the vehicle on the logistics optimization, and then propose a MTSP with constraints of time window and capacity of each salesman. We design a novel hybrid algorithm by combining the minimum spanning 1-tree with ACO to find the optimal solution. In addition, we improve the pheromone update rules to increase the search efficiency of ACO algorithm. The experiments show that the novel hybrid algorithm achieves a shorter path than the other algorithms.

**TABLE OF CONTENTS**

1. Introduction
2. RELATED WORK
3. PROBLEM DEFINITION AND FORMULATION
4. ANT COLONY OPTIMIZATION
5. Implementation
6. Results Discussion
7. Conclusion and Future Work
8. References

Introduction

Logistics distribution industry becomes more and more important due to the rapid development of the e-business. Nowadays, there are many challenges in logistics distribution industry. For example, Nov. 11th is a peak date for online shopping that all the products online have a big discount and people go on a shopping spree, so there are billions of express items needed to be delivered during that time. Therefore, it is important to deliver the products to customer within a short time. Service composition is a way which can combine all the logistics services processes efficiently. Another challenge is the balance between paths and items and it is more difficult.The logistics services process in China is to deliver all items from the head office to each brand. At the same time, the optimization of time cost and road distance, the amount of items, the number of transport vehicles as well as their carrying capacity are taken into account. The associate editor coordinating the review of this manuscript and approving it for publication was Shouguang Wang . MTSP is a generalization of the well-known traveling salesman problem (TSP), where at least two salesmen could be used to deliver items in the solution. Obviously, the logistics distribution question is the same as a multiple traveling salesman problem with constraints. Moreover, MTSP could be extended to a wide variety of vehicle routing problems (VRP) according to incorporating a number of additional side-constraints. In addition, finding a quick and efficient way to transport goods and materials in earthquake relief work or military operation is significant. There are many solutions to solve TSP like Tabu search, simulated annealing, genetic algorithms. The ant colony system (ACS) is improved by this paper which is also a well-behaved solution. It is a kind of the group of metaheuristic methods. This idea was firstly proposed in the early 90s by Dorigo . ACO is generated by simulating the real behavior of ants in nature and it is widely used to solve practical problems Many different transforms of ACO have been proposed. On the basic of these, a new optimal ACO with some improvements is proposed in the paper.

First of all, the minimum spanning 1-tree can be applied to construct the path in MTSP. Secondly, the improved model and pheromone update rules make the algorithm more effective. Our main contributions are as followings: (1) Redefining multiple traveling salesman problem with time window and capacity of each salesman, which comes from real life problem such as logistics industry; (2) Building an improved pheromone model with new pheromone update rules; (3) Combining the minimum spanning 1-tree with ACO based on the relationship between the optimal TSP tours and spanning tree. The rest of the paper is organized as follows. Section II shows the related work about TSP and ACO on solving TSP. In the section III, the definition of MTSP and possible solutions for MTSP are discussed. Section IV discusses the algorithm of ant colony system. Section V proposes an optimal ACO algorithm on the basic of previous section. Section VI shows the evaluation and experiment of the improved algorithm. Section VII concludes the paper and outlines the future work.

RELATED WORK

TSP is a typical branch of metaheuristic algorithm and there is a lot of research on it. We list a number of representative works recently, and there is an overview article by Panwar and Gupta , where earlier works are discussed in detail. They conclude some TSP for solving soft computing techniques. (1) Ant Colony Optimization has been widely used in proving the complexity of combinatorial problem. The movement of each ant is effected by the intensity of pheromones contained on the path. The ACO algorithm can calculate good results in a short time after a few iterations . (2) In ], Ding et al. try to look for the shortest Hamiltonian cycle in all clusters, then remove a selected edge in every cycle to build an intra-cluster path. At last, attach the whole intra-cluster paths in a particular sequence to organize a complete tour. A novel method called Two-level Genetic Algorithm for the Cluster TSP is proposed, which is a good choice to alter an extensive TSP into a CTSP that can be settled by the TLGA. This method is well applied to large-scale TSPs. (3) In , the conventional particle swarm optimization (PSO) is improved. This method takes the heuristic factor, crossover operator and adaptive disturbance factor into account. Then, the authors propose a novel hybrid discrete PSO algorithm which can improve the search performance in convergent speed and precision. This method can be applied to solve the problem of path optimization in TSP. (4) In , random keys are introduced to solve the coding problem of genetic algorithm. Random keys play an important role in the problems where permutations of the integers are required as well as feasibility problems caused by traditional one-or two-point crossover. Also, there are some other algorithms related to TSP. For example, in , Xu et al. redefine a general colored traveling salesman problem and propose a Delaunay-triangulation-based Variable Neighborhood Search algorithm. The new algorithm performs well in large-scale problems.

PROBLEM DEFINITION AND FORMULATION

TSP is a typical combinational optimization problem that belongs to NP complete problem [10]. The aim is to find the shortest path by visiting every node once and then returning to the start node in a complete weighted graph G with n nodes and n(n-1) edges. Furthermore, MTSP could use more than one salesman which is different from the TSP with only one salesman. When considering the carrying capacity and time windows, this problem becomes more meaningful in real life. The MTSP with constraints can be defined as follows. First, given a set of nodes, arrange m salesmen at a single depot node. Then, finding tours for all m salesmen, and they all need to start and end at the depot. Meanwhile, the other nodes can be visited exactly once and the cost of visiting all nodes will be least. The two constraints are: every node is visited in a fixed time window and every salesman should take items as many as possible [32]. Specifically, for a city i, the salesman should visit it in a time window Ei, Li , where Ei is the earliest time that the salesman could visit and the salesman should visit city i before Li . And for every salesman, the maximum items they can carry is M. Comparing the TSP with the MTSP, it is obvious that MTSP is more adequate to simulate real life situations, and it is capable of handling plenty of salesmen problem. MTSP can be applied in the following context: print scheduling, workforce planning, transportation planning, mission planning, production planning, satellite systems, etc. Each context includes numerous types of real applications. Consequently, there are various solutions proposed for MTSP. At first, some accurate solutions like cutting plane, branch and bound, and Lagrange relaxation were proposed. Then, heuristics are applied to this problem. Moreover, evolutionary algorithm, simulated annealing, tabu search, genetic algorithms and neural networks also have a good ability on working out the MTSP

ANT COLONY OPTIMIZATION

Ant colony optimization is a heuristic algorithm which is introduced into many combinatorial optimization problems due to it is one of the highest performance computing methods for MTSP [33]. Many efforts have been made on ant colony optimization techniques in different areas.

A picture containing timeline

Description automatically generated

Timeline

Description automatically generated with medium confidence

## Diagram Description automatically generated

Diagram

Description automatically generated

It takes inspiration from ants’ social behaviors of finding the best way between food and the ant nest. Ants can communicate with each other about amount of food and the distance of route. The process can be divided into two parts and a positive feedback mechanism is applied by the ants randomly and releases pheromone in the routes. A global pheromone updating rule is applied when all ants have finished their tours. Pheromone accumulates at a higher rate on the shorter path, so that more ants begin to choose the shorter path which has a higher intensity 106874 VOLUME 8, 2020 M. Wang et al.: Ant Colony Optimization With an Improved Pheromone Model for Solving MTSP of pheromone. At the second part, ants affect each other by pheromone which is the main factor to lead the ants to find shorter route. Then the process is iterated to find the global optimal solution. In ACO, artificial ant replaces the real ant. They both can affect surrounding environment such as the intensity of pheromone in the path

Implementation

Main.py

import tsplib95

import matplotlib.pyplot as plt

import networkx as nx

import random

from colony import Colony

from solver import Solver

problem = tsplib95.load\_problem('bays29.tsp')

G = problem.get\_graph()

solver = Solver()

colony = Colony(1, 3)

sales = 5

ans = solver.solve(G, colony, sales, start=13, limit=50, opt2=20)

print(sum(s.cost for s in ans), ans)

# draw

colors = ['black', 'blue', 'green', 'red', 'pink', 'orange']

plt.figure(dpi=300)

\_, ax = plt.subplots()

pos = problem.display\_data or problem.node\_coords

nx.draw\_networkx\_nodes(G, pos=pos, ax=ax, node\_color=(0.4157, 0.3529, 0.3490))

nx.draw\_networkx\_labels(G, pos=pos, labels={i: str(i) for i in range(1, len(G.nodes) + 1)}, font\_size=8, font\_color='white')

for i in range(len(ans)):

    solution = ans[i]

    path = solution.path

    nx.draw\_networkx\_edges(G, pos=pos, edgelist=path, arrows=True, edge\_color=colors[i])

    # nx.draw\_networkx\_edges(G, pos=pos, edgelist=path, arrows=True, edge\_color=[random.random() for i in range(3)])

# If this doesn't exsit, x\_axis and y\_axis's numbers are not there.

ax.tick\_params(left=True, bottom=True, labelleft=True, labelbottom=True)

plt.show()

ant.py

from solution import Solution

import itertools

import bisect

import random

class Ant:

    def \_\_init\_\_(self, alpha, beta):

        self.alpha = alpha

        self.beta = beta

        self.sales = None

        self.graph = None

        self.n = None

    def tour(self, graph, sales, start, opt2):

        self.graph = graph

        self.sales = sales

        self.n = len(graph.nodes)

        solutions = [Solution(graph, start, self) for \_ in range(sales)]

        saleses = [(self.n - 1) // sales for i in range(sales)]

        for i in range((self.n - 1) % sales):

            saleses[i] += 1

        unvisited = [i for i in range(1, self.n + 1) if i != start]

        for i in range(sales):

            for j in range(saleses[i]):

                next\_node = self.choose\_destination(solutions[i].current, unvisited)

                solutions[i].add\_node(next\_node)

                unvisited.remove(next\_node)

            solutions[i].close()

        if opt2:

            self.opt2\_update(graph, opt2, sales, saleses, solutions)

        return solutions

    def opt2\_update(self, graph, opt2, sales, saleses, solutions):

        for i in range(sales):

            for j in range(opt2):

                k = saleses[i] + 1

                while True:

                    a = random.randint(0, k - 1)

                    b = random.randint(0, k - 1)

                    if a != b:

                        break

                if a > b:

                    a, b = b, a

                dist\_a = graph.edges[solutions[i].nodes[a], solutions[i].nodes[a + 1]]['weight']

                dist\_b = graph.edges[solutions[i].nodes[b], solutions[i].nodes[(b + 1) % k]]['weight']

                dist\_c = graph.edges[solutions[i].nodes[a], solutions[i].nodes[b]]['weight']

                dist\_d = graph.edges[solutions[i].nodes[a + 1], solutions[i].nodes[(b + 1) % k]]['weight']

                if dist\_a + dist\_b > dist\_c + dist\_d:

                    solutions[i].nodes[a + 1:b + 1] = reversed(solutions[i].nodes[a + 1: b + 1])

                    solutions[i].cost += (dist\_c + dist\_d - dist\_a - dist\_b)

                    solutions[i].path = []

                    for l in range(k):

                        solutions[i].path.append((solutions[i].nodes[l], solutions[i].nodes[(l + 1) % k]))

    def choose\_destination(self, current, unvisited):

        if len(unvisited) == 1:

            return unvisited[0]

        scores = self.get\_scores(current, unvisited)

        return self.choose\_node(unvisited, scores)

    def choose\_node(self, unvisited, scores):

        total = sum(scores)

        cumdist = list(itertools.accumulate(scores))

        index = bisect.bisect(cumdist, random.random() \* total)

        return unvisited[min(index, len(unvisited) - 1)]

    def get\_scores(self, current, unvisited):

        scores = []

        for node in unvisited:

            edge = self.graph.edges[current, node]

            score = self.score\_edge(edge)

            scores.append(score)

        return scores

    def score\_edge(self, edge):

        weight = edge.get('weight', 1)

        if weight == 0:

            return 1e200

        phe = edge['pheromone']

        return phe \*\* self.alpha \* (1 / weight) \*\* self.beta

A screenshot of a computer

Description automatically generated with medium confidence

Results Discussion

Chart

Description automatically generated

Ant -1

Chart, scatter chart

Description automatically generated

Ant-2

Chart, scatter chart

Description automatically generated

Ant -3

Chart, radar chart, scatter chart

Description automatically generated

Ant-4

Chart, radar chart, scatter chart

Description automatically generated

Ant -5

Chart, radar chart, scatter chart

Description automatically generated

Ant -6

Chart, radar chart, scatter chart

Description automatically generated



Conclusion and Future Work

In this paper, 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

REFERENCES

[1] X. Sun, S. Wang, Y. Xia, and W. Zheng, ‘‘Predictive-trend-aware composition of Web services with time-varying quality-of-service,’’ IEEE Access, vol. 8, pp. 1910–1921, Dec. 2020. [2] S. Wang, D. You, and M. Zhou, ‘‘A necessary and sufficient condition for a resource subset to generate a strict minimal siphon in S 4PR,’’ IEEE Trans. Autom. Control, vol. 62, no. 8, pp. 4173–4179, Aug. 2017. [3] S. Pang, M. Wang, S. Qiao, X. Wang, and H. Chen, ‘‘Fault diagnosis for service composition by spiking neural P systems with colored spikes,’’ Chin. J. Electron., vol. 28, no. 5, pp. 1033–1040, Sep. 2019. [4] M. Drigo, V. Maniezzo, and A. Colorni, ‘‘The ant system: Optimization by a colony of cooperation agents,’’ IEEE Trans. Syst. Man, Cybern. B, Cybern., vol. 26, no. 1, pp. 29–41, Feb. 1996. [5] Y. Feng, M. Zhou, G. Tian, Z. Li, Z. Zhang, Q. Zhang, and J. Tan, ‘‘Target disassembly sequencing and scheme evaluation for CNC machine tools using improved multiobjective ant colony algorithm and fuzzy integral,’’ IEEE Trans. Syst., Man, Cybern. Syst., vol. 49, no. 12, pp. 2438–2451, Dec. 2019