OPTIMIZED ROUTE PLANNING ALGORITHM FOR PACKAGE DELIVERY SYSTEM

PROJECT REPORT

Submitted by

BL.EN.U4ECE16112	REETIKA MONAVARTHI
BL.EN.U4ECE16327	MALLA POOJTIH
BL.EN.U4ECE16069	J SAI RUCHITHA

In partial fulfilment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

ELECTRONICS AND COMMUNICATION ENGINEERING

AMRITA SCHOOL OF ENGINEERING, BANGALORE

AMRITA VISHWAVIDYAPEETHAM

BANGALORE 560035

June 2020

AMRITA VISHWAVIDYAPEETHAM

AMRITA SCHOOL OF ENGINEERING, BANGALORE, 560035

BONA FIDE CERTIFICATE

This is to certify that the project entitled **"OPTIMIZED ROUTE PLANNING ALGORITHM FOR PACKAGE DELIVERY SYSTEM "** submitted by

BL.EN.U4ECE16113	REETIKA MONAVARTHI
BL.EN.U4ECE16327	MALLA POOJITH
BL.EN.U4ECE16069	J SAI RUCHITHA

in partial fulfilment of the requirements of the award of the **Degree Bachelor of Technology** in **"ELECTRONICS AND COMMUNICATION** Engineering" is a bona fide record of the work carried out under my (our) guidance and supervision at Amrita School of Engineering, Bengaluru.

Nandi Vardhan H R
Assistant Professor
Department of ECE

Dr. N.S.Murty
Professor & Chairman
Department of ECE

This project report was evaluated by us on…….....

EXAMINER 1
Ms. S. Lalitha
Assistant Professor (Sr.Gr.)
Department Of ECE

EXAMINER 2
Dr. Neelima N.
Assistant Professor (Sr. Gr.)
Department Of ECE

ACKNOWLEDGEMENT

Any achievement big or small should have a catalyst and a constant encouragement and advice to bring about the best efforts to do this project work. The satisfaction that accompanies the successful culmination of any task would be incomplete without mentioning those who made it possible because success is the epitome of hard work, determination, concentration and dedication. We would like to avail this opportunity to express our sincere thanks and gratitude to all the people who have helped us through our project.

We first give our sincere thanks to **AMMA** without whose grace we wouldn't have reached this stage.

We would like to express our sincere gratitude towards our Principal, **Dr. S. R. Nagaraja** for giving opportunity to manifest our ideas into real-time project.

We are grateful to **Prof. N.S.Murty**, Head of the Department of Electronics and Communication Engineering, for permitting us to make use of the facilities available in the department to carry out the project successfully.

We place on record and warmly acknowledge the continuous encouragement, invaluable supervision, timely suggestions and inspired guidance offered by our guide **Mr. Nandi Vardhan H R** Asst. Professor, Department of Electronics and Communication Engineering in bringing this project to a successful completion. Lastly, we also like to thank our evaluators **Ms. S. Lalitha** and **Dr. Neelima N** for their rich experience filled discussions and suggestions about the project.

We would also like to express our sincere gratitude towards all the teachers of Electronics and Communication Department who have always been a great source of help, support and strength throughout the project. Last but not the least we express our sincere thanks to all our friends and family who have patiently extended all sorts of help for accomplishing this undertaking.

Thank You All.

ABSTRACT

The Purpose of this Paper is to give computational comparison of five heuristic algorithms in terms of quality and computation time. The final result of the computational tests is the conclusion that there are simple and easily implemented heuristic procedures that will produce high quality solutions to the TSP in a moderate amount of computer time. We also compare the operator of pursued approach which give the best result for finding the shortest path in a shortest time for moving toward the goal.

This paper presents five algorithms namely, Dijkstra's algorithm, genetic algorithm, Clarke and Wright savings algorithm, simulated annealing algorithm and tabu search algorithm to solve the TSP problem. Then these different algorithms are compared in the perspectives of time complexity, space complexity, difficulty level of realization, the advantages and disadvantages of the calculation results, and the coding part is done in python to get the results.

CONTENTS

LIST OF FIGURES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Traveling salesman problem (TSP) is the optimization problem of finding a shortest closed tour that visits all the given nodes only once and returns to the starting node. It is considered as a classical NP-hard problem, which has extremely large search spaces to evolve and is very difficult to get the result. Classical methods for solving TSP like Dijkstra's algorithm, Brute-Force search algorithm and Branch and Bound algorithm usually result in exponential computational complexities and makes it hard to obtain a solution.

Several problems in different fields like traffic, communications, military, logistics, etc., can be re-modelled to TSP problem models, thus we try to find efficient and effective methods for TSP problems with great attempts. Therefore, few deterministic algorithms such as branch and bound , Brute force search methods are applicable only to small sized TSP problems. In the real world TSP problems which have large input size cannot be practically solved by these algorithms as the computation time needed is too long,

A metaheuristic is a higher-level procedure used to find or generate a heuristic that can provide a sufficiently good solution to an optimization problem, especially with limited computation capability.
Metaheuristic optimization deals with optimization of problems using metaheuristic algorithms. Optimization is everywhere, from engineering design to economics. With limited money, time and resources, the optimal utility of these available resources is essentially important.

These algorithms are sufficiently complex to provide adaptive search results, and usually can be embedded with other heuristic algorithms to speed up their performances for optimizing problems.
The most used optimization algorithms are genetic algorithms, simulated annealing algorithm, Clarke and Wright’s savings algorithm, Ant colony optimization and Tabu search algorithm.

MOTIVATION

- Identifies the most Cost-Effective Routes
- Reduces Mileage, Fuel costs, Vehicle Maintenance cost
- Increase Response Time to last-minute scheduling changes
- Increase Customer Satisfaction
- Decreases Time spent planning routes
- Reduces Avoidable Transportation Delays
- Accurate calculation of distance and time required for delivering the products

PROBLEM STATEMENT

A Comparative study based on different optimization techniques to find cost and time efficient route.

SCOPE OF THE PROJECT

Each and every algorithm has some important parameters that could vary the outputs. All such parameter could be selected and a new hybrid algorithm can be devised to get more optimized results

CHAPTER 2

CHAPTER 2: LITERATURE SURVEY

2.1 PAPER 1: A Practical Delivery Route Planning System

This paper demonstrates a system that provides a practical solution for scheduling and planning parcel delivery routes. Given a parcel delivery workload, e.g., the number of parcels to be delivered and the sizes of the parcels, the system tries to identify a set of delivery routes such that the workload is satisfied and the total delivery cost is minimized.

This process is called last-mile delivery, which delivers parcels from distribution centres to final delivery destinations such as fixed pick-up points and ad-hoc home addresses, This is the most complex and expensive part of the whole delivery chain, which often gives negative impacts on pollution and congestion especially in highly populated areas. It uses a workload file which consists of a list of parcels that need to be delivered, where each parcel is associated with (1) a source, e.g., the distribution centre where the parcel is currently placed, (2) a destination to be delivered, e.g., a pick-up point or a home address, (3) a size information, e.g., in the form of length × width × height, (4) a weight, and (5) a preferred delivery interval, e.g., morning, afternoon, or a whole day. Based on the workload file, for each distribution centre, the system computes a set of routes to delivery all the parcels that are associated with the distribution centre with the least total travel distance. Then, each route is assigned to a vehicle. When computing the routes, the system also considers the capacities of vehicles such that the parcels to be delivered in one route do not exceed the capacity of a vehicle. The company provides a list of its distribution centres and fixed pick-up points. A workload file that consists of all parcels need to be delivered in one week is also provided.

The Input component takes road as an input, a list of distribution centres, and a list of fixed pickup points are made as a graph, where vertices represent road intersections, the distribution centres, or the fixed pickup points and edges represent road segments connecting vertices.

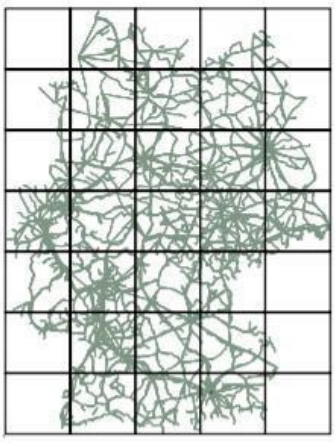
The Route planning component consists of two steps: route generation and route optimization. In the route generation step, based on a workload, we first group parcels to be

delivered by their sources, i.e., distribution centres. For each distribution centre, we run a route generation algorithm to identify the set of optimal routes to deliver all parcels for the distribution centre. The routes are computed using Open Source Routing Machine which only considers distance.

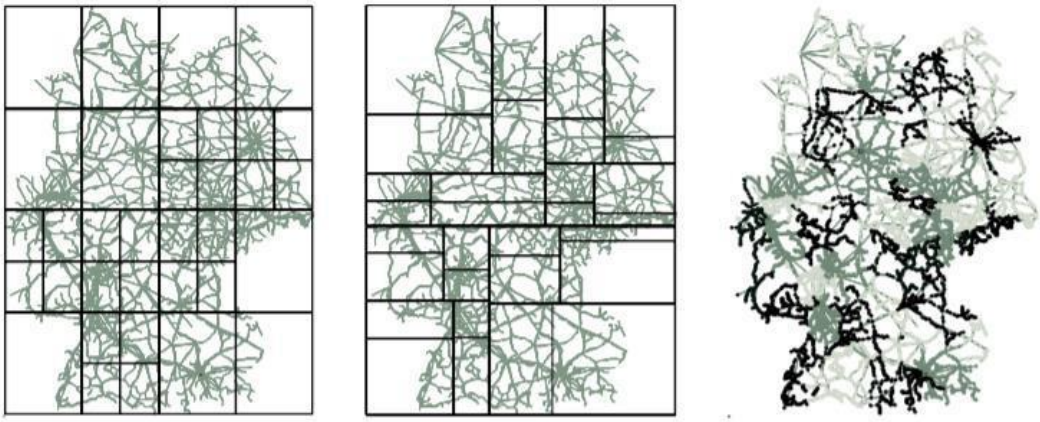
2.2 PAPER 2: Partitioning Graphs to Speed Up Dijkstra's Algorithm

This paper mainly emphasizes on the limitations of Dijkstra's Algorithm. In this paper, the main limitation regarding the speed of the algorithm is considered. Mainly they considered the actual algorithm using a 128bit data arc and implemented it. And secondly, they implemented it using the designed algorithm.

In this algorithm, initially the entire nodal graph was divided into number of arcs and then in each arc they checked for the shortest path in them. But, in some cases they failed to find a shortest path in each arc and so they went for a two-level implementation of the algorithm. Considering a 128-bit data arc, they found 64 first level regions and 8 second level regions. In each level the data is calculated in bits per arc. Finally, on comparing the designed algorithm with the actual Dijkstra's algorithm, they found that the designed algorithm was 545 times faster than the actual algorithm w.r.t to 128 bit data arc with a processing range of 1.3ms per search.



A 5×7 grid partitioning of Germany



(a) Quadtree (34 regions) (b) Kd-Tree (32 regions) (c) METIS (32 regions)

FIGURE 1: GERMANY WITH THREE DIFFERENT REGIONS

2.3 PAPER 3: Optimization of Logistics Route Based on Dijkstra's Algorithm

This paper emphasises on the developing e-commerce business that involves major transportation issues when it comes to the delivery of the logistical goods. So, in this paper they developed a model based on Dijkstra's algorithm to calculate the shortest path from the warehouse to the distribution centres. This to done so as to reduce to cost on the transportation and delivery of the products.

In this paper, they considered a model of Dijkstra's with 6 nodes, keeping V0 as the warehouse (primary node). They calculated the distance according to Dijkstra's algorithm and specified the optimized route for each of these.

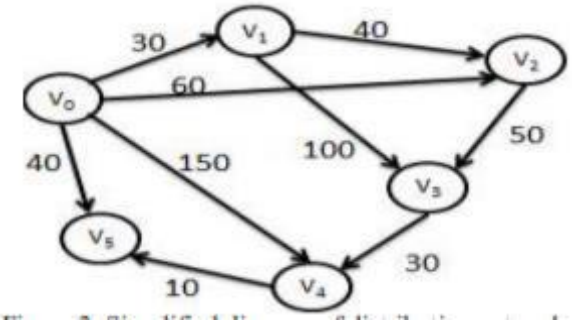


FIGURE 2: SIMPLIFIED DIAGRAM OF DISTRIBUTED NETWORK

The optimized route for this network is thus calculated using Dijkstra's algorithm and the shortest path is specified.

2.4 PAPER 4: Adaptive Ant Colony Optimization Algorithm

An adaptive ant colony algorithm is proposed to overcome the premature convergence problem in the conventional ant colony algorithm. The adaptive ant colony is composed of three groups of ants: ordinary ants, abnormal ants and random ants. Each ordinary ant searches the path with the high concentration pheromone at the high probability, each abnormal ant searches the path with the high concentration pheromone at the low probability, and each random ant randomly searches the path regardless of the pheromone concentration. Three groups of ants provide a good initial state of pheromone trails together. As the optimization calculation goes on, the number of the abnormal ants and the random ants decreases gradually. In the late optimization stage, all of ants transform to the ordinary ants, which can rapidly concentrate to the optimal paths. Simulation results show that the algorithm has a good optimization performance, and can resolve travelling salesman problem effectively. Ant colony algorithm is a new evolution algorithm, which can resolve many optimization problems, such as function optimization problems, combinatorial optimization problems and so on.

The basic idea of the ant colony algorithm is to use some artificial ants, which are similar to the real ants in nature, to find a minimum cost path, which is a legal solution for given problem. The behaviours of the artificial ants are inspired from real ants. Artificial ants could lay some pheromone on the paths which they passed. As time goes on, the pheromone can also evaporate. Artificial ants choose their path with respect to the probabilities that depend on pheromone trails which have been previously laid by the artificial ants. In this way, artificial ants are easy to choose the paths with high concentration pheromones and lay more pheromone on the paths which they choose. Thus, the paths with high concentration pheromones can attract more ants to choose them. The positive feedback principle of the ant pheromone can guide the ants to find the optimal paths, so it has fast convergence and global optimization performances. The artificial ants are divided into three groups. The artificial ants in the first group are ordinary ants, which find the path according to the pheromone positive feedback principle. The artificial ants in the second group are abnormal ants, which search

the path with the high concentration pheromone at the low probability, and search the path with the low concentration pheromone at the high probability. And the artificial ants in the third group are the random ants, which search the path regardless of the pheromone concentration. As the iteration goes on, the ant number of the first group and the second group gradually decreases. At the end of the optimization process, all of the ants become the ordinary ants, which can ensure the fast convergence of the algorithm.

2.5 PAPER 5: Solving travelling salesman problem using multiagent simulated annealing algorithm with instance-based sampling

This paper basically emphasises on the draw backs of implementing Simulated Annealing (SA) Algorithm for Travelling Sales man problem. Accordingly, it states that, the main drawback of SA algorithm being its slow implementation time and slow convergence rate. To overcome this problem, the SA algorithm can be implemented using Parallel SA algorithms, which are totally problem dependent. The slow convergence rate is due to the random generation of candidate solutions. So, to overcome the above-mentioned problems of SA algorithm, they came up with a algorithm called Multi-Agent SA algorithm with instance-based sampling (MSA-IBS). In this, a population of agents run SA algorithm collaboratively. Agents generate candidate solutions with the solution components of instances in current population. MSA-IBS achieves better results by taking advantage of learning ability from population-based algorithm.

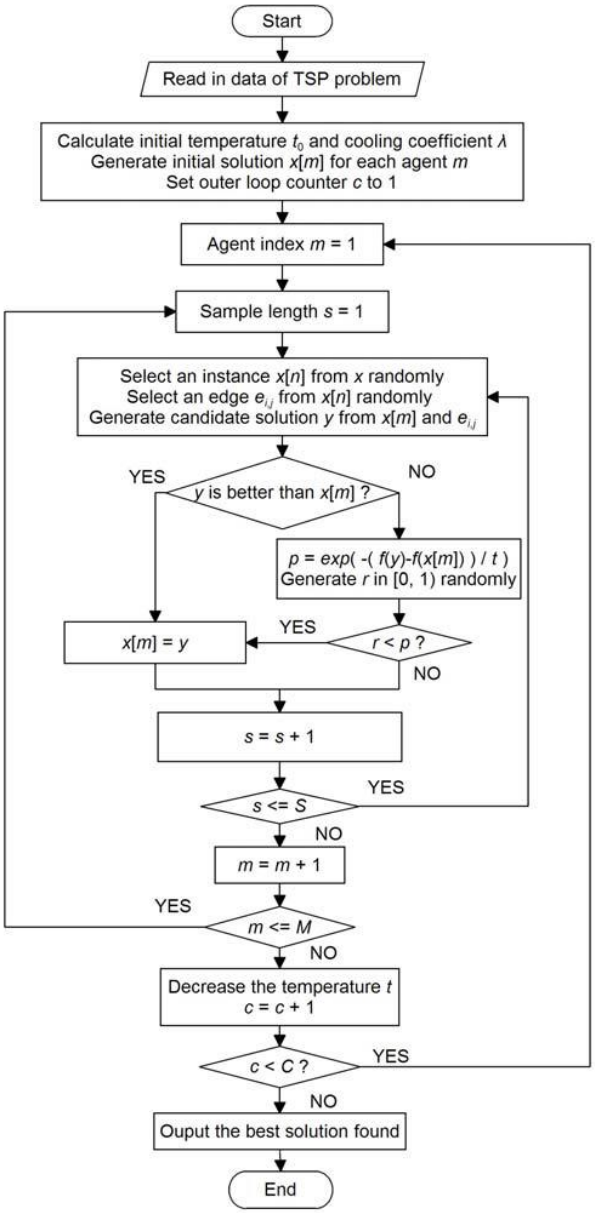


Fig.3 MSA-ISB ALGORITHM

2.6 Paper 6: Tabu Search Implementation on Traveling Salesman Problem and Its Variations: A Literature Survey

This paper emphasises a study of Tabu search algorithm based on different algorithms like Travelling Sales Man Problem (TSP) and its variations like Vehicular Routing Problem (VRP). In this it considers different cases in the implementation of Tabu search algorithm like different problem size. Also considering different aspects in the algorithm like Initial solution generation methods like General Randomized Adaptive Search Process (GRASP), Route First Cluster Second (RFCS), Cluster First Route Second (CFRS), Randomized Insertion (RandIns), Nearest Neighbour (NN) to compares the outputs.

Table 1. Problem sizes considered in published literature.

Problem Size	Years					
	Before 1996	1996-2000	2001-2005	2006-2010	After 2010	Total
100 or less	[8-10]	[11-13,20-22,26-31]	[7,14-16,23,24]	[17-19,25]	--	25
101 - 150	--	[32]	[33,34]	[35]	--	4
151 - 200	[36-38]	[39-41]	--	[17,42]	--	8
201 - 250	[43]	--	--	[44,45]	--	2
251 - 300	--	[46,47]	[34]	[48-50]	51	7
301 - 350	--	--	--	--	--	0
351 - 400	--	[52]	--	[53]	[54]	3
401 or more	[55,56]	[57-59]	[5-7,62]	[60,61]		11

Fig.4 Tabu outputs depending on the problem size

Table 2. Methods to generate initial solutions.

Method	Years					
	Before 1996	1996-2000	2001-2005	2006-2010	After 2010	Total
GRASP	--	[30]	--	--	--	1
RFCS	--	[26]	--	[42]	[54]	3
CFRS	--	--	--	[42]	--	1
RandsIns	--	[12,28,40,57]	[34,62,62]	[19,25,60,61]	[51]	12
NN	--	[13,22,29,30,40,41]	[6,16,23,33,64,65]	[17,19,35,53]	--	15
Sweep	--	[58]	--	[35,44,49,53]	--	5
GENI	[36]	[46]	--	--	--	2
C&W	[37]	[11,59]	--	[17,49,50]	--	6

Fig.5 Tabu outputs depending on the initial solutions generated

2.7 Paper 7: A Heuristic Approach Based on Clarke-Wright Algorithm for Open Vehicle Routing Problem

The paper emphasises on altering the actual Saving's Algorithm, that was devised by Clarke and Wright. In the general Savings algorithm, the routes are completely dependent on the capacity of the truck. But, this limitation in the capacity leads to repetitive movement to the warehouse. To avoid this problem, they came up with a solution like route merging procedure and Two-phase selection procedure. Route merging process works in such a way that all the routes are collectively merged optimizing the cost. In two phase selection procedure, initially it is sorted in descending order and then the savings list is generated, like in genetic algorithm, the values are randomly selected like selection of chromosomes in Genetic Algorithm and then the savings list is used. In this way the selection is done for the routes and then the cost is calculated as in the actual Savings Algorithm.

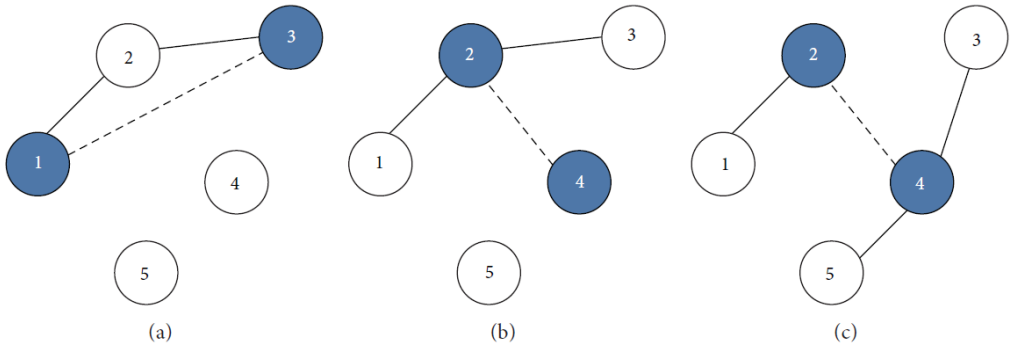


Fig.6 Merging procedure

Gene (n)	1	2	3	4	5	6	7	8	9	10
$s_{i,j}$	$s_{2,6}$	$s_{3,6}$	$s_{2,4}$	$s_{3,5}$	$s_{2,3}$	$s_{4,6}$	$s_{5,6}$	$s_{2,5}$	$s_{3,4}$	$s_{4,5}$
Savings (s_n)	40.64	40.64	32.36	32.36	30.58	28.28	28.28	20.00	20.00	11.72

(a)

n	1	2	3	4
$s_{i,j}$	$s_{2,6}$	$s_{3,6}$	$s_{2,4}$	$s_{3,5}$
Savings (s_n)	40.64	40.64	32.36	32.36
P_n	0.28	0.28	0.22	0.22
q_n	0.28	0.56	0.78	1.00

New gene (n)	1
$s_{i,j}$	$s_{3,6}$
Savings (s_n)	40.64

Random spin (r) = 0.38

Fig.7 Two phase selection procedure

2.8 Paper 8: AN IMPROVED GENETIC ALGORITHM FOR TSP

This paper emphasises on the limitations of Genetic algorithm. In order to overcome the limitations, a new operator called the untwist operator is introduced. This operator helps in reducing the total cost and also helps in faster convergence. In this algorithm, the entire genetic algorithm runs as usual starting with population initialization, fitness calculation, selection, crossover, mutation and then untwist operator is implemented in the later stages of Genetic Algorithm. By implementing the untwist operator, initially it reduces the total cost. It basically swaps two city positions and checks for the cost for all possible combinations and chooses the solution that has least cost.

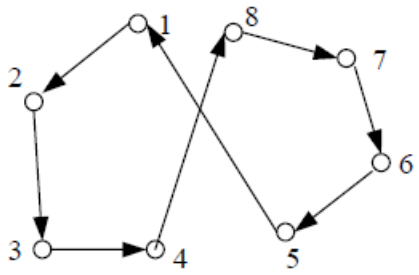


Figure 1 The twisted route

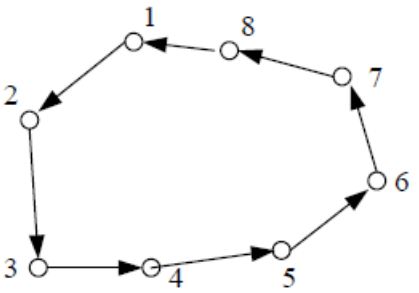


Fig.8 Twisted Routes

Table 1. Experimental parameters and result.

City Number	member of every population	Max generations	Length
50	40	2000	6.1439
100	80	2500	8.519
150	120	2800	10.1994

Fig.9 Output for the sample data

What is Route Optimization?

Route optimization is the process of finding the best path between two or more locations with a fixed order in a road or rail network. The criterion according to which a path is the best can vary. You may be looking for the shortest path (by distance), the fastest (by travel time), but also the most scenic or the safest path. Anything is possible as long as it can be specified by assigning some quantity, a *generalized cost*, to each road segment, and looking for the *least-cost path*.

How does route optimization work?

Route optimization typically uses algorithms. The reason for this is because the complexity of route optimization means that humans can't easily compute all the different parameters to find an optimal route, especially in a short amount of time.

Where route optimization is used?

There are a lot of day-to-day activities, where route optimization is utilized. Also, there are a lot of MNC's that are still working on these route optimization algorithms like: Google, Uber, and many others.

How are Route optimization algorithms are implemented?

An efficient route optimization is done only when a lot of criteria are considered. There are a number of algorithms to solve this problem. The most famous ones are Dijkstra's algorithm, and its accelerated version, the A* algorithm. The following image illustrates how the A* algorithm finds the fastest path between two locations.

What is Travelling Salesman Problem (TSP)?

The traveling salesman problem (TSP) is an algorithmic problem tasked with finding the shortest route between a set of points and locations that must be visited. In the problem statement, the points are the cities a salesperson might visit. The salesman's goal is to keep both the travel costs and the distance travelled as low as possible.

Where Travelling Salesman Problem (TSP) is used?

Focused on optimization, TSP is often used in computer science to find the most efficient route for data to travel between various nodes. Applications include identifying network or hardware optimization methods. Rather than focus on finding the most effective route, TSP is often concerned with finding the cheapest solution. In TSPs, the large amount of variables creates a challenge when finding the shortest route, which makes approximate, fast and cheap solutions all the more attractive.

How to solve Travelling Salesman Problem (TSP)?

There a numerous number of algorithms that can solve TSP problem. Some of them are: Branch and Bound Algorithm, Brute Force algorithm, Clarke Wright Saving's Algorithm, Simulated Annealing Algorithm, Genetic Algorithm and Tabu Search Algorithm etc. Each of these algorithms follow a particular approach and maintains particular parameters to solve this TSP problem.

2.5 DIJKSTRA'S ALGORITHM:

Dijkstra's algorithm is an algorithm for finding the shortest paths between nodes in a graph, which may represent road networks. It was conceived by computer scientist Edsger W. Dijkstra in 1956. The algorithm exists in many variants. Dijkstra's original algorithm found the shortest path between two given nodes, but a more common variant fixes a single node as the "source" node and finds shortest paths from the source to all other nodes in the graph.

APPLICATIONS:

- It is used in finding Shortest Path.
- It is used in geographical Maps
- To find locations of Map which refers to vertices of graph.
- Distance between the location refers to edges.
- It is used in IP routing to find Open shortest Path First.
- It is used in the telephone network.

LIMITATIONS:

- It do blind search so wastes lot of time while processing.
- It cannot handle negative edges.
- This leads to acyclic graphs and most often cannot obtain the right shortest path.

2.6 BRUTE FORCE SEARCH ALGORITHM:

Brute Force search is also known as exhaustive search. It is also known as **generate and test**, it is used in general problem-solving technique and algorithmic paradigm that consists of systematically listing all possible solutions and checking whether each path satisfies the problem's statement.

Examples:

1. A brute-force algorithm to find the divisors of a natural number ‘n’ would enumerate all integers from 1 to n, and check whether each of them divides n without remainder.
2. A brute-force approach for the eight queens puzzle would examine all possible arrangements of 8 pieces on the 64-square chessboard, and, for each arrangement, check whether each (queen) piece can attack any other.

While a brute-force search is simple to implement, and will always find a solution if it exists, its cost is proportional to the number of candidate solutions – which in many practical problems tends to grow very quickly as the size of the problem increases (combinatorial explosion).Therefore, brute-force search is typically used when the problem size is limited, or when there are problem-specific heuristics that can be used to reduce the set of candidate solutions to a manageable size. The method is also used when the simplicity of implementation is more important than speed.

Brute forcing is nothing but an exhaustive search method in which you try all the possibilities to reach the solution of a problem.

To all the problem in this world including real world problem there is an answer which can be achieved by brute forcing, it’s another topic of debate that it might take years/ages to reach that answer. As brute force method is pretty slow and this being the reason why computer scientists go mad after developing a faster algorithm, because in today's world we have everything but TIME.

ADVANTAGES:

- This method is used by default to solve some problems such as sorting, searching, matrix multiplication, binomial expansion etc.
- used for solving smaller instances or modules of a larger problem.
- brute force algorithm is that at the end of it you can definitely find the optimal solution
- There are a few approximation algorithms that runs in polynomial time which can produce a solution close to optimal solution but you cannot be certain that your solution is optimal. Whereas brute force though it may take exponential time, you can certainly get the optimal solution.

DISADVANTAGE:

- Brute-force algorithm is not the one which doesn't necessarily require a lot of memory. It generally requires a lot of processing time.
- It is very hardware intensive.
- Brute force attacks try as many possible answers as possible, this takes a lot of processing power.
- Since, at the moment processing time is more precious than memory, a brute-force solution is considered to be inferior. If there is an algorithm that is more suitable to your needs in terms of memory and processing time it should always be preferred. However, for certain problems there are still no algorithms that solve the problem exactly besides brute-force ones.

Why Brute Force is used over other methods:

Bruce forcing is sufficient when there isn't any other known efficient way to solve the problem. For example, problems that are NP-complete have no known efficient solution, and one will have to resort to brute forcing, probably coupled with some optimization techniques to reduce the search space.

Another situation where a brute force algorithm can be sufficient is when the problem size is small enough that it will run faster than an asymptotically faster algorithm due to less overhead.

2.7 BRANCH AND BOUND:

- Branch and bound is a systematic method for solving optimization problems
- B&B is a rather general optimization technique that applies where the greedy method and dynamic programming fail.
- However, it is much slower. Indeed, it often leads to exponential time complexities in the worst case.
- On the other hand, if applied carefully, it can lead to algorithms that run reasonably fast on average.
- The general idea of B&B is a BFS-like search for the optimal solution, but not all nodes get expanded (i.e., their children generated). Rather, a carefully selected criterion determines which node to expand and when, and another criterion tells the algorithm when an optimal solution has been found.

ALGORITHM :

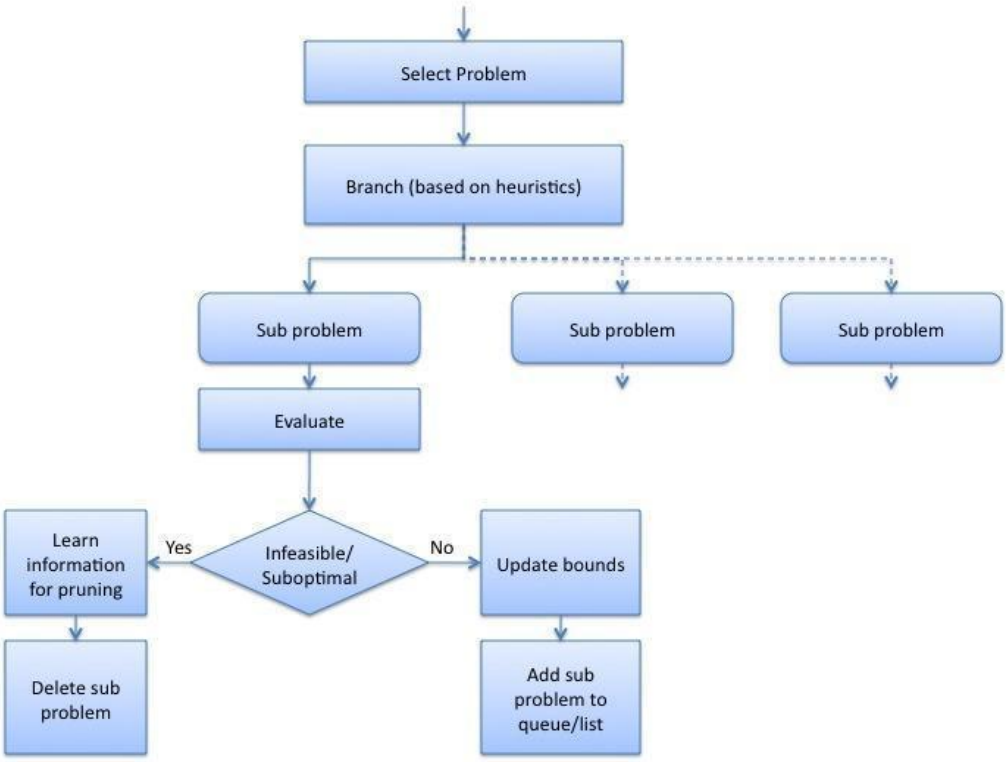


FIGURE 10: BRANCH AND BOUND ALGORITHM

Branch-and-Bound is a general technique for improving the searching process by systematically finding all the solutions and listing them and discarding the impossible solutions.

Branch-and-bound usually applies to those problems that have finite solutions, in which the solutions can be represented as a sequence of options. The first part of branch-and-bound is branching. This requires several choices to be made so that the choices branch out into the solution space. In this method, the solution space is organized as a treelike structure. Figure given below shows an instance of TSP and a solution tree, which is constructed by making choices on the next cities to visit.

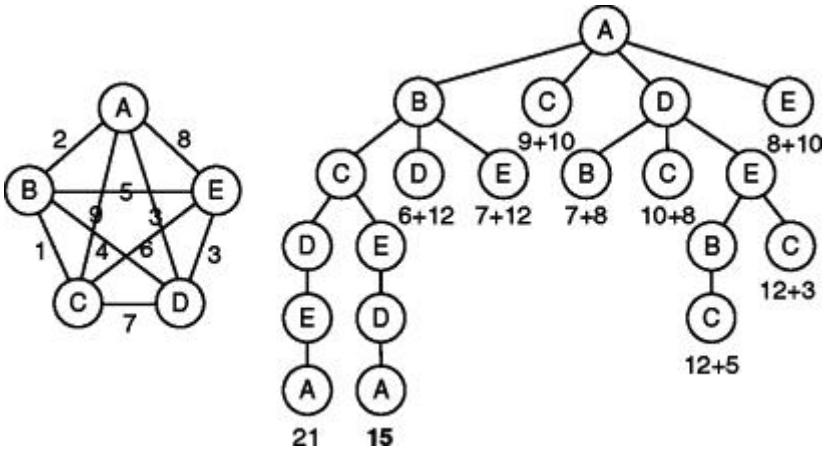


FIGURE 11: TRAVELLING SALESMAN (TSP)

Branching out is done so that all possible paths are covered making sure no path is left out. But because the target problem is usually NP-complete or even NP- hard, the solution space is often too vast to traverse. The Branch-and-Bound algorithm handles this problem by bounding and pruning. Bounding refers to setting a bound on the solution (Example: the route length for TSP), and pruning means trimming off branches in the solution tree whose solution quality is estimated to be poor. Bounding and pruning are the essential concepts of the branch-and-bound technique, because they are used to effectively reduce the search space. We demonstrate in the above Figure how branch-and-bound works for the TSP problem.

Branch & Bound discovers branches within the complete search space by using estimated bounds to limit the number of possible solutions. The different types (FIFO, LIFO, LC) define different 'strategies' to explore the search space and generate branches.

FIFO (first in, first out): always the oldest node in the queue is used to extend the branch. This leads to a **breadth-first** search, where all nodes at depth dare visited first, before any nodes at depth d+1 are visited.

LIFO (last in, first out): always the youngest node in the queue is used to extend the branch. This leads to a **depth-first** search, where the branch is extended through every 1st child discovered at a certain depth, until a leaf node is reached.

LC (lowest cost): the branch is extended by the node which adds the lowest additional costs, according to a given cost function. The strategy of traversing the search space is therefore defined by the cost function.

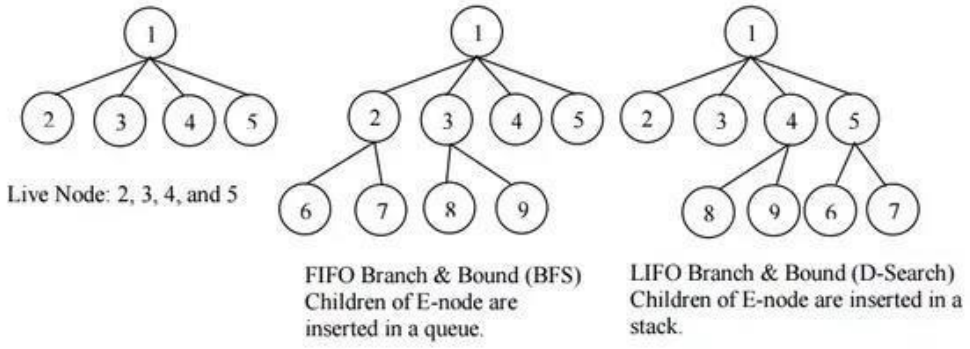


FIGURE 12: BRANCH AND BOUND ALGORITHM

The only difference is in the implementation of live nodes. In LC branch and bound, the first node we start exploring is the one which promises us the best solution at that moment. For example, in 0/1 Knapsack Problem, using LC Branch and Bound, the first child node we will start exploring will be the one which offers the maximum cost out of all.

In FIFO branch and bound, as is visible by the name, the child nodes are explored in First in First out manner. We start exploring nodes starting from the first child node. In LIFO branch and bound, we explore nodes from the last. The last child node is the one to be explored first.

ADVANTAGES:

- Finds an optimal solution (if the problem is of limited size and enumeration can be done in reasonable time).
- As it finds the minimum path instead of finding the minimum successor so there should not be any repetition.
- The time complexity is less compared to other algorithms.

DISADVANTAGES:

- The load balancing aspects for Branch and Bound algorithm make it parallelization difficult.
- Extremely time-consuming: the number of nodes in a branching tree can be too large.
- The algorithm finds the first complete schedule and then tries to improve it.
- Often developing the "promising" branches may lead to a huge number of off springs that finally may not give an improvement. Thus, the size of the tree may grow exponentially without improving the best solution obtained.
- The Branch and Bound algorithm is limited to small size network. In the problem of large networks, where the solution search space grows exponentially with the scale of the network, the approach becomes relatively prohibitive.

2.8 ANT COLONY OPTIMIZATION:

Ant Colony Optimisation has been inspired by the foraging behaviour of real ants. Ants randomly explore the surroundings of the anthill; when they find food, they return to the nest depositing a pheromone trail, a trace of a chemical substance that can be smelled by other ants. Ants can follow various paths to the food source and back, but it has been observed that only the shortest path remains in use, since ants prefer to follow stronger pheromone concentrations. Pheromone reinforcement is autocatalytic, since the shortest path, the least time will be taken to travel back and forth, and therefore, while ants on longer paths are still in transit, the ants on the shortest path can restart the route again, reinforcing the pheromone trail on the shortest path. Over time, the majority of the ants will travel on that path, while a

minority will still choose alternative paths. The behaviour of this minority is important, since it allows to explore the environment to find even better solutions, which initially were not considered. The choice of the path is therefore probabilistic and, while it is strongly influenced by the pheromone intensity, it still allows for random deviations from the current best solution.

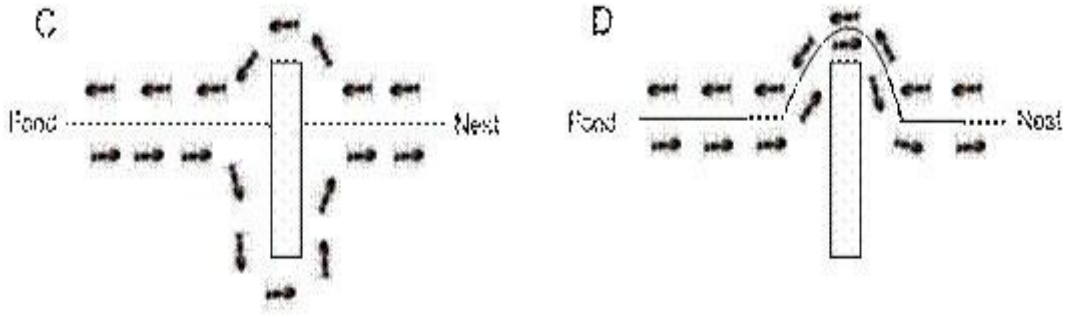


FIGURE 13: ANT COLONY

What is Ant Colony Optimization?

- Ant Colony Optimization (ACO) is a population-based, general search technique for the solution of difficult combinatorial problems.
- ACO follows Stigmergy.(Stigmergy is an interaction and coordination of organisms in nature modifying the environment)
- Ants release pheromone.
- Ants deposit pheromone while walking from nest to food sources and

  vice versa ants tend to choose, in probability, paths marked by strong pheromone concentrations.

  The two main characteristics of stigmergy that differentiate it from other forms of communication are the following.
- Stigmergy is an indirect, non-symbolic form of communication mediated by the environment insects exchange information by modifying their environment; and
- Stigmergic information is local: it can only be accessed by those insects that visit the locus in which it was released (or its immediate neighbourhood).

- Ant Colony Optimization takes elements from real ant behaviour to solve more complex problems than real ants.

- In ACO, artificial ants are stochastic solution construction procedures that probabilistically build solutions exploiting,
- (artificial) pheromone trails which change dynamically at run time to reflect the agents' acquired search experience
- heuristic information on the problem instance being solved.

Mathematical Model of Ant Colony Optimization :

The trail level τ_{ij} of a move depends on the pheromone level, and it represents a dynamic indication a posterior of its goodness. In other words, if the artificial ant smells a strong pheromone trail leading to a node, it knows that it is a promising direction to explore. When the constructive procedure has finished and artificial ants have computed a set of solutions, the pheromone information associated to some of the edges i–j are updated according to the following equation:

The mathematical model can be expressed as

$(\tau_{i,j})^k=1/L$, kth ant travels on the edge i,j ; 0 ,otherwise L is the length of the path

Ants construct their solutions in parallel. At the end of each constructive phase (iteration) the entire set of computed solutions is used to update the pheromone trail. In this case $\Delta\tau_{ij}$ is computed according to the following formula:

$\tau_{i,j}^k = \tau_{i,j}^k$, without evaporation

$\tau_{i,j}^k = (1- \rho)\tau_{i,j}^k + \tau_{i,j}^k$, with evaporation

where $\tau_{i,j}^k$ is the amount of trail laid on edge i–j by ant k.

Probabilities,

$P_{i,j}= (\tau_{i,j})^{\alpha} (\eta i_{,j})^{\beta} / ((\tau_{i,j})^{\alpha} (\eta_{i,j})^{\beta})$

$\eta i,j=1/Li,j$, Quality of the edge i-j.

Where p_{ij} is the probability of moving to j from i, and Ω is the set of nodes which are feasible to be visited from i. The parameters α and β weight the influence of trails and visibility.

Pheromone update:

- Use positive feedback to reinforce components of good solutions. Better solutions give more feedback.
- Use pheromone evaporation to avoid unlimited increase of pheromone trails and allow forgetting of bad choices.
- Pheromone evaporation, $0 < \rho <= 1$

ACO applications in routing:

- Build routing tables to direct data traffic optimizing some measure of network performance
- A routing table entry gives for node and destination node of a data packet the next node to move to.
- Routing is difficult because routing costs are dynamic.
- Adaptive routing is difficult because changes in control policy result in changes in the performance measure

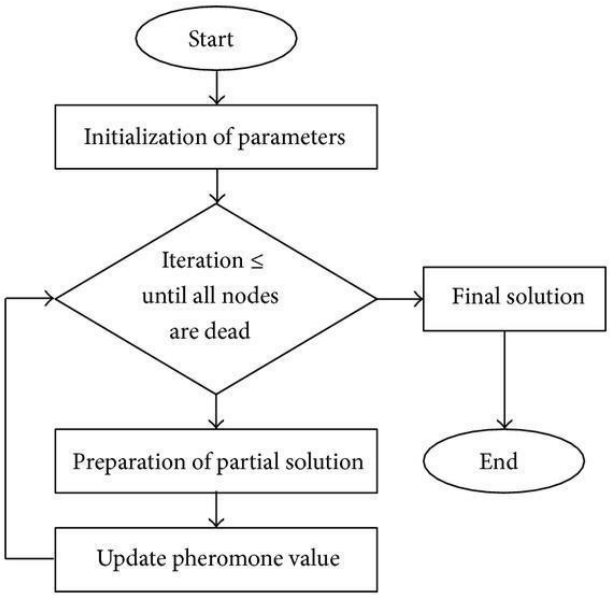
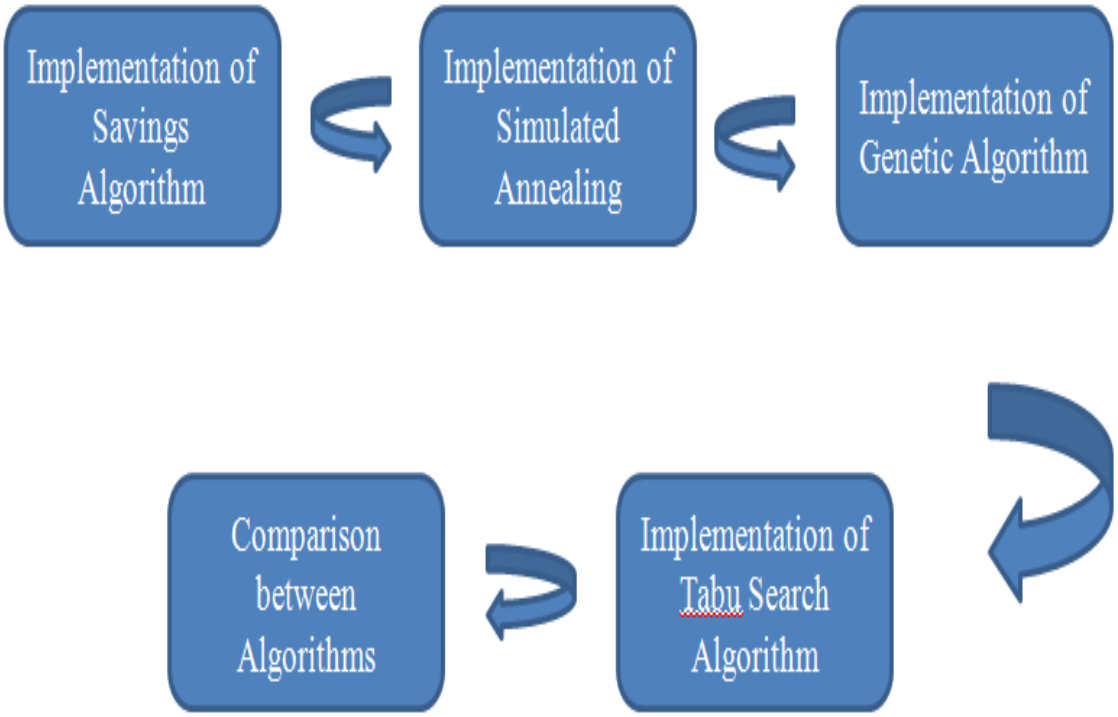


FIGURE 14: ANT COLONY OPTIMIZATION MODELLING

The advantage of ACO based algorithm over traditional optimisation algorithms is the ability to produce a good suboptimal solution in a very quick time, as it has been shown in experimental cases for the Travelling Salesman Problem

ACO is a typical adaptive algorithm since it can transfer information from past environment to new environment and quickly adapt to dynamic changes. In addition, ACO has strong robustness and handles extreme conditions reasonably. In order to better meet dynamic environment, a great number of strategies are introduced to enhance ACO for resolving the DVRP.

PROJECT PLAN



2.9 CLARKE AND WRIGHT SAVING'S ALGORITHM:

The savings procedure of Clarke and Wright is the most widely known heuristic for the VRP. The procedure begins with each customer being served by a single tour. Cost savings $S_{ii} = C_{oi} + C_{oj} - c_{ij}$ can be obtained by satisfying the demands of customers i and j using one vehicle from the depot 0 (fig. 15(a)). These savings are sorted in decreasing order. The procedure merges customers i and j corresponding to the highest saving S_{ij} without violating the capacity restriction
until no further merges are possible.

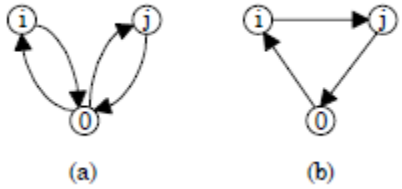


FIGURE 15: FIGURE OF SAVING'S ALGORITHM

In figure 15(a) customers i and j are visited on separate routes. An alternative to this is to visit the two customers on the same route. The above example illustrates the cost savings by combining 2 different routes into a single route. Denoting the transportation cost between two given points i and j by c_{ij}, the

Total transportation cost in figure 15(a) is:

$$D_a = c_{0i} + c_{i0} + c_{0j} + c_{j0}$$

Equivalently, the transportation cost in figure 15(b) is:

$$D_b = c_{0i} + c_{ij} + c_{j0}$$

By combining the two routes one obtains the savings S_{ij} :

$$S_{ij} = D_a - D_b = c_{i0} + c_{0j} - c_{ij}$$

There are two types of the savings algorithm, a sequential and a parallel version. In sequential method only one route is built at a time (excluding routes with only one customer), while in the parallel version more than one route may be built at a time.

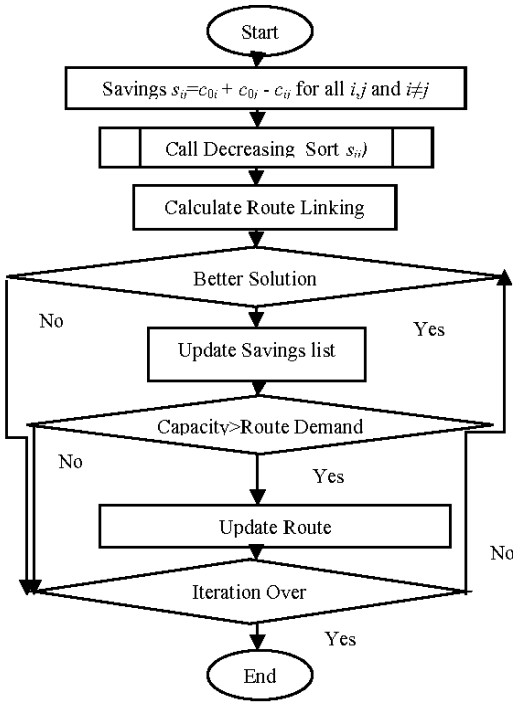


FIGURE 16: SAVINGS ALGORITHM

The first step of the savings algorithm is to find savings for all pairs of customers, and then all pairs of customer points are sorted in descending order of the savings. Then the route linking is done between the customers with the capacity constraint. If neither i nor j have already been assigned to a route, a new route is initiated with i and j. If either i or j has already been taken in an already existing route and if that point is not interior to that route, the link (i, j) is added to that same route. If both i and j are already included in two different existing routes and neither point is interior to its route, the two routes should be merged. This process is continued till the savings list is exhausted or the maximum capacity of the vehicle is reached.

LIMITATIONS:

- The local optimum achieved may be from the global optimum
- The quality of the final solution depends critically on initial starting solution

2.10 SIMULATED ANNEALING (SA):

The process of annealing a material such as metal or glass by increasing its temperature to a high value and then gradually reducing the temperature, allowing local regions of order to grow outward, increasing ductility and reducing stresses in the material, the algorithm randomly perturbs the original path to a decreasing extent according to a gradually decreasing logical "temperature". This is the analogy used in the Simulated Annealing algorithm.
In this algorithm, the equivalent of temperature is a measure of the randomness by which changes are made in the path, aiming to minimise it. When the temperature is high, more random changes are made, avoiding the risk of being trapped in a local minimum then pinpointing on a near-optimal minimum as the temperature reduces. The temperature falls in a stepwise manner which can be a function of our choice.

MATHEMATICAL MODELLING OF SA:

We first define the energy function as,

$$S(m) = -T_0 \log\left[\frac{\sigma_M(m)}{\mu_M(m)}\right]$$

T0 is a fixed positive number termed the ambient temperature (e.g. T=1). We obtain the probability density function

$$\sigma_M(m,T) = \rho_M(m)\exp\left(-\frac{S(m)}{T}\right)$$

$$\sigma_M(m,T) = \rho_M(m)\exp\left(-\frac{-T_0 \log \frac{\sigma_M(m)}{\rho_M(m)_o}}{T}\right)$$

$$\sigma_M(m,T) = \rho_M(m)\exp\left(-\frac{-T_0 \log \frac{\sigma_M(m)}{\rho_M(m)_o}}{T}\right)$$

Where, $\sigma_M(m)$ -> peaks (as pdf), $\rho_M(m)$ -> constant, $T_0=1$

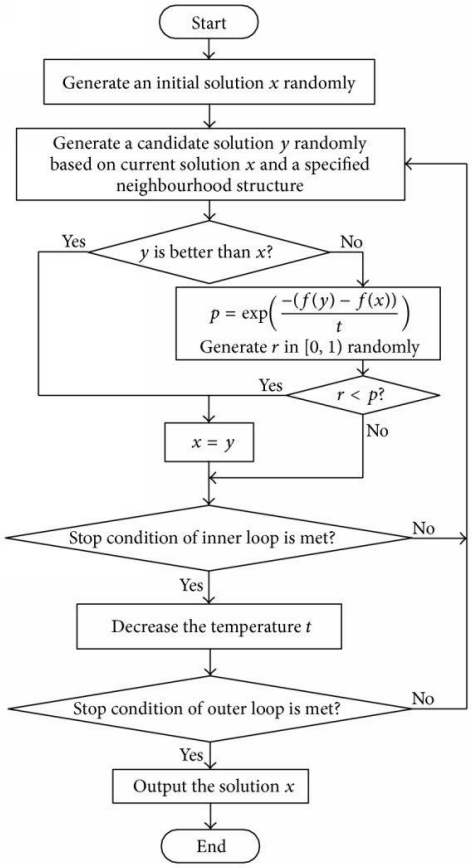


FIGURE 17: SIMULATED ANNEALING ALGORITHM

LIMITATIONS:

- Repeatedly annealing with a $1/\log k$ schedule is very slow, especially if the cost function is expensive to compute

- For problems where the energy landscape is smooth, or there are few local minima, SA is overkill --- simpler, faster methods (e.g., gradient descent) will work better. But usually don't know what the energy landscape is.
- Heuristic methods, which are problem-specific or take advantage of extra information about the system, will often be better than general methods. But SA is often comparable to heuristics.
- The method cannot tell whether it has found an optimal solution. Some other method (e.g. branch and bound) is required to do this.

2.11 GENETIC ALGORITHM:

Genetic Algorithm (GA) is a search-based optimization technique based on the principles of Genetics and Natural Selection. It is used to find optimal or near-optimal solutions for difficult problems which would take a lifetime to solve.

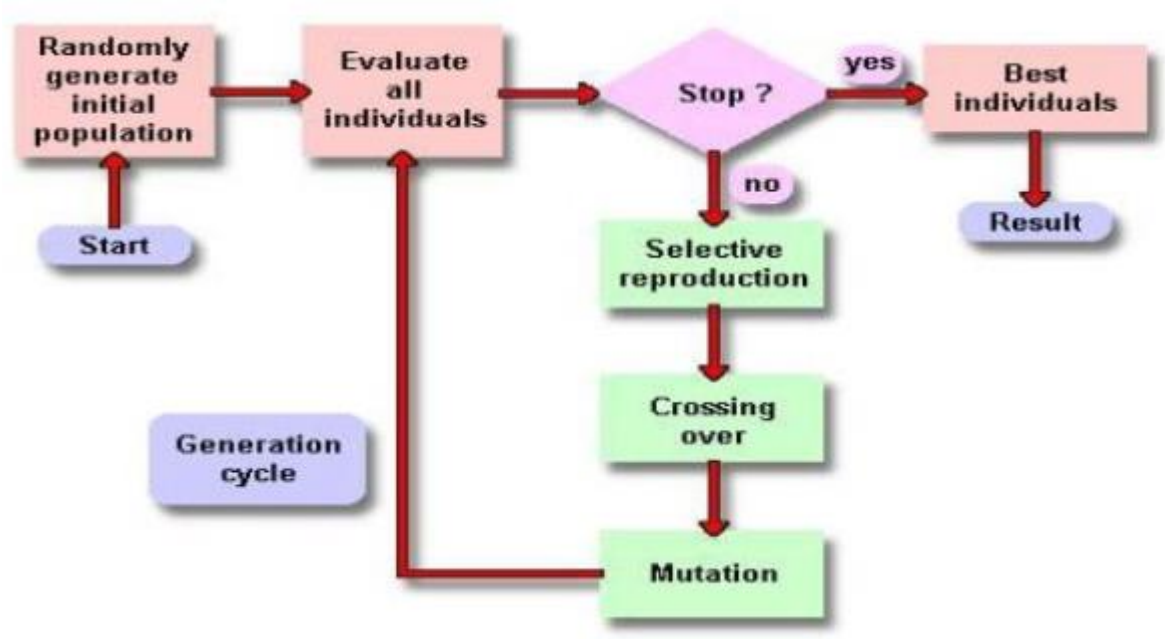


FIGURE 18: GENETIC ALGORITHM

Initialization is the first Step of Genetic Algorithm. In which individual solution is randomly generated to form an initial population. The size of the population depends on the nature of the problem and it contains several possible solutions.

Let $C = \{1,2,\cdots i,\cdots n\}$ be the cities to be visited. Every chromosome can be represented as follows

$$X = (X_1, X_2, \cdots X_i, \cdots X_n),$$

For TSP, this chromosome represents the route

$$X_1 \to \cdots \to X_i \cdots \to X_n \to X_1$$

.To ensure the validity of chromosome, the following conditions are needed:

1) $X_i \in C, 1 \le i \le n$

2) $X_i \ne X_j, i \ne j, 1 \le i, j \le n$

3) $X_i = X_j, i = j, 1 \le i, j \le n$

The **Fitness function** is used to evaluate the quality of every individual. Since the aim of traveling salesman problem is to get a route which minimizes the total distance, the fitness function f (X) of every chromosome X can be defined as follows:

$$f(X) = \sum_{i=1}^{n-1} D(X_i, X_{i+1}) + D(X_n, X_1)$$

Where

n is the total number of city,

X_i is the city number in position i ,

$D(X_i, X_j)$ is the distance from city X_i to X_j , and $D(X_i, X_j) = D(X_j, X_i)$

Selection is the next process in which evaluation of all chromosomes from a population is done by using the fitness function. According to Darwin's Theory of evaluation only fittest individuals are selected and are considered for further process. Types of Selection Are:

a) Roulette Wheel Selection (RWS) b) Tournament Selection c) Stochastic Universal Sampling (SUS) d) Rank Selection

The chromosomes undergo random selection and they are ready for crossover and to generate a new set of offspring. The crossover operator is analogous to reproduction and biological crossover. In this more than one parent is selected and one or more off-springs are produced using the genetic material of the parents. Crossover is usually applied in a GA with a high probability – *pc*.

Types of Crossover:

a) Single Point Crossover b) N-Point Crossover c) Uniform Crossover d) Partially Matched Crossover

Mutation may be defined as a small random tweak in the chromosome, to get a new solution. It is used to maintain and introduce diversity in the genetic population and is usually applied with a low probability – *pm.* If the probability is very high, the GA gets reduced to a random search. It is the part of the GA which is related to the "exploration" of the search space. It has been observed that mutation is essential to the convergence of the GA while crossover is not.

Type of Mutation:

a) Flip Bit Mutation b) Random Resetting c) Swap Mutation d) Scramble Mutation

e) Inversion Mutation

The Survivor Selection determines which individuals are to be removed and which are to be kept in the next generation. It is important to ensure that the fitter individuals are not removed out of the population, while at the same time diversity should be maintained in the population.

Some GAs use Elitism. It means the current fittest member of the population is always propagated to the next generation and ensures under no circumstance can the fittest member of the current population be replaced.

The termination condition of a Genetic Algorithm is important in knowing when a GA stops. It has been observed that the GA progresses very fast with better solutions coming in every few iterations, but later it tends to saturate where the improvements are very small. We usually need a termination condition such that our solution is close to the optimal, at the end of the run.

Usually, we keep one of the following termination conditions −

- When there is no improvement in the population for X iterations.
- Reaching an absolute number of generations.
- Objective function value has reached a certain pre - defined value.

LIMITATIONS:

- One major obstacle of genetic algorithms is the coding of the fitness (evaluation) function so that a higher fitness can be attained and better solutions for the problem at hand are produced.

- Along with making a good choice of fitness function, the other parameters of a Genetic Algorithm like population size, mutation and crossover rate must also be chosen with care. A high frequency of genetic change or poor selection scheme will result in disrupting the beneficial schema and the population may enter error catastrophe, changing too fast for selection to ever bring about convergence.

- It is not advisable to use Genetic algorithms for analytical problems. Though Genetic algorithms can find accurate solutions to these kind of problems, traditional analytic methods can find the same solutions in less time with few computational steps.

2.12 TABU SEARCH ALGORITHM:

Tabu search is one of the most widely used metaheuristic algorithms to solve TSP. It guarantees to give a near optimal solution to TSP. The word tabu or taboo comes from Tongan, a word from Polynesian language, where it was used to indicate things that are restricted from being touched. The word also means that something is "banned as constituting a risk". The tabu search is a higher-level metaheuristic for solving optimization problems. The tabu search algorithm starts at an initial solution and then progresses to a neighbouring solution. A neighbouring solution is generated by a set of justifiable moves. At each iteration the method progresses to a better solution in the neighbourhood of the current solution.

Tabu Search algorithm is to necessitate already recorded heuristic from coming back to recently visited areas of the search space. This way of approach is to preserve a temporary memory for all the changes of recent moves within the search space to prevent future moves from annulling those changes. Tabu search is based on local search. The search starts with an initial solution for the problem. That initial solution is considered as a current solution and searches for the best solution. After this step, it yields the best solution from the neighbourhood as the current solution and the search process continues.

There are three main strategies in Tabu search algorithm. They are Forbidding strategy, Freeing Strategy. Short-Term strategy and its parameters are Local Search procedure, Neighbourhood Structure, Aspirations Conditions, Size of tabu list, Stopping Rule.

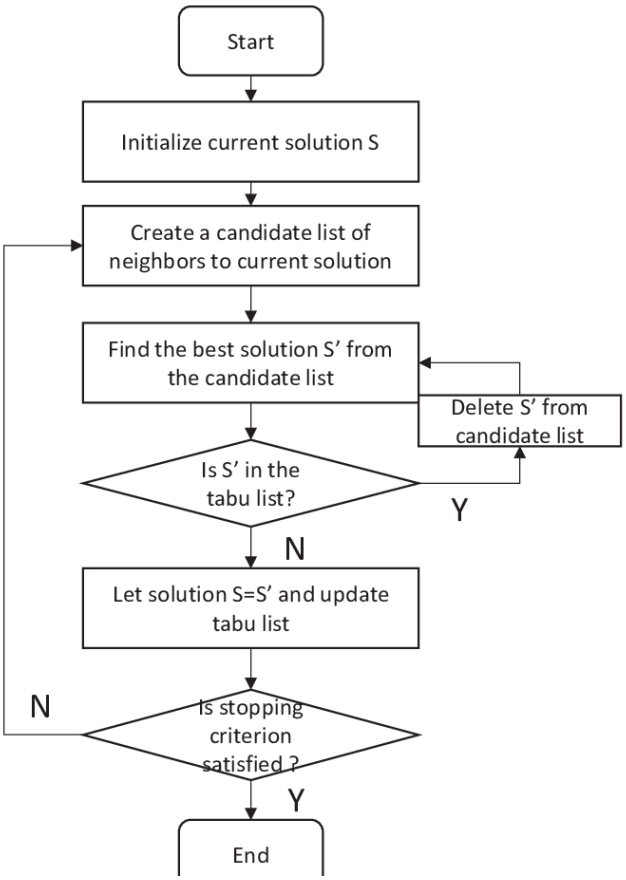


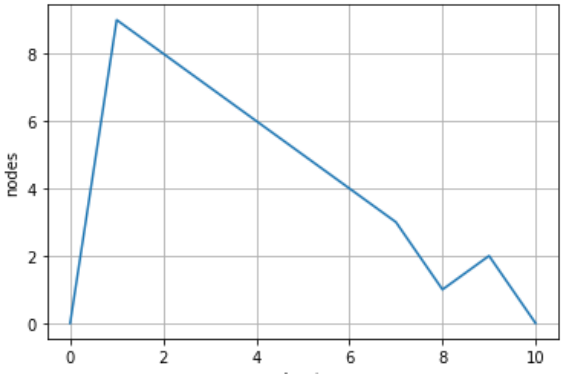
FIGURE 19: TABU SEARCH ALGORITHM

LIMITATIONS:

- Tabu list construction is problem specific
- No guarantee of global optimal solutions

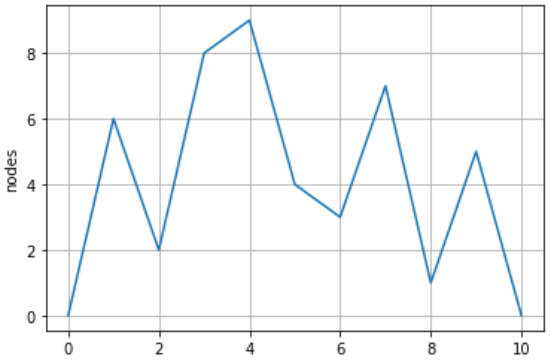
CHAPTER 3

COMPARISION AND RESULTS



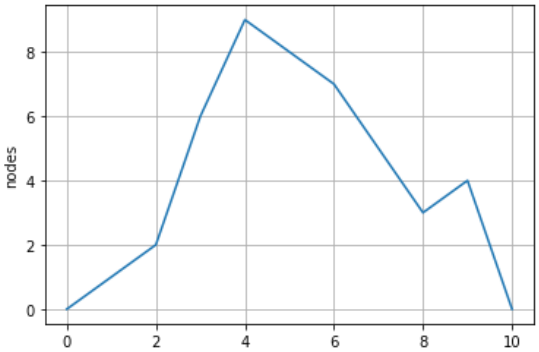
PATH: [0,9,8,7,6,5,4,3,1,2,0]

FIGURE 20: SAVINGS GRAPH FOR 10 NODES



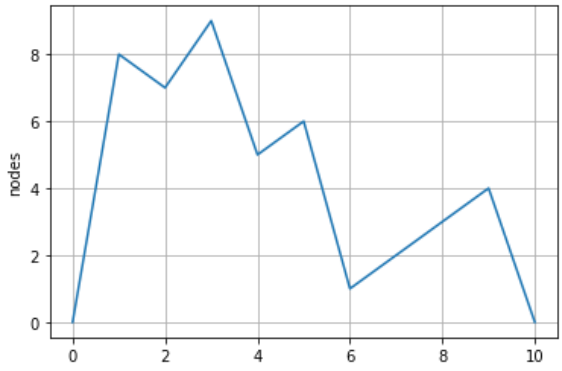
PATH: [0,6,2,8,9,4,3,7,1,5,0]

FIGURE 21: SIMULATED GRAPH FOR 10 NODES



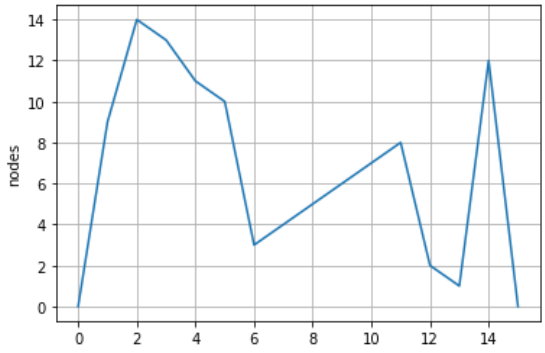
PATH: [0,1,2,6,9,8,7,5,3,4,0]

FIGURE 22: GA GRAPH FOR 10 NODES



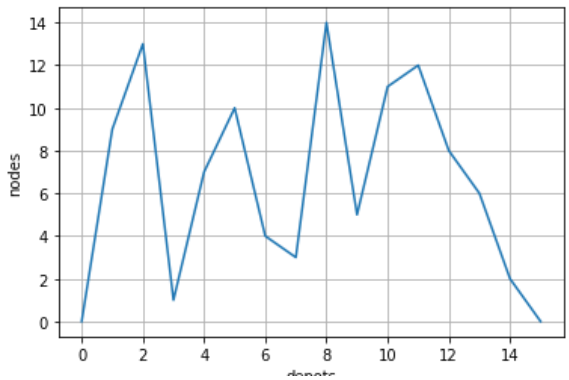
PATH: [0,8,7,9,5,6,1,2,3,4,0]

FIGURE 23: TABU GRAPH FOR 10 NODES



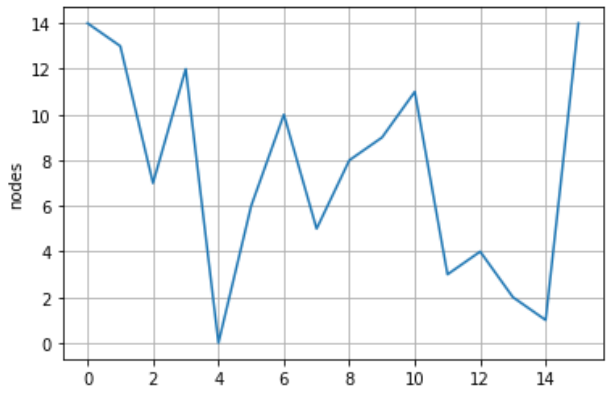
PATH:
[0,9,14,13,11,10,3,4,5,6,7,8,2,1,12,0]

FIGURE 24: SAVINGS GRAPH FOR 15 NODES



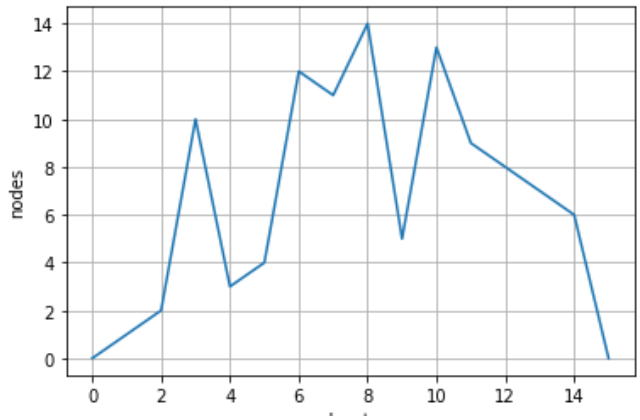
PATH:
[0,9,13,1,7,10,4,3,14,5,11,12,8,6,2,0]

FIGURE 25: SA GRAPH FOR 15 NODES



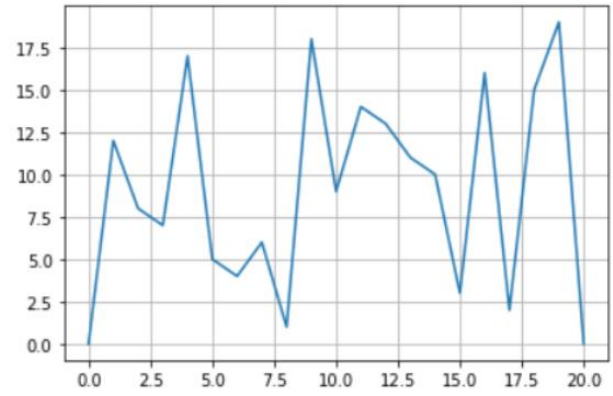
PATH:
[14,13,7,12,0,6,10,5,8,9,11,3,4,2,1,14]

FIGURE 26: GA GRAPH FOR 15 NODES



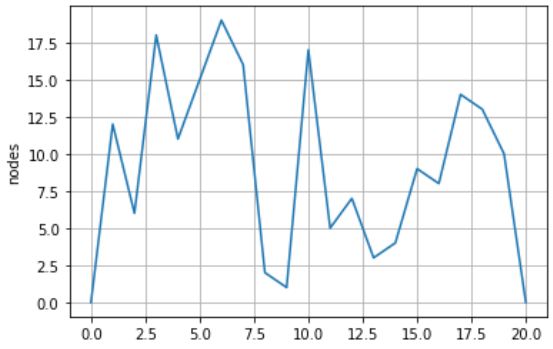
PATH:
[0,1,2,10,3,4,12,11,14,5,13,9,8,7,6,0]

FIGURE 27: TABU GRAPH FOR 15 NODES



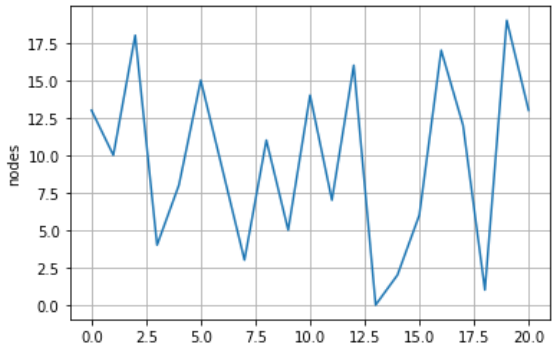
PATH:
[0,12,8,7,17,5,4,6,1,18,9,14,13,11,10
,3,16,2,15,19,0]

FIGURE 28: SAVINGS GRAPH FOR 20 NODES



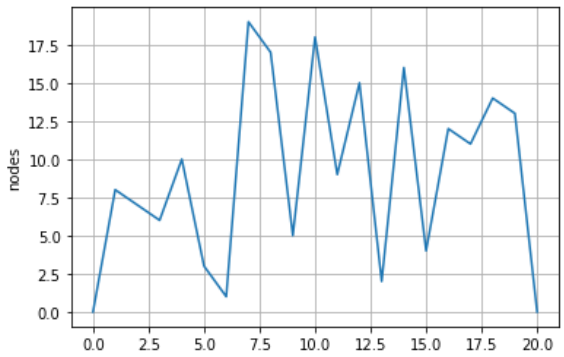
PATH:
[0,12,6,18,11,15,19,16,2,1,17,5,7,3,4,9,8,14,13,10,0]

FIGURE 29: SA GRAPH FOR 20 NODES



PATH:
[13,10,18,4,8,15,9,3,11,5,14,7,16,0,2,6,17,12,1,19,13]

FIGURE 30: GA GRAPH FOR 20 NODES



PATH:
[0,8,7,6,10,3,1,19,17,5,18,9,15,2,16,4,12,11,14,13,0]

FIGURE 31: TABU GRAPH FOR 20 NODES

SPECULATIONS:

	10 NODES				15 NODES				20 NODES			
	SAVINGS	SA	GA	TABU	SAVINGS	SA	GA	TABU	SAVINGS	SA	GA	TABU
COST	324	391	246.9228	379	398	487	408.4883	346	359	500	570.4049	337
TIME	2.669	2.2424	10.7281	23.4339	3.1998	3.3399	6.3772	11.6635	4.8962	3.8833	9.0519	34.0865
DELIVERIES	41	37	40	38	61	53	69	63	71	78	74	74

Table 1: Speculation table comparing all algorithms

	10 NODES				15 NODES				20 NODES			
	SAVINGS	SA	GA	TABU	SAVINGS	SA	GA	TABU	SAVINGS	SA	GA	TABU
COST	324	367.4	257.6677	351.6	398	490	429.3781	336.6	359	500	560.3633	301.5
TIME	2.669	3.22936	4.51125	15.88313	3.1998	2.43021	10.21899	14.83305	4.8962	2.62768	55.64977	37.43491
DELIVIERS	41	35	41	34	61	50	70	60	71	78	72	69

Table 2: Speculation table for average of all observations

10 NODES:

1. As we can see from the above table, when the nodes present in the delivery route are 10, cost is reduced the most in GA when compared to Savings, SA and Tabu search algorithms but the time required is almost four times that of time required in Savings and SA. From the observation, we can conclude that we can use Savings and SA algorithms when the time constraint we have is limited by trading off for the cost.
2. Tabu search is not suitable when the nodes are less.

3. When cost factor need not be minimised to the least possible value and the time factor is limited, we can see that the Savings algorithm gives favourable results compared to the GA.

15 NODES:

1. When the nodes are increased by 5, we can note that the cost is the least in Tabu Search algorithm but also the time required for the algorithm to give results is more than the other algorithms.
2. We can infer that GA is the better algorithm when the cost factor can be compromised a bit as the number of deliveries made by using the GA are more than the other three algorithms during a very less time compared to Tabu search algorithm.
3. Savings algorithm also gives significant results when time constraint is considered to be less.

20 NODES:

1. When the number of nodes is further increased by 5, Tabu search is the most effective algorithm with a good number of deliveries made, but the time required is also considerably more than the other algorithms.
2. We can also see that Savings gives almost equal number of deliveries in a very less amount of time with a little amount of cost added.
3. If the cost factor can be compromised a little, Simulated Annealing gives a greater number of deliveries in a very less amount of time.

Speculation table for average of all Observations:

10 NODES:

1. As we can see from the above table 2, the cost is most reduced in the GA and the time taken is very close to the Savings and SA algorithms and the deliveries made are also higher compared to the other algorithms.
2. Also, the Tabu Search algorithm is not very suitable when the nodes are less in number as the time taken is more than the other algorithms.

15 NODES:

1. When the nodes are increased by 5, we can note that the cost is the least in Tabu Search algorithm but also the time required for the algorithm to give results is just only a little more than the other algorithms. So, we can conclude that, based on different datasets, the parameters may vary accordingly.
2. Also, the Savings algorithm gives effective results in a very less amount of time with only a little more cost than Tabu search algorithm.

20 NODES:

1. When the number of nodes is further increased by 5, Tabu search is the most effective algorithm with a good number of deliveries made, but the time required is also considerably more than the other algorithms.
2. We can also see that the Savings Algorithm gives almost equal number of deliveries in a very less amount of time with a little amount of cost added.
3. If the cost factor can be compromised a little, Simulated Annealing gives a greater number of deliveries in a very less amount of time.
4. When the goal is to make more number of deliveries even if we have to compromise on the cost factor, in a very less amount of time, Simulated Annealing gives great results.

CONCLUSION:

The purpose of this study is to investigate some of the machine learning heuristics for solving Traveling Salesman Problem (TSP). GA, Savings, Simulated Annealing and Tabu search Algorithms have been implemented in python for comparison purposes. Then these algorithms are compared in time complexity, the advantages and disadvantages of the calculating results, and difficulty level of realization, etc. This method of comprehensive evaluation is relatively simple and useful.

REFERENCES

[1] Möhring, R. H., Schilling, H., Schütz, B., Wagner, D., & Willhalm, T. (2005). Partitioning graphs to speed up Dijkstra's algorithm. *Lecture Notes in Computer Science*, *3503*, 189–202. https://doi.org/10.1007/11427186_18

[2] Zhang, X., Chen, Y., & Li, T. (2015). Optimization of logistics route based on Dijkstra. *Proceedings of the IEEE International Conference on Software Engineering and Service Sciences, ICSESS*, *2015-November*(1), 313–316. https://doi.org/10.1109/ICSESS.2015.7339063

[3] Ping, G., Chunbo, X., Yi, C., Jing, L., & Yanqing, L. (2015). Adaptive ant colony optimization algorithm. *Proceedings - 2014 International Conference on Mechatronics and Control, ICMC 2014*, (Icmc), 95–98. https://doi.org/10.1109/ICMC.2014.7231524

[4] Xia, M. (2009). A modified ant colony algorithm with local search for capacitated vehicle routing problem. *PACIIA 2009 - 2009 2nd Asia-Pacific Conference on Computational Intelligence and Industrial Applications*, *2*(1), 84–87. https://doi.org/10.1109/PACIIA.2009.5406543

[5] Pillac, V., Guéret, C., & Medaglia, A. L. (2012). An event-driven optimization framework for dynamic vehicle routing. *Decision Support Systems*, *54*(1), 414–423. https://doi.org/10.1016/j.dss.2012.06.007

[6] Rodriguez, C., Salazar, J., & Viacava, G. (2019). Improvement in delivery times of a logistic operator, 1–6.

[7] Nacional, C. (2004). Relationship between Genetic Algorithms and Ant Colony Optimization Algorithms. *Quality*, *11*(4), 1–16. https://doi.org/10.1109/MCI.2006.329691

[8] Gitz-Johansen, A., Holm, M. E., Kirkeby, L. V., Kristiansen, D., Ostenfeld, A. S., Schou, M. K., & Yang, B. (2019). A practical delivery route planning system. *Proceedings - IEEE International Conference on Mobile Data Management*, *2019-June*(Mdm), 349–350. https://doi.org/10.1109/MDM.2019.00-37

[9] Nacional, C. (2004). Relationship between Genetic Algorithms and Ant Colony Optimization Algorithms. *Quality*, *11*(4), 1–16. https://doi.org/10.1109/MCI.2006.329691

[10] He, Z., Wei, C., Jin, B., Pei, W., & Yang, L. (2004). Salesman Problem. *Update*, 1152–1156.

[11] Lysgaard, J. (1997). *Clarke & Wright 's Savings Algorithm*. *September*, 1–7.

[12] Allwright, J. R. A., & Carpenter, D. B. (1989). A distributed implementation of simulated annealing for the travelling salesman problem. *Parallel Computing*, *10*(3), 335–338. https://doi.org/10.1016/0167-8191(89)90106-3

[13] Delin, L., Lixiao, Z., & Zhihui, X. (2011). Heuristic simulated annealing genetic algorithm for traveling salesman problem. *ICCSE 2011 - 6th International Conference on Computer Science and Education, Final Program and Proceedings*, *Iccse*, 260–264. https://doi.org/10.1109/ICCSE.2011.6028630

[14] Wang, C., Lin, M., Zhong, Y., & Zhang, H. (2015). Solving travelling salesman problem using multiagent simulated annealing algorithm with instance-based sampling. *International Journal of Computing Science and Mathematics*, *6*(4), 336–353. https://doi.org/10.1504/IJCSM.2015.071818

[15] Wang, Y. (2014). An improved genetic algorithm for TSP. *WIT Transactions on Engineering Sciences*, *87*(August), 853–862. https://doi.org/10.2495/AMITP20131011

[16] Liu, F., & Zeng, G. (2009). Study of genetic algorithm with reinforcement learning to solve the TSP. *Expert Systems with Applications*, *36*(3 PART 2), 6995–7001. https://doi.org/10.1016/j.eswa.2008.08.026

[17] Razali, N. M., & Geraghty, J. (2011). Genetic algorithm performance with different selection strategiesin solving TSP. *Proceedings of the World Congress on Engineering 2011, WCE 2011*, *2*, 1134–1139.

[18] Basu, S. (2012). Tabu Search Implementation on Traveling Salesman Problem and Its Variations: A Literature Survey. *American Journal of Operations Research*, *02*(02), 163–173. https://doi.org/10.4236/ajor.2012.22019

[19] Alkallak, I. N., & Sha'ban, R. Z. (2008). Tabu Search Method for Solving the Traveling salesman Problem Isra Natheer Alkallak Ruqaya Zedan Sha ' ban. *Journal of Computational Mathematics*, *5*(2), 141–153.