INTERNATIONAL JOURNAL OF RESEARCH AND INNOVATION IN APPLIED SCIENCE (IJRIAS)
ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS | Volume VIII Issue III March 2023

Travelling Salesman's Problem-Based Model Application for Mail Routing and Delivery with Minimal Cost

Ndifon Godswill Brendan, Ceasar E. Eko

Department of Computer Science, Faculty of Physical Sciences, University of Calabar, Calabar, Nigeria

Received: 05 March 2023; Accepted: 11 March 2023; Published: 08 April 2023

Abstract:- In ancient history, messages were hand-delivered using a variety of methods, including runners, homing pigeons, and riders on horseback. Before the introduction of mechanized courier services, foot messengers physically ran miles to their destinations. However, over the years, the employed methods have changed for the better, but not optimal enough. As in a traveling salesman problem, a delivery agent has a set of cities to visit, a set of parcels or goods of variable weight, a medium of movement, etc, with the specific objectives to minimize the cost of visiting the cities vis-a-vis the constraints. The problem was reduced to a graph problem and handled by linear programming. The researcher developed a model or a scenario whereby courier mail or goods can be delivered by touring cities, using the constraints available in courier services as in the traveling salesman problem. object-oriented design model was adopted for use. The researcher had to develop a platform, including the Google map tool to view the entire route spread depending on where the courier man intends to visit. The weight of the parcels, location for which the parcel(s) is to be delivered, medium of transport, amount to be paid, longitude, and latitude were also considered to estimate the cost-effectiveness of the dispatch. HTML was used to design the interface while javascript was used to code the system. The result of the study was very revealing as there were indications of the fact that dispatches were anticipated faster by 46% than before. The minimal cost was efficient enough with better optimality and the visual effects on graphs, gave encouraging perceptions for users.

Keywords: Traveling salesman, Mail Routing, Minimal Cost, problem-based Model, Graph Problem, Binary Tree, Algorithm,

I. Introduction

A courier is a person or company that delivers messages, packages, and or mail**.** Couriers are distinguished from ordinary mail services by features such as speed, security, tracking, signature, specialization, and individualization of express services, which are optional for most everyday mail services. As a premium service, couriers are usually more expensive than standard mail services, and their use is normally limited to packages where one or more of these features are considered important enough to warrant the cost. Courier services operate on all scales, from within specific towns or cities, to regional, national, and global services. Large courier companies include DML, OCS, FedEX, EMS INTERNATIONAL, TNT, UPS, and Aramex. These offer services worldwide, typically via a hub and a spoke model. Therefore, the researcher is trying to model a scenario whereby a courier mail can be delivered by touring cities without repeating any of the cities, without having in mind the constraint of using a binary tree to represent the cities.

II. Review of traveling salesman problem

The Travelling Salesman Problem (often called **TSP**) is a classic algorithmic problem in the field of computer science and operational research. It focused on optimization. In this context, a better solution often means a cheaper solution. TSP is a mathematical problem. It is most easily expressed as a graph describing the locations of a set of nodes.

The traveling salesman problem was defined in the 1800s by the Irish mathematician W.R Hamilton and by the British mathematician Thomas Kirkman. Hamilton icosian game was a recreational puzzle based on finding a Hamiltonian cycle. The general form of the TSP appears to have been first studied by mathematicians during the 1903s in Vienna and at Harvard, notably by Karl Menger. Menger defines the problem, considers the obvious brute force algorithm, and observes the non-optimality of the nearest neighbor heuristics.

The traveling salesman problem describes a salesman who must travel between N cities. The order in which he does this is something he does not care about, as long as he visits each once during his trip, and finishes where he was t first. Each city is connected by other close by cities, nodes, airplanes, or by road or railway. Each of those links between the cities has one or more weights or the cost attached. The cost describes how difficult it is to traverse this edge on the graph, and may be given, for example, by the cost of an airplane ticket or a train ticket, or perhaps by the length of the edges, or time required to complete the traversal. The salesman wants to keep both the travel costs, as well as the distance he travels as low as possible.

The traveling salesman problem is typical of a large class of hard optimization problems that have intrigued mathematicians and computer scientists for years. Most important, it has applications in science and engineering. For example, in the manufacture of a

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS | Volume VIII Issue III March 2023

circuit board, it is important to determine the best order in which a laser will drill thousands of holes. An efficient solution to this problem reduces production costs for the manufacturer. Back in the days when salesmen traveled door-to-door hawking vacuums and encyclopedias, they had to plan their routes, from house to house or city to city. The shorter the route, the better. Finding the shortest route that visits a set of locations is an exponentially difficult problem: finding the shortest path for 20 cities is much more than twice as hard as 10 cities. Mathematically, traveling salesman problems can be represented as graphs, where the locations are the nodes and the edges (or arcs) represent direct routes between the nodes. The weight of each edge is the distance between the nodes. The goal is to find the shortest path not minding the constraints.

In courier services, parcels, goods, messages, etc. are delivered to different locations at a go or in batches based on the certainly given constraints. For example, delivery to the governor's office of a state should have bottlenecks that are different from deliveries from city to city. The distance, the weight of the parcels, the cost of the delivery, the routes, etc. are all constraints to be considered in this work to minimize the cost of the delivery. this paper aims to develop a route mapping application to minimize the cost of delivery and improve the services of courier agents. The study will focus on research on how to get the shortest path between a set of offices or a town. Within the Travelling salesman problem model (TSPM) framework, a graph problem will emerge and be decomposed using the adjacency matrix method. A mapped frame that would act as a guide to the dispatcher and the shortest path created for use with Google map assistance. To achieve this, the following objectives are established. The objective of this paper is

- Develop a platform for dispatch components input
- Generate a map guide for the route using Google map and the TSPM
- Estimate the cost of delivery concerning the constraints

This research is important as it allows us to find out the challenges encountered in the courier delivery industry, and the relevant steps to improve the present system using computerized modules. It will enable the dispatch riders to adopt some state-of-the-art innovations with the solutions implemented in technology using data structures like the graph to reduce distance, time, resources, and total cost of services. This research will focus on the services of United Parcel Services (UPS) and a few courier mail vendors around Calabar and the South-south geopolitical zone of Nigeria. The offices across major cities will serve as major nodes of the trees and the recipient addresses will be the child nodes; each with its attributes. Part of the limitation is the number of available offices for the collection of data; and the willingness of the organization in giving out information that will help the research. This work is going to be based on a traveling salesman model for mail routing and delivery with minimal cost.

2.1 Courier service

The Travelling Salesman Problem is one of the most studied problems in mathematical optimization. The possibility to apply this problem to various human activities is what makes it one of the most recognizable mathematical problems without an ideal solution so far.

The formulation of the problem is simple. The Travelling Salesman needs to pass through several towns and return to the starting point of his travel, making the shortest trip possible. Instead of examining the shortest route, one could examine the one that costs the least, one that allows the greatest flow of information, etc.

By its nature, the TSP falls into the category of NP-complete problems; meaning no algorithm that would provide a solution for the problem in terms of polynomial time but if the solution exists, one could test it and conclude whether it is an optimal one. n this paper, we consider the Courier Delivery Problem (CDP), a variant of the Vehicle Routing Problem with Time Windows (VRPTW) with uncertain service times and probabilistic customers. The operations of a courier delivery/pick-up company that serves a dense urban area motivated the problem. In this situation, travel times between locations are relatively short and therefore can be assumed constant when compared to the variation in service times at each location. In a business district, for example, a driver might have several drops and pick-ups in multiple offices at the same address. We, therefore consider a routing problem with uncertainty due to unknown service times and the probabilistic nature of the customers, i.e. daily delivery requests from potential customers are unknown beforehand but they usually become available in the morning.

For many practical reasons, it seems beneficial to create regular or consistent routes for the CDP that assign the same driver to the same set of customers to serve them at roughly the same time. Such consistent routes are easy to adapt to the realization of the daily uncertainty and help courier companies realize the important goal of personalization of services, making the driver the contact person whenever the Customer needs service. This regularity in schedules also increases driver familiarity with their routes and territories, which improves driver efficiency (Sungur[a], 2009).

The courier mail problem variants related to this work are VRP with stochastic demands, with stochastic customers and a Model and Algorithm for the Courier Delivery Problem with Uncertainty (Sungur[a], 2009), and with stochastic service and travel times (VRPSSTT). A major contribution to the courier mail problem of distribution comes from Bertsimas (1992), where a priori

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS | Volume VIII Issue III March 2023

solutions use different recourse policies to solve the courier mail problem of distribution and bounds, and derived asymptotic results and other theoretical properties.

Several models and solution procedures for the courier mail problem on distribution allow recourse actions to adjust an a priori solution after revealing the uncertainty. The literature proposed Different recourse actions such as skipping non-occurring customers, returning to the depot when the capacity is exceeded, or completely rescheduling for occurring customers (Jaillet 1988; Bertsimas et al. 1990; Waters 1989). Recent work by Morales (2006) uses robust optimization for the vehicle routing problem with stochastic demands (VRPSD) with recourse. It considers that vehicles replenish at the depot, computes the worst-case value for the recourse action by finding the longest path on an augmented network, and solves the problem with a tabu search heuristic. Sampling methods are also popular in solving stochastic VRP (Birge and Louveaux 1997). Recently, Hvattum et al. (2006) develop a heuristic method to solve a dynamic and stochastic vehicle routing problem (VRP) problem, which generated sample scenarios, solved heuristically, and combined to form an overall solution. Compared with stochastic customers and demands, the VRP with stochastic service and travel times has received less attention. Laporte et al. (1992) propose three models: the chance-constrained model, the 3-index recourse model, and the 2-index recourse model, and present a general branch-and-cut algorithm for all three models.

Recent work on the CDP has modeled customer service for fixed route delivery systems under stochastic demand (Haughton and Stenger 1998). Later, Haughton (2000) develops a framework for quantifying the benefits of route re-optimization, again under stochastic customer demands. Zhong et al. (2007) propose an efficient way of designing driver service territories considering uncertain customer locations and demand. Their method uses a two-stage model to construct core service territories at the strategic level and assigns customers in the non-core territories daily to adapt to the uncertainty at the operational level. Approximation equations of the distance traveled are the basis for the territory model;

The operational level makes it possible for all drivers to share their capacities by introducing the concept of "flex-zone". This approach however does not consider customer time windows. Gro¨er et al. (2008) introduced the Consistent VRP (ConVRP) model. The objective is to obtain routes such that the same drivers visit the same customers at roughly the same time on each day that the customers need service. They develop an algorithm, ConRTR (ConVRP Record-to-Record travel), which first generates a template and from it generates daily schedules by skipping non-occurring customers and inserting new customers.

Ben-Tal and Nemirovski (1998, 1999) and El-Ghaoui et al. (1998) introduced robust optimization methodology for convex programs, which was extended to integer programming by Bertsimas and Sim (2003). The general approach of robust optimization is to optimize against the worst instance due to data uncertainty by using a min-max objective. A data variation is immune to this uncertainty; this typically results in solutions that exhibit little sensitivity. Thus, robust solutions are good for all possible data uncertainty. Robust solutions are likely to be efficient, since they tend not to be far from the optimal solution of the deterministic problem and significantly outperform the deterministic optimal solution in the worst-case (Goldfarb and Iyengar 2003; Bertsimas and Sim 2004).

The robust optimization methodology assumes the uncertain parameters belong to a given bounded uncertainty set. For good general uncertainty sets and optimization problems, the resulting robust counterpart can have a comparable complexity to the original problem. This nice complexity result however does not carry over to robust models of problems with recourse, where LPs with polyhedral uncertainty can result in NP-hard problems (Ben-Tal et al. 2004). An important question, therefore, is how to formulate a robust problem that is not less difficult to solve than its deterministic counterpart. In particular, Sungur (2009) and Sungur et al. (2008) show that obtaining robust solutions for VRP with demand and travel time uncertainty is not more difficult than obtaining deterministic solutions.

2.2 Random Binary Tree

A random binary tree is a binary tree selected at random from some probability distribution on binary trees. It uses two different distributions, which are binary trees formed by inserting nodes one at a time according to a random permutation, and binary trees chosen from a uniform discrete distribution in which all distinct trees are equally likely. It is also possible to form other distributions, for instance by repeated splitting. Adding and removing nodes directly in a random binary tree will in general disrupt its random structure, but the treap and related randomized binary search tree data structures use the principle of binary trees formed from a random permutation to maintain a balanced binary search tree dynamically as nodes.

2.2.1 Expected depth of a node

For any fixed choice of a value x in a given set of n numbers, if one randomly permutes the numbers and forms a binary tree from them as described above, the expected value of the length of the path from the root of the tree to x is at most $2 \log n + O(1)$. Where "log" denotes the natural logarithm function and the O introduces big O notation. For, the expected number of ancestors of x is by the linearity of expectation equal to the sum, over all other values y in the set, of the probability that y is an ancestor of x. And a value y is an ancestor of x exactly when y is the first element to be inserted from the elements in the interval $[x,y]$. Thus, the values

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS | Volume VIII Issue III March 2023

that are adjacent to *x* in the sorted sequence of values have a probability of 1/2 of being an ancestor of *x*, the values that are one step away have a probability of 1/3, etc. Adding these probabilities for all positions in the sorted sequence gives twice a Harmonic number, leading to the bound above. A bound of this form holds also for the expected search length of a path to a fixed value *x* that is not part of the given set.

2.2.2 Treaps and randomized binary search trees

In applications of binary search tree data structures, it is rare to insert the values without deletion in a random order, limiting the direct applications of random binary trees. However, algorithm designers have devised data structures that allow the binary tree to perform insertions and deletions at each step maintaining as an invariant the property that the shape of the tree is a random variable with the same distribution as a random binary search tree.

If a given set of ordered numbers is assigned numeric priorities (distinct numbers unrelated to their values), these priorities may be used to construct a Cartesian tree for the numbers, a binary tree that has as its order traversal sequence the sorted sequence of the numbers and that is heap-ordered by priorities. Although we know of more construction that is efficient algorithms, it is helpful to think of a Cartesian tree as being constructed by inserting the given numbers into a binary search tree in priority order. Thus, by choosing the priorities either to be a set of independent random real numbers in the unit interval or by choosing them to be a random permutation of the numbers from 1 to *n* (where *n* is the number of nodes in the tree. in addition, by maintaining the heap ordering property using tree rotations after any insertion or deletion of a node, it is possible to maintain a data structure that behaves like a random binary search tree. We refer to such a data structure as a treap or a randomized binary search tree

2.2.3 Uniformly random binary trees

The number of binary trees with *n* nodes is a Catalan number: for $n = 1, 2, 3 \ldots$ these numbers of trees are 1, 2, 5, 14, 42, 132, 429, 1430, 4862, 16796… (Sequence A000108 in the OEIS).

Thus, if one of these trees is selected uniformly at random, its probability is the reciprocal of a Catalan number. Trees in this model have expected depth proportional to the square root of *n*, rather than to the logarithm. however, the Strahler number of a uniformly random binary tree, a more sensitive measure of the distance from a leaf in which a node has Strahler number *I* whenever it has either a child with that number or two children with a number $i - 1$, is with high probability logarithmic.

Due to their large heights, this model of equiprobable random trees is not generally used for binary search trees, but it has been applied to problems of modeling the parse trees of algebraic expressions in compiler design (where the above-mentioned bound on Strahler number translates into the number of registers needed to evaluate an expression and for modeling evolutionary trees. In some cases, we can automatically transfer the analysis of random binary trees under the random permutation model to the uniform model.

2.2.4 Random split trees

Devroye&Kruszewski (1996) generate random binary trees with *n* nodes by generating a real-valued random variable *x* in the unit interval (0,1), assigning the first *xn* nodes (rounded down to an integer number of nodes) to the left sub-tree, the next node to the root, and the remaining nodes to the right sub-tree, and continuing recursively in each sub-tree. If we chose *x* uniformly at random in the interval, the result is the same as the random binary search tree generated by a random permutation of the nodes, as any node is equally likely to be chosen as root; however, this formulation allows other distributions to be used instead. For instance, in the uniformly random binary tree model, once a root is fixed, each of its two sub-trees must also be uniformly random; we can generate the uniformly random model by a different choice of distribution for *x*. As Devroye and Kruszewski show, by choosing a beta distribution on *x* and by using an appropriate choice of shape to draw each of the branches, we can use the mathematical trees to create realistic-looking botanical trees.

2.3 Objective Function Selection Issue

The most commonly used objective function is the minimization of the distance, i.e. the total number of miles driven. If we disregard the fact that the distances obtained from the GPS or a Google map typically do not copy exactly the distance according to the road network, the criterion of the minimum of the distances traveled does not always have to be satisfactory. In some cases, the carrier may prefer the criterion of minimum time or cost. In both cases, however, it is much more difficult to obtain data for the input matrix of the coefficients. The matrix of objective function rates between the starting point marked with index 1 and all the customers may be found based on the addresses of customers, for example using the route planner at www.mapy.cz. In this route planner, we have two choices at first. We can choose between finding the shortest or fastest connections to the places. In both cases, we obtain data on the connection length in km and, at the same time, the duration of the connection in minutes. In total, we can get up to 4 matrixes of the objective function rates: the matrix of distances in km found for the shortest connection, the matrix of time in minutes found for the shortest connection, the matrix of distances in km found for the fastest connection and the matrix of time

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS | Volume VIII Issue III March 2023

in minutes found for the fastest connection. The matrices found for the same type of objective function (by the route planner) express one connection along the same route in two different units (km and minutes). In contrast, when comparing two matrices with the same units (but found by the route planner differently) we can easily find out that the connection between two points may take a different path (different distance and time for the same connection). Each of these matrices can therefore give a different optimal solution! For a sample solution of one selected day when 15 customers were to be served (Figure 1).

Figure 1: Customers location (blue point represents starting point)

2.4 The branch and bound method

Given a set of cities and the distance between every pair of cities, the problem is to find the shortest possible route that visits every city exactly once and returns to the starting point.

For example, consider the below graph. A TSP tour in the graph is A -> B -> C -> D -> B -> A. The cost of the tour in figure 2 is 10 + 25 + 40 + 25 + 10 = 100.

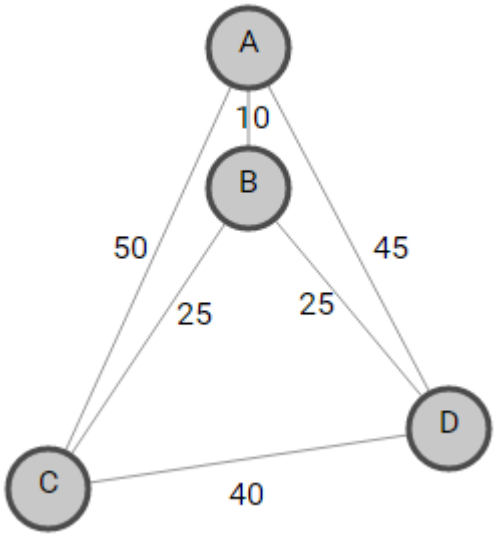

Figure 2: Graph of tour

The term Branch and Bound refer to all state space search methods in which all the children of the E-node are generated before any other live node can become the E-node. E-node is the node, which is being expended. State space tree can be expended in any method i.e. breath first search or depth-first search. Both start with the root node and generate other nodes. A node that has been generated and all of whose children are not yet been expanded is called a live node. A node is a dead node, which has been generated but cannot be expanded further. In this method, we expand the node, which is most promising, which means the node which promises that expanding or choosing it will give us the optimal solution. So we prepare the tree starting from the root then we expand it.

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS | Volume VIII Issue III March 2023

We have a cost Matrix defined by

$$C_{(i,j)=} W_{(i,j)}$$

if there is a direct path from C_i to C_j

$$C_{(i,j)=} \infty$$

if there is no direct path from C_i to C_j

For example, consider below cost matrix M,

	C0	C1	C2	C3	C4
C0	INF	20	30	10	11
C1	15	INF	16	4	2
C2	3	5	INF	2	4
C3	19	6	18	INF	3
C4	16	4	7	16	INF

Here,
C(0, 2) = 30

C(4, 0) = 16

C(1, 1) = INFINITY

Below is the state space tree for the above TSP problem, which shows the optimal solution marked in green.

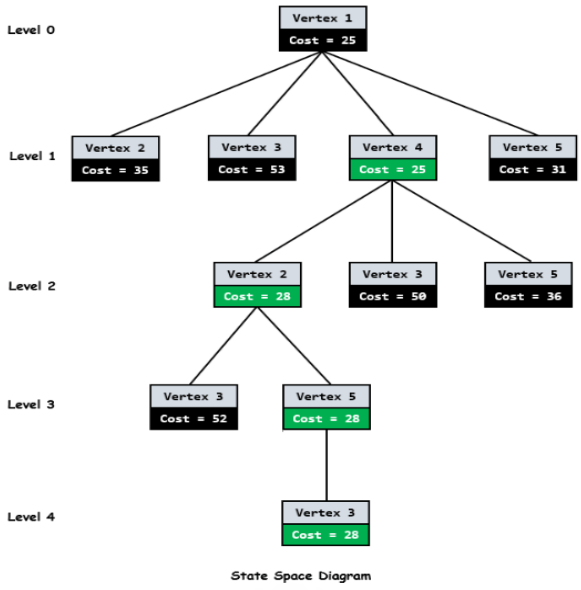

Figure 3: State Space Diagram

As we can see from above Figure 3, every node has a cost associated with it. When We go from city i to city j, the cost of node j will be the sum of the cost of parent node i, the cost of the edge (i, j), and the lower bound of the path starting at node j.

As the root node is the first node to be expanded, it doesn't have any parent. So the cost will be only the lower bound of the path starting at the root.

Now how do we calculate the lower bound of the path starting at any node?

In general, to get the lower bound of the path starting from the node, we reduce each row and column in such a way that there must be at least one zero in each row and Column. For doing this, we need to reduce the minimum value from each element in each row and column.

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS | Volume VIII Issue III March 2023

Let us start from the root node.

We reduce the minimum value in each row from each element in that row. The minimum in each Row of cost matrix M is marked in blue [10 2 2 3 4] below.

INF	20	30	10	11
15	INF	16	4	2
3	5	INF	2	4
19	6	18	INF	3
16	4	7	16	INF

After reducing the row, we get below the reduced matrix.

INF	10	20	0	1
13	INF	14	2	0
1	3	INF	0	2
16	3	15	INF	0
12	0	3	12	INF

We then reduce the minimum value in each column from each element in that column. The minimum in each Column is marked by blue [1 0 3 0 0]. After reducing the column, we get below the reduced matrix. This matrix will be further processed by child nodes of the root node to calculate their lower bound.

INF	10	17	0	1
12	INF	11	2	0
0	3	INF	0	2
15	3	12	INF	0
11	0	0	12	INF

The total expected cost at the root node is the sum of all reductions.

Cost = **[10 2 2 3 4] + [1 0 3 0 0] = 25**

Since we have discovered the root node C0, the next node to be expanded can be any node from C1, C2, C3, and C4. Whichever node has the minimum cost, that node will be expanded further. Therefore, we have to find out the expanding cost of each node.

The parent node (C0) has a below-reduced matrix –

INF	10	17	0	1
12	INF	11	2	0
0	3	INF	0	2
15	3	12	INF	0
11	0	0	12	INF

Let us consider the edge from 0 -> 1.

1. As we are adding edge (0, 1) to our search space, we set outgoing edges for city 0 to infinity and all incoming edges to city 1 to infinity. We also set (1, 0) to infinity.

So in the reduced matrix of a parent node, we change all the elements in row 0 and column 1 and at index (1, 0) to INFINITY (marked in red).

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS | Volume VIII Issue III March 2023

INF	10	17	0	1
12	INF	11	2	0
0	3	INF	0	2
15	3	12	INF	0
11	0	0	12	INF

The resulting cost matrix is:

INF	INF	INF	INF	INF
INF	INF	11	2	0
0	INF	INF	0	2
15	INF	12	INF	0
11	INF	0	12	INF

2. We try to calculate the lower bound of the path starting at node 1 using the above-resulting cost matrix. The lower bound is 0 as the matrix is already in reduced form. i.e. all rows and all columns have zero value.

Therefore for node 1, the cost will be

Cost = cost of node 0 + cost of the edge(0, 1) + the lower bound of the path starting at node 1

= 25 + 10 + 0 = 35

Let us consider the edge from 0 -> 2

1. Change all the elements in row 0 and column 2 and at index (2, 0) to INFINITY (marked in red).

INF	10	17	0	1
12	INF	11	2	0
0	3	INF	0	2
15	3	12	INF	0
11	0	0	12	INF

The resulting cost matrix is:

INF	INF	INF	INF	INF
12	INF	INF	2	0
INF	3	INF	0	2
15	3	INF	INF	0
11	0	INF	12	INF

2. Now calculate the lower bound of the path starting at node 2 using the approach discussed earlier. the resultant matrix will be –

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS | Volume VIII Issue III March 2023

INF	INF	INF	INF	INF
1	INF	INF	2	0
INF	3	INF	0	2
4	3	INF	INF	0
0	0	INF	12	INF

Therefore for node 2, the cost will be

Cost = cost of node 0 + cost of the edge(0, 2) + lower bound of the path starting at node 2

= 25 + 17 + 11 = 53

Let us consider the edge from 0 -> 3.

1. Change all the elements in row 0 and column 3 and at index (3, 0) to INFINITY (marked in red).

INF	10	17	0	1
12	INF	11	2	0
0	3	INF	0	2
15	3	12	INF	0
11	0	0	12	INF

The resulting cost matrix is:

INF	INF	INF	INF	INF
12	INF	11	INF	0
0	3	INF	INF	2
INF	3	12	INF	0
11	0	0	INF	INF

2. Now calculate the lower bound of the path starting at node 3 using the approach discussed earlier. The Lower bound of the path starting at node 3 is 0 as it is already in reduced form. i.e. all rows and all columns have zero value.

Therefore for node 3, the cost will be

Cost = cost of node 0 +

cost of the edge(0, 3) +

The lower bound of the path starting at node 3

= 25 + 0 + 0 = 25

Similarly, we calculate the cost for 0 -> 4. Its cost will be 31.

Now we find a live node with the least estimated cost. Live nodes 1, 2, 3, and 4 have costs of 35, 53, 25, and 31 respectively. The minimum among them is Node 3 having costs 25. Therefore, node 3 will be expanded further as shown in the state space tree diagram. After adding its children to the list of live nodes, we again find a live node with the least cost and expand it. We continue the search until a leaf is encountered in a space search tree. If a leaf is encountered, then the tour is completed and we will return to the root node.

The proposed method, which is using Branch & Bound, is better because it prepares the matrices in different steps. At each step, the cost matrix is calculated. From the initial point, we come to know what can be the minimum cost of the tour. The cost in the initial stages is not the exact cost but it gives some idea because it is the approximated cost. At each step, it gives us a strong reason that which node we should travel the next and which one not. It gives this fact in terms of the cost of expanding a particular node.

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS | Volume VIII Issue III March 2023

III. System Analysis And Design

The analysis model is a concise, precise abstraction of what the desired system must do, and not how it will be done after the study of the existing system is completed. This includes system study and requirement analysis. Interacting with the courier company regarding their requirements and expectations from the system requirement analysis.

3.1 Review of Existing System

Originally, the Courier man goes to the courier office to check his/her traveling schedule for the day. After getting the Travelling Schedule from the courier office to know how many cities to be visited, He now starts the delivery of the parcel without checking the shortest route, what the courier person knows is that he/she is to deliver the parcel and he does it at random, he/she goes to a particular location twice. With this traditional way of delivery, we can now observe that the courier person does not have a clue of the shortest path or even the distance he/she is to cover because delivery was done at random.

3.2 Methodology Adopted

The method used in carrying out this research work was Object Oriented Design Model.

Object-oriented design is a popular technical approach for analyzing and designing an application, system, or business by applying object-oriented programming, as well as using visual modeling throughout the development life cycles to foster stakeholder communication and product quality. Object-oriented programming in this context is a programming paradigm based on the concept of object, which may contain data, in the form of fields, often known as attributes; and code, in the form of procedures, often known as a method. In computer programs context of OOP is designed by making them out of objects that interact with one another. There is significant diversity of OOP languages, the most popular ones are class-based, meaning that objects are instances of classes, which typically also determined their type.

In the process of carrying out this research, a feasibility study was adopted to show the workability of the existing system. Requirement analysis was also drawn to show the components of the existing system so that the component of the proposed system can be derived or added to the existing system like battery, system, internet, etc.

The proposed system architecture of the system was drawn to show all the components that will make up the system and finally, the class diagram of the architecture was also gotten, showing all, the modules that make up the system like the manager module, customer module, and Roll module to determine the total of customer or good to be delivered

3.2.1 Feasibility study

A feasibility study is a necessary activity at this stage was carried out to make a judgment on whether the application system is possibly worthy of construction or not. The three main aspects that were considered in this study are the cost, time, and distance as in the case of this research work, which is the main cause of project failure. Thus, the feasibility study helped in identifying the risk factors that may be faced while developing the system, and in planning for analyzing these risks.

3.2.2 Requirement specification

Before this proposed platform can be deployed to any courier company, a few considerations must first be addressed, such as an internet connection, and a complete system to run the platform with a good battery backup plan to protect the system from power failure.

3.2.3 System architecture

The Courier man goes to the courier office as shown in figure 4 to check his/her traveling schedule for the day. After getting his Travelling Schedule from the courier office to know how many cities to be visited, He/she now checks the shortest route from the proposed system of the cities to be visited without repeating any cities that he/she has toured and then uses the Google map platform to Choose the travel mode (walking, car or bicycle), from the courier office, the courier man moves to the garage to Load consignment and also taking in consideration the weight of the package, the location to be delivered, the cost of delivery of the Invoke Package.

The courier man can decide to have the new application installed on his or her phone so that he or she can easily have access to finding out the shortest route without making the mistake of the tour a particular route twice.

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS | Volume VIII Issue III March 2023

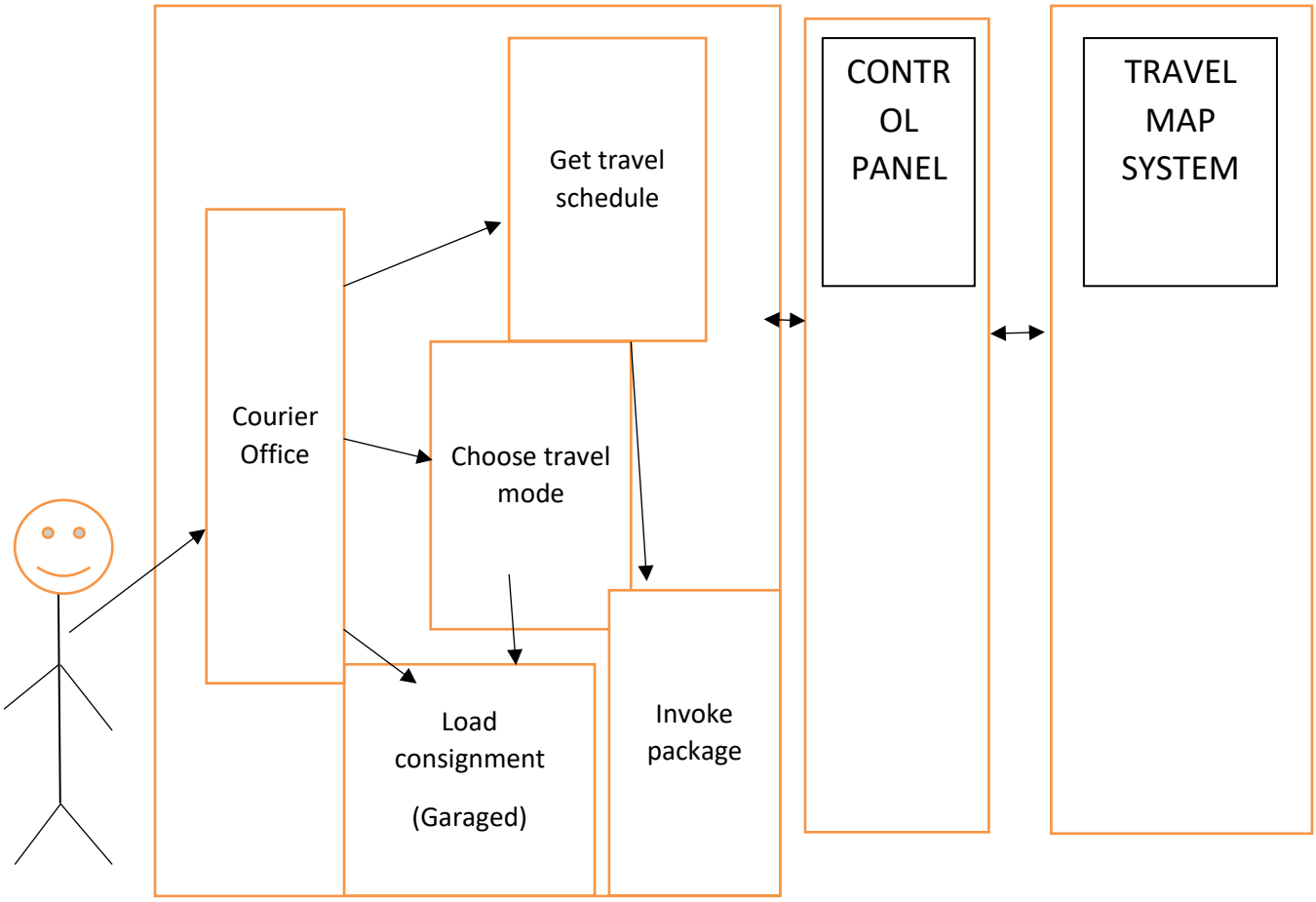

Figure 4: SYSTEM ARCHITECTURE

3.3 Entities Relationship Diagram

This entity relationship Diagram represents the model of the courier management system entity as shown in figure 5. The entity-relationship diagram of the courier management system shows all the visual instruments of database tables and the relations between customers, managers, couriers, delivery, etc. it used structured data to define the relationships between structured data groups of the courier management system functionalities. The main entities and attributes of a courier management system are shown in figure 5

Description of a courier management system Database:

1. The details of the courier are stored in courier tables respectively with all tables.
2. Each entity as shown in figure 5 contains a primary key and unique keys.
3. The entity office and shipment have bonded with the courier and customer entities with a foreign key.
4. There is one to one and one-to-many relationships available between shipment, manager, delivery, and courier.
5. All the entities are normalized and reduced the duplicity of records
6. We have implemented indexing on each table of the courier management system table for fast query execution.

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS | Volume VIII Issue III March 2023

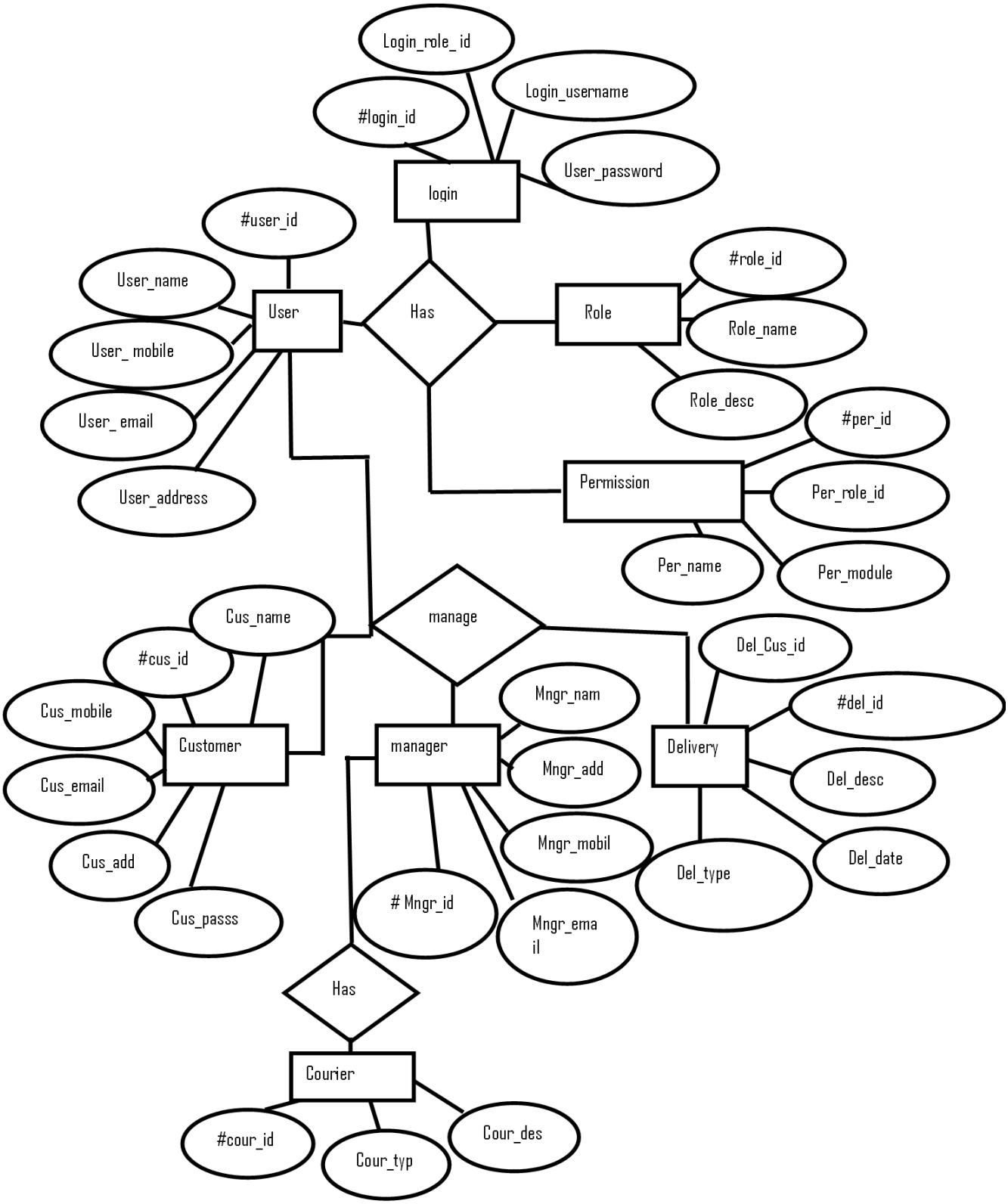

Figure 5 ER DIAGRAM FOR COURIER MANAGEMENT SYSTEM source: www.freeproject.com

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS | Volume VIII Issue III March 2023

3.4 System Implementation

Branch and bound techniques were used to find the most optimal solution. Each solution path has a cost associated with it. Branch and bound techniques keep track of the solution and its cost and continue searching for a better solution further on. The entire search space is usually searched. Each path and hence each arc (rather than the node) has a cost associated with it. The assumption made is that the cost increases with path length. Paths that are estimated to have a higher cost than paths already found are not pursued. These techniques are used in conjunction with depth-first and breadth-first searches as well as iterative deepening. Recall that the central idea of backtracking is to cut off a branch of the problem's state-space tree as soon as we can deduce that it cannot lead to a solution. This idea can be strengthened further if we deal with an optimization problem. An optimization problem seeks to minimize or maximize some objective function (a tour length, the value of items selected, the cost of an assignment, and the like), usually subject to some constraints. Note that in the standard terminology of optimization problems, a feasible solution is a point in the problem's search space that satisfies all the problem's constraints. To this end, a simulation of a Google map was a model showing that a salesman has a list of cities, each of which he must visit exactly once. There are direct roads between each pair of cities on the list and also to find and calculate the route that the salesman should follow for the shortest trip that both starts and finishes at any one of the cities.

The interface and beautification of the application were done using HTML, and other codes were embedded into the HTML. The HTML document is displayed in a browser. A browser is an application that is installed on the client's machine. The browser reads the HTML code and displays the page/map as instructed making it an interface. User interface design focuses on anticipating what users might need to do and ensuring that the interface has an element that is easy to access, JavaScript was to called Google Maps with its API key in it, which is displayed on a browser. it was also used to calculate the distance and find the shortest path. CSS was used to link the script or to call the next action that the user wants to be done.

IV. Results and Discussion

In the Traveling salesman problem, a rule-governed states that a traveling salesman cannot repeat a tour that He/ She has already toured meaning that if you use a certain route to your destination, then you must find an alternative route back to your departure point. In carrying out this research, the researcher had to develop a platform using Google Maps to view the entire route depending on where the courier man intends to visit. But in using Google Maps, the researcher needed a Google Maps API key which was gotten online and the animation marker was declared using the JQuery library. In carrying out the research also the weight of the parcels, the location for the parcels to be delivered, the amount to be paid, and longitude and longitude were also considered. HTML was used to design the interfere while javascript was used to code the scripting of the platform both calculation of distance and declaration of function and variable was done using javascript, maximum of nine (9) cities or locations are allow to be visited in a day or instance and if the number of cities to be visited exceed 9, A popped up will appear as shown in figure 6 below

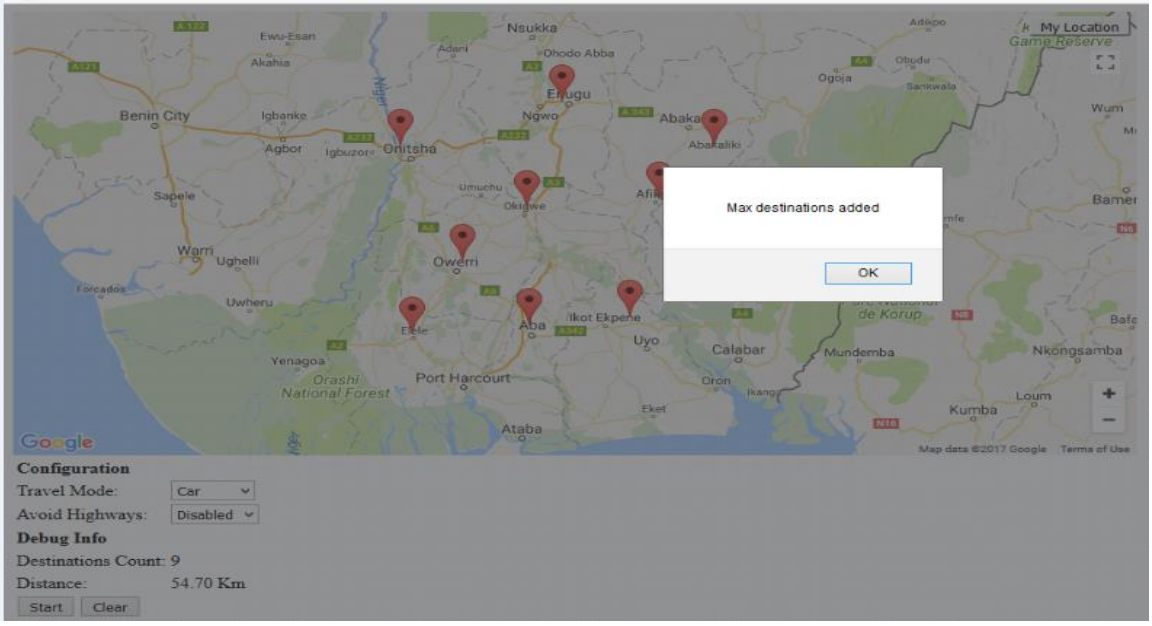

Figure 6: selected more cities than expected

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS | Volume VIII Issue III March 2023

When the courier man goes to the office to get his delivery for the day, He/ She will need to print or see the screen view to be taken by him as the case may be. In doing so, He/ She need to Log in to the system as shown in figure 7 and fill in his username and password, and also filled in all the location he/ she wishes to visit so that the shortest path and the distance can be calculated. Immediately the courier man who happens to be the traveling salesman, in this case, logged into the Google map platform as shown in figure 8, as you can see in it because no location has been selected, the destination count is zero (0) and distance has not been calculated, meaning that it is in its default mode. From the map, you can also decide which travel mode to use whether it is a car, walking, or bicycle as the case may be. The traveling salesman can also decide to avoid the highway if He chooses to.

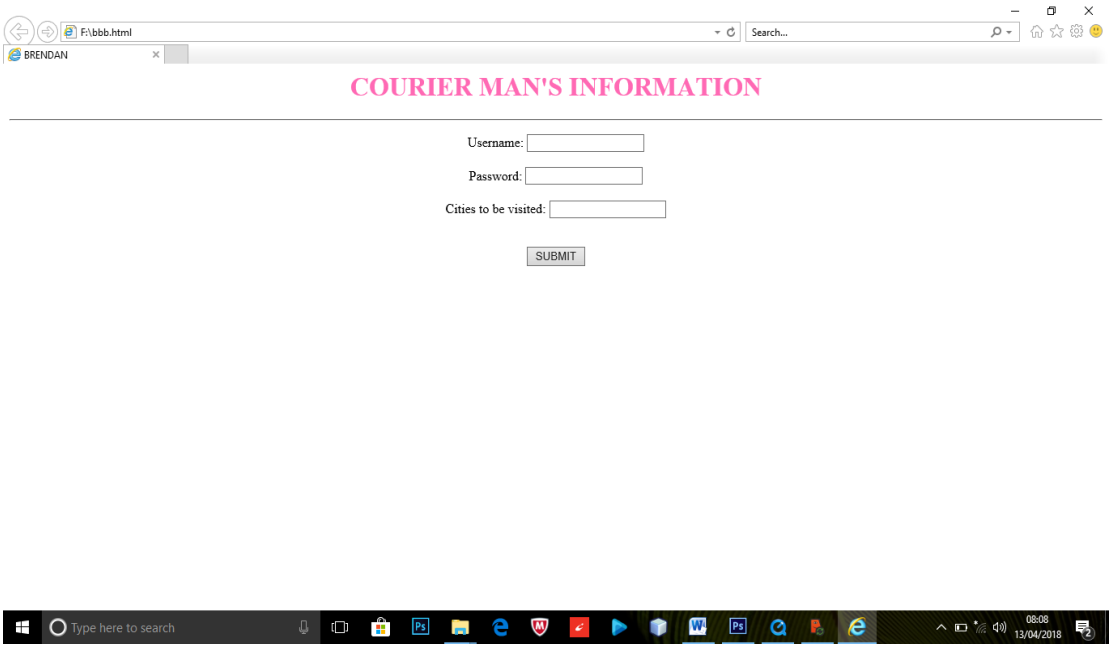

Figure 7: Welcome interface (The Login)

Figure 8: The map

Since the locations to be visited were spelled out at the login, Google maps now displace the selected cities as shown in figure 9. As you can see from the map, the destination count is five (5) since we choose five locations i.e Port Harcourt, Onitsha, Enugu, Uyo, and Calabar and we choose a car as our travel mode but remember the avoiding highway is optional. If the traveling salesman decides to cancel any tip then He can now click on the **clear** to return to the login (figure 7). After clicking on start, the Google Map will now call on the marker to draw a line linking the cities or locations that have been selected already as shown in figure 10, since we are finding the shortest path to a particular location without touring a location twice, the Google map through the help of a marker draw a straight line to show you the shortest path even as we have so many paths to a particular location before it zooms for us to see the village or route the salesman will pass through. As you can also see, the distance has been calculated.

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS | Volume VIII Issue III March 2023

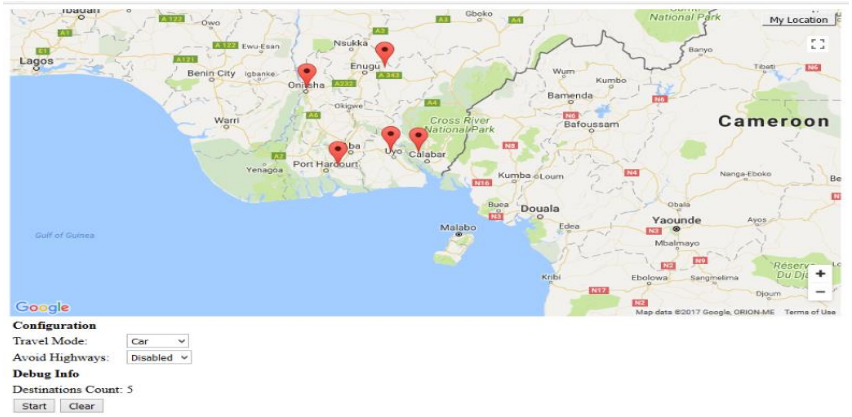

Figure 9: Selected cities in red color

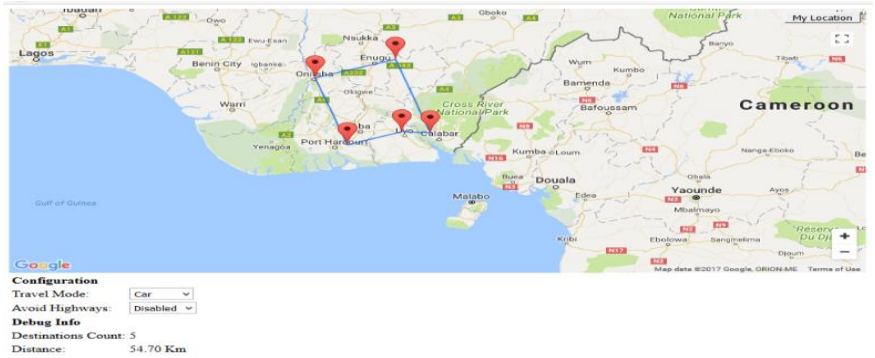

Figure 10: mapping out the coordinates of the selected cities

After then, it will zoom the map showing all the routes to be taken from one location to another as shown in figure 11, As we see from the map, from Calabar to Uyo is a straight road but from Uyo to Port Harcourt, The shortest Path is to pass through ikot ekpene then to Port Harcourt before Aba to Owerri express road before heading to Onitsha, from Onitsha we have a straight road to Enugu but in all this, we have other routes to this location but the shortest route for delivery is what we have shown you. Then a graph will happen to show the graphical representation of all the axis or locations to be visited as shown in figure 12

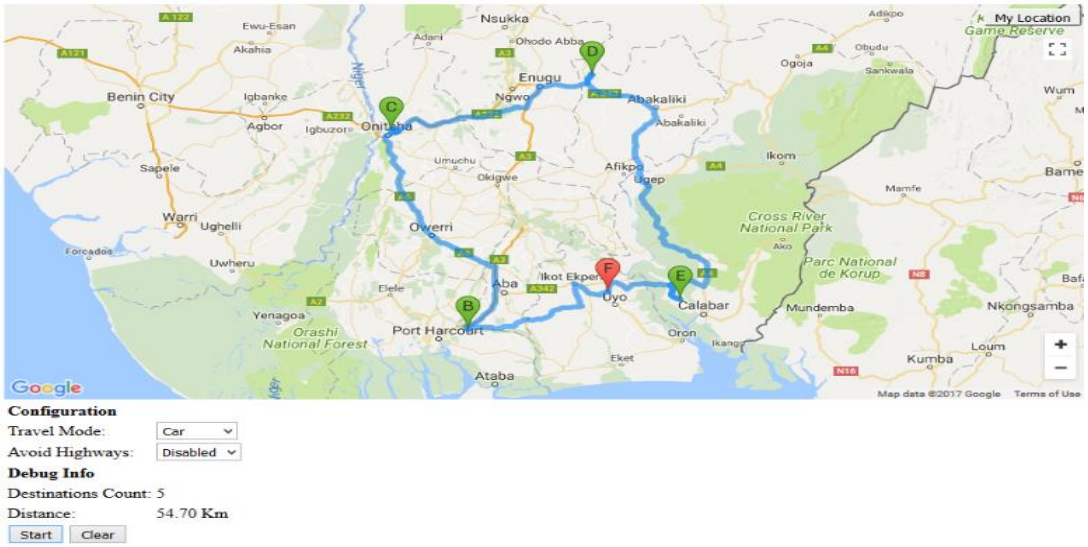

Figure 11: Zooming the coordinates to show the shortest path

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS | Volume VIII Issue III March 2023

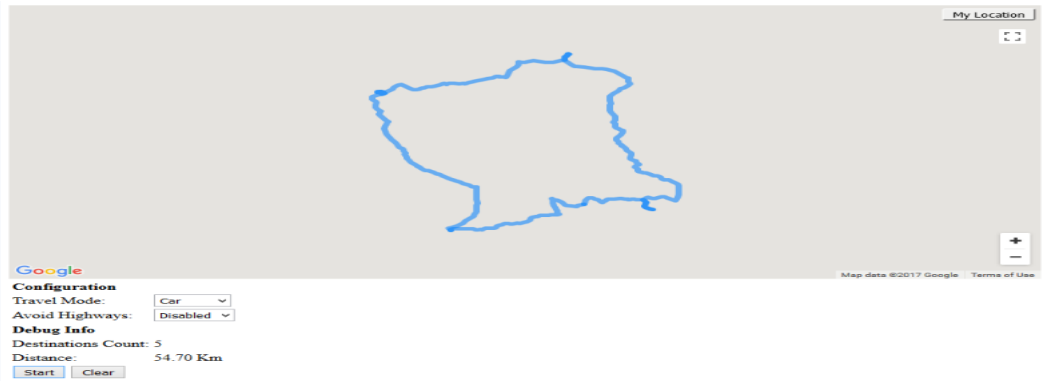

Figure 12: Graph Decomposition

Finally, the google map will load to its normal form showing the total distance that will be covered in the course of the journey as shown in figure 13

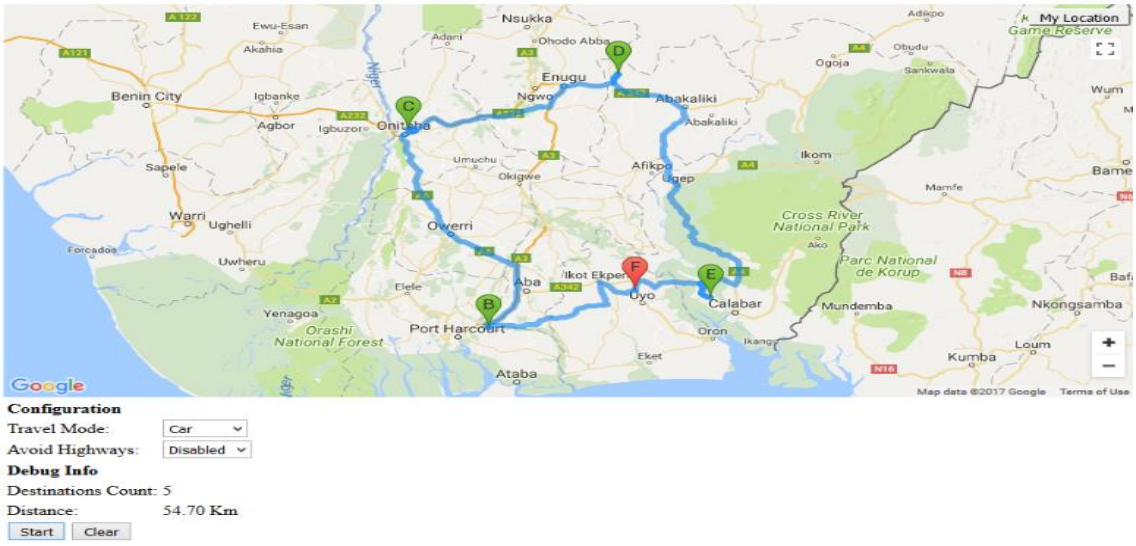

Figure 13 the results with the distance cover

V. Conclusions

At the end of this research, an application was built to show that if given a list of cities a traveling salesman can travel within them without touring any cities twice. Based on the result of the study it was concluded or shown that the aim and objective of the researcher were met. That is to say, a traveling salesman can travel through cities with minimal cost without repeating the route or cities twice with it distance calculated.

VI. Recommendation for future research

Based on the limitations inherent in this study, the researcher will strongly recommend that:

I. A real data-based study should be conducted in the study area to help validate the workability of the new System.
II. Courier company Research and Evaluation units should carry out an assessment study on the system to seriously investigate the possible effect if cooperated.

Reference

1. Aldous, D. (1996), "Probability distributions on cladograms", in Aldous, David; Pemantle, Robin, Random Discrete Structures, The IMA Volumes in Mathematics and its Applications, 76, Springer-Verlag, pp. 1–18.
2. Ben-Tal, A. and Nemirovski A. (1998). Robust convex optimization.Mathematics of Operations Research 23, 769–805.
3. Ben-Tal, A. and Nemirovski A. (1999). Robust solutions to uncertain programs.Operations Research Letters 25, 1–13.

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS | Volume VIII Issue III March 2023

4. "Cyclopædia, or, An universal dictionary of arts and sciences: Alguazil - anagram". digicoll.library.wisc.edu. Retrieved 2017-03-10.
5. "City Council Districts Archived January 17, 2010, at the WaybackMachin.." City of Sandy
6. Devroye, Luc (1986), "A note on the height of binary search trees", Journal of the ACM, 33 (3): 489–498, doi:10.1145/5925.5930
7. Devroye, L; Kruszewski, P. (1996), "The botanical beauty of random binary trees", in Brandenburg, Franz J., Graph Drawing: 3rd Int. Symp., GD'95, Passau, Germany, September 20-22, 1995, Lecture Notes in Computer Science, **1027**, Springer-Verlag, pp. 166–177, doi:10.1007/BFb0021801, ISBN 3-540-60723-4.
8. Drmota, M. (2009), Random Trees: An Interplay between Combinatorics and Probability, Springer-Verlag, ISBN 978-3-211-75355-2.
9. Devroye, L; Kruszewski, P. (1995), "A note on the Horton-Strahler number for random trees", Information Processing Letters, **56** (2): 95–99, doi:10.1016/0020-0190(95)00114-R.
10. Flajolet, P.; Raoult, J. C.; Vuillemin, J. (1979), "The number of registers required for evaluating arithmetic expressions", Theoretical Computer Science, **9** (1): 99–125, doi:10.1016/0304-3975(79)90009-4.
11. Hibbard, T. N. (1962), "Some combinatorial properties of certain trees with applications to searching and sorting", Journal of the ACM, **9** (1): 13–28, doi:10.1145/321105.321108.
12. https://next.ft.com/content/7acf8a4e-9ef3-11e4-ba25-00144feab7de
13. Hofmann, J. J.. "Lexicon Universale: ANA-". www.uni-mannheim.de. Retrieved 2017-03-10.
14. https://www.seas.gwu.edu/~bell/csci212/Branch_and_Bound.pdf
15. http://research.ijcaonline.org/volume65/number5/pxc3885866.pdf
16. IlgazSungur[a], Yingtao Ren[a], Fernando Ord´on˜ez[a]et, 2009
17. Knuth, D. M. (1973), "6.2.2 Binary Tree Searching", The Art of Computer Programming, **III**, Addison-Wesley, pp. 422–451.
18. Knuth, Donald M. (2005), "Draft of Section 7.2.1.6: Generating All Trees", The Art of Computer Programming, **IV**
19. IlgazSungur[a], Yingtao Ren[a], Fernando Ord´on˜ez[a]et (2009)**.**A Model and Algorithm for the Courier Delivery Problem with Uncertainty.Research supported by NSF under grant CMS-0409887 and by METRANS under grant 06-11
20. M. Mataija, M. RakamarićŠegić, F. Jozić: Solving the traveling salesman problem using the Branch and... ZbornikVeleučilišta u Rijeci, Vol. 4 (2016.), No. 1, pp. 259-
21. Retrieved from https://en.wikipedia.org/wiki/Courier#cite_note-1, 2017
22. Sungur, I. (2009). The Robust Vehicle Routing Problem: A novel approach to uncertainty in a classical transportation problem with real-life application.VDM Verlag Dr. Mller.
23. Sungur, I., F. Ord´on˜ez, and M. M. Dessouky (2008).A robust optimization approach for the capacitated vehicle routing problem with demand uncertainty.IIE Transactions 40, 509–523.
24. Springs.Retrieved on July 4, 2009.
25. " SEC 2014 Form 10-K Item 1 Business Overview.
26. United Parcel Service.Retrieved on May 17, 2016.
27. [ab]"UPS Fact Sheet". Retrieved June 24, 2016