SIMULATION OF TRAVELLING SALESMAN PROBLEM

A THIRD YEAR PROJECT REPORT

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF B.Sc. IN COMPUTATIONAL MATHEMATICS

BY

1. Samrajya Raj Acharya (22617)
2. Abhay Sharma (22635)
3. Diya Shrestha (22638)

SCHOOL OF SCIENCE
KATHMANDU UNIVERSITY
DHULIKHEL, NEPAL

SUBMITTED TO:

Dr. Ganga Ram Phaijoo

Department of Mathematics

December 18, 2022

CERTIFICATION

This project entitled "SIMULATION OF TRAVELLING SALESMAN PROBLEM" is carried out under my supervision for the specified entire period satisfactorily, and is hereby certified as a work done by following students

1. Samrajya Raj Acharya (Regd No :026592-19)
2. Abhay Sharma (Regd No : 026610-19)
3. Diya Shestha (Regd No : 026613-19)

in partial fulfillment of the requirements for the degree of B.Sc. in Computational Mathematics, Department of Mathematics, Kathmandu University, Dhulikhel, Nepal.

Prof.Dr.Ram Prasad Ghimire
Department of Mathematics
School of Science, Kathmandu University
Dhulikhel, Kavre, Nepal
Date: December 18, 2023

APPROVED BY:

I hereby declare that the candidate qualifies to submit this report of the Mathematics Project (COMP-311) to the Department of Mathematics.

Head of the Department
Department of Mathematics
School of Science
Kathmandu University
Date: December 18, 2023

CERTIFICATION

This project entitled "SIMULATION OF TRAVELLING SALESMAN PROBLEM" is carried out under my supervision for the specified entire period satisfactorily, and is hereby certified as a work done by following students

1. Samrajya Raj Acharya (Regd No :026592-19)
2. Abhay Sharma (Regd No : 026610-19)
3. Diya Shestha (Regd No : 026613-19)

in partial fulfillment of the requirements for the degree of B.Sc. in Computational Mathematics, Department of Mathematics, Kathmandu University, Dhulikhel, Nepal.

Mr. Sushil Nepal
Lecturer
Department of Computer Science and Engineering
School of Science, Kathmandu University
Dhulikhel, Kavre, Nepal
Date: December 18, 2023

APPROVED BY:

I hereby declare that the candidate qualifies to submit this report of the Mathematics Project (COMP-311) to the Department of Mathematics.

Head of the Department
Department of Mathematics
School of Science
Kathmandu University
Date: December 18, 2023

ACKNOWLEDGMENTS

This research was carried out under the supervision of **Prof.Dr.Ram Prasad Ghimire** and **Mr.Sushil Nepal**. We would like to express our sincere gratitude towards our supervisors for providing us with excellent supervision, guidance and suggestion for accomplishing this work.We would also like to thank **Mr.Harish Chandra Bhandari** for guiding us. We are thankful to the entire faculty of Department of Mathematics for encouraging, supporting and providing this opportunity.

We would also like to thank everyone who helped us directly and indirectly during the duration of completing our project work.

ABSTRACT

The Traveling Salesman Problem (TSP) is a classical combinatorial optimization NP-hard problem, which is simple to state but very difficult to solve. Although a global solution for the Travelling Salesman Problem does not yet exist, there are algorithms for an existing local solution.There are also necessary and sufficient conditions to determine if a possible solution does exist when one is not given a complete graph. This report gives an introduction to the Travelling Salesman Problem with an algorithm to visualise the optimal distance between the nodes. Given a list of nodes, we find the shortest path that visits all nodes at once.

Keywords: TSP, NP-hard, algorithm, graph, nodes

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

CHAPTER 1

INTRODUCTION

1.1 Background

Travelling Salesman Problem (TSP) is a classical and most widely studied problem in Combinatorial Optimization. This problem belongs to a category referred to as NP-complete ie. if its solution can be guessed and verified in polynomial time [2]. It has been studied intensively in both Operations Research and Computer Science since the 1950s as a result of which a very large number of methods were studied to solve this problem. The idea of the problem is to find the shortest route of salesman starting from a given city, visiting n cities only once and finally arriving at the origin city . The problem can be sketched on graphs with each city becoming a node[1]. Assuming a complete weighted graph, edge lengths correspond to the distance between the attached cities .

This problem can be formulated as:

- There are n points and ways between all of them with known lengths (which means it is simple to find the shortest way between any two points). The target is to find the shortest way, which passes all points right at once, start and end in the same point. It means the target is to find the shortest possible round trip.

The TSP occurs in countless forms with some applications of engineering that include Vehicle routing scheduling problems, integrated circuit designs, physical mapping problems, and constructing phylo-genetic trees[4].

Following are the constraints of a TSP problem:

- The total length of the loop should be a minimum.
- The salesperson cannot be at two different places at a particular time.
- The salesperson should visit each city only once.

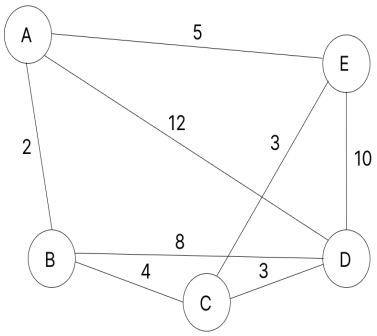

Figure 1.1: A graph with weights on its edges

According to the above figure, the problem lies in finding the shortest path to pass through all vertices at-least once.
For example, both Path1 (ABCDEA) and Path2 (ABCEDA) pass through all the vertices. However it can be seen that the total length of Path1 is 24, whereas the total length of Path2 is 31.

The real problem is not to find the simplest way to looking for the shortest round trip, one that exists and to find the optimal round trip[5]. There is, however, a problem, a major one. In this case it is necessary to try out all possible ways between all points, which is exactly the problem. Even a "layman" has to see this solution has to be abnormally time-consuming. In a case of a few points (e.g. four or five) the problem is not so complicated. Even though everybody can solve this with a pen and piece of paper, some proficient arithmetician can solve this problem even without any tools and help. In case of more point (e.g. eight or nine) it is possible to use some simple algorithm even this one with try out all of possible solutions[1]. But what if there is a case where there is the need to solve traveling salesman problem with thousands or tens thousands of points?

In table 1 it is possible to see that when the traveling salesman problem has only ten points the count of possible ways is really a huge number. This is such a large quantum of computations that it is impossible to solve it in real time. There were a great number

of scientists and other smart people in history who tried to find universal optimization algorithm for this problem, however, nobody was successful[4].

Count of points	Count of possible ways
4	24
5	120
6	720
7	5 040
8	40 320
9	362 880
10	3 628 800
11	39 916 800
12	479 001 600
13	6 227 020 800
14	87 178 291 200
15	1 307 674 368 00
16	20 922 789 888 000
17	355 687 428 096 000
18	6 402 373 705 728 000
19	121 645 100 408 832 000
20	2 432 902 008 176 640 000
25	15 511 210 043 330 985 984 000 000

Table 1.1: Count of permutation for solving TSP with exact algorithm

1.2 Related work

The traveling salesman problem was firstly officially defined around 1800 by Irish mathematician W. R. Hamilton and British mathematician Thomas Kirkman. Hamilton developed some kind of game for solving traveling salesman problem - "Icosian game", which is the so called Hamilton's game. The principle of this game is simple, the "player" has to find Hamilton's circle in the set of points. This problem got the final form around 1930 when professors began teaching about it at the University of Vienna and at Harvard University.

In the 1950s and 1960s this problem gained popularity in scientific circles not only in Europe but in the USA as well. Remarkable articles were published by George Dantzig, Delbert Ray Faulkerson and Selmer M. Johnson from RAND Corporation in Santa Mon-

ica. They transformed this problem onto a linear programming problem and developed method "cutting plane" for solving it. With this new method it was possible to solve the traveling salesman problem with 49 cities with an optimal solution which means this method found the shortest possible way around all points. For the next decades this problem was studied by many other scientists, mathematicians, computer scientists, physicists and many other scientists and enthusiasts[4].

Richard M. Karp proved that the problem of Hamilton's circle is NP-hard in 1972. This detection gave more apparent difficulty for finding optimal route.

A mathematician named Karl Menger, first studied the TSP problem in its general form in Vienna in the year 1920. In 1954, G. Dantzig et al. solved a non-trivial TSP with 49 cities. As the years passed, the number of cites for which the TSP has been solved gradually increased[5].

Research Team	**Year**	**Size of Instance**
G. Dantzig, R. Fulkerson, and S. Johnson	1954	49 cities
M. Held and R.M. Karp	1971	64 cities
P.M. Camerini, L. Fratta, and F. Maffioli	1975	100 cities
M. Grötschel	1977	120 cities
H. Crowder and M.W. Padberg	1980	318 cities
M. Padberg and G. Rinaldi	1987	532 cities
M. Grötschel and O. Holland	1987	666 cities
M. Padberg and G. Rinaldi	1987	2,392 cities
D. Applegate, R. Bixby, V. Chvátal, and	1994	7,397 cities

Table 1.2: Historical benchmarks

1.3 Objectives

The key objectives of Traveling Salesman Problem (TSP) is:

- to be an educational resource to help visualise, learn, and develop different algorithms for the travelling salesman problem in a way that's easily accessible
- to figure out the syntax and semantics of how JavaScript, HTML and CSS works as a whole
- to understand the basic framework, current applications and future scope of the algorithm
- to demonstrate the functionality of 'Branch and Bound' and 'Simulated Annealing' algorithm and site design using JavaScript as the core programming language

1.4 Limitations

There are several limitations to the Traveling Salesman Problem (TSP):

- It is NP-hard, meaning that it is computationally infeasible to solve exactly for large instances. As a result, heuristics and approximation algorithms are often used to find good, but not necessarily optimal, solutions to the TSP.
- It is a combinatorial optimization problem, which means that it becomes exponentially more difficult to solve as the number of cities increases. This makes it challenging to apply the TSP to real-world problems involving large numbers of cities.
- It assumes that the cost of traveling between any two cities is known and fixed. In practice, however, the cost of travel may vary based on factors such as time of day, mode of transportation, and availability of tickets.
- It does not consider additional constraints that may be present in real-world travel situations, such as the need to visit certain cities at specific times or the availability of accommodation.
- It does not take into account the possibility of non-linear routes, such as those involving ferries or flights.

- Despite these limitations, it remains a widely studied problem due to its practical importance and the insights it provides into the nature of optimization and complexity.

1.5 Future Works

The traveling salesman problem (TSP) involves finding the shortest possible route that visits a given set of cities and returns to the starting city. The problem is NP-hard, meaning that it is computationally infeasible to solve exactly for large instances. As a result, much research has been devoted to finding approximate solutions to the TSP, as well as developing efficient heuristics for finding good solutions quickly.

One area of ongoing research in the TSP is the development of better heuristics and approximation algorithms. These algorithms are designed to quickly find good, but not necessarily optimal, solutions to the TSP. Examples of such algorithms include local search algorithms, such as simulated annealing and tabu search, as well as metaheuristics, such as genetic algorithms and ant colony optimization.

Another area of research in the TSP is the development of exact algorithms for solving small instances of the problem. While it is not possible to solve large instances of the TSP exactly, it is possible to solve small instances to optimality using techniques such as branch and bound and dynamic programming.

Finally, there is ongoing research into the use of machine learning techniques for solving the TSP. These approaches aim to use data-driven methods to learn good solutions to the TSP, rather than relying on traditional mathematical optimization techniques. Examples of such approaches include neural network-based models and reinforcement learning algorithms.

CHAPTER 2

METHODOLOGY

2.1 Simulated Annealing

By definition, Simulated Annealing is a generic probabilistic meta-algorithm used to find an approximate solution to global optimization problems. Simulated Annealing is inspired by the concept of annealing in metallurgy which is a technique of controlled material cooling used to reduce defects (M. Sureja and V. Chawda, 2012) .

To use Simulated Annealing, the system must first be initialised with a particular configuration. The SA algorithm then starts with a random solution. With every iteration, a new random nearby solution is formed. If the solution is better, then it will replace the current solution[4].

If it is worse, then based on the probability of the temperature parameter, it may be chosen to replace the current solution. As the algorithm progresses, the temperature tends to decrease, thus making sure that worse solutions have a lesser chance of replacing the current solution.

2.1.1 Algorithm

Step 1:

Initialize – Start with a random initial placement.Initialize a very high "temperature".

Step 2:

Move – Perturb the placement through a defined move.

Step 3:

Calculate score – calculate the change in the score due to the move made.

Step 4:

Choose – Depending on the change in score, accept or reject the move. The prob of acceptance depending on the current "temperature".

Step 5:

Update and repeat– Update the temperature value by lowering the temperature. Go back to Step 2. The process is done until "Freezing Point" is reached.

2.1.2 Principle

Let $\mathbf{s} = \mathbf{s}_0$

$For\ \mathbf{k} = \mathbf{0} through k_{max}(exclusive)$:

$T \leftarrow temperature(k/k_{max})$

$pick a random neighbour, s_{new} \leftarrow neighbour(s)$

$If P(E(s), E(s_{new}), T) random(0, 1), move to$

$the new state$:

$s \leftarrow s_{new}$

$Output$: $the final states$

2.1.3 Flowchart

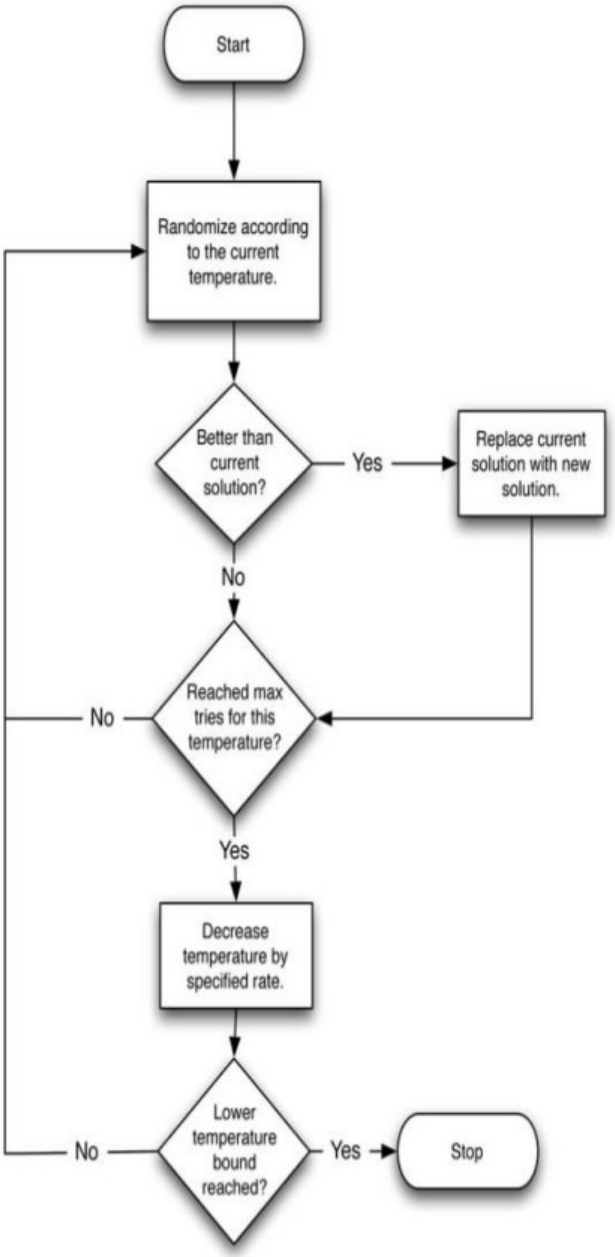

Figure 2.1: Flowchart of Simulated Annealing

As per the figure, if the energy of the new state is less than that of its previous one, then the change is accepted unconditionally and the system is updated. But if the energy is greater, than the latest configuration has the probabilistic chance or being accepted based on the temperature parameter.

2.1.4 Advantages

- Simulated annealing is easy to code and use.
- It does not rely on restrictive properties of the model and hence is versatile.
- It can deal with noisy data and highly non-linear models.
- Provides optimal solution for many problems and is robust.

2.1.5 Disadvantages

- A lot of parameters have to be tuned as it is meta-heuristic.
- The precision of the numbers used in its implementation has a significant effect on the quality of results.
- There is a trade off between the quality of result and the time taken for the algorithm to run.

2.2 Branch and Bound

There were at least three groups that independently discovered the Branch and bound technique. It was first applied by to solve asymmetric TSP. This exceptionally noteworthy paper also introduced quite a lot of other innovations. Land and Doig provided a self explanatory general description to solving integer programming problems by linear programming[5]. Little finally described and named this approach as Branch and Bound in an application to the TSP. Branch and bound is a method that uses a state space search where expansion of any of the children may take place when all the sub-problems of a node are created. Although similar to backtracking method it employs a breadth first search algorithm-like search.

The technique adopted by this method is to divide a problem into a number solvable sub-problem. It solves a series of sub problems of which each may have numerous possible solutions and where the chosen sub-problem for one solution may influence the possible solutions of later sub-problems.

A branch-and-bound algorithm comprises of a complete computation of all node solutions, where large subsets of ineffective nodes are discarded, by means of upper and lower approximated bounds of the optimize quantity[2]. To keep away from the complete computation of all partial graphs, a practical solution is first found and its value noted as an upper bound for the optimum. Computations are done as the distance exceeds the upper bound. The value of a new found cheaper solution is used as the new upper bound when found.

2.2.1 Algorithm

Step 1:

PUSH the root node into the stack.

Step 2:

If stack is empty, then stop and return failure.

Step 3:

If the top node of the stack is a goal node, then stop and return success.

Step 4:

Else POP the node from the stack. Process it and find all its successors. Find out the path containing all its successors as well as predecessors and then PUSH the successors which are belonging to the minimum or shortest path.

Step 5:

Exit.

2.2.2 Principle

Assuming an objective function is required to be minimized and assuming that there is a method for obtaining a lower bound on the cost of whichever solution amongst those in the set of solutions corresponding to some subset. If the subset with the best solution found so far costs less than the lower bound, then exploring that subset further is aborted.

Let S represent some subset of solutions.
L(S) = a lower bound on the cost of whichever solution belonging to S
Let C = best solution cost found so far
If C <= L(S), abort exploring S since it does not have any better solution.
If C > L(S), explore S further since it may have a better solution

2.2.3 Flowchart

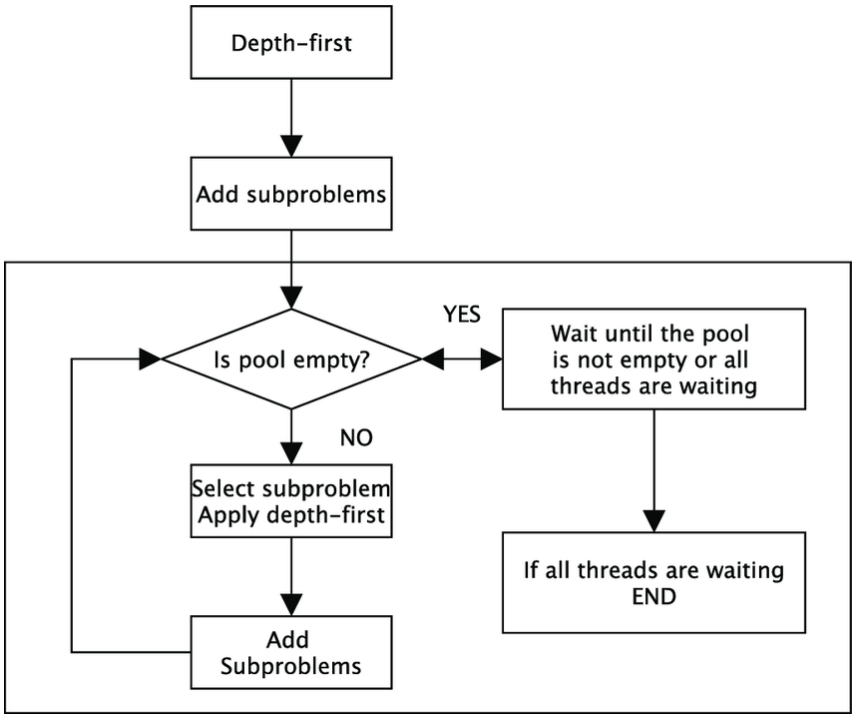

Figure 2.2: Flowchart of Branch and Bound

2.2.4 Advantages

- 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.

2.2.5 Disadvantages

- The load balancing aspects for Branch and Bound algorithm make it parallelization difficult.
- 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.

2.3 Simulated Annealing vs Branch and Bound

2.3.1 Speed Differences

Branch and Bound Algorithm takes less time to execute than the Simulated Annealing Algorithm. It is an algorithmic technique which finds the optimal solution by keeping the best solution found so far. If partial solution can't improve on the best it is abandoned, by this method the number of nodes which are explored can also be reduced. When the number of cities is increased, the execution time to find the shortest path through Branch and Bound module increases.

Algorithm	No of cities	Execution Time (in min)
Simulated Annealing	4	1.3523
	6	0.5742
	10	0.4734
	12	0.3046
Branch and Bound	4	0.0114
	6	0.0347
	10	0.0645
	12	0.0987

Figure 2.3: Comparison of Execution Time

2.3.2 Performance Evaluation

Simulated annealing can be used for very hard computational optimization problems where exact algorithms fail. Its benefits are its easy implementation and its possibility of finding a global optimal even after finding a local minimum, as it accepts solutions that are worse than the best candidate. It uses the objective function of an optimization problem instead of the energy of a material. Implementation of this algorithm is surprisingly simple. If graphs that have at least as many edges as they have nodes then the average time complexity of simulated annealing for a typical graph with n nodes is O(n4).

Branch and bound algorithms are used to find the optimal solution for combinatory, discrete, and general mathematical optimization problems. This algorithm can solve the TSP problem more efficiently because it does not calculate all possibilities. The worst case complexity of any branch-and-bound algorithm is exponential.

Algorithm	Feasible Solution	Optimal Result	Ease of Implementation	Simplicity
Simulated Annealing	☒	☐	☒	☒
Branch and Bound	☒	☒	☐	☐

Figure 2.4: Comparison of Performance Evaluation

2.4 Future Works

Future development of this problem is at least really unclear. As it was said there is not yet any polynomial complex algorithm for solving this problem, however, it is not impossible to find someone. Every year many organizations around the world make contests and give grants to find the best possible solution. In this time the progress of development in information technology is so fast and with it goes hand in hand the progress of the development of software solutions, not excepting traveling salesman problem. It can be compared how powerful computers were 10 years ago and how powerful are they today. It is possible to say computers are more powerful day by day. Maybe it is worth considering if in (near) future there could be a computer so powerful that it will be possible to solve the traveling salesman problem (and many others) just with the exact algorithm without any revolutionary optimize algorithm.

2.5 Language Used for solving TSP

To visualize the simulation of Travelling Salesman Problem, three programming languages has been used.

- **HTML** to define the content of web pages
- **CSS** to specify the layout of web pages
- **JavaScript** to program the behavior of web pages

2.5.1 HTML

HTML (HyperText Markup Language) is the most basic building block of the Web. It defines the meaning and structure of web content. Other technologies besides HTML are generally used to describe a web page's appearance/presentation (CSS) or functionality/behavior (JavaScript).

"Hypertext" refers to links that connect web pages to one another, either within a single website or between websites. Links are a fundamental aspect of the Web. By uploading content to the Internet and linking it to pages created by other people, you become an active participant in the World Wide Web.

2.5.2 CSS

CSS (Cascading Style Sheets) is a stylesheet language used to describe the presentation of a document written in HTML or XML (including XML dialects such as SVG, MathML or XHTML). CSS describes how elements should be rendered on screen, on paper, in speech, or on other media.

CSS is among the core languages of the open web and is standardized across Web browsers according to W3C specifications. Previously, the development of various parts of CSS specification was done synchronously, which allowed the versioning of the latest recommendations[1].

2.5.3 JavaScript

JavaScript (JS) is a lightweight, interpreted, or just-in-time compiled programming language with first-class functions. While it is most well-known as the scripting language for Web pages, many non-browser environments also use it, such as Node.js, Apache CouchDB and Adobe Acrobat. JavaScript is a prototype-based, multi-paradigm, single-threaded, dynamic language, supporting object-oriented, imperative, and declarative (e.g. functional programming) styles[3].

JavaScript's dynamic capabilities include runtime object construction, variable parameter lists, function variables, dynamic script creation (via eval), object introspection (via for...in and Object utilities), and source-code recovery (JavaScript functions store their source text and can be retrieved through toString()).

Libraries Used

- **jQuery**

  jQuery is a lightweight, open-source JavaScript library that helps us build interactive web pages with animations, visual effects, and advanced functionality. It is the most popular JavaScript library, used by around 70 million websites worldwide.

- **vis.js**

  Vis.js is a dynamic, browser based visualization library. The library is designed to be easy to use, handle large amounts of dynamic data, and enable manipulation of the data.

Functions Used

- **dragEnd**

  dragEnd function is used to move the nodes and/or drag the nodes so as to change the distance between the respective nodes.

- **tinyQueue**

  tinyQueue function is based on the binary heap priority queue concept. Both of the algorithms use this function to find the best optimal path.

- **rand**

  rand function is used to assign random temperature values for each nodes.

CHAPTER 3

RESULTS AND DISCUSSIONS

3.1 Index Page

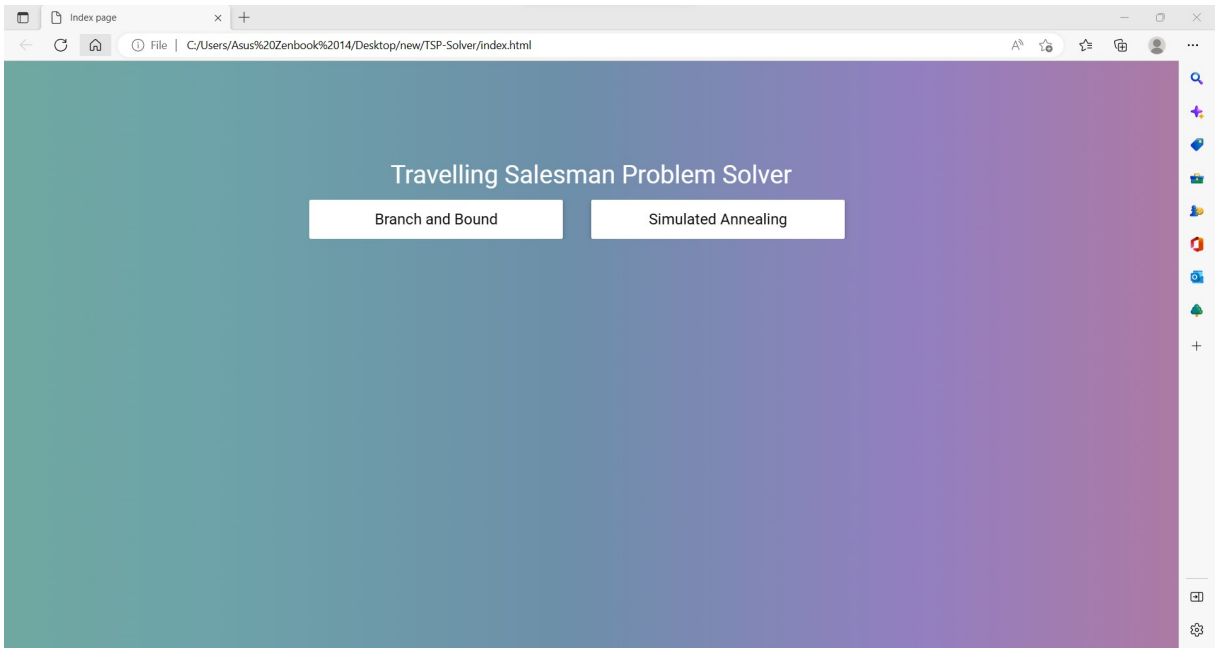

Figure 3.1: First responsive page having two algorithms to choose from

The portal webpage through which we choose the algorithm to execute the Traveling Salesman Problem. It contains the two algorithms used that is: Simulated Annealing and Branch and Bound.

3.2 Simulated Annealing

3.2.1 Generating Nodes

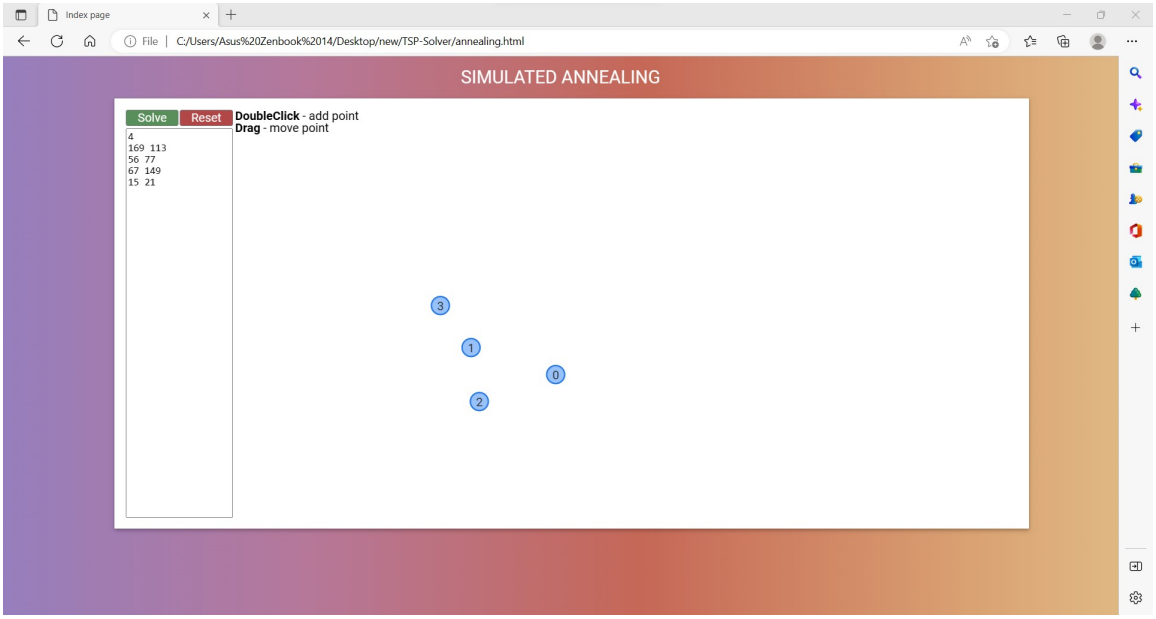

Figure 3.2: Introducing the nodes as required

Here, the coordinates of the nodes are entered or the user can place the nodes as desired by clicking on the nodes and moving around the screen.

3.2.2 Executing the Algorithm

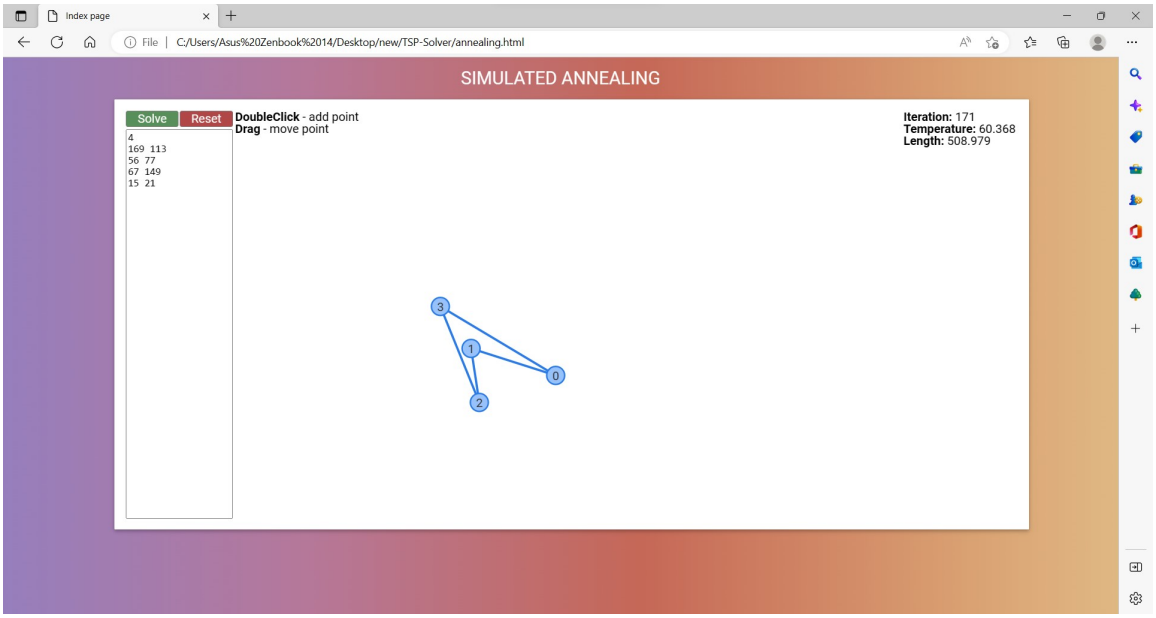

Figure 3.3: Executing the Algorithm

Once, we have entered the desired coordinates,we can start the algorithm and it finds the shortest path possible.

3.2.3 Final Result

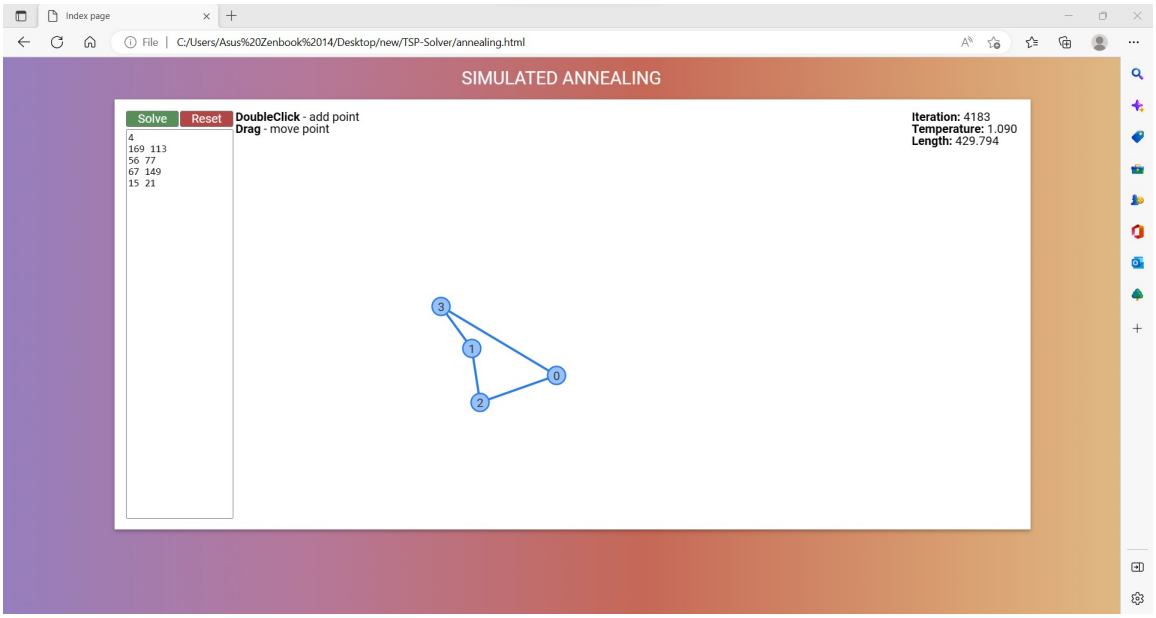

Figure 3.4: The shortest path generated

After the execution of the simulated annealing model,the shortest path is screened.

3.3 Branch and Bound

The process is same for the Branch and Bound method. start from entering nodes,executing the algorithm and finally acquiring the result.

3.3.1 Generating Nodes

Figure 3.5: Introducing the nodes as required

3.3.2 Executing the Algorithm

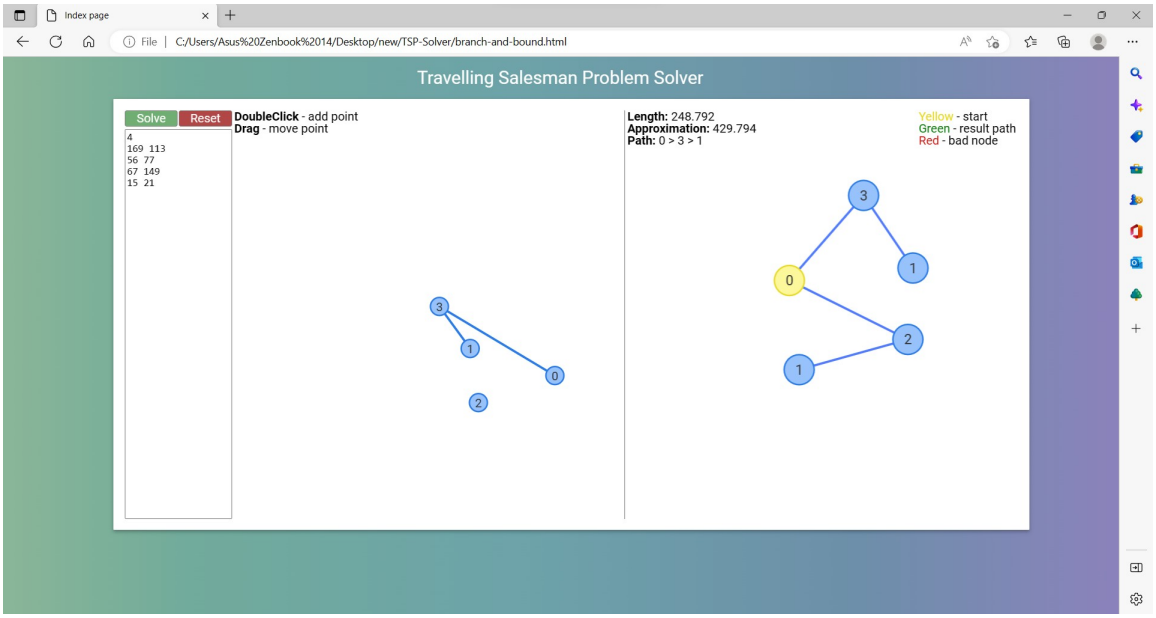

Figure 3.6: Executing the Algorithm

3.3.3 Final Result

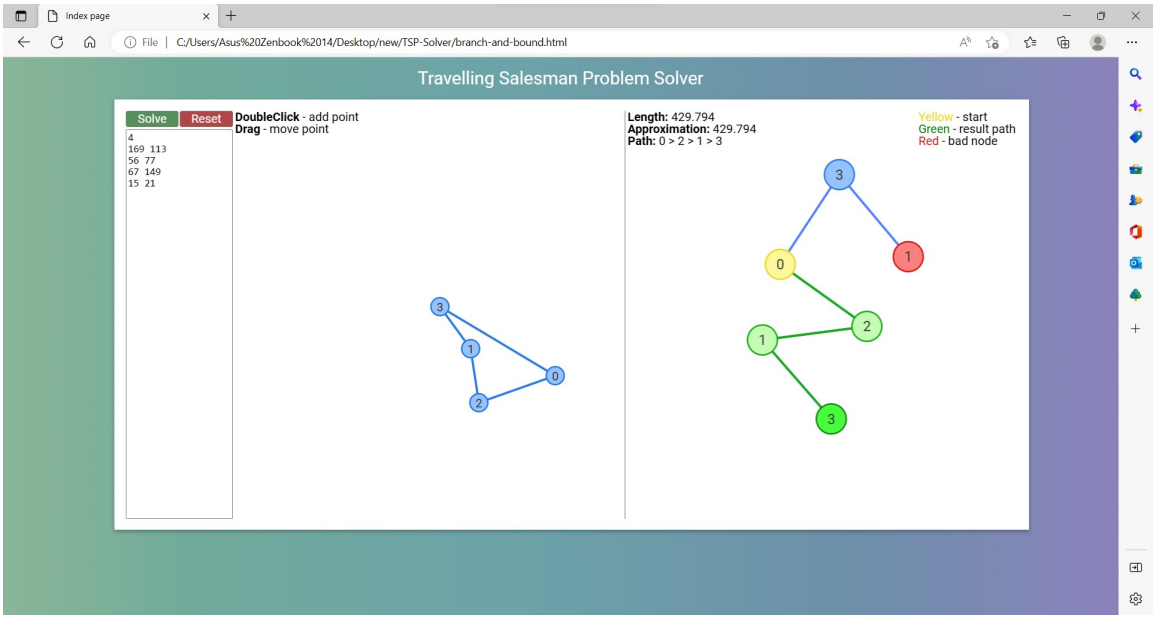

Figure 3.7: The shortest path generated

CHAPTER 4

CONCLUSION

There are many other algorithms for the Traveling Salesman Problem. Researchers are constantly trying to improve the results that have been around for years. In 1976, Nicos Christofides created an algorithm that gives a local solution which is at most 1.5 times the optimal global solution. This record was recently surpassed in September of 2012 by Andr as Seb o and Jens Vygen who created an algorithm that gives a local solution which is at most 1.4 times the optimal global solution. Furthermore, it is conjectured that one can find an algorithm that will give a local solution which is at most 4/3 times the optimal global solution.

Over the course of this project, we got to expand our knowledge on coding, since most of us are new to the subject and are very passionate about it. We manage to understand how the TSP's most common algorithms work and how they can be applied to solve many different types of problems. This helped us get a better understanding of how we can experiment with other projects. Thanks to this project, we can be prepared for future coding classes and know what to expect of them thanks to the challenge and our amazing mentors.

Whether we travel by a stage coach or a space ship, the Traveling Salesman Problem is relevant today and will continue to be relevant tomorrow.

REFERENCES

[1] Tolga Bektas, *The multiple traveling salesman problem: an overview of formulations and solution procedures*, omega **34** (2006), no. 3, 209–219.

[2] Steven G Krantz, *Essentials of mathematical thinking*, Chapman and Hall/CRC, 2017.

[3] Nitesh M Sureja and Bharat V Chawda, *Random travelling salesman problem using sa*, International Journal of Emerging Technology and Advanced Engineering **2** (2012), no. 4, 621–624.

[4] Hamdy A Taha, *Operations research: an introduction*, vol. 790, Pearson/Prentice Hall Upper Saddle River, NJ, USA, 2011.

[5] J William, *Cook. in pursuit of the traveling salesman. mathematics at the limits of computation*, (2012).