TRAVELLING SALESMAN PROBLEM

Juan Daniel Morales Arias
Juan Daniel Morales Arias
Universidad EAFIT
Medellin, Colombia
jmoral33@eafit.edu.co

Daniel Rendon Montaño
Daniel Rendon Montaño
Universidad EAFIT
Medellín, Colombia
drendon9@eafit.edu.co

Cristyan Sepúlveda Vásquez
Cristyan Sepúlveda Vásquez
Universidad EAFIT
Medellín, Colombia
bsepulv3@eafit.edu.co

ABSTRACT

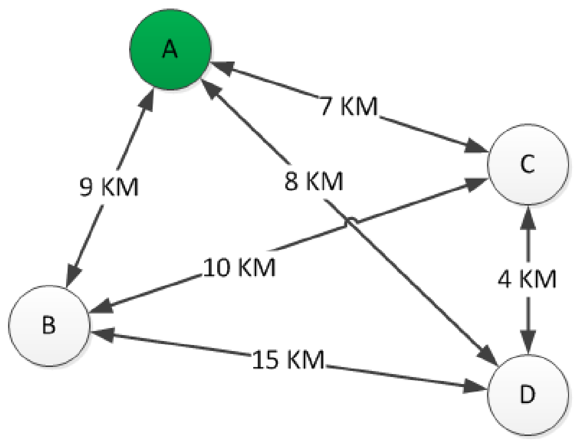
In this paper we describe some possible solutions for the travelling salesman problem. Such as brute force method, greedy algorithms and dynamic programming. We will also explain which data structure was chosen and why.

Keywords
Brute force, dynamic programming, greedy algorithms, map, list, matrix, run time.

1. INTRODUCTION

To solve this issue, we'll put forward some proposed alternatives in different algorithms. We'll evaluate each and every single one of them, in order to examine and determine which is the most optimal when giving a solution to this problem.

The traveling salesman problem has a big application in logistics and products delivering. Due to this issue's big O notation, it has been very effective in Academic solution with complex algorithms learning.



TSP example, cost routes [4]

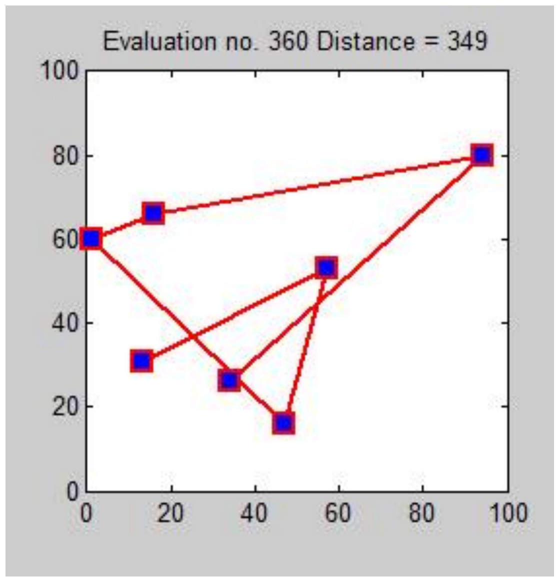
2. THE PROBLEM

Our problem is basically that we need to find the optimal path to use in order to travel around all the vertices and go back to the beginning. For a large amount of vertices and arcs, this will mean thousands of iterations, which also include computing resources. We want to find a way to solve this problem efficiently.

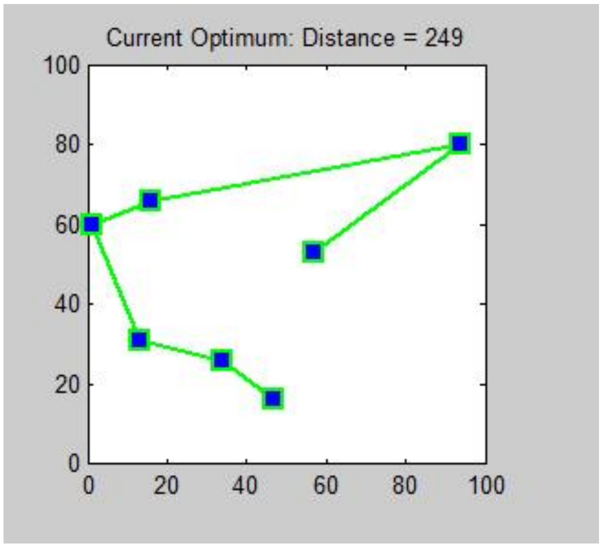
3. RELATED WORK

Brute force or proof by exhaustion:

This method as its name says, means to be inefficient. Consists in validating all the possible cases or instances of a scenario. Applying it to our travelling salesman problem will mean that we must check EVERY path in our map so we can decide which one we will choose. Its run time is O (n!). If our map (graph) has several amount of arcs and vertices, this will be a mess.



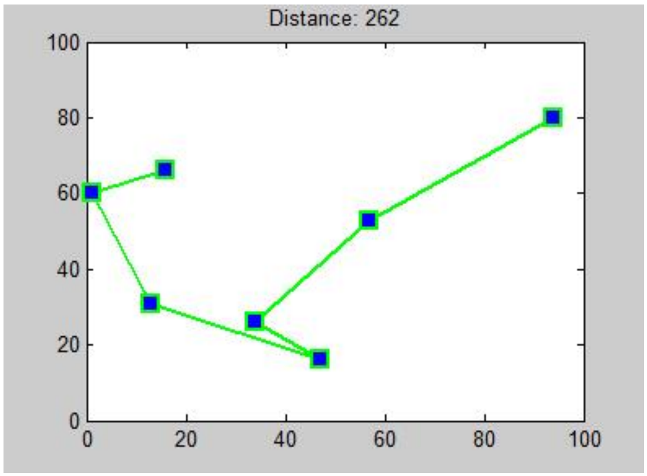
Solution to a symmetric TSP using brute force search [5]



Current optimum solution [6]

Greedy Algorithms:

This kind of algorithms will search the solution based on a heuristic which consists in selecting step by step the best option to find the best path between the beginning and the end. So going to our problem, this may cause less headaches than the previous solution because its run time decreases dramatically. O (n*n) [2]. But these algorithms have a HUGE problem, they in general may not produce an optimal solution.



Nearest Neighbor algorithm for a TSP [7]

3. ALGORITHM

We want to use dynamic programming to solve the TSP. This algorithm design demands a data structure to save previous solutions, meaning that we could use them to solve either other paths to calculate, or the same path that we are looking for if we wanted to look for it again [3] O(2^n*n^2).

5. DATA STRUCTURES

We will definitively use a graph with an adjacency list to represent our city's map. They use memory proportionally to the number vertices, which might save a lot of memory if the adjacency matrix is scattered.

We'll use a Hash map, that includes an adjacency list, due to the fact that the time to access is O(1), compared to a list which is O(n)

6. REFERENCES

[1] Anany Levitin, Design and Analysis of Algorithms p97.

[2] Anany Levitin, Design and Analysis of Algorithms p315.

[3] Anany Levitin, Design and Analysis of Algorithms p283.

[4] http://www.ingenieriaindustrialonline.com/herramientas-para-el-ingeniero-industrial/investigaci%C3%B3n-de-operaciones/problema-del-agente-viajero-tsp/

[5][6][7] https://en.wikipedia.org/wiki/Travelling_salesman_problem