Bachelor’s Thesis

Information and Communications Technology

2021

Diego Peribáñez Mortera

THE TRAVELING SALESMAN PROBLEM AND ITS APPLICATION TO CAR SHARING SERVICES

TURKU AMK

TURKU UNIVERSITY OF APPLIED SCIENCES

BACHELOR’S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Embedded Systems

2021 | 22 pages

Diego Peribáñez Mortera

THE TRAVELING SALESMAN PROBLEM AND ITS APPLICATION TO CAR SHARING SERVICES

With a rapidly growing population across the globe and the ease of car ownership, the challenges of the individual transport vehicles is showing every day. Individual transport vehicles not only cause a lack of space, but they also produce air, noise and visual pollution. A soluction to the problem of city road saturation is car sharing.

The purpose of the thesis is to explain the possible application of algorithms to transportation sharing services and its benefits. This research covers the history, evolution and derivations of these pathfinding algorithms.

The objectives have been achieved through an extensive research on both pathfinding problems and algorithms. Some of the documents consulted were from the early 20th century, which provide an overview of the evolution of the pathfinding problems.

The result is a comprehensive analysis on pathfinding algorithms and the derivations of the Traveling Salesman Problem. The reader will have a better understanding of the subject even with very little knowledge on programming.

The thesis serves as an introduction to pathfinding problems for new programmers. With very basic programming skills, any reader will be able to replicate the algorithms explained in the thesis.

KEYWORDS:

Traveling Salesman Problem, sharing economy, multiplatform development

CONTENTS

FIGURES

TABLES

LIST OF ABBREVIATIONS (OR) SYMBOLS

API	Application Programming Interface
BFS	Breadth First Search
CS	Computer Science
CTP	Canadian Traveller Problem
DFS	Depth First Search
LPA*	Lifelong Planing A*
TSP	Traveling Salesman Problem
SSSP	Single-Source Shortest Path

1 INTRODUCTION

Traveling around the world is, perhaps, what characterizes the human kind the most. Thanks to that, we as a species have shared knowledge that otherwise could have been forgotten, preventing the ancient civilizations from developing the first cities.

Because traveling is such an important activity, humans have developed the best means of transport. Although going faster or being able to transport more passengers or products is not the only improvement to traveling. This thesis aims to present methods of achieving more efficient routes between two or more points.

The cost of each travel can be measured in time or distance, and reducing one or the other can lead up to important benefits. For example, for a transport company, reducing the time of each travel could mean that they would require fewer vehicles, while reducing the distance could mean reducing the cost of driving each vehicle.

To calculate the best possible route it is necessary to apply an algorithm to the data available. Knowing distances, speed limits, traffic information, obstacles, or any other information of the graph and applying almost any pathfinding algorithm is faster and more efficient than choosing a route by hand. The computation cost of determining the best route is subject to the amount of nodes in the graph, so the more nodes there are, the harder for a person to calculate them without the use of a computer.

This thesis is structured as follows. The Chapter ”Pathfinding Algorithms” introduces the algorithms, or procedures, to solve the problem of finding a route and its history. The Chapter ”Car sharing services” explains the evolution of the means of transportation and it focus on the advantages of sharing transportation platforms. Finally, the Chapter ”Use of pathfinding algorithms for car sharing services” covers the advantages of applying pathfinding algorithms to share the transportation services available.

2 PATHFINDING ALGORITHMS

2.1 History of pathfinding algorithms

Throughout the 20th century, with the widespread of motorized vehicles, the need of newer and better roads became a general problem for the fastest growing countries. These countries needed to design modern infrastructures to interconnect the entire country, as the problem of lacking proper roads applied to both roads and cars. First of all, the resources for building roads were very limited and expensive. There were too many cities and villages with no motorized vehicles that would require new roads. Second, the vehicles had very limited capabilities so they would have a hard time driving off-road, or even driving for several hours on paved roads.

The governments built few roads that would cross the entire country, only connecting the most important cities. This way, a truck driver could drive from city A to city B, if there was a direct road, with no major issues. However, not all cities had a direct road connecting each other, meaning that , for example, to travel from city A to city C, it could probably require to drive by city B doubling the distance (Figure 1).

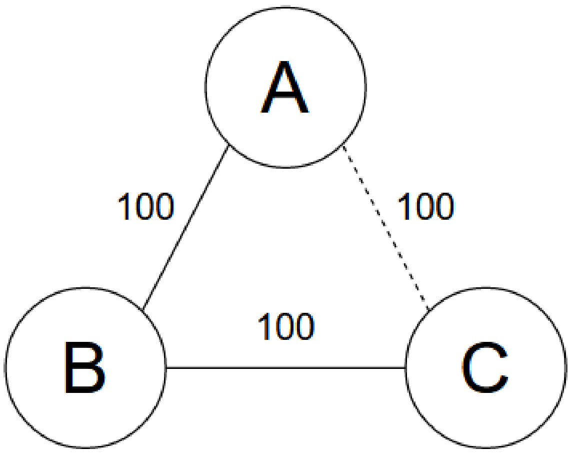

Figure 1. Interconnection of cities.

The problem of interconnecting points in a map can be treated as an optimization problem in graph theory. If every city or population is represented as nodes, and the roads as the edges connecting the nodes, every map can be represented as a graph and thus, the optimal paths can be obtained between two or more points. With this simplification, the study of graph theory expanded considerably.

To solve the problem of interconnecting as many points in a map as possible while also using less resources, mathematicians tried to determine the most efficient pathfinding algorithms. These algorithms, even though simple to apply, may take much time depending on the size and nature of the data available.

To speed up this process, the computers evolved and engineers managed to almost duplicate their speed with every new model while lowering the costs. The computers slowly stopped being reserved for government buildings and entered the market for private companies. It was then when most of the population could access these machines and benefit from their computation power.

2.2 Exact and approximate algorithms

There are two different approaches to determine the reliability of the solution provided by an algorithm. The first is an exact algorithm, which finds the optimal solution to a given problem. The second is an approximate algorithm, which although does not always gives the best solution, it gives an approximation that could be as good.

It is always best to give the optimal solution instead of an approximation but, depending on the data provided and its size, finding the optimal solution could take more computing time and resources than what one could afford.

An approximation is better than an exact solution when the time needed to calculate a solution is crucial, which is the case of real-time systems and when accuracy is crucial. In the case of real-time systems, there is already a classification depending on the deadline: hard, firm and soft. An specific example of hard real-time system is a self-driving car, which needs to process its environment information in order to choose the best action, and not processing in time a critical information could result in fatal accident.

2.3 Shortest Path Problem

The Shortest Path Problem answers the question of what is the shortest path (by time, distance, or any other measure) between nodes in a graph. It is an optimization problem in graph theory and there have been numerous algorithms depending on the variation:

- The Single-pair shortest path problem. The goal is to find the shortest path between two given nodes. This is the most known variation as it is the most common problem.
- The Single-source shortest path problem. Given an initial node, the goal is to find the shortest paths to the rest of the nodes.
- The Single-destination shortest path problem. Similar to the previous variation, the Single-destination finds the shortest paths from all the nodes to a given destination.
- The All-pairs shortest path problem. The result is the shortest paths among all the nodes. This variation has the highest spatial and time complexity.

The last three variations have their own algorithms with higher efficiency. For the single-pair shortest path problem, the most relevant algorithm is Dijkstra's.

The single-source and single-destination problems, require the same computing time and can both be solved using the same simple algorithm. In the case of a symmetric graph, the shortest paths from a pair of nodes (node A being initial or final node, respectively, and any other node in the graph) is the same in both. However, given an asymmetric graph, the shortest path between a pair of nodes is different although the time and spatial complexity remains the same.

Finally, for the all-pairs shortest path problem there are two algorithms used to find the optimal solution. These algorithms are the Floyd-Warshall algorithm and Johnson's algorithm. For sparse graphs, the latest algorithm is, in most cases, faster.

2.4 Traveling Salesman Problem

The Traveling Salesman Problem, abbreviated as TSP, is a problem consisting of finding the shortest route covering every city and going back to the starting point. In the most common version of this problem, the cost of traveling between two cities is symmetrical, as both ways share the same cost.

Although its exact origin is unclear, the Traveling Salesman Problem (also referred as TSP) can be traced back to the early 19th century. It was mathematically formulated in the 1800s by William R. Hamilton and Thomas Kirkman, and a general formulation of TSP also appeared in a handbook from 1832, including examples tours.

Later, in the 1930s, a new mathematical formulation by Merrill M. Flood appeared as a solution to a bus routing problem. Note that almost 90 years ago there were already attempts to create optimal navigation routes for motorized vehicles.

The importance of the Traveling Salesman Problem comes down to the use of pathfinding algorithms for a real life problem, such as the case of delivery trucks which have a very limited time to deliver every package before new packages arrive to the center. In this example, the delivery truck needs to travel to a series of directions exactly once, in the shortest period of time, and come back to the delivery center.

Classification of TSP

- sTSP: symmetric traveling salesman problem, also referred as sTSP, states that the cost *d* between the node *r* and the node *s* is the same in both ways, being $d_{rs} = d_{sr}$ for every pair of interconnected nodes.
- aTSP: asymmetric traveling salesman problem, also referred as aTSP, states that if at least one pair of interconnected nodes does not have equal cost in both ways, let it be $d_{rs} \neq d_{sr}$, it shall be classified as asymmetric.
- mTSP: multi traveling salesman problem, also referred as mTSP, states that let be *m* salesmen located at the same initial node, there shall be a solution in which each intermediate node is visited exactly once with minimized cost.

Applications of TSP

According to Matai et al., (2010) in Traveling Salesman Problem, Theory and Applications, the TSP can be applied not only to road maps but to a wide variety of fields. Few examples of this are explained in this reference book, and will be numbered here.

1. Drilling of printed circuit boards.
2. Overhauling gas turbine engines.
3. X-Ray crystallography.
4. Computer wiring.
5. The order-picking problem in warehouses.

Efficiency of TSP

To be able to find an exact solution wih n cities, it requires to check (n – 1)! possible tours. Even for a reduced set of points, evaluating every possible tour is not practical as it requires a great amount of computing time (Table 1).

Table 1. TSP complexity using brute force.

NODES	COMPLEXITY
4	6
8	5040
12	39,916,800
24	25,852,016,738,884,976,640,000
36	10,333,147,966,386,144,929,666,651,337,523,200,000,000

The rapid increase in complexity is an obstacle that researchers have solved by developing new specific models.

Following the Held and Karp model (Held and Karp, 1962), it would be possible to easily find optimal routes for a small instance of TSP by storing the values of all possible optimal routes. Storing all of these values requires a space complexity of $(n - 1)2^{n-2}$, which means that for larger instances this model is not feasible.

Table 2. TSP complexity using the Held and Karp model.

NODES	COMPLEXITY
4	12
8	448
12	11,264
24	96,468,992
36	601,295,421,440
48	3,307,330,976,350,208
70	20,365,205,457,375,344,984,064

In Table 1, for smaller instances with 4 nodes, the spatial complexity is rather acceptable. As the node quantity increases, so does the spatial complexity and, even with small instances such as a 12-node graph, the complexity becomes more and more concerning.

However, for larger instances there is also a model developed by Dantzig, Fulkerson and Johnson called Cutting Plane Method, a model based on linear inequality systems (Dantzig et al., 1954).

If an acceptable solution is not necessarily the best one, there might be a known acceptance range. Knowing this acceptance range, a developer can adopt an heuristic to any chosen algorithm.

An heuristic is an information used to decide whether the current path complies with the goal. An example of heuristic is the maximum cost for a path, so if the algorithm exceeds this cost and it has not found the solution, it stops looking further into that specific path and switches to the next one.

The benefit of using heuristics is a drastic increase in speed when working with larger instances. And with this increase in speed it also comes a decrease in space complexity. The price to pay for this benefit is accuracy, as it will not be as accurate as an exact solution. But an approximate solution might be good enough depending on the context and needs.

2.5 Canadian Traveller Problem

The Canadian Traveller Problem (abbreviated as CTP from now on) is a generalization of the Shortest Path Problem. The CTP is applied to partially observable graphs, which is progressively revealed by exploring it.

The CTP has been formulated by C. H. Papadimitriou and M. Yannakakis in Shortest Paths Without a Map (Theoretical Computer Science, 84 (1) (1991), pp. 127-150) and, since then, it has become of great interest by researchers to formulate more specific variations of it.

Formulated after listening to the problematic of Canadian drivers, who had to travel through Canada while having blocked roads due to the snow, it became of interest in operations research as there are variations which include a probability for each edge to be or not to be available in the graph at any given time.

There are variations in which each edge has a probability of being available or not. These probabilities helps deciding whether to explore further a path or switch to another one. It corresponds to the true nature of the problem, as Canadian travelers do not know when a path might be blocked or not. These variations became of interest in operations research under the name of Stochastic Shortest Path Problem with Recourse, or SSPPR.

It is therefore more than a mere problem to find the shortest path between nodes. The CTP has a wide range of applications, including operations research, artificial intelligence and machine learning due to its statistical nature.

2.6 Dijkstra’s Algorithm

According to (Thorup, M., 1999), Dijkstra’s algorithm has become the base of newer developments in Single-Source Shortest Path (SSSP) algorithms for the last 60 years. These SSSP algorithms were basically extensions or modifications of Dijkstra’s algorithm, adjusting it to newer fields and needs that appeared progressively.

Given an undirected graph, the Dijkstra’s algorithm starts from an initial node and saves the distance to each node. If a node can be reached from multiple routes, the algorithm discards all the routes except from the shortest. This is similar to the A* algorithm, in which the algorithm once it knows the new path is bad, it will not check how bad it is.

With the heuristic updated, it now discards all the current paths with weight or distance longer than the new heuristic. It keeps expanding the remaining paths until they too reach the weight or distance equal or superior to h, or until they reach the goal node. If a new path reaches the goal node with lower weight or distance than the current heuristic, this heuristic will be updated and the previous best path from the initial node to the goal node will also be updated.

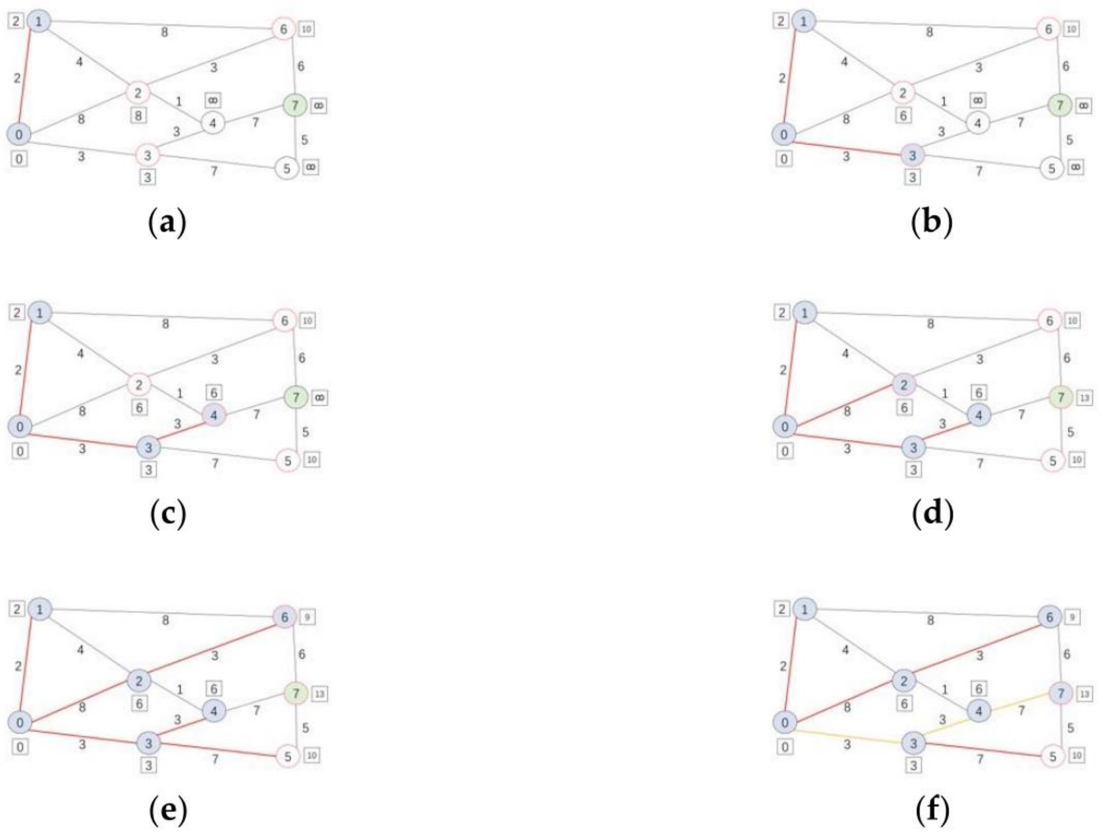

Figure 2. An example of Dijkstra's algorithm. (Pan and Pun-Cheng, 2020, p. 2).

As seen in Figure 2, the initial node is Node 0 and the goal node is Node 7. In the first iteration, the node with the smallest cost and directly connected to the initial node is updated with the cost of the actual shortest path (Node 1 shortest path cost is 2 as the edge connecting both Node 0 and Node 1 has a cost of 2 and the shortest path from the initial node and Node 0 is 0). In the next iterations, the algorithm expands the route of the path with the smallest cost. In (f) the algorithm stops and return the shortest path from Node 0 to Node 7, as there are no more paths to expand.

2.7 Evolution of Dijkstra’s Algorithm

As stated above, Dijkstra’s algorithm led to the research of new algorithms, each of them developed to cover very specific problems. An example to this is the Bellman-Ford algorithm, which covers graphs with negative edge weights, unlike Dijkstra’s.

A* (A Star)

The A* algorithm is an informed algorithm based on Dijkstra’s algorithm. An informed search algorithm starts its iterations with an initial information, usually an heuristic with a maximum cost of the path. The heuristic information is used to choose what node to explore next.

The main difference with Dijkstra’s algorithm is the use of heuristics. The behavior of Dijkstra’s would be the same as setting the heuristic of the A* algorithm to 0, it would explore paths with unacceptable costs and would take them as the best current solution.

The speed of this algorithm depends heavily on the heuristic function (Dechter, R. and Pearl, J. 1972), in charge of determining. An admissible heuristic would make the A* algorithm both complete and optimal, which means that it always finds the shortest path if there is one.

The A* algorithm can be further improved by the use of two or more heuristic functions, as there are more decisions to make than just to keep expanding the current path or not. One of those heuristic functions could choose candidate nodes, while another could choose the best node out of that list. This could speed up the algorithm in the case of dense graphs, where the amount of edges is close to the maximum number of edges.

D* (D Star)

Also referred to as Dynamic A*, this algorithm works very similar to A* in the first run. But, unlike A*, the D* algorithm is capable of quickly recalculate the optimal path from the current node to the goal node once the graph changes. This incremental search property makes it more suitable than the A* algorithm for real-time applications. Or simply when the data processed can change over time.

Another difference between these two algorithms is that D* starts from the goal node, back to the initial node.

A variation of this algorithm called D* Lite can achieve better performance while being easier to code. D* Lite is not based on D* although both share the same behavior, but it is based on the LPA* (Lifelong Planning A*) which is the incremental version of A*, so it can also be used when the graph suffers unexpected changes.

BFS

The Breadth First Search algorithm is used in trees and graphs structures to search through all the nodes, starting from the tree root (or any node in the graph). This algorithm processes all the neighbor nodes first, switching to the next depth level once there are no neighbor nodes left.

Unlike Dijkstra's, the BFS algorithm cannot be used to find the shortest path using weighted graphs. It does not consider an already visited node, so it would simply return a path from the initial node to the goal node, thus not being suitable to finding optimal paths between nodes.

DFS

In contrast with the BFS, the Depth First Search algorithm first search through the first neighbor of each node. Once it reaches the end of the tree or graph, it switches to the next previous possible neighbor until there are no more nodes to visit.

The clearest benefit from using DFS over BFS is when the data we want to extract is located in the lowest levels of the tree. But, like BFS, the DFS algorithm does not take into account weighted structures.

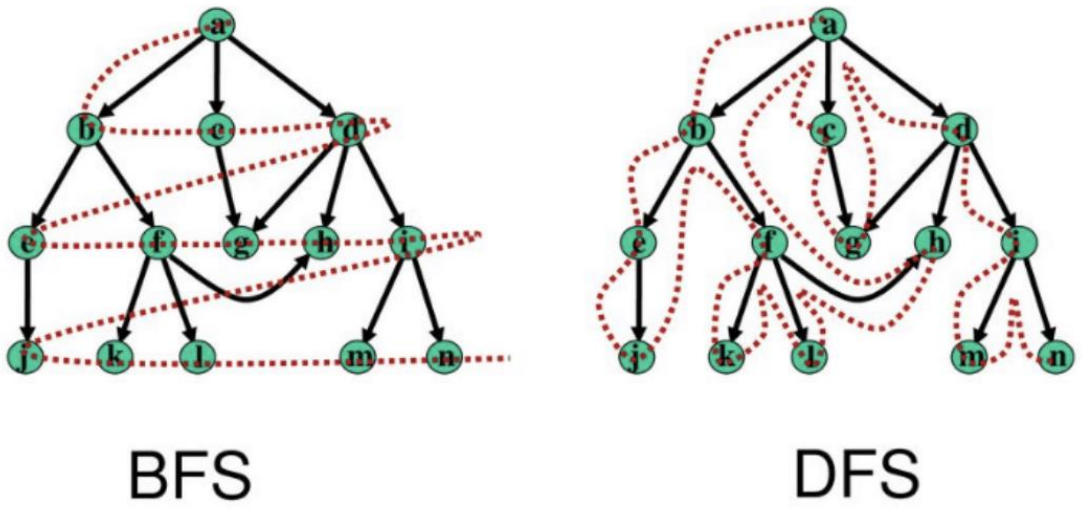

Figure 3. BFS vs. DFS from Discussion 11, 2018 Spring.

3 CAR SHARING SERVICES

3.1 About the taxi service in the 21st century

The taxi service dates back way before the development of motorized vehicles. In the early 17th century there were horse-drawn vehicles available for hire in Paris and London. The Hackney Carriage Act from 1635 regulated this kind of service in London, with requirements in height, seats, etcetera.

It evolved for centuries, and had a major impact in the mobility of big cities, leaving their mark in each urban history. New York City can not be pictured without their bright yellow cabs, or London without their black hackney carriages as they are called.

And because this service is so important in most cities, it is crucial that all the drivers provide a minimum of comfort for the customers. This leads to rigorous examinations to become a taxi driver, often memorizing all the streets, alleys and avenues in the city map. The experience of each driver can speed up the ride, knowing the day of the week, and the hour of the day, a driver could easily determine which route is the best to avoid most traffic. But it takes a lot of time and experience to get to that level. What if there was a better way of getting real-time information?

3.2 About transportation in a competitive market

For centuries, the hackney carriages and taxis have monopolized the public transportation. There appeared alternatives for mass transportation, such as the bus or the subway. These alternatives were a lot cheaper, but had fixed routes and were not as fast as a taxi.

However, to become a taxi driver one had to get through very demanding examinations. To name one example, the theoretical examination to become a taxi driver in London requires to study more than 320 routes and more than 25,000 streets. Without help of any map or device. This requirement plus the fact that the number of taxi licenses issued by each company or city is very limited pushed for a changed in the last decade.

Uber appeared as a startup in 2007, starting their beta launch in 2010 and officially launched in 2011. This company pursued the idea of sharing the cost of transportation from two points, as in the bus or the subway, while also having the benefits of choosing the route, as in a taxi. The early idea resulted in losses, as it required a professional driver and the vehicle available was a black luxury car, two things that made it more expensive than a regular cab. It was in early 2012 when Uber let users to register not only as customers, but also as drivers. But this still required a luxury car, something not many people willing to work in transportation had. A few months after, in July, UberX launched with a series of important changes, such as allowing new drivers to use their private non-luxury cars, and having a background check, minimum standards of safety, insurance for both the car and the driver, etcetera.

By 2014, Uber operated worldwide. As stated in the beginning of this point, the taxi service monopolized the public transportation, and the companies behind it would not give up their monopoly. Even before starting their operations outside the US, Uber faced several lawsuits regarding the legality of their operations, and the lack of control of their drivers' behavior.

This alternative to the traditional taxi soon met its own competitors. The Uber effect spred all over the world, and alternatives to the alternative arrived. This includes Lyft, DiDi, Ola… but almost all of them faced the same issues as Uber.

3.3 A better response for the daily commute

Sharing transportation goes way back to the widespread of personal vehicles. In the point History of pathfinding algorithms it has been explained how not all the cities or urban needed wide, quality modern roads at first. There were not enough motorized vehicles that would benefit from such expensive investment. So before paved roads, drivers would have to deal with muddy or rocky roads. This situation was not ideal for any kind of mass transport, specially the bus service. What many workers did back then was to either purchase a car, or find someone who owned one and could transport them to their destiny. This situation could be the first instance of car sharing in history. Another similar situation would be not necessarily that the worker is located somewhere with poorly made roads, but that a coworker is able to pick up them since they both work in the same area or even the same building.

It is not always possible to share vehicle with workers living in the same area, who share the same office area and same schedule. One of the reasons of this is because of insufficient communication. It is time to use the Internet to solve this.

The main idea of Uber is, from your mobile device, ask to get picked up from somewhere, and be transported somewhere else. The nearest driver will then be warned about a possible trip and it is up to him or her to accept to take your request. This system of driver-passenger is crucial for a car sharing application, a platform should not warn users who are not interested. In this case, a nearby driver is the most interested user and, if he or she does not accept the request or does not answer between a narrow time period, another nearby driver will be notified.

With the driver-passenger structure in mind, it is time to disclose the limitations of offer and demand for both drivers and passengers. Let it be user A, recently registered in the platform, wanting to publish his or her daily commute from his house to his office. After publishing the details of the vehicle such as number of seats or maximum number of passenger willing to take and other details such as schedule, the user publishes the trip. The user could only publish one 2-way trip each day to avoid spam, as that is what all the workers will be doing: from home to work, and from work to home.

Let it be user B, also recently registered in the platform. His or her intention is to find someone willing to take him or her from home to the office. The ideal situation would be to be picked up at home, but for privacy reasons or because the street may be reserved to pedestrian, the user will be able to choose wherever he or she wants to be picked up and left at. Adding the location and schedule, the application shall find the best trips already published. User B will then proceed to choose the best route available. Let it be user A the driver in the trip.

User A gets a notification of someone interested in being picked up. With the location and schedule, user A decides whether to pick user B up or not. If user A accepts, user B gets a notification with the confirmation and both users will met at the location and time set in the publication. If user A denies the trip, user B gets a notification to pick another trip if available.

3.4 Friendly with the planet, friendy with the people

The benefits of mass transportation are clear: it reduces the CO2 footprint from the fuel, it reduces the harmful impact of car manufacturing, it reduces the toxic and non-recyclable materials derived from the vehicle maintenance, it reduces the noise pollution and it decreases traffic making it faster and easier to get from one point to another.

However, it is not always possible to use public mass transportation. Sometimes the routes available from point A to point B are rather slow and, in bigger cities, it could take hours to get to the destiny. If this is the case, sharing a private car is as good as the public transportation. The impact of the pollution made by the car remains the same, but it heavily decreases per capita when the car is not empty.

Additionally, sharing a vehicle increases the social interaction of its passengers. Several studies throughout the century have stated that the average social interactions are slowly decreasing, even more among the younger generations. This decline has a negative impact in how lonely people may feel overall, and researchers had claimed that younger generations are, in average, feeling lonelier than their parents or grandparents.

In summary, sharing transportation is beneficial for everyone directly or indirectly.

4 USE OF PATHFINDING ALGORITHMS FOR CAR SHARING SERVICES

After discussing the efficiency of different algorithms, it is time to discuss the scalability of some of these algorithms applied to a practical problem of the 21st century: car sharing platforms.

For a car sharing service, it is important to understand what kind of data is going to be available. The number of seats, the coverage of the service, the personal information collected and the restrictions of each area in which the platform will operate in.

Given the average number of seats and the nature of the vehicle that will be shared, it is possible to determine the algorithm that best suits the needs. In the case of a 5-seats car, that is the most common kind of vehicle, there would be one seat for the driver, and four seats for the passengers. If all the passengers share the same destination, the graph can be pictured as seen in the Figure 4.

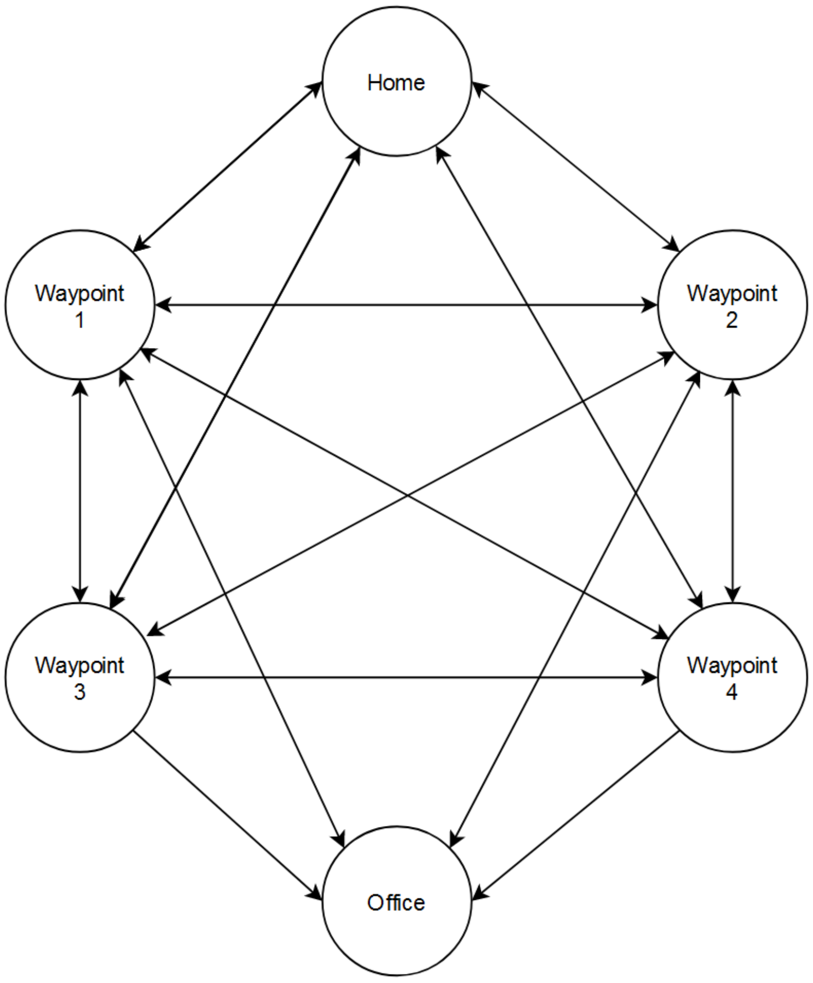

Figure 4. Maximum graph for a car sharing travel.

In the Figure 4 the number of nodes is 6 and the number of edges is 14. To simplify things, the graph will be undirected taking for granted that the distance between node A and node B is exactly the same in both ways, algthough in reality, in most cases there is at least a small variation.

Even though Figure 4 looks like a complete graph, there is no edge connecting node Home and node Office together, as there is no need to calculate the distance between the initial and final nodes if there are no waypoints.

However, it is possible to create a complete graph and add a fictional value to the edge connecting node Home and node Office as shown in Figure 5. This fictional value shall be large enough to not be taken into account when running the algorithm.

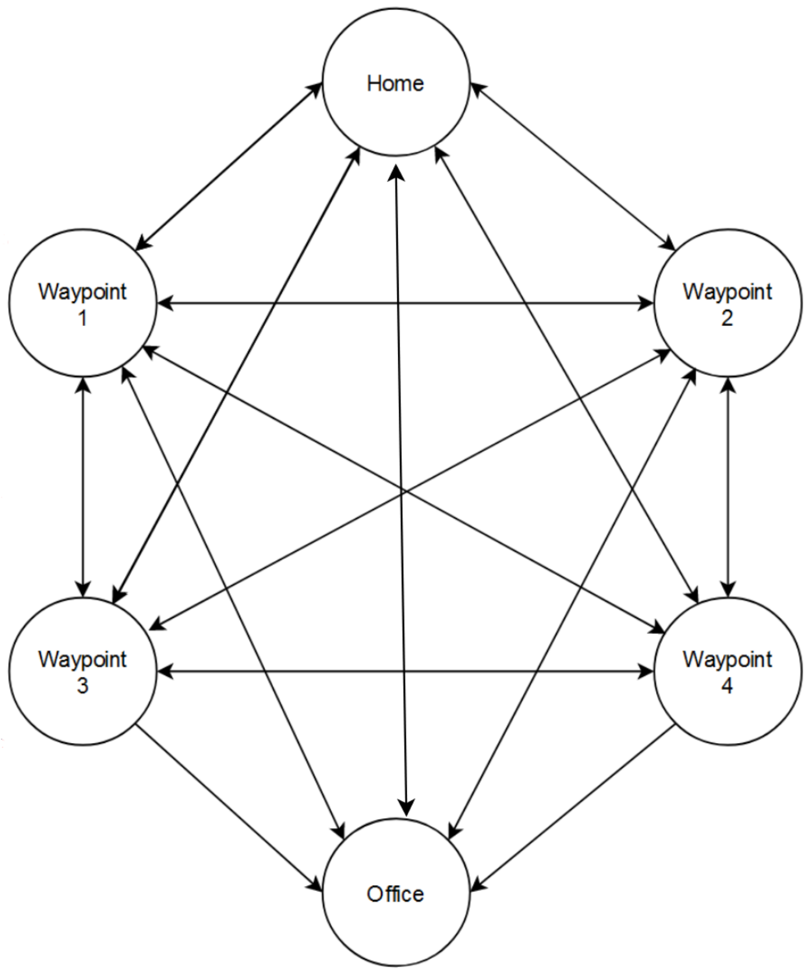

Figure 5. Maximum complete graph for a car sharing travel.

So given these 6 nodes, and applying the traditional brute force method used to solve a Traveling Salesman Problem, the complexity would be of 120, whereas applying the Held and Karp model the complexity would drop down to 80. It is reasonable to think that the brute force method is not suitable for this task.

However, this example might not be feasible if the vehicle sharing service is applied to a school bus, which picks up children at their houses and takes them to school. In this case, the number of passenger can easily exceed twice as much as the maximum number of passengers in Figure 5.

Given the fictional number of 50 passengers that a school bus would be able to pick up and applying again the brute force method, the complexity skyrockets at more than 3 novemdecillions. An unimaginable number that stays out of question. In the case of more refined models, such as the Held and Karp model, the complexity gets up to not much more than 13 quadrillions. Still an unimaginable number as well but at least it is not impossible to calculate.

The Held-Karp algorithm is not the most efficient model for large instances of a TSP problem. As it has been mentioned previously, when the number of nodes gets higher, it is necessary to make use of the Cutting Plane Method (Dantzig et al., 1954). Examples of larger instances solved by the Dantzig-Fulkerson-Johnson method can be found in (Applegate et al., 2003), where numerous algorithms were applied using the Cutting Plane Method on a publicly available library of TSP datasets from Princeton University.

As the complexity for smaller instances of TSP applying Held-Karp algorithm results appealing, it is not feasible for larger instances as shown in Table 2. The performance of the Cutting Plane Method (Dantzig et al., 1954) might be poorer in relative terms, but in absolute terms there is not much of a difference, while it is still feasible in the worst case scenario.

5 DISCUSSION

In this section, the importance of car sharing is discussed. In the chapters of this thesis, it has been presented the history and evolution of pathfinding algorithms. These algorithms date back when there were no computers and it was unimaginable to think of a device that would automatically calculate large equations.

But as the technology advanced, the data entrusted to the computers to be computed has exponentially increased. For pathfinding algorithms there is a noticeable difference between the data available to cross a country back in the 1950s, and the data available to cross a city today. Going from fewer roads, with no major traffic, to a congested city with a large road network has an impact on the amount of information that can be gathered.

Filtering useless information is a good strategy to decrease the computing time of the algorithms, as they would spend less time processing information that has no positive impact on the result.

With the technology integrated in newer cars, it is even easier to develop applications that helps finding the fastest route on the go. These new produced cars usually come with an integrated infotainment, which is composed of a screen, control buttons and an integration system with the smartphone. It is this integration with the smartphone that makes it even easier to be connected on the road. This infotainment system can be considered, then, a bridge between the driver and the world.

With an application developed to share the vehicle, it is possible to connect it to the car if its infotainment is based on Android Auto or Apple CarPlay. It can be used therefore as a navigation system while driving, making it smoother.

6 CONCLUSION

This thesis focused on the pathfinding algorithms, their history, evolution and how they can be used to improve the current situation caused by the excessive amount of active cars in modern cities.

Throughout the 20th century, the pathfinding algorithms have helped city planners plan the construction of roads to interconnect everyone. This planning aimed to reduce the resources needed to build and maintain these roads. With all the population in a country interconnected, a new problem appeared as the private vehicle became more affordable to the general public.

With more and more private vehicles, the space needed to park them and drive them remained almost the same. The cities tried to adapt their roads to this growth. However, it might not be necessary to do so.

In the discussion on car sharing services and the application of algorithms to reduce the amount of private cars and the time on the road, it was shown that these problems have a solution. This solution does not require an investment in infrastructure nor public transport, it does not require to promote the use of bicycles either. This solution offers each driver the possibility to contribute to the benefit of the society, to help lower the pollution of their city with no sacrifice.

The technology is already available to the public. There are tools available for free, such as Google Maps, that based on the traffic information they show the fastest route to a destination. Although there is public transportation, it is not necessarily the best option for some citizens, and it is these citizens who have a chance to contribute to decongest the roads by sharing theirs with travellers or travelling with someone willing to share their own vehicle.

Summing up, technology can be used by the public to contribute positively to the society. There are applications for the second hand market, for taxi services, for sharing transportation when travelling between cities. It is, therefore, time to offer a useful tool aimed at eliminating rid of the single-passenger travel.

REFERENCES

Applegate, D., Bixby, R., Chvátal, V. and Cook, W. (2003) *Implementing the Dantzig-Fukerson-Johnson algorithm for large traveling salesman problems*. Mathematical Programming, 97, p. 91-153.

Dantzig, G. B., Fulkerson, D. R. and Johnson, S. M. (1954) *Solution of a Large-scale Traveling Salesman Problem*. Operations Research, 2, p. 393-410.

Dreyfus, S. E. (1969) *An Appraisal of Some Shortest Path Algorithm*. Operations Research, June 1969.

Hahsler, M. and Hornik, K. (2007) TSP – *Infrastructure for the Traveling Salesperson Problem*. Journal of Statistical Software, Vol. 23, Issue 2.

Held, M. and Karp, R. M. (1962) *A Dynamic Programming Approach to Sequencing Problems*. Journal of SIAM, 10, p. 196-210.

Matai, R., Singh, S. P. and Mittal, M. L. (2010) *'Traveling Salesman Problem: An Overview of Applications, Formulations and Solution Approaches'*, in Donald Davendra (Ed.) *Traveling Salesman Problem, Theory and Applications*. London, England: IntecOpen, pp. 1-24.

Math Dept. of University of Waterloo (2015) University of Waterloo web page http://www.math.uwaterloo.ca/tsp/problem/index.html/ Retrieved 15-04-2021

Michael, T. (2017) The Sun web page. https://www.thesun.co.uk/news/3307245/the-knowledge-taxi-test-london-black-cab-drivers-exam/ Retrieved 18-04-2021

Pan, T. and Pun-Cheng, S. C. (2020). *A Discussion on the Evolution of the Pathfinding Algorithms*. Preprints 2020, 2020080627.

Papadimitriou, C. H. and Yannakakis, M. (1991) *Shortest Paths Without a Map*. Theoretical Computer Science, 84.

Thorup, M. (1999) *Undirected single-source shortest paths with positive integer weights in linear time*. Journal of the ACM 46 (1999), p. 362-394.

Twenge, Jean M., Spitzberg, Brian H. and Campbell, W. Keith (2019) *Less in-person social interaction with peers among U.S. adolescents in the 21st century and links to loneliness*. SAGE journals, Vol. 36 Issue 6.

UK Government (2008) UK Government Legislation web page. https://www.legislation.gov.uk/ukpga/Will4/1-2/22/ Retrieved 18-04-2021