Carpooling Implementation For Car Traffic Reduction

Pedro Quintero
Universidad EAFIT
Colombia
pquinterol@eafit.edu.co

Santiago Duque
Universidad EAFIT
Colombia
sduqueg@eafit.edu.co

Mauricio Toro
Universidad EAFIT
Colombia
mtorobe@eafit.edu.co

ABSTRACT
To write the abstract, you should answer the following questions in a paragraph: What is the problem? Why is the problem important? Which are the related problems?

Humanity is facing its worst problem of all times, which is setting at risk our existence and that the rest of the living creatures in the world, beside that, human growth its increasing exponentially and with it, the increase in cars used per person, this just makes worst the global warming problem and along it, the traffic in our cities. Modern circumstances have created many problems, and a solution that solves both problems previously mentioned simultaneously is needed, to ensure a drastic reduction in pollution and make life in cities more comfortable. In the purpose of solving both of this problem a carpooling approach that helps employees get into their job is an effective short-term solution for this problem. Why is it effective? Taking advantage that many users have to get to a same place in a determined time, there's is a big probability that some of them pass close to others, using this hypothesis and an algorithm that ensures that the time to get into company is no more than 10% of the time previously spent by each user, it will let us reduce the number of cars in our streets and the pollution as well, at the same time that employees get an acceptable delay in time to get into jobs.

1. INTRODUCTION
In the past few decades carbon dioxide has reached its highest levels in human history and increases in cars corresponding the growth of human population has contributed to it in a huge way. That's why a program that can helps us reduce the number of cars in our streets is needed.

2. PROBLEM
We need to develop a program that can takes as inputs the coordinates of users houses and outputs a minimum route number of carpooling route to get into the target place, without exceeding a 10% inverse in the time each user got into the target place previously.

3. RELATED WORK

3.1 Getting You Faster to Work – A Genetic Algorithm Approach to the Traffic Assignment Problem

The last decades saw a constant increase in road traffic demand. Trips tend to concentrate on a few central spots, like major roads or large intersections during the rush hour periods. This is very likely to cause congestion in the road network. This paper has demonstrated that genetic algorithms area feasible approach to the route assignment problem in road networks. This optimization is performed with a global view of a microscopic traffic model.[1]

3.2 Order Assignment and Routing for Online Food Delivery: Two Meta-Heuristic Methods

There is a critical need to improve service level (in terms of delivery time) with less delivery cost. For this end, the key problems include the assignment of the orders to the delivery staff and routing of the orders for each staff. The hierarchical method reduces the total traveling distance by at least 11 percent for the instances, which implies a reduced customer waiting time and delivery cost.[4]

3.3 Multi-Objective Optimization for Reducing Delays and Congestion in Air Traffic Management

Nowadays, with the increasing traffic density of the European airspace, air traffic management includes the planning and monitoring way to facilitate the work of the air traffic controllers. This article presents the use of an evolutionary multi-objective algorithm for optimizing a schedule on the time of overflight of the way-points. An evaluation function takes around 3 seconds with the characteristic function method. At this point of the research, we know that the mean fitness of the population increases along the generations.[2]

3.4 An Algorithmic Approach for Environmentally-Friendly Traffic Control in Smart Cities

With the constant development and urbanization of human society in recent years, the density of urban residential areas is increasing all over the world. To serve these increasingly high-density urban populations, governments everywhere seek sustainable development approaches [13] that balance commercial and personal needs with environmental protection. Discussed that a fully central optimum solution is likely to be NP-hard and showed our fully distributed algorithm has a bounded performance.[3]

4. Closest Adjacent Vertex Route Carpooling

In what follows we explain the data structure and algorithms for this solution.

4.1 Data Structure

This algorithm uses an implementation of various data structures needed to get a reasonable solution for both the car reductions in our streets and users time to get to their job. This Data structures are:

1) HashMap of Priority Queues (Main Data Structure): A HashMap of Priority Queues, where each priority queue saves instances of Edges objects will represent a matrix where each key represents the source vertex, the Edges within the priority queue contains the respective adjacent node, the heuristic function (edge cost plus cost to the target location).

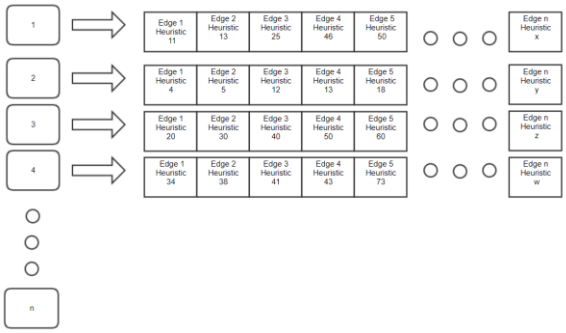

Figure 1: Matrix representation with HashMap pf Priority Queues..

2) Priority Queue of Vertex:
This priority queue contains all the vertex of the graph. It organizes the vertex within it on descending order. It orders it elements by the respective distance from each vertex to the target.

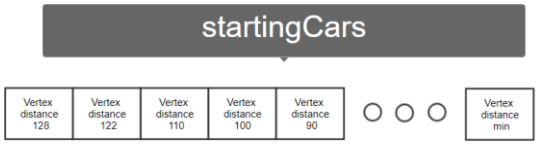

Figure 2: Priority Queue with the furthest Vertex.

3) HashMap of Vertex:
This HashMap contains instances of the class Vertex, which contains all the information of each vertex (x coordinate, y coordinate, name and the distance to the company).

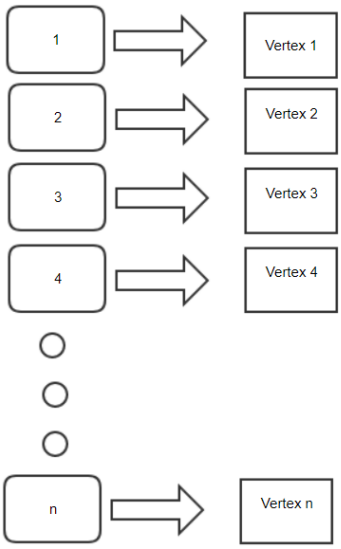

Figure 3: HashMap that contains each Vertex with its corresponding id as its key.

4.2 Operations of the data structure

1) HashMap of Priority Queues:
   1)Insertion
   2)Search.
   3)Get-Remove(poll)
2) Priority Queue of Vertex:
   1)Insert
   2)Get-Remove(poll)
3) HashMap of Vertex:
   1)Search
   2)Get
   3)Insert

4.3 Design criteria of the data structure

The objective to implement this type of data structure is its efficiency when organizing and searching for data, in exchange of memory usage, which will increase by the amount of data structures used. This data structure will increase our file reading time, this caused by the insertion and sorting of the data structures.

Priority Queues will let us organize and search data in an efficient way, being the first element of it, the element that we need, making the sear O(1) in the best case and O(n) in its worst.

HashMaps let us do the needed search in O(1).

4.4 Complexity analysis

Method	Complexity
Insertion(P.Q)	O(logn)
Search	O(1)
Poll(get-remove)	O(logn)
Insertion(H.M)	O(1)

Table 1: Table to report complexity analysis

4.5 Execution time

Measure (I) execution time and (II) memory used by the operations of the data structure, for the data set found in the .ZIP file.

Measure the execution time and memory used 100 for each data set and for each operation of the data structure. Report the average values.

Operation	Time Taken
Creation	72ms
Analysis	5ms
Printing	48ms

Table 2: Execution time of the operations of the data structure for each data set.

4.6 Memory used

Report the memory used for each data set

	Datset U=11	Dataset U=205	
Memory	6MB	25MB	

Table 3: Memory used for each operation of the data structure and for each data set data sets.

4.7 Algorithm

Design one algorithm to solve the problem and make a figure. Do not use figures from the Internet, make your own.

In order to solve the problem, we implement and algorithm which taking as parameters the data structures previously discussed, follows the next steps:

1) Takes the first Vertex (Car) in the startingCars Priority Queue, which has the Vertex in descending order from furthest to closest, using the distance between the target vertex with each vertex.
2) Next, The Vertex is marked as picked up.
3) Then, The Vertex id is taken and using it as a key for the Hash Map of Priority Queue, it gets the Priority Queue, which contains the edges with its corresponding adjacent vertex in ascending order.
4) After that, the first element (Edges object) of the Priority Queue is taken if it's heuristic value (cost to get to it plus the cost of the edges vertex to the target) does not exceeds the vertex p and that of its predecessors (if has)
5) Then, using its vertex id, it sets the vertex as picked up in the Hash Map of Vertex.
6) Consequentially, it uses the vertex id (taken from the Edges object) and repeats the process described in steps 3, 4, 5 until the total capacity is reached or there is no vertex who's addition does not exceeds the p, if so, a route is created.
7) Finally, all the steps are repeated until the startingCars Priority Queue has no elements, meaning that all routes had been set up.

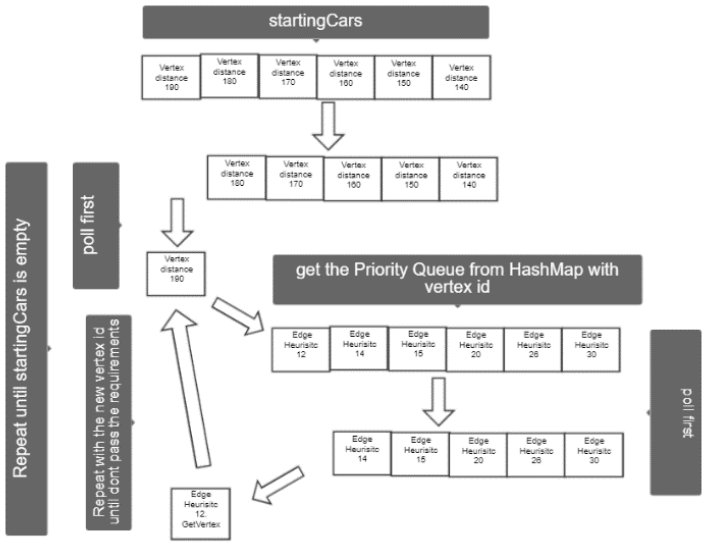

Figure 4: Step by step explanation of the algorithm procedures.

4.8 Complexity analysis of the algorithm

Calculate the complexity of the algorithm for the worst case, best case and average case.

Case	Complexity
Best	O(n log n)
Average	O(n log n)
Worst	O(n^2 log n)

Subproblems	Complexity
Poll from startingCars	O(log n)
Get the closest vertex towards the target	O(log n)
Get the route	O(n b log n)
Total complexity	O(n b log n)

Table 4. Complexity of each subproblem that is part of the algorithm. Let n be the number of vertexes, b the number of edges it visits to make the route.

4.9 Design criteria of the algorithm

This algorithm was the result of a deep research in needed data structures methods to optimize the time complexity of the algorithm in exchange of memory usage and let the users have a logical answer as fast as possible.

The average solution for this problem will take O(n^2) complexity, which is not optimal, for that reason in exchange of reading time, all the data structures where organized in the file reading period to ensure a low time needed to make the analysis and get the answer.

4.10 Execution times

Measure (I) execution time and (II) memory used by the operations of the data structure, for the data set found in the .ZIP file.

Measure the execution time and memory used 100 for each data set and for each operation of the data structure. Report the average values.

	U 11 p1.1	U 11 p1.2	U 205 p1.1	U 205 P1.2
Best	0ms	2ms	4ms	0 ms
Average	0ms	3ms	6ms	1ms
Worst	0ms	4ms	8ms	2ms

Table 5. Execution time of the operations of the data structure for each data set.

4.11 Memory consumption

Report the memory used for each data set

	Dataset U=11 p1.1	***Dataset U=205 p1.2***	***Dataset U=205 p1.1***
Memory consumption	10 MB	30 MB	25 MB

Table 6. Memory consumption with different datasets

5. CONCLUSIONS

To write the conclusions, proceed in the following way. 1. Write a paragraph with a summary, the most important issued of the report. 2. In another paragraph, explain the most important results that you obtained with the last data structure you designed. 3. Compare your first solution with the last solution. 4. At last, explain future work, a future continuation of this project. You can also mention in the conclusions, technical problems that you had during the development of the data structure and its implementation and you solved them.

Implementing algorithms that helps us solve our daily problems is an optimal decision in order to let people focus in different daily problems and reduce their stress, furthermore, as global warming problems and challenges rice, different approaches to help reduce the pollution as much as possible are really needed but solutions that help make peoples life's more comfortable are needed as well, here is where digital programs come to action as a powerful tool, and as demonstrated in this project, they can helps us solve both problems at once, probably not getting the really

optimal solution but getting a logical and good solution in exchange of answer time.

Through the project we developed various approaches to the solution, some of them proving optimal but not effective, as they take more time to give and answer, that was the case of a simple matrix where each row represents a vertex and the columns the corresponding adjacent nodes, which took O(n^2) time complexity, but gave the best solution.

In order to solve the O(n^2) time complexity problem the development of a preordered matrix with constant search methods was needed and from both options that we had (Binary Search trees and Priority Queues), Priority Queues proved to be the best option, but we the issue of not giving the optimal solution but a logical and good one in good time.

In conclusion, sometimes its better to get a good and fast solution that the very best one, if takes too long, and that with this non optimal solutions of some societies problems, we can make a huge reduction in the pollution generation in our societies, proved with this project, where the average car reduction was of the 50%.

5.1 Future work

Futures works for this project are many, as it opens the possibilities to create an app which will be more user friendly and contain more characteristics of a social network between users. Some of this future works are:

1)Development of an interactive and real life responding algorithm.

2)Research of a more optimal way to get into the solution with out damaging the time complexity.

3)Develop of a GUI for users visualization of the routed.

ACKNOWLEDGEMENTS

We want to especially thank our fellow classmates and friends Stiven Agudelo and his team, which whom we discussed and shared ideas of how to solve the problem and the effectiveness of the data structures proposed, the development of this project was much easier by their help.

REFERENCES

Reference sourced using ACM reference format. Read ACM guidelines in http://bit.ly/2pZnE5g

As an example, consider this two references:

1. International Scientific Conference on Mobility and Transport Transforming Urban Mobility, mobil.TUM 2016, 6-7 June 2016, Munich, Germany A Matching Algorithm for Dynamic Ridesharing Maximilian Schreieck , Hazem Safetli, Sajjad Ali Siddiqui, Christoph Pflügler, Manuel Wiesche, Helmut Krcmar Chair for Information Systems, Technical University of Munich, Boltzmannstraße 3, 85748 Garching, Germany

2. Emilio Ferrari, Ricardo Manzini, Arrigo Pareschi, Alberto Regattieri .The Car Pooling Problem: Heuristic Algorithms Based on Savings Functions, Jun 2003.