JK Shared Vehicle Strategic System (JK-SVSS algorithm)

Kevin Alexander Herrera
Universidad EAFIT
Colombia
kaherrerag@eafit.edu.co

Jose Joab Romero
Universidad EAFIT
Colombia
jjromeroh@eafit.edu.co

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

ABSTRACT

The high traffic brings great problems to the quality of life of people because it affects various factors such as air quality and decreased productivity and mobility of the population; The strategy "Carpooling" is significantly functional and effective on environments such as companies and universities, the previous strategy is that people share their vehicle thus reducing the number of private vehicles that circulate daily, for this an algorithm is implemented that determines a possible minimum route in which a person can pick up others in a way that prevents them from going in their own car individually, with the condition that time can not increase more than a certain percentage or quantity compared to the original and analyzing the mobility restrictions. In this way you can see a considerable reduction in the number of vehicles (205 vehicles can be reduced around 45 with a time of 1.5 higher, depending on the situation) in such a way that the previous problems are reduced, thus streamlining the processes.

KEYWORDS

Graph, Pathfinding, Table Hash, Carpooling, Complexity, Memory, Traffic, Data structure, Algorithm, vehicle, Minimum path.

ACM CLASSIFICATION KEYWORDS

Theory of computation → Design and analysis of algorithms and problem complexity→ Data structures and algorithms → Graph algorithms analysis → Shortest paths.

1. INTRODUCTION

The high levels of traffic in cities are one of the main causes of problems in various aspects in relation to the quality of life of people, this is because it drastically affects the environment, productivity and mobility within cities.

The following report shows how, based on a strategy called "Carpooling", the number of vehicles that circulate in the city can be reduced considerably, the strategy is for people to use their vehicle in a shared way with people who are heading to the same destination, thus avoiding the use of their individual vehicles, however, the following conditions must be met: the travel time must not exceed a certain percentage with respect to the original trip and respect the mobility restrictions of each user with their respective vehicle ; for this it is necessary to implement an algorithm that indicates which people can pick up others in the most efficient way regarding mobility-time.

2. PROBLEM

The problem faced by the strategy "shared vehicle" is the design and implementation of an algorithm that manages to find efficiently and quickly the group of people who can go together to the same destination without significantly increasing the travel time ; It is done with the objective of establishing methods that can be applied in companies and other entities to reduce the use of private vehicles and therefore reduce traffic.

3. RELATED WORK

3.1 Traveler agent problem:

Given a list of cities and the distances between each pair of cities, which is the shortest possible route that each city visits exactly once and at the end returns to the origin city?

Solution: the linear and integer programming algorithm; branch and dimension.

The first way to attack the traveler's problem is to use a linear and integer programming algorithm. This is an exact algorithm but with a very high cost of time. In many cases, the use of this program is unfeasible because the cost of time is very high even though it always arrives to obtain the solution of the problem.

Only the TSP should be expressed as a whole linear programming problem and programmed.

The formula is the following:

$$\text{minimizar} \quad \sum_{i,j=0}^{n} c_{ij}x_{ij}$$

$$\begin{aligned}
\text{sujeto a} \quad (1) \quad & \sum_{i=0}^{n} x_{ij} = 1 && j = 0, \ldots, n \\
(2) \quad & \sum_{j=0}^{n} x_{ij} = 1 && i = 0, \ldots, n \\
(3) \quad & u_i - u_j + nx_{ij} \leq n - 1 && 1 \leq i \neq j \leq n \\
& 0 \leq x_{ij} \leq 1, && x_{ij} \in Z, \forall i, j
\end{aligned}$$

Figure 1: Traveler agent equation.

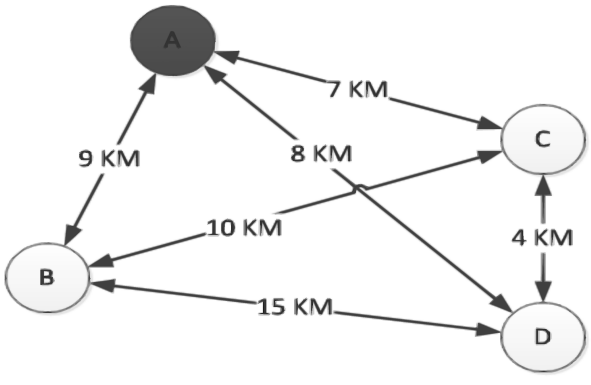


Figure 2: Cities and their distances as a graph.

3.2 Persecution in video games:

In video games like Pac Man, a situation is presented where the main character, the player, is chased by enemies who try to catch him. The problem is in making these enemies locate and pursue the player.

For this, the Pathfinding A * algorithm is used. This algorithm is an improvement of the Dijkstra algorithm and works considering the scenario as a grid with many cells: we have the starting point within one and our objective point in another. The algorithm consists in calculating a "general cost" for each cell that surrounds the cell that is being analyzed, the cell with the lowest "general cost" is chosen and then it will become the analysis cell, with the parent cell as the cell we just left.

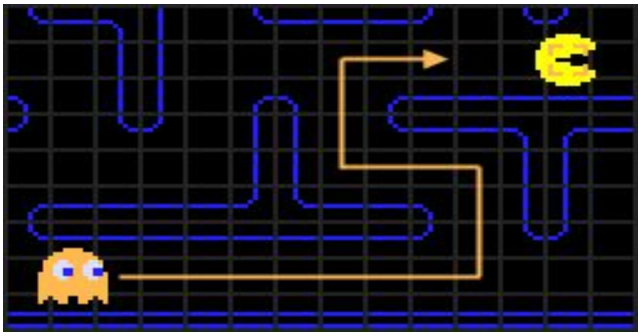


Figure 3: Pathfinding in the Pac Man game.

3.3 Intelligent guided parking system:

In a parking lot you want to establish a system that allows you to give the closest place to your customers so they can park your vehicle, so that the service is more efficient and faster; for this you need a machine that delivers a ticket to the client indicating the corresponding location. Therefore, the Dijkstra algorithm has been used to take into account the available and occupied places and calculate which is the closest place to park.

Figure 4: Intelligent parking system.

3.4 Road planning and control assistance tracking global trajectory to autonomous vehicle:

Autonomous vehicles have a built-in GPS system that allows them to locate themselves, a dilemma that arises from the origins of this idea is to indicate to the vehicle the shortest path to reach its destination, for which various road search algorithms have been implemented shorts and combinations of them with the aim of solving this problem; currently there are several versions of "pathfinding" algorithms but they are not yet optimal enough.

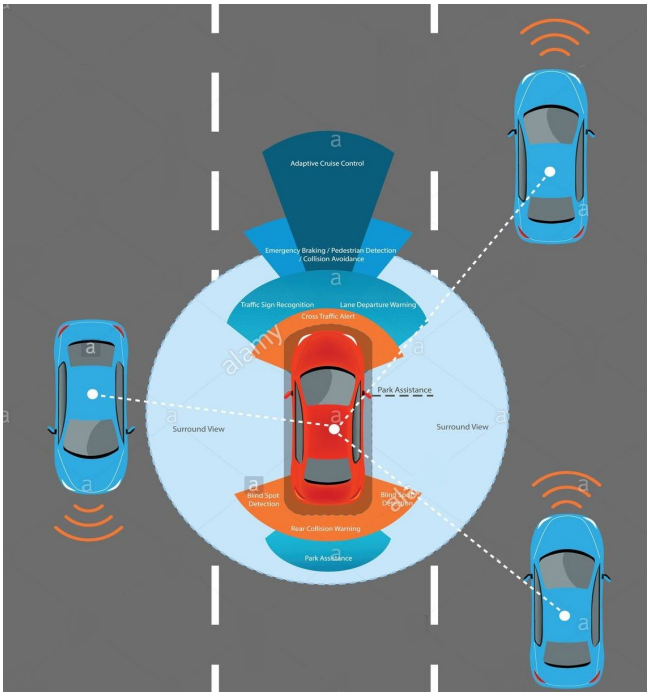
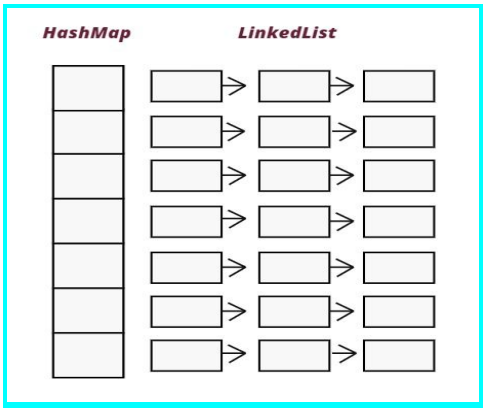


Figure 5: Autonomous Car GPS System

4. HASH TABLE - LINKEDLIST

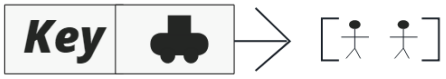
4.1 Data Structure:

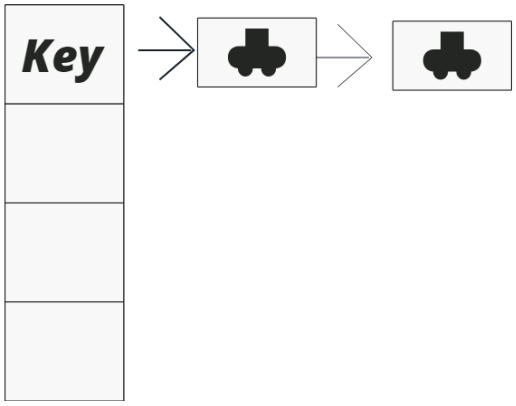


Graph 1: Graph of the data structure.

4.2 Operations of the data structure:

Graph 2: Operation to add data.





Gráfica 3: Estructura para almacenar carros que contienen personas.

4.3 Design criteria of the data structure:

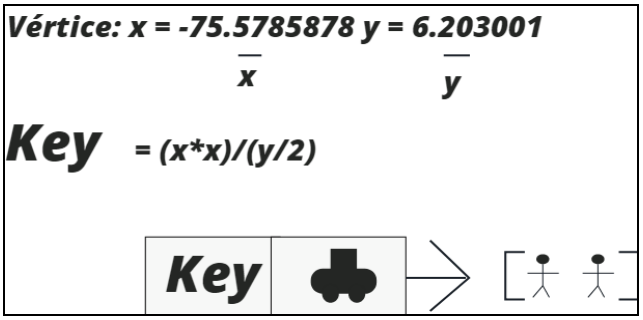
A Hash table linked to a linked list is implemented due to the efficiency of the operations of the same, granting a great facility and performance (memory and time use) that allows the versatility within the algorithm when adding or consulting elements , in this way complexity and execution times are reduced.

4.4 Complexity analysis:

Method	Complexity (n = Vertices)
LeerArchivo	O(n^2)
agrupadorDeCarros	O(n)
EscribirArchivo	O(n)

Table 1: Complexity report of the methods.

4.5 Algorithm:



Graph 4: Method for grouping people in cars.

4.6 Complexity analysis of the algorithm:

Subproblem	Complexity
Leer Archivo y extraer los vértices	O(n^2)
separar los vértices en el hashmap y asignarlo a un carro	O(n)
Escribir el archivo	O(n)
Complejidad Total	O(n^2)

Table 2: complexity of each of the sub-problems that make up the algorithm. Let n be the number of vertices.

4.7 Design criteria of the algorithm:

Before the implementation of the algorithm, it was analyzed that there is a direct relationship between the X and Y coordinates of a vertex and its distance-time to other vertices, which allows establishing a series of conditions that speed up the processing of information, which are the basis for the algorithm. In this way, using data structures that are coupled to the architecture it is possible to implement a design that allows the algorithm to be more optimal, faster and more efficient (memory complexity).

4.8 Execution time:

	Runtime (ms)
Best case	192 ms
Average case	197 ms
Worst case	256 ms

Table 3: Execution times of the algorithm.

4.9 Memory consumption:

	Data set 1	*Data set 2*
Memory consumption:	0.152 MB	0.982 MB

Table 4: Memory consumption of the algorithm with different data sets.

4.10 Analysis of the results:

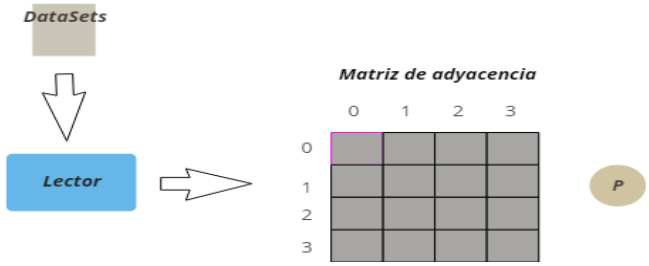
	HashMap	**LinkedList**
Memory	7.5 MB	3.47 MB
search (ms)	145 ms	113 ms
grouping (ms)	190 ms	198 ms

5. STRATEGY CARPOOLING SYSTEM

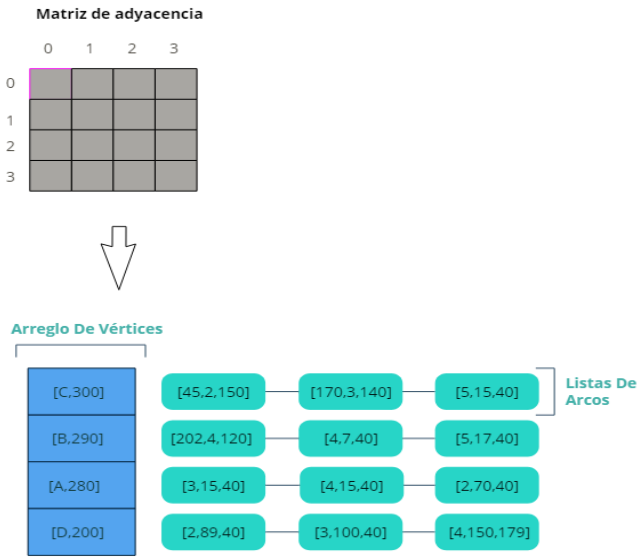
5.1 Data Structure:

Graph 5: Graph of the Data structure.

5.2 Operations of the data structure:



Graph 6: Graph of the reading method that extracts the P and the vertices of the dataset and passes them to an adjacency matrix



Graph 7: Graph of the Method Create vector and ordered linked lists which are extracted from the adjacency matrix

5.3 Design criteria of the data structure:

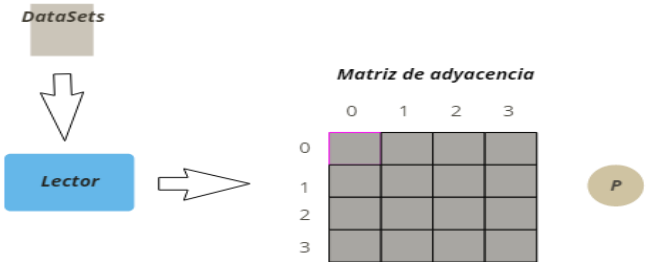
4 data structures that work together are implemented, such structures are: a hash table, a linked list, and two fixes; Each structure has a particular objective and were chosen for efficiency and performance in the operations in which they were required, so that it takes advantage of the potential of each structure in a meaningful way, thus granting the algorithm versatility and acceptable complexity.

5.4 Complexity analysis:

Method or function	**Complexity (n = # vertices)**
LeerArchivo	O(n^2)
AsignarObjetoEnArreglo	O(n)
AsignarEnObjeto	O(n^2)
OrdenarArregloVertices	O(n * log(n))
OrdenarLinkedList	O(n * log(n))
DireccionEnHashMap	O(n)
AsignarCoches	O(n)
Escribir	O(1)

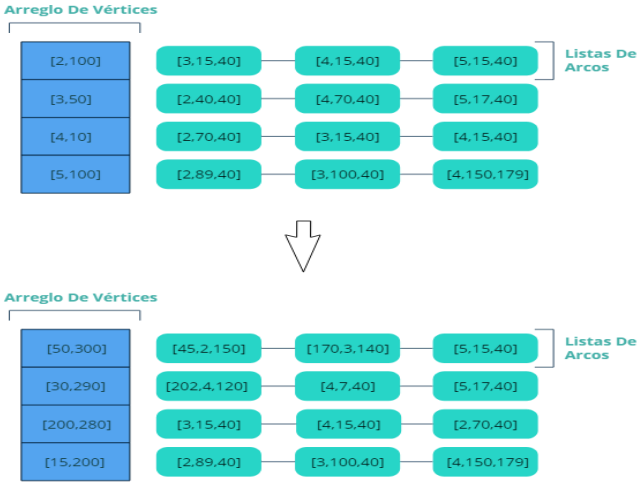
Table 6: Complexity report of the methods.

5.5 Algorithm:



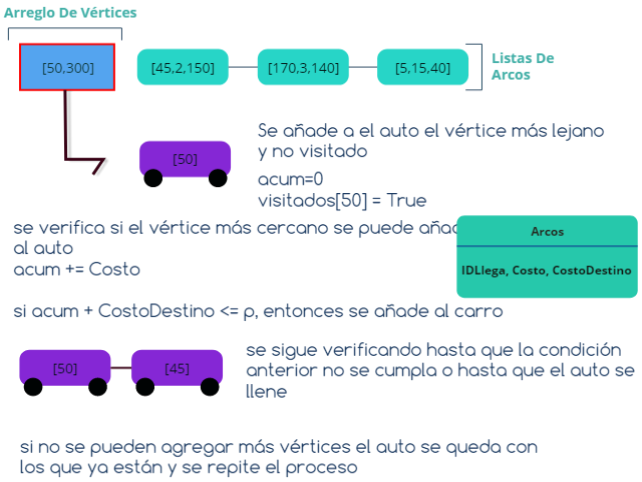
Graph 8: the algorithm uses the reader method to extract the P and the vertices of the dataset to be transferred to an adjacency matrix.

Graph 9: Representation of vertices and arcs



Graph 10: From the adjacency matrix an array of vertices is created, which will also contain a list of each of its arcs, after which the array of vertices is ordered from the farthest to the closest to the destination point and the arcs are also ordered from the closest to the farthest

Graph 11: A hashmap is created that will help us find the position of a vertex in the array of vertices (now ordered) and also creates an array of visited to be able to mark those vertices that have already been placed in a vehicle



Graph 12: Explanation of the method to assign the vertices to the vehicles.

5.6 Complexity analysis of the algorithm:

Sub Problem	**Complexity (n = # vertices)**
Leer archivo y extraer los vértices	O(n^2)
Crear vector y listas enlazadas ordenadas	O(n^2)
Formación de rutas y escritura del archivo respuesta	O(n)
Total Complexity	**O(n^2)**

Table 7: complexity of each of the sub-problems that make up the algorithm. Let n be the number of vertices.

5.7 Design criteria of the algorithm:

The information was analyzed in search of a heuristic that would allow to design an algorithm that from it that will be able to solve the problem, in this way it was observed that it is possible from time to establish a logic that allows to identify quickly and efficient an approximate solution to the optimal one by means of the search of the furthest point and verifying if this one can happen through another point fulfilling the conditions and of this way to continue until finalizing the points.

5.8 Execution time:

The following table shows the time it takes the algorithm to solve the problem with different data sets where U is the number of vertices and P the extra percentage of time.

Execution Time (ms)	*U=4 P=1.7*	*U=11 P=1.1*	*U=205 P=1.1*	*U =205 P=1.3*
Best Case	160 ms	175 ms	352 ms	370 ms
Average case	162 ms	182 ms	358 ms	377 ms
Worst case	165 ms	189 ms	365 ms	384 ms

Table 8: Algorithm execution times for different data sets.

5.9 Memory consumption:

	U=4 P=1.7	*U=11 P=1.3*	*U=205 P=1.1*
Memory Consumption (MB)	9.45 MB	14.1 MB	42 MB

Table 9: Memory consumption of the algorithm with different data sets.

5.10 Analysis of the results:

	Memory (MB)	**Time (ms)**	**Solution (# Cars)**
U=205 P=1.1	42 MB	358 ms	78
U=205 P=1.3	40 MB	377 ms	50
U=11 P=1.1	15 MB	182 ms	5

Table 10: Analysis of the results obtained with the implementation of the algorithm.

6. CONCLUSIONS

Throughout the development of the project various problems were presented and different aspects were analyzed, in this way the project advanced in several stages and went through different solutions, in such a way that it was identified that working with the times was more efficient than analyzing the coordinates and work with them; the heuristic was significantly improved, thus allowing the development of a more efficient and versatile algorithm, for which it was necessary to link four different data structures, which is known to be complicated, but the same working in unison give the algorithm characteristics as greater speed and less complexity. Finally, the following is highlighted and asked to take into account: the algorithm does not grant the optimal solution but a very close one.

6.1 Future work:

The improvement in the implementation of the algorithm presented in this report is planned for the future, in search of improvements that allow us to get even closer to the optimal solution, reduce time, complexity and establish aspects that allow the algorithm to perform in more real environments such as cities in such a way that they take into account aspects such as accidents on the roads, or routes in poor condition, that is, phenomena that cause an increase in time on a regular route.

ACKNOWLEDGEMENTS

- Special thanks to the government of Colombia and Icetex for allowing access to education, their solidarity and commitment, among others.
- Thanks to Edison Valencia, professor of the Systems Engineering Department of the EAFIT University for their collaboration and comments that allowed the improvement of the project
- We thank Jhonatan Sebastian Acevedo and Manuel Alejandro Gutierrez Mejia for their contribution and help towards the project.

REFERENCES

1. Es.wikipedia.org. (2019). *Problema del viajante*. [online] Available at: https://es.wikipedia.org/wiki/Problema_del_viajante [Accessed 3 Mar. 2019].
2. Eio.usc.es. (2019). [online] Available at: http://eio.usc.es/pub/mte/descargas/ProyectosFinMaster/Proyecto_774.pdf [Accessed 3 Mar. 2019].
3. Rodríguez Puente, R. and Lazo Cortés, M. (2019). *Shortest path search: current applications*. [online] Availableat:https://www.researchgate.net/profile/Rafael_Rodriguez_Puente/publication/295861940_Busqueda_de_caminos_minimos_aplicaciones_actuales_Shortest_path_search_current_applications/links/56cf1f5a08ae4d8d649f9b09.pdf [Accessed 4 Mar. 2019