ISSN 1870-4069

Transport Routes Optimization in Martinez de la Torre, Veracruz City, Using Dijkstra's Algorithm with an Additional Parameter

Hugo Lucas-Alvarado, Eddy Sánchez-DelaCruz, R. R. Biswal

Technological Institute of Misantla, Veracruz, Mexico

esanchezd@itsm.edu.mx**

Abstract. A solution to the shortest path problem in Martinez de la Torre, a city in the state of Veracruz, Mexico, is presented. Dijkstra's algorithm was applied to find the shortest route and additional parameter that is used to choose the optimal route was also added. The shortest path problem is a specific case of optimization problems and can be addressed using graph theory. From a computational point of view, graphs are data structures that can be treated as arrays, tables or lists. On the other hand, several organizations in Martinez de la Torre face the problem of finding the optimal route to reduce time and cost, and to offer a better service to its customers. This article addresses the problem of finding an optimal route and implementing the solution to a small portion (area of interest) of Martinez de la Torre City, to test the algorithm and analyze the results, and ultimately leading to a detailed analysis that covers the whole city.

Keywords: Route optimization, Dijkstra's algorithm, shortest route problem, graph, adjacency matrix.

1 Introduction

Optimization problems are presented in diverse industries such as banks, education, freight transport, among others. Many important optimization problems are best analyzed using graphical (graphs) or network representation. Some specific network models are: shortest path problem, maximum-flow problems, minimum spanning tree problems, etc. [6].

Nowadays, systems that provide the best route in a specific geographic area are useful. In many places there are various needs that require these types of systems, to mention a few: distribution networks, road programs, telecommunications networks, electricity networks, among others. Martinez de la Torre, Veracruz is no exception, therefore, in this research we intend to construct a graph representing that city and implementing the Dijkstra's algorithm by adding another parameter to determine the shortest path. For purposes of testing,

** Corresponding author is Eddy Sánchez-DelaCruz.

a zone of interest has been segmented. The graph is constructed taking into consideration, Melchor Ocampo (a district belonging to Martinez de la Torre). It is important to mention that the implementation of Dijkstra's algorithm in this project is useful to find the shortest route.

However, not only is distance important, there are also other parameters to consider when choosing the best route in order to reduce cost and time. It is thus necessary to add them to manipulate them by Dijkstra's algorithm. These parameters are: the sense of streets, severe flooding during the rainy season, restricted access to heavy transport, vehicular flow, breadth of streets, sections in repair, traffic lights, local traffic regulations, insecurity level, among others. The optimization of the routes will allow, various sectors in the Martinez de la Torre City, to be benefited, since they would have at their disposal a useful tool to find the best route.

The present article is structured as follows: section 1 deals with introduction, the problem is mentioned in section 2, Dijkstra's algorithm is described in section 3, section 4 mentions previous work related to Dijkstra' algorithm and optimization of routes, section 5 describes the tools used as well as the methodology to be followed, section 6 shows the experiments performed and the results obtained thereof. Finally, the conclusion and future work are presented in section 7.

2 Problem Formulation

Formally, the route optimization problem can be represented by graphs. Here we mention some important concepts related to the problem that is approached in this investigation.

- **Simple graph.** A simple graph $G = (V, E)$ consists of a set V of vertex (or nodes) and a set E of edges, each edge is a set of two vertices. A simple graph edge has the form $\{v_i, v_j\}$ and for a edge such as this, $\{v_i, v_j\} = \{v_j, v_i\}$. $|V|$ denotes the number of vertices and $|E|$ denotes the number of edges.
- **Directed graphs o digraph.** It consists of a non-empty set V of vertices and a set E of edges (also called arcs). In a digraph, each edge is of the form $\{v_i, v_j\}$ and in this case, $\{v_i, v_j\} \neq \{v_j, v_i\}$.
  It is said that an edge e in a graph that is associated with the pair of vertices u and w is incident on u and w, and it is said that u and w are incidents over e and are adjacent vertices.
- **Weighted graph.** A graph with numbers in the edges is called a weighted graph. If the edge e is labeled k, it is said that the weight of the edge e is k. In a weighted graph, the length of the route is the sum of the weights of the edges on the route.
- **Route (or path).** Let v_0 and v_n vetices in a graph. A path from v_0 to v_n is an alternating sequence of $n + 1$ vertices and n edges that begins at vertex v_0 and ends at vertex v_n,
  $$(v_0, e_1, v_1, e_2, v_2, ..., v_{n-1}, e_n, v_n),$$
  where the edge e_1 is incident on the vertices v_{i-1} and v_i for $i = 1, ..., n$.

– **Graph representation.** Computationally, graphs can be represented by adjacency lists and adjacency and incidence matrices.

In Martinez de la Torre, municipality of Veracruz, there are often situations such as: excessive expenses in the distribution of products by citrus packers, inability to meet the demand for LP Gas distribution, inconsistencies in fuel consumption of distribution vehicles, inefficient taxi service. Among other road problems, these cases have a common denominator: the problem of finding the best route. By solving this problem, it is possible to have more precise control of both time and cost. On the other hand, there are advanced cartographic tools such as Google Maps, which provide relevant information that can be used to build a system to determine the best route. In this research we intend to construct a graph representing the Martinez de la Torre City and implement the Dijkstra's algorithm by adding other parameters to determine the shortest route.

3 Dijkstra's Algorithm

In weighted graphs it is often desired to find the shortest path between two given vertices, that is, the sequence of vertices that must be visited to arrive from a source node to a destination node, whose length is minimal. Dijkstra's algorithm solves this problem efficiently. For the application of Dijkstra's algorithm it is assumed that the weights are positive numbers and that we want to find the shortest path from vertex a to vertex z. Let $L(v)$ be the vertex label v. At any point, some vertices have temporary tags and the rest are permanent tags. Let T be the set of vertices that have temporary tags. If $L(v)$ is the permanent label of vertex v, then $L(v)$ is the length of a shortest path from a to v. At the beginning, all vertices have temporary tags. Each iteration of the algorithm changes the state of a label from temporary to permanent; the algorithm ends when z receives a permanent label [2].

Algorithm 1 finds the length of a shortest path from vertex a to vertex z in a weighted connected graph. The weight of the edge (i, j) is $w(i, j) > 0$, and the vertex label is $L(x)$. When finished, $L(z)$ is the length of the shortest path from a to z.

The above algorithm finds the length of the shortest path from a to z. In most applications, you also want to identify the shortest route. A slight modification to achieve this is to add $L(v)$, the name of the previous vertex (in the path), that is, the label of a vertex is "$L(v), v_a$", where $L(v)$ is the minimum length from a to v, and v_a is the previous vertex in the path. This ensures that on completion, z (destination node) has a label indicating the minimum length from a and it is also possible to move backwards on the labels to find the shortest route. Subsequently, we refer to this modification as the *extended version of the Dijkstra's algorithm*.

In the algorithm, line 5 runs $O(n)$ times. Within the "while" cycle, line 9 takes a time $O(n)$. The body of the cycle "for" (line 12) takes a time $O(n)$.

Algorithm 1 Dijkstra's algorithm

Require: A weighted connected graph in which all weights are positive; vertices a to z

Ensure: $L(z)$, the length of the shortest route from to a to z

```
1: dijkstra(w,a,z,L)
2: {
3:    L(a) = 0
4:    for all vertices x ≠ a do
5:       L(x) = ∞
6:    end for
7:    T = Set of all vertices  //T is the set of all vertices whose shortest distances
      from a have not been found
8:    while z ∈ T do
9:       select v ∈ T with L(v) minimum
10:      T = T − v
11:      for  each x ∈ T adjacent to v do
12:         L(x) = min{L(x), L(v), +w(v, x)}
13:      end for
14:   end while
15: }
```

Since lines 9 and 12 are nested in the "while" cycle that takes a time $O(n)$, the total time for lines 9 and 12 is $O(n^2)$. Then Dijkstra's algorithm runs at a time $O(n^2)$.

For manipulation of an additional parameter (slow traffic), a condition indicating the availability of the edge according to the value of said parameter has been introduced, prior to line 9 of the algorithm. If the edge is not available, then the nature of the algorithm is to continue looking for a vertex with $L(v)$ minimum. In the worst case, if that vertex is not found, that is, if there is no alternate path, the algorithm ignores that parameter, so that it returns a shorter path.

4 Previous Works

The following information is about research that addresses the routes optimization problem, as well as the applications and improvements made to Dijkstra's algorithm:

Edsger W. Dijkstra, solved two problems related to graphs, one of them is to find the total minimum length path between two given nodes P and Q. The solution provided is widely used and is known as Dijkstra's algorithm [1].

Wang Shu-Xi, analyzed the Dijkstra's algorithm and identified some aspects that were not considered in the original problem. Based on this analysis, he proposed improvements to the algorithm and validated it by experiments. The algorithm improvements allow: to control the output mechanism and avoid entering an infinite cycle in case of vertices without connection, save information

about all adjacent vertices and not only the previous vertex in the route, and two (or more) vertices are labeled simultaneously to find the shortest path [5].

Strehler et al., identified, among other things, the need to determine the shortest route and travel planning for use in electric and hybrid vehicles, since it is necessary to regulate speed and range, as well as choose stops for recharging the vehicle battery in a strategic way, the above because the time required for the recharge is greater than the vehicles that use fossil fuels. In the study mentioned, they developed an appropriate model to find the shortest route for such vehicles based on Dijkstra's algorithm and considering that a route may contain cycles leading to a recharge station. The study also considered cases where a charging station could be occupied by another vehicle [4].

Galán-García et al. state that although it is possible to calculate the shortest route using the Dijkstra's algorithm. However, in real life situations this is not always the chosen one. In order to obtain more realistic simulations of traffic flow in intelligent cities, the authors present PEDA (Probabilistic Extension of Dijkstra's algorithm) as an extension to the Dijkstra's algorithm in which they introduce probabilistic changes in the weight of the edges and also in decisions while choosing the shortest path. As an application to the example, they present ATISMART*, which is a model that provides simulations of traffic flow in smart cities using PEDA. When PEDA is applied to the traffic flow, more realistic simulations are obtained, in contrast to the ATISMART model, which uses Dijkstra's algorithm (without probabilistic extension) and therefore saturates those tracks that belong to the shortest route, leaving some map areas almost empty [3].

5 Materials and Methods

5.1 Materials

To obtain the cartographic information, Google Maps, a web application server belonging to Alphabet Inc was used. It offers scrollable map images as well as satellite photographs of the world. In addition, it allows to obtain information about a specific place, to measure the distance between two points, to consult information on vehicular traffic, public transport, routes, among others. For the present investigation, this tool was used to obtain information about Martinez de la Torre, a Veracruz City, from which the graph was constructed.

Through Google Maps, we obtained the Map of Martinez de la Torre City. According to the INEGI Intercensal Survey of 2015, the municipality of Martinez de la Torre has a total population of 110 415 habitants (1.4 % of the state population), a population density of 277.0 hab/km^2 and an area of 815.13 km^2 (0.6 % of the state territory). These data help us to understand that there is a considerable population in the city, which leads to road congestion. Figure 1 a) shows the location of Martinez de la Torre in the state of Veracruz and Fig. 1 b) shows the map of the city. To perform the initial tests, the area of interest shown in Fig. 2 has been selected, in the same image the graph generated is shown.

In the graph, the nodes represent the intersections or ends of streets (numbered from 0 to 37) and the weights of the edges represent the distance (in meters) between the nodes.

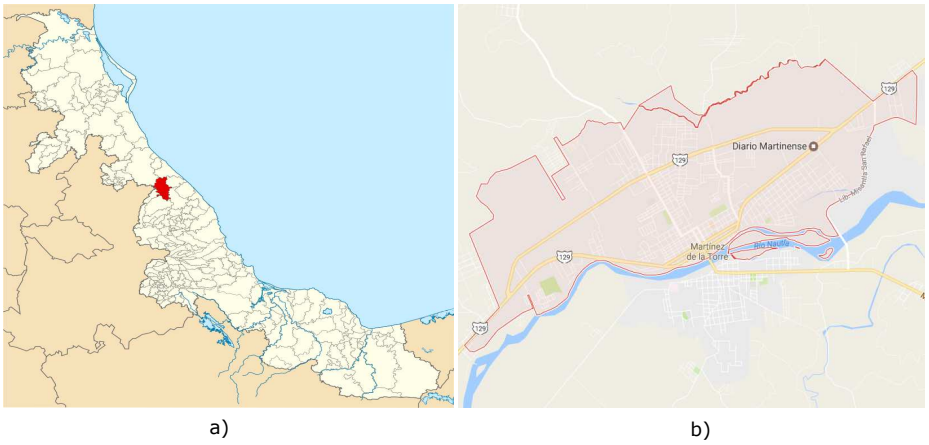

Fig. 1. (a) Location of Martinez de la Torre in the state of Veracruz and (b) Map of the Martinez de la Torre City.

The algorithm is implemented on the information of the graph of the selected region, in addition the algorithm was extended, to consider an additional parameter, as described in the last paragraph of section 3: *Dijkstra's algorithm*. For the implementation, NetBeans IDE 8.2 and Java language were used. The program reads a text file containing the adjacency matrix of the graph, implements the algorithm and finds the shortest path between two points. Object-oriented programming is used, so nodes (or vertices) are objects capable of storing multiple parameters.

5.2 Methodology

Figure 3 shows the flowchart of the methodology to be followed. The process consists of four phases.

As mentioned earlier, the model is intended to be developed for Martinez de la Torre City, and with the aim of having a visual panorama of the problem. In phase one the map of the city was acquired. In the second phase, an area of interest was chosen within Martinez de la Torre City. The basic data were obtained and as there were intersections (nodes) and distance between them (weights of the edges), a graphic representation was created (graph) and finally represented by an adjacency matrix. In phase three the programming of the algorithm in Java was carried out, the *extended version of the Dijkstra's algorithm* mentioned in section 3 is used, which allows to find not only the distance

Fig. 2. Area of interest: Melchor Ocampo District.

of the shorter route (sum of weights) but also the shortest route (sequence of vertices that form the route); and secondly, the adaptations needed to handle some parameters. In this phase the tests were also performed with the original algorithm and adding parameters. In phase four, the results are obtained and analyzed, to determine the correct operation. The data is checked manually and the results are compared with the results of the Google Maps tool.

6 Experiments and Results

For the experiments carried out, a computer with the following specifications was used: Lenovo laptop, YOGA 510-14IKB model, Windows 10 Home Single Language operating system, Intel(R) Core(TM) i7-7500U CPU 2.70 GHz 2.90 GHz, 8.00 GB of RAM, 1.00 TB hard disk capacity, 64-bit operating system and x64 processor.

Strategically, routes have been chosen that could compromise the algorithm or show the management of an additional parameter, specifically those that could have more than one "shorter" route, one-way streets and streets that present road chaos in hours peak. Table 1 shows the results obtained in 8 tests. The columns *Src* and *Dst* indicate the source and destination nodes respectively, the *Dist* column contains the total distance (sum of the weights) of the shortest route and the *Path (without parameters)* column shows the vertices of that route, for tests that consider only the distance (see Fig. 2).

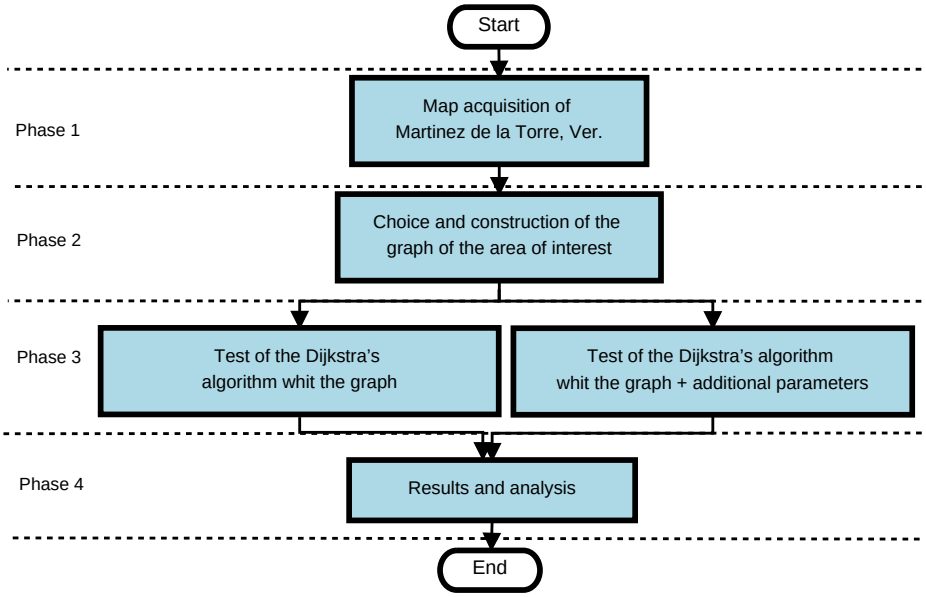

Fig. 3. Proposed methodology.

Table 1. Inputs and outputs of the algorithm in the tests performed.

No.	Src	Dst	Dist	Path (without parameters)	$Dist_p$	Path (with parameters)
1	31	37	289	31-30-33-36-37	289	31-30-33-36-37
2	0	37	665	0-4-10-16-22-26-29-32-35-36-37	850	0-1-2-6-12-25-31-30-33-36-37
3	0	4	65	0-4	311	0-1-5-4
4	0	35	448	0-4-10-16-22-26-29-32-35	879	0-1-2-6-12-25-31-30-33-36-35
5	35	0	448	35-32-29-26-22-16-10-4-0	877	35-36-37-31-25-12-6-2-1-0
6	3	35	760	3-8-14-20-25-31-30-29-32-35	761	3-8-14-20-25-31-30-33-36-35
7	27	30	312	27-26-29-30	563	27-23-17-11-12-25-31-30
8	12	25	415	12-25-20-14-13	415	12-25-20-14-13

Returning to the example of our area of interest, some edges have been marked that show road congestion in peak hours, namely: $(0, 4)$, $(4, 10)$, $(10, 16)$, $(16, 22)$, $(22, 26)$, $(26, 29)$, $(29, 32)$, $(32, 35)$. The $Dist_p$ column contains the total distance of the shortest route and the *Path (with parameters)* column shows the vertices of that route, for the tests that consider a parameter apart from the distance (see Fig. 2).

Note that the routes 2, 3, 4, 5, 6 and 7 are those that, pass through streets that have traffic congestion in the test without parameters. As can be seen, the shorter routes that evade these streets have been chosen, therefore, the expected

results have been achieved, i.e. the shortest route in a section of the city is found considering an additional parameter besides the distance.

It is important to note that additional parameters may belong to both nodes and edges, eg. road congestion and sections in repair are exclusive parameters of the edges (streets), while the presence of traffic lights and the step 1x1 belong to the nodes (intersections). The edge e can be treated as an object with n parameters (one of which is the weight), that is, $e = (p_1, p_2, ..., p_n)$ and the vertex v as an object with m parameters, that is, $v = (p_1, p_2, ..., p_m)$.

7 Conclusion and Future Work

It has been possible to construct the graph of an area of interest of Martinez de la Torre, Veracruz City. The Dijkstra's algorithm has been used to find the shortest route and also, by manipulating other parameter, it has been possible to determine the optimal route. It was possible to create the complete city graph, to determine routes considering various parameters, including travel time. In addition to a real application of the Dijkstra's algorithm, this work is a useful tool for the different organizations that face the problem of finding the optimal route.

It is intended, in a later stage, to construct the complete city graph, to implement the algorithm and to add all the parameters that are of interest for the choice of the optimal route. In addition, it will be necessary to add information from the intersections and streets, which are represented by nodes and edges respectively.

To take advantage of this research, implementing the current model to find the best route in a real application, is not ruled out, to mention a few, in local distribution networks and in the taxi service. To provide greater functionality and to facilitate the conditions to operate dynamically, it is also possible to integrate GPS technology, so that a user (a driver, for example) has a route orientation to follow and reorganize routes from the current location of the vehicle when new destinations are required.

References

1. Dijkstra, E.W.: A note on two problems in connexion with graphs. Mathematisch Centrum (1959) 4
2. Johnsonbaughc, R.: Matemáticas discretas. Pearson Educación (2005) 3
3. José L. Galán-García, Gabriel Aguilera-Venegas, M.A.G.G.: A new probabilistic extension of dijkstra's algorithm to simulate more realistic traffic flow in a smart city. ELSEVIER (2014) 5
4. Martin Strehler, Soren Merting, C.S.: Energy-efficient shortest routes for electric and hybrid vehicles. ELSEVIER (2017) 5
5. Shu-Xi, W.: The improved dijkstra's shortest path algorithm and its application. IWIEE (2012) 5
6. Winston, W.L.: Investigación de operaciones Aplicaciones y algoritmos. CENGAGE Learning (2005) 1