SAFETY-BASED PATH FINDING ALGORITHM FOR REDUCTION OF SEXUAL HARRASSMENT.

Samuel Rico
Universidad Eafit
Colombia
Sricog@eafit.edu.co

Gregorio Bermúdez
Universidad Eafit
Colombia
gbermudezo@eafit.edu.co

Andrea Serna
Universidad Eafit
Colombia
asernac1@eafit.edu.co

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

ABSTRACT

Sexual harassment is a daily concern for women in Medellín. 6 out of 10 women reported that they consider that the city is not safe for them [1]. Due to this, is important to implement alternatives, making the city a safer place to live. This project, implements Dijkstra algorithm, which provides safer routes to the women, avoiding risk and uncomfortable situations for them. This algorithm managed to calculate an optimal path regarding either harassment risk or distance as its main source of guidance. It has an average running time of 0.17 seconds and The time complexity is O(V+E log E). We can conclude that for this project to work correctly there must be a balance between distance and harassment risk, because if there is tendency towards only one of them it can be unsustainable.

Keywords

Constrainted shortest path, street sexual harassment, secure-path identification, crime prevention.

1. INTRODUCTION

According to SISC [2] in the last month (February 2022) the violence towards women has increased in Antioquia's capital. Femicide is the biggest indicator that violence is rising. There have been 5 cases in the last 19 days and that is a number which worries the authorities. Nowadays, the police is investigating, hoping to find someone they could use as guilty. But is this really the solution? Women home harassment was increasing thanks to the pandemic [3], but since the confinement is over this has move to the city. What should be worrying the government in the first place is finding a suitable way for avoiding street sexual harassment amongst citizens.

Safeness has always been an issue in Antioquia, since the 80s, people levels of concern towards crimes has been rising [4]. Even though the crime rates have decreased lately in Medellin [5], people still feel worried about getting robbed. As an attempt to solve this problematic, this project intends to create a safety-based pathfinding algorithm that protects citizens from high crime zones in Medellin.

1.1. Problem

The problem we are trying to solve is to calculate the shortest path without exceeding a weighted-average risk of harassment and, to find the path with lowest level of danger without exceeding a distance.
It is important to return a short path without exceeding a given sexual harassment risk, because it reduces the probability of having uncomfortable and dangerous situations for women, but it is also important to return a path without exceeding a distance, because it may occur that the safest path is not convenient in terms of distance.

1.2. Solution

We chose Dijkstra algorithm for solving the street sexual harassment issue in the city. This algorithm is optimal to handle graphs with big number of nodes and find the shortest path in weighted and directed graphs as Medellin map. It is thought to get the shortest path from one starting node to all other nodes in the graph, thus we managed to modify the algorithm to stop if it finds the shortest distance to the target node, that represents the destination. We decided to create a program that manages to get a constrained path that balance both distance and risk for women's security. Another reason to choose Dijkstra is because of the execution time. In our tests, it did not take more than 0.3 seconds, finding multiple routes.

We have two options when creating a route between two places; the first one is to create a route that has the shortest path without exceeding certain harassment risk, and the second one is to create a route that has the safest path without exceeding certain distance. Aside the two main algorithms, that compares the previous two to the ultimate shortest or safest path without any secondary restriction.

1.3. Article structure

In what follows, in Section 2, we present related work to the problem. Later, in Section 3, we present the data sets and methods used in this research. In Section 4, we present the algorithm design. After, in Section 5, we present the results. Finally, in Section 6, we discuss the results, and we propose some future work directions.

2. RELATED WORK

In what follows, we explain four related works to path finding to prevent street sexual harassment and crime in general.

2.1 SafeStreet: empowering women against street harassment using a privacy-aware location based application

SafeStreet is a mobile application created to avoid sexual harassment. It was developed by the Department of

Computer Science and Engineering of Dhaka University. The application has a Safe Route Search functionality, which allows the user to give the location, destination, and preferred time of travel. SafeStreet, returns a safe path, and an unsafe path, if exists. It measures the safeness of a route depending on the records provided by other users [6]. The Safe Route Search algorithm wasn't specified in the article.

2.2 CROWDSAFE: Crowd Sourcing of Crime Incidents and Safe Routing on Mobile Devices

CROWDSAFE, is a mobile application created by the Department of Computer Science of Virginia Polytechnic Institute and State University. It allows users to search and report new crime information. It has s Safety Router, which provides the safest route between two locations, based on the reports given by other users. This implements the A* and Dijkstra shortest path algorithms, depending on the performance [7].

2.3 SafeRoute: Learning to Navigate Streets Safely in an Urban Environment

This study is an attempt to embrace emerging technologies such as machine learning, in long-existing problematics such as pathfinding. It was developed by researchers of University of California, Santa Barbara. This permitted the creation of a safety-based pathfinding platform in which information given by other users are processed by a deep RL algorithm, returning the better option regarding safety and travel time [8].

2.4 Safety-based path finding in urban areas for older drivers and bicyclists:

As its title states, this study works with an algorithm aiming for better conditions for elderly drivers and bicyclist. This paper written by researchers in Texas Transportation Institute focus on how to create a methodology that merges safety and pathfinding. Considering both, driver and traffic attributes, they came up with a multi-objective process that studies amongst the safe paths which one is the best one [9].

3. MATERIALS AND METHODS

In this section, we explain how data was collected and processed and, after, different constrained shortest-path algorithm alternatives to tackle street sexual-harassment.

3.1 Data Collection and Processing

The map of Medellín was obtained from Open Street Maps (OSM)[1] and downloaded using Python OSMnx API[2]. The (i) length of each segment, in meters; (2) indication wheter the segment is one way or not, and (3) well-known binary representation of geometries were obtained from metadata provided by OSM.

For this project, we calculated the linear combination that captures the maximum variance between (i) the fraction of households that feel insecure and (ii) the fraction of households with income below one minimum wage. These data were obtained from the quality of life survey, Medellín, 2017. The linear combination was normalized, using the maximum and minimum, to obtain values between 0 to 1. The linear combination was obtained using principal components analysis. The risk of harassment is defined as one minus the normalized linear combination. Figure 1 presents the risk of harassment calculated. Map is available at Github[3].

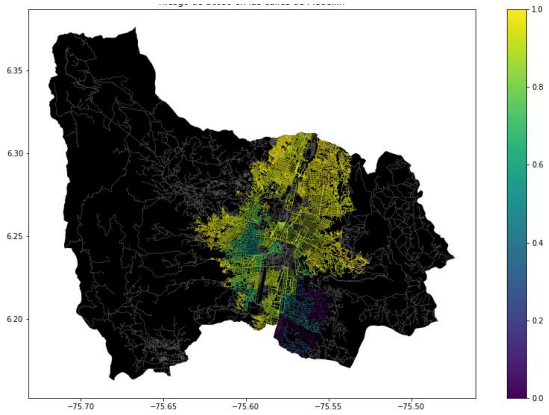

Figure 1. Risk of sexual harassment calculated as a lineal combination of the fraction of households that feel insecure and the fraction of households with income below one minimum wage, obtained from Life Quality Survey of Medellín, in 2017.

3.2 Constrained Shortest-Path Alternatives

In what follows, we present different algorithms used for constrained shortest path.

3.2.1 Extended Dijkstra Algorithm

Dijkstra algorithm is used to find the shortest path in a graph. It starts at the starting point node and calculates the movement cost to each connected node. It marks, or save, the node where the movement cost is the minimum. It repeats this process until it reaches the target node, or the destination. This algorithm requires that every node must be visited.

[1] https://www.openstreetmap.org/

[2] https://osmnx.readthedocs.io/

[3] https://github.com/mauriciotoro/ST0245Eafit/tree/master/proyecto/Datasets/

Thus, the search time can be long depending on the number of nodes. Therefore, it can be a good idea to use the Extended Dijkstra Algorithm, in which the algorithm is applied from both directions, this is, from the starting point to the target and vice versa. With this implementation, the algorithm must continue until the search from the starting point and the search from the destination overlap [10].

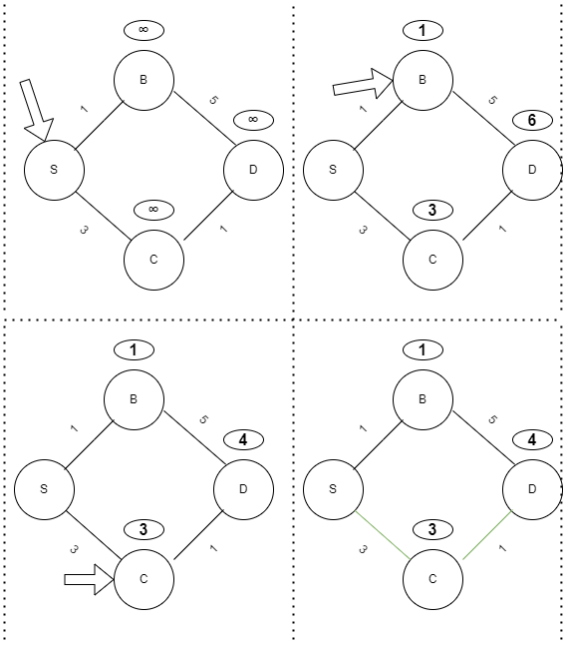

Figure 2. Dijkstra Algorithm.

3.2.2 A Star Algorithm

Also called A* Algorithm. It uses the combination of heuristic and searching based on the shortest path. Each adjacent cell of the current cell is evaluated by the heuristic distance to the goal cell, plus the length of the path from the initial state to the goal state through the selected cells. As in the previous algorithm, it chooses the "next cell" based on the minimum value obtained. This algorithm has some modifications, such as Basic Theta* and Phi* algorithms, which may be better when there is free space between connected cells, because they consider searching in every angle. [11].

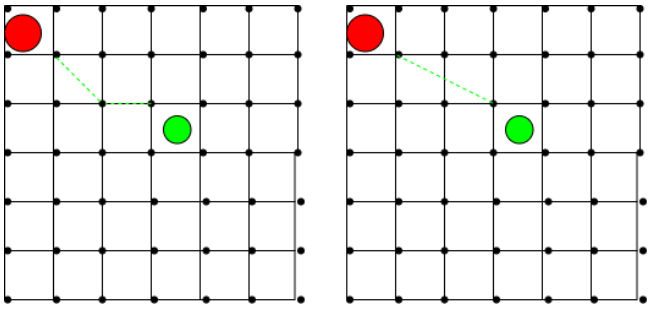

Figure 3. A Algorithm (left) and a modified A* Algorithm (right).*

3.2.3 Breadth first search Algorithm:

BFS algorithm is an algorithm used to examine nodes and edges from a graph. It uses first-in-first-out method in order to find the best path. It works by assigning every node a specific place in the process, every new one goes at the back of the queue. By doing this it examines every path until it arrives to the final node and then performs backtracking for confirm the path. This algorithm ensure that the first path found is the shortest one [12].

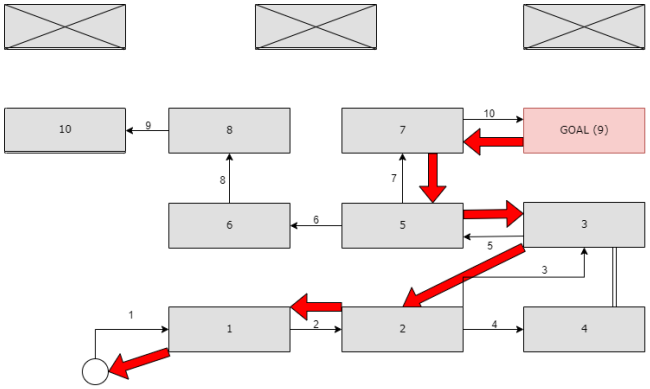

Figure 4. BFS Algorithm.

3.2.4 Depth first search Algorithm:

DFS algorithm is a very similar algorithm to the BFS algorithm, both find a route from a root node to a final node. The main difference between these two is the processed used by one and other. As already mentioned, BFS algorithms use FIFO processing of the data, unlike DFS algorithms use LIFO processing for finding an answer. Due to this DFS algorithm find a path but it isn't ensured to be the best path regarding distance [13].

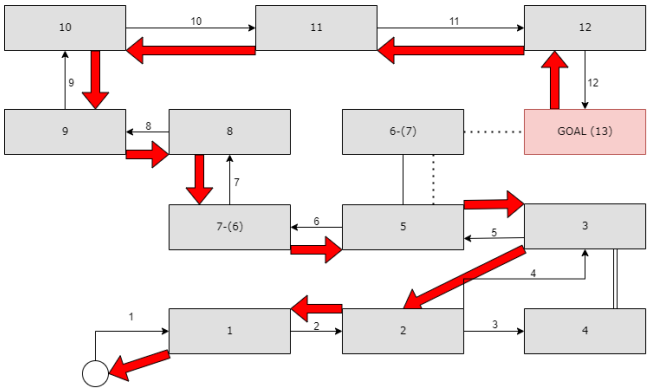

Figure 5. DFS Algorithm.

As seen in the image the path found by the algorithm is not the best. This is thanks to the stacking technique. When it arrives to the node five, it must assign an order to the next two options of pathing. Since the longest path was placed at the end of the stack (7) it continues to analyze this track. The suitable path is indicated by the dotted line (6-lost node).

4. Algorithm Design and Implementation

In what follows, we explain the data structures and the algorithms used in this work. The implementations of the data structures and algorithms are available at Github[4].

4.1 Data Structures

The data was given in a csv file. Thus, we use a Pandas DataFrame to clean in. After this, we implemented an adjacency list with dictionaries to represent the graph
The keys are the coordinates of each node of the graph, and the values are the coordinates of the adjacent nodes, with their distance and harassment risk.

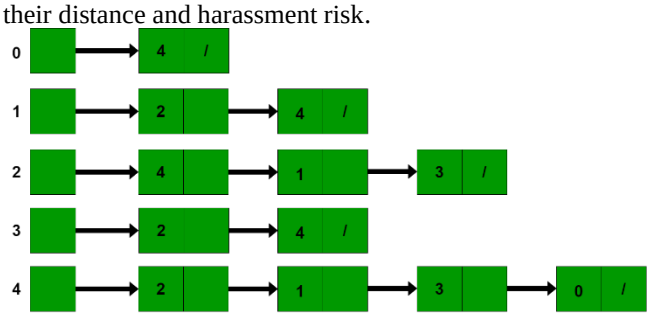

Figure 6. Graph as an Adjacency List.

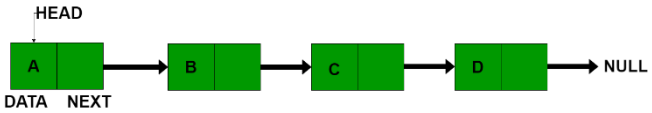

Figure 7. Linked List.

4.2 Algorithms

In this work, we propose algorithms for the constrained shortest-path problem. The first algorithm calculates the shortest path without exceeding a weighted-average risk of harassment *r*. The second algorithm calculates the path with the lowest weighted-average risk of harassment without exceeding a distance *d*.

4.2.1 First and Second Algorithm: Dijkstra Algorithm

As our problem was to find the shortest path on a weighted graph (distance and harassment risk), we chose Dijkstra Algorithm as the solution.

We used a dictionary, representing the distance from the starting point to every other node, and a priority queue with all the 'unvisited' nodes, meaning that the minimum distance to those nodes was not found yet. At first, that queue had the same number of nodes of the graph, and in the dictionary, the distance to all the nodes was assign to a large number, except from the starting node, that was assign to 0.

Then, we pop the node **v in** the queue that had the minimum distance value in the dictionary, and we updated the distance of the adjacent nodes of **v**, if the distance found plus the distance given in the graph was less than the distance in the dictionary. This process was repeated while the queue was not empty, or if **v** was equal to the target, meaning that the shortest distance of the target was found.

The same algorithm was used to calculate the path with the minimum harassment risk. *Figure 2.*

4.4 Complexity analysis of the algorithms

Dijkstra algorithm implemented with adjacency list has a time complexity of O(V+E log E) (V being the intersections and E the streets). Since in the worst case the while loop would have to go pass all the nodes for adding them to the priority queue, inside the while loop the algorithm traverse through all the neighbors of all the nodes meaning this segment complexity is O(V+E). And finally the complexity for adding an element to a priority queue is O(log E). And the result is shown below:

Algorithm	Time Complexity
Dijkstra Algorithm	O(V+E log E)

Table 1: Time Complexity of the Dijkstra algorithm, where V is the number of nodes and E is the number of edges.

Data Structure	Memory Complexity
Adjacency List	O(V)

Table 2: Memory Complexity of Adjacency list where V is the number of vertices.

4.5 Design criteria of the algorithm

Low time and simplicity for the user are the most important things for us. To have simplicity, we thought that receiving just a string location (not coordinates) was the best for the user. To achieve this, we implemented a function that converts the string location input to coordinates [16].

As our problem was to find the lowest cost path from a starting vertex to target vertex (the cost depends on the user needs) and the adjacency list was "big", we thought Dijkstra algorithm, with a little modification was a decent idea. As previously said, the executions times are very low, which shows that the algorithm was a good choice.

At first, we did not implement the priority queue and the algorithm did not stop even if it found the shortest distance to the target, and the tests time executions were around 2 minutes. It affirms that the implementation of the priority queue is fundamental in terms of time, even more with the size of the adjacency list.

Once the path is done with a recursive function, we thought that the library "GmPlot" was the best choice to show the map and the path. The documentation of the library is very clear, and it has all the needed functions, such as the construction of the polygon, representing the limits of the map. To see the path correctly, an API key is needed.

5. RESULTS

In this section, we present some quantitative results on the shortest path and the path with lowest risk.

5.1.1 Shortest-Path Results

In what follows, we present the results obtained for the shortest path without exceeding a weighted-average risk of harassment *r* in Table 3.

Origin	**Destination**	**Shortest Distance**	**Without Exceeding *r***
Universidad EAFIT	Universidad de Medellín	6130.017	0.84
Universidad de Antioquia	Universidad Nacional	2192.955	0.85
Universidad Nacional	Universidad Luis Amigó	1467.791	0.85

Table 3. Shortest distances without exceeding a weighted-average risk of harassment *r*.

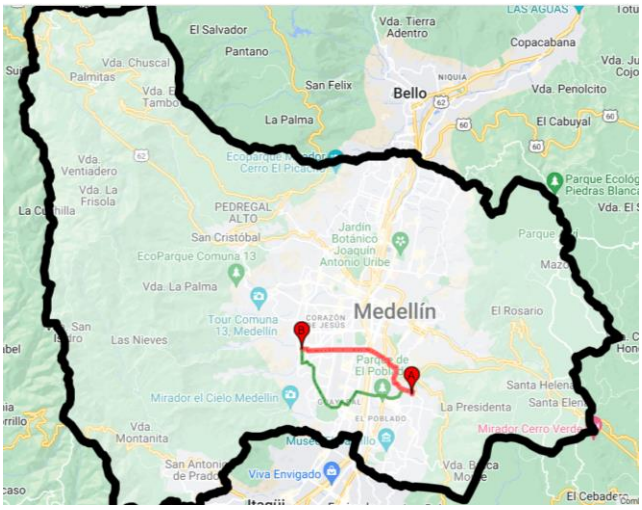

Figure 8. Visual example of two possible routes from Inter-continental Medellin (Point A) to Centro Comercial Los Molinos (Point B). Red being the shortest path (6642.151m) and green being shortest path not exceeding risk of 0.65 (8111.014m).

5.1.2 Lowest Harassment-Risk Results

In what follows, we present the results obtained for the path with lowest weighted-average harassment risk without exceeding a distance *d* in Table 4.

Origin	**Destination**	**Lowest Harassment**	**Without Exceeding *d***
Universidad EAFIT	Universidad de Medellín	0.7291	7,000
Universidad de Antioquia	Universidad Nacional	0.8435	7,000
Universidad Nacional	Universidad Luis Amigó	0.8514	6,500

Table 3. Lowest weighted-average harassment risk without exceeding a distance *d* (in meters).

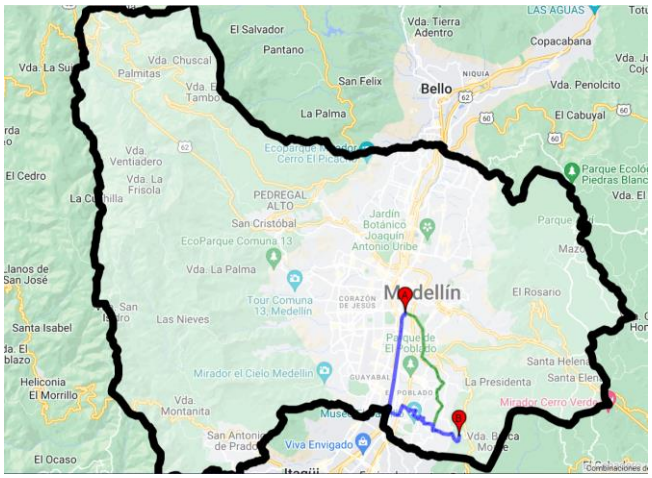

Figure 9. Visual example of two possible routes from metro station "Exposiciones" (Point A) to the restaurant "El Bosque era Rosado"(Point B). Blue being the safest path (0.2402) and green being safestest path not exceeding a distance of 9700m (0.2982).

5.2 Algorithm Execution-Time

In Table 4, we explain the relation of the average execution times for the queries presented in Table 3.

	Average execution times (s)
Universidad EAFIT to Universidad de Medellín	0.18 s
Universidad de Antioquia to Universidad Nacional	0.16 s
Universidad Nacional to Universidad Luis Amigó	0.17 s

Table 4: Execution times of the Dijkstra algorithm for the queries presented in Table 3.

6. CONCLUSIONS

The results obtained indicate that the shortest path is hardly ever similar to the lower harassment-risk path, which shows the big problematic of the city. This project, with multiple upgrades such as a more user-friendly interface, can lead to a better security, not only for the people in Medellin but also for the tourists.

In terms of the efficiency of the algorithm, considering that the project will be only used in Medellin, we think that the times are pretty good. It may get a little better, but the difference will not be that much.

As previously said, the time and the simplicity of the user are the most important things, thus, a more user-friendly interface probably is the most urgent thing to change.

6.1 Future work

We found very interesting working with problematics related to the city and its wellbeing. We would like to work in other situations like optimizing heavy traffic or health systems. As well as being able to turn all these projects into web sites that make the solution even more user-friendly.

ACKNOWLEDGEMENTS

The authors are grateful to Prof. Juan Carlos Duque, from Universidad EAFIT, for providing data from Medellín Life Quality Survey, from 2017, processed into a Shapefile.

REFERENCES

1. Ruta N. Reducir El Acoso Callejero: El Reto De La Secretaría De Mujeres. https://www.rutanmedellin.org/es/programas-vigentes/2-uncategorised/592-reto-de-mujeres
2. Semana. (2022, February 21). Aumenta la Violencia contra mujeres en Medellín: Van cinco asesinatos en 15 días. Semana.com Últimas Noticias de Colombia y el Mundo. Retrieved February 23, 2022, from https://www.semana.com/nacion/articulo/aumenta-la-violencia-contra-mujeres-en-medellin-van-cinco-asesinatos-en-15-dias/202231/
3. La Pandemia en la sombra: Violencia contra Las Mujeres Durante El Confinamiento. ONU Mujeres. (n.d.). Retrieved February 23, 2022, from https://www.unwomen.org/es/news/in-focus/in-focus-gender-equality-in-covid-19-response/violence-against-women-during-covid-19
4. Violencia y crimen en medellín: Pasado, presente y futuro. Medellín Cómo Vamos. (n.d.). Retrieved February 23, 2022, from https://www.medellincomovamos.org/boletin/violencia-y-crimen-en-medellin-pasado-presente-y-futuro
5. Investigación Desarrollada en Medellín Revela El impacto de la pandemia sobre la criminalidad en la ciudad. Alcaldía de Medellín. (n.d.). Retrieved February 23, 2022, from https://www.medellin.gov.co/irj/portal/medellin?NavigationTarget=contenido%2F10823-Investigacion-desarrollada-en-Medellin-revela-el-impacto-de-la-pandemia-sobre-la-criminalidad-en-la-ciudad

6. Mohammed Eunus Ali, Shabnam Basera Rishta, Lazima Ansari, Tanzima Hashem, and Ahamad Imtiaz Khan. 2015. SafeStreet: empowering women against street harassment using a privacy-aware location based application. In Proceedings of the Seventh International Conference on Information and Communication Technologies and Development (ICTD '15). Association for Computing Machinery, New York, NY, USA, Article 24, 1–4. DOI:https://doi.org/10.1145/2737856.2737870
7. Sumit Shah, Fenye Bao, Chang-Tien Lu, and Ing-Ray Chen. 2011. CROWDSAFE: crowd sourcing of crime incidents and safe routing on mobile devices. In Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems(GIS '11). Association for Computing Machinery, New York, NY, USA, 521–524. DOI: https://doi.org/10.1145/2093973.2094064
8. Sharon Levy, Wenhan Xiong, Elizabeth Belding, and William Yang Wang. 2020. SafeRoute: Learning to Navigate Streets Safely in an Urban Environment. ACM Trans. Intell. Syst. Technol. 11, 6, Article 66 (December 2020), 17 pages. DOI:https://doi.org/10.1145/3402818
9. Chandra, S. (2014, September 18). Safety-based path finding in urban areas for older drivers and bicyclists. Transportation Research Part C: Emerging Technologies. Retrieved February 23, 2022, from https://www.sciencedirect.com/science/article/pii/S0968090X14002344
10. M. Noto and H. Sato, "A method for the shortest path search by extended Dijkstra algorithm," Smc 2000 conference proceedings. 2000 ieee international conference on systems, man and cybernetics. 'cybernetics evolving to systems, humans, organizations, and their complex interactions' (cat. no.0, 2000, pp. 2316-2320 vol.3, doi: 10.1109/ICSMC.2000.886462.
11. Duchoň, F., Babinec, A., Kajan, M., et al. 2014. Path planning with modified a star algorithm for a mobile robot. Procedia Engineering. https://www.sciencedirect.com/science/article/pii/S187770581403149X.
12. Khan Academy. (n.d.). The breadth-first search algorithm (BFS) (article). Khan Academy. Retrieved February 23, 2022, from https://www.khanacademy.org/computing/computer-science/algorithms/breadth-first-search/a/the-breadth-first-search-algorithm
13. Elgabry, O. (2017, October 12). Path finding algorithms. Medium. Retrieved February 23, 2022, from https://medium.com/omarelgabrys-blog/path-finding-algorithms-f65a8902eb40
14. Brilliant.org. 2022. *Dijkstra's Shortest Path Algorithm | Brilliant Math & Science Wiki*. [online] Available at: <https://brilliant.org/wiki/dijkstras-short-path-finder/> [Accessed 17 April 2022].
15. Anon. 2022. Dijkstra's algorithm for adjacency list representation: Greedy Algo-8. (April 2022). Retrieved April 17, 2022 from https://www.geeksforgeeks.org/dijkstras-algorithm-for-adjacency-list-representation-greedy-algo-8/
16. Tuanavu. 2016. Coursera-university-of-michigan/geojson.py at master · Tuanavu/Coursera-University-of-michigan. (January 2016). Retrieved May 19, 2022 from https://github.com/tuanavu/coursera-university-of-michigan/blob/master/python_for_everybody/4_using_databases_with_python/code/geojson.py