PATH FINDING ALGORITHM FOR SEXUAL HARASSMENT PREVENTION, APPLIED TO MEDELLÍN-COLOMBIA.

Carlos Alberto Mazo Gil
Universidad Eafit
Colombia
camazog1@eafit.edu.co

Rafael Urbina Hincapié
Universidad Eafit
Colombia
riurbinah@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 persistent issue for the citizens of Medellín. The biggest concern is that sexual harassment can be: touching, comments, persecution, masturbation, observation, and so much more [1]. Because of the numerous ways the aggressor has to impact the victim, it becomes harder for the authorities to enforce the law. Due to this, it is important to implement new methods and alternatives that can help the citizens reduce the risk. This project aims to create an algorithm that can provide safer and shorter routes for the citizens of Medellin.

Key words

Shortest route, street sexual harassment, identification of safe routes, crime prevention

1. INTRODUCTION

According to Cacua[3], from 2012 to 2018, Medellín has been a city where women and children live in a context of violence, mainly because of sexual harassment and sexual crimes, this happens due to the ineffectiveness of the policies that the city has in place. Due to this, women and children live, walk and travel with fear [3], it is also important to note that the LGBTQ+ community is also affected, as every citizen is. This in turn, creates a generalized fear.

There are other alternatives apart from the law enforcement. Social alternatives that attempt to solve this issue, for example: "Ciudades seguras" or "Safe cities" is a project in some of the Latin-American countries to fight this problem [2], their focus is on how women can protect themselves from sexual harassment. The idea of creating a path-find algorithm for avoiding sexual harassment is an alternative on how the citizens of Medellín can protect themselves or at least prevent the risk without relying only on what the authorities can do.

In this project we contribute by creating a safety-based pathfinding algorithm that reduces both distance and risk of sexual harassment, we intend to reduce both the distance and the risk as weighted-averages since the safest path without constraints can be so long that is not worth taking in the day to day life, in the other hand, the shortest path without constraint can have more risk than a person would want to take.

1.1. The problem

The problem we attempt to solve is to create three different paths that reduce both the distance and the risk of sexual harassment, to solve this problem is important to consider different methods or different parameters of distance and risk, both options are valid, these three different paths can give the end user more than one option to choose, either he wants a safer path ,a shorter path, or a balanced path.

1.2 Solution

Our solution is based on Dijkstra's algorithm, one of the most used algorithms for shortest route path finding, we choose Dijkstra since it's one of the most supported ones for this type of research, A star algorithm was another option but since the computations in A star are larger and it's similar to Dijkstra, we choose Dijkstra instead, we will explain Dijkstra and three more algorithms in section 3. The solution involves combining the length between each node and the harassment risk to form a length and risk value that helps us determine the optimum path. In the first scenario we opt for a value = log(length) + risk. Where length is taken as the logarithm, this way both variables have a closer relevance in terms of the weight.

1.3 Structure of the article

Next, in Section 2, we present work related to the problem. Then, in Section 3, we present the datasets and methods used in this research. In Section 4, we present the algorithm design. Then, in Section 5, we present the results. Finally, in Section 6, we discuss the results and propose some directions for future work.

2. RELATED WORK

Below, we explain four works related to finding ways to prevent street sexual harassment and crime in general.

2.1Preventing Sexual Harassment Through a Path Finding Algorithm Using Nearby Search.

This application was made by Omdena, Omdena is an innovation platform for building AI solutions to real-world problems, in 2020, they published an application that proposed something similar to our goal but was mainly implemented in India, they created a Path finding algorithm that took heat maps in to account, these heat maps had values to simulate a crime index, in this way, the algorithm could predict the fastest route but also the safest one(The algorithm choose first the safest path, then if two paths are in the same safety-ness level, the algorithm would choose the shortest one), the algorithm was built taking two ideas, first the Euclidean shortest distance between two points, and Bresenham’s drawing method which tries to draw a straight line from points that might have different elevations.

2.2 Safest path via safe zones.

This is a publication from students of: the university of Melbourne, Aalborg university and Latrobe university, they attempt to create an algorithm that can only show the optimal route outside “Safe zones”, the goal focuses on something that related work had not done by that time, they used Euclidean variations and spatial network variants to solve the problem, in the results they find a more efficient way to solve this problem by using hyperbolas to prune the search space, they then conclude that their method is more efficient than the Euclidean approach for this case of traveling outside of “Safe zones”.

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

SafeStreet is a project published in 2015 that attempts to create a crowd-powered privacy-aware location based mobile application for the widespread cases of street harassment that are not handled correctly from law-enforcement agencies, some examples of these are: commenting, catcalling. SafeStreet allows women to capture and share their experiences in streets, this creates a data base that shows what other women have encountered in certain parts of a city.

2.4 CROWDSAFE

Is a novel solution for crime reporting, it uses portable smart devices to enable real-time location-based crime incident searching and reporting, it has a wider spectrum since it not only takes sexual harassment but all the criminal activity. the mobile application also provides other features like safe path-finding router and crime analytics. It is mainly implemented in metropolitan Washington DC.

3. MATERIALS AND METHODS

In this section, we explain how the data were collected and processed, and then different alternative path algorithms that reduce both the distance and the risk of sexual street harassment.

3.1 Data collection and processing

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

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

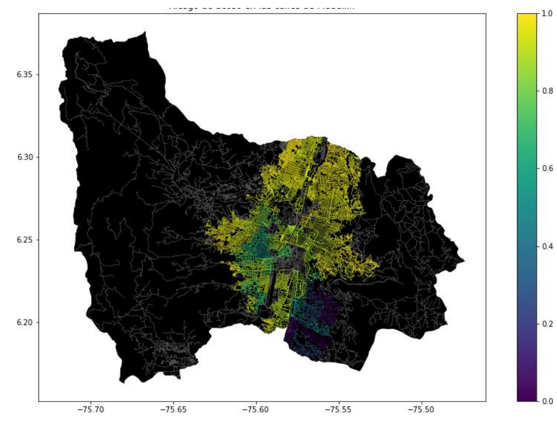

Figure 1. Risk of sexual harassment calculated as a linear combination of the fraction of households that feel unsafe and the fraction of households with income below one minimum wage, obtained from the 2017 Medellín Quality of Life Survey.

3.2 Algorithmic alternatives that reduce the risk of sexual street harassment and distance

In the following, we present different algorithms used for a path that reduces both street sexual harassment and distance.

3.2.1 A star algorithm or A*

Is a path search algorithm used mainly to find the closest route from point "a" to point "b" in a metric or topological grid space, the algorithm uses two main components, it uses heuristic searching and searching based on the closest path to the goal, the algorithm has versatility since the distances from cell to cell can be adopted as other units of measure

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

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

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

apart from distance, it can be: time, safety, complexity and so on, because of this, the A* algorithm can be used in a wide number of fields. In the A* algorithm every cell is evaluated by the following equation: f(v)=h(v)+g(v) [9].

where h(v) is the heuristic distance of the cell in evaluation to the goal cell, and g(v) is the distance from the initial point, to the goal cell, passing strictly through the collection of cells in evaluation, then, from each cell, it calculates the f(v) of every adjacent cell and chooses the cell that has the lowest f(v) value, repeating this process until it gets to the goal will provide the shortest route possible from the initial point to the goal point.

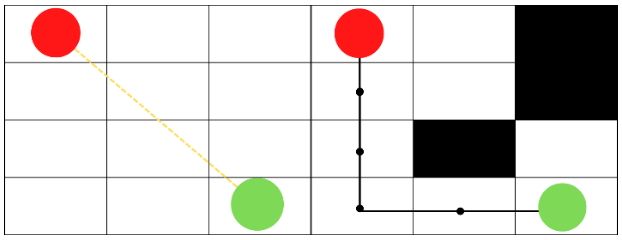

Figure 2. A* algorithm initial to goal cell projection (Left), Algorithm applied in a constrained grid (Right).

3.2.2 Dijkstra's algorithm

It is an algorithm to determine the shortest path, given an origin vertex, to the rest of the vertices of a graph whose edges have been associated with a series of weights, the minimum path of a vertex that goes from X to Y, taking into account where the sum of the weights on the arcs is the smallest possible among all the paths from X to Y.

The complexity of Dijkstra's algorithm is O (n^2) where n is taken as the number of vertices; furthermore, this algorithm is based on optimization by saying that if the path from X to Y passes through Z then Z must be the shortest path. From Z to Y and so on until you reach Y.

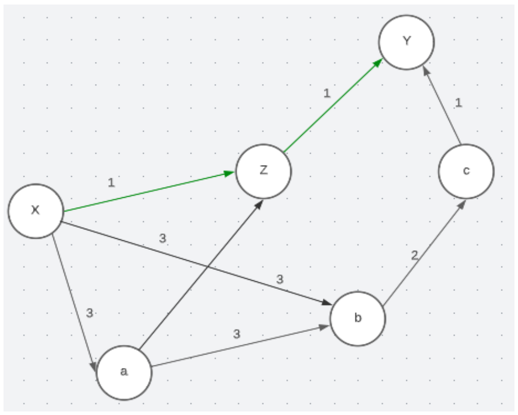

Figure 3. An example to represent this algorithm is how to go from X to Y passing through Z, as it is the path with the least weight

3.2.3 Bellman algorithm

The Bellman-ford algorithm solves the one origin shortest path problem in which the weights of the arcs can be negative. The Bellman-Ford algorithm for shortest paths is also almost completely intuitive and returns a Boolean indicating whether there is a negative weight cycle that is accessible from the source. Therefore, when there is a cycle, the algorithm says that there is no solution except when there is none; the algorithm produces the shortest paths and their weight. Furthermore, its basis is as follows: given a graph with n vertices, the greatest number of edges there can be in a shortest path is n-1. The shortest path has more than n-1 so it must have a negative circuit. Similarly, a negative circuit graph will have the shortest path of n or more arcs.

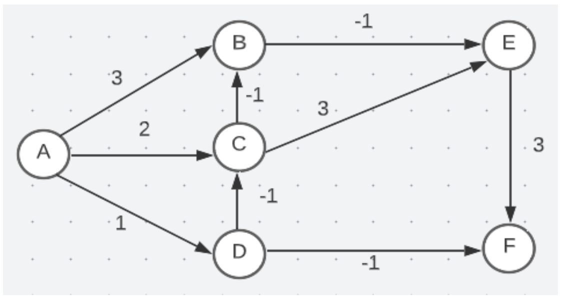

Figure 4. An example to represent bellman's algorithm

3.2.4 Floyd algorithm

Floyd-Warshall is an algorithm to find the shortest path between all pairs of vertices in a weighted graph, that is, in a balanced way, this algorithm is also known as Floyd's algorithm, Roy-Floyd algorithm, Roy-Warshall algorithm or WFI algorithm, unfortunately this algorithm does not work for negative cycle graphs. The Floyd-Warshall algorithm follows a dynamic programming approach to find the shortest possible path.

"The Floyd algorithm, given the matrix L of adjacency of graph g, calculates a matrix D with the length of the minimum path that joins each pair of vertices." (Jiménez, Parra, & Torres, 2014)

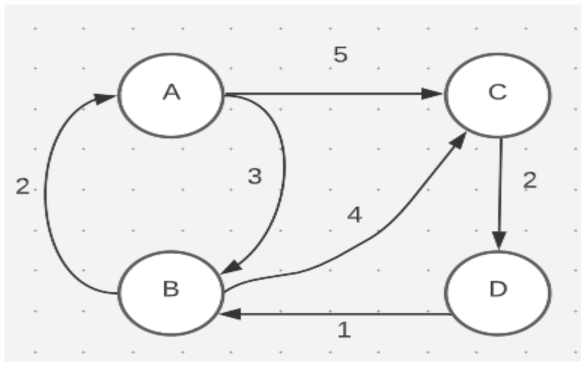

Figure 5: An example to represent Floyd's algorithm

4. ALGORITHM DESIGN AND IMPLEMENTATION

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

4.1 Data Structures

The data structure that we choose to represent the city of Medellín is based on a adjacency list implemented with dictionaries, this list allows the nodes to have unique keys for every neighbor that they have and have the value associated with each arc, every node is the string of the pairs longitude and latitude, this allows us to locate any node only by knowing it's coordinates instead of naming every node with an arbitrary name. In terms of code we implemented the whole data structure with a class called Graph, the reason is that it becomes easier to use, with the class Graph we could even import the script for future projects without creating a new one or having it in the same algorithm script, in this way we can just import the file and create a new object of the class Graph, inside of the class Graph we first create a method called construct_graph that allow us to create an empty dictionary with every key(Node), then we create a method called get_outgoing_edges, this method is used to do the connections between the neighbors, thus assigning the corresponding weights and getting the dictionary structure complete, at the end its something like the following: dict[node_start][node_end] = {Weight}. The data structure is presented in Figure 2. (Keep in mind these nodes represented as letters are in fact coordinates and every value associated with every arc is a combination of length and risk (Weight).)

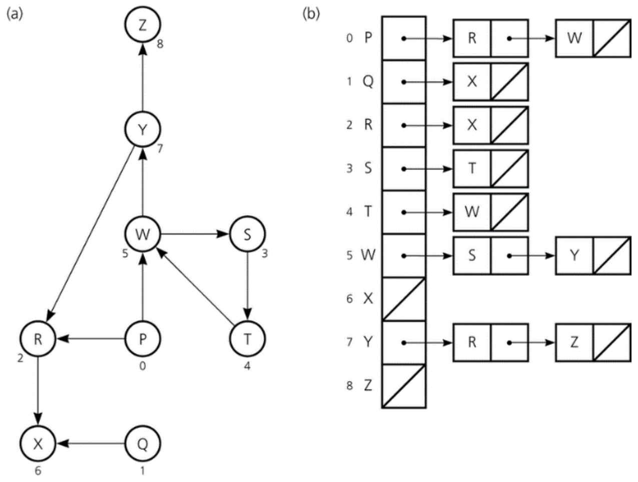

Figure 6: An example street map is presented in (a) and its representation as an adjacency list in (b).

4.2 Algorithms

In this paper, we propose an algorithm for a path that minimizes both the distance and the risk of street sexual harassment.

4.2.1 Algorithm for a pedestrian path that reduces both distance and risk of sexual street harassment

Since our problem is to find the shortest path taking length and risk into consideration, we first set every node value as close to infinity and the start node value to zero, this is done so that the algorithm can start from the start node and take each neighbor value once it updates from infinity to the real value, this approach has some similarities with the BFS that searches first all the neighbors before taking on a deeper level. After the algorithm starts to visit the current node neighbors, update their real value, and remove them from the unvisited nodes list when visited, the algorithm creates the tentative value to every tentative path, then with the tentative values of every arc and finding the value from the start node to every node, the algorithm chooses the shortest path. The algorithm is exemplified in Figure 3.

[4] https://github.com/RafaelUrbina/st245-002

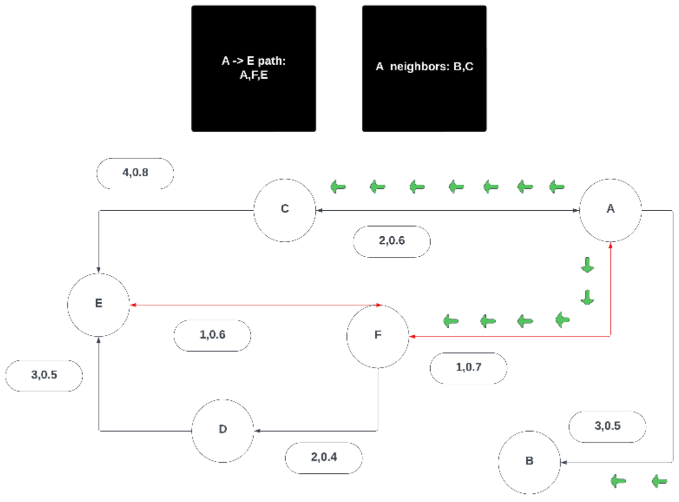

Figure 7: Calculation of a path that reduces both distance and risk of harassment.

4.2.2 Calculation of two other paths to reduce both the distance and the risk of sexual street harassment

As we discussed in the introduction section, the algorithm used for this work is the Dijkstra algorithm, initially we worked with a combination of both variables (Length and risk), this combination was set as the sum of: log(Length)+risk, for the other two path calculations we compute the shortest path and the safest one separately, this approach lets us compare between the two extreme paths and one that falls somewhere in the middle, it's important to note that sometimes the safest can be very similar to the combination or the other way around, the shortest can be very similar to the combination, the reason for this, can be that the algorithm has not enough constraints or it needs more node data, because of this we present two paths in this section.

The algorithm is exemplified in Figure 4 and Figure 5.

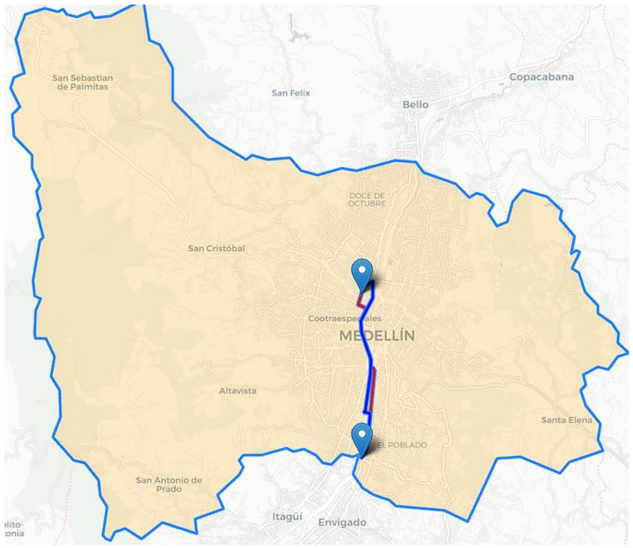

Figure 4: Map of the city of Medellín showing three pedestrian paths that reduce both the risk of sexual harassment and the distance in meters between the EAFIT University and the National University. The red line being the shortest path, the blue one being the safest and the yellow the combination of both. (In this case the safest path is almost the same as the combination of both)

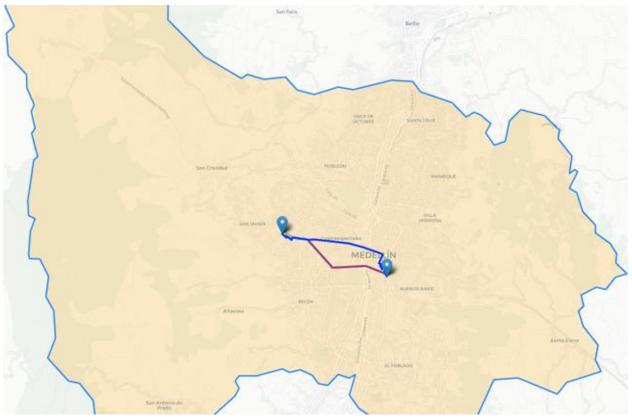

Figure 5: Map of the city of Medellín showing three pedestrian paths that reduce both the risk of sexual harassment and the distance in meters between the "Inicio via Las palmas" and "El danubio – La america" Quarter. The red line being the shortest path, the blue one being the safest and the yellow the combination of both.

4.3 Algorithm complexity analysis

In our implementation of the Dijkstra algorithm we first iterate through all the unvisited nodes(All nodes) to assign the maximum value to them and let the starting node have a value of zero, this sets the complexity at O(V) complexity to this point, V being the number of vertices, after ,we check that while we have unvisited nodes we iterate through the unvisited nodes to check which neighbor has the shortest path and then remove it from the unvisited nodes, since we have two loops one inside the other the complexity at the end becomes O(V^2), It's also important to mention that the complexity could be lowered with a min-priority queue, the complexity could become O(V + E Log(V)).

Algorithm	Time complexity
Dijkstra	O(V^2)

Table 1: Time complexity of Dijkstra algorithm, where V is the number of vertices and E are the edges.

Data Structure	Complexity of memory
Adjacency list	O(V)

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

4.4 Algorithm design criteria

Our problem was to find the best path according to the user needs, this is done by searching the best path from the starting node to the target node, since the data base used had a considerable amount of vertices, we choose an algorithm that was well tested by other authors and that has a well-documented implementation and problem solutions, also between the ones explained in section 3, Dijkstra is one that uses less calculations than some of the other three.

Although our algorithm was selected mainly for its simplicity, we found out about the priority queue very late in the stages of the project, implementing a priority queue can shorten the Average run times drastically, so this is one of the main things to do in future work.

For the map we used folium as the main source to plot every route that was shown in this project. Folium is a library for python that allows easy geolocation plotting without having to access the internet, it works with geopandas to find the coordinates.

We choose folium instead of GmPlot because we needed to work on the project locally (without internet) on some occasions, also we had already worked with folium before.

5. RESULTS

In this section, we present some quantitative results on the three pathways that reduce both the distance and the risk of sexual street harassment.

5.1 Results of the paths that reduces both distance and risk of sexual street harassment

Next, we present the results obtained from *three paths that reduce both distance and harassment,* in Table 3.

Origin	Destination	Distance	Risk
Eafit(Length)	Unal	7966.87	60.14
Eafit(Combi)	Unal	9671.59	35.10
Eafit(Risk)	Unal	9680.52	35.10

Table 3: Distance in meters and risk of sexual street harassment (between 0 and 1) to walk from EAFIT University to the National University.

Origin	Destination	Distance	Risk
Palmas(Length)	La america	5603.60	67.66
Palmas(Combi)	La america	6309.87	61.29
Palmas(Risk)	La america	6365.76	61.28

Table 4: Distance in meters and risk of sexual street harassment (between 0 and 1) to walk from Via las palmas to La america quarter.

5.2 Algorithm execution times

In Table 5 and Table 6, we explain the ratio of the average execution times of the queries presented in Table 3.

Calculation of the execution time for the queries presented in Table 3.

Calculation of v	Average run times (s)
v = Length	163.56s
v = Combination	164.24s
v = Risk	166.48 s

Table 5: *Dijkstra* execution times for each of the three calculator paths between EAFIT and Universidad Nacional.

Calculation of the execution time for the queries presented in Table 4.

Calculation of v	Average run times (s)
v = Length	163.45s
v = Combination	162.80s
v = Risk	161.85 s

Table 6: *Dijkstra* execution times for each of the three calculator paths between Via las palmas and La America quarter.

6. CONCLUSIONS

The results show that the shortest and safest path are very different in terms of the route taken, the combination of them tends to be very similar to one of them, either it follows the same route of the safest or the shortest path and has usually a few turns that are different, but in general tends to take the same main roads. One example of this is the path from Via las palmas to La America quarter, in this one the combination changes slightly at the start and then at the end of the route.

In terms of user application the project could be used by the citizens of Medellín after implementing a few changes, two of the most important ones are: Creating a way to input the locations for their name, currently our model its taking the nodes as a string pair of coordinates, if the user could input the name of the desired location the application would become easier, second, the Average runtimes are a little high for the daily use since it takes approximately 2 minutes to finish, this could be solved with the priority heap, that is one of the main changes for future work.

6.1 Future work

Two of the main factors we already talked about are the implementation of a friendly interface that takes destinations by its name, the second one would be implementing the priority heap to decrease the average runtimes substantially and make the project usable in a daily basis, it's also important to mention that these social problems need a lot of contribution in the city of Medellín, another one that could help the city is finding an optimal way to redistribute the routes, of the public transport in the city, since they have proved to be very inefficient over the years.

ACKNOWLEDGEMENTS
The authors thank Professor Juan Carlos Duque, Universidad EAFIT, for providing the data from the 2017 Medellín Quality of Life Survey, processed in a *Shapefile*.

REFERENCES

[1]
María Claudia López Gi López Gil. 2018. Acoso sexual callejero: Evaluación de su percepción cultural en el Valle de Aburrá y análisis de género de las formas de sanción en Colombia, en las últimas dos décadas. *Revista Indisciplinas* 4, (May 2018), 79-100. DOI:https://doi.org/10.24142/indis

[2]
Ana Milena Montoya Ruiz and Angela María Correa Londoño. 2018. Ciudades seguras y sin violencias para las mujeres y las niñas, avances y retos de la ciudad de Medellín, Colombia. *Perspectiva Geográfica* 23, 2 (December 2018). DOI:https://doi.org/10.19053/01233769.7384

[3]
Shaaron Dennisse Cacua Jaimes. 2019. Política De Seguridad Y Convivencia Ciudadana Y Política De Espacio Público Desde La Narrativa De Las Mujeres En El Municipio De Medellín 2012-2018. *Facultad de Derecho, Ciencias Políticas y Sociales, Departamento y Área Curricular de Ciencia Política* (June 2019).http://repositorio.unal.edu.co/handle/unal/78986

[4]
Daniel Ma. 2020. Building a Path Finding Algorithm to Prevent Sexual Harassment. *Omdena*. Retrieved August 4, 2022 from https://omdena.com/blog/path-finding-algorithm/

[5]
Saad Aljubayrin, Jianzhong Qi, Christian S. Jensen, Rui Zhang, Zhen He, and Zeyi Wen. 2015. The safest path via safe zones. *2015 IEEE 31st International Conference on Data Engineering* (2015). DOI:https://doi.org/10.1109/icde.2015.7113312

[6]
Mohammed Eunus Ali, Shabnam Basera Rishta, Lazima Ansari, Tanzima Hashem, and Ahamad Imtiaz Khan. 2015. SafeStreet. *Proceedings of the Seventh International Conference on Information and Communication Technologies and Development* (2015). DOI:https://doi.org/10.1145/2737856.2737870

[7]
Fenye Bao, Ing-Ray Chen, Chang-Tien Lu, and Sumit Shah. 2011. Leveraging Ubiquitous Computing for Empathetic Routing: A . Retrieved August 5, 2022 from https://dl.acm.org/doi/abs/10.1145/3411763.3451693

[8]
Samuel Rico Gómez, Mauricio Toro, Andrea Serna, and Gregorio Bermúdez. 2022. Safety Based Pathfinding Algorithm For Reduction of Sexual Harassment. (2022). DOI:https://doi.org/10.31219/osf.io/x5av2

[9]
František Duchoň, Andrej Babinec, Martin Kajan, Peter Beňo, Martin Florek, Tomáš Fico, and Ladislav Jurišica. 2014. Path Planning with Modified a Star Algorithm for a Mobile Robot. *Procedia Engineering* 96, (2014), 59-69. DOI:https://doi.org/10.1016/j.proeng.2014.12.098

[10]
Andrew Goldberg and Robert Tarjan. 1996. Expected Performance of Dijkstra's Shortest Path Algorithm. (1996). DOI:https://doi.org/https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.39.7441&rep=rep1&type=pdf

[11]
Simar Herrera, Octavio Salcedo, and Adriana Gallego. 2014. Efficiency Graphs fllgorithmic's applications oriented to GMPLS networks. (2014). DOI:https://doi.org/http://www.scielo.org.co/scielo.php?script=sci_abstract&pid=S0121-11292014000100009

[12]
Fadhil Mukhlif and Abdu Saif. 2020. Comparative Study On Bellman-Ford And Dijkstra Algorithms. (2020). Retrieved 2022 from https://www.researchgate.net/publication/340790429_Comparative_Study_On_Bellman-Ford_And_Dijkstra_Algorithms

[13]
Alexey Klochay. 2021. Implementing Dijkstra. Retrieved September 22, 2022 from https://www.udacity.com/blog/2021/10/implementing-dijkstras-algorithm-in-python.html

[14]
Hackerearth. 2022. Shortest Path Algorithms. Retrieved November 01, 2022 from https://www.hackerearth.com/practice/algorithms/graphs/shortest-path-algorithms/tutorial/