RESEARCH OF AN ALGORITHM ABLE TO PREVENT THE STREET HARASSMENT IN MEDELLIN

Alejandro Ríos
Universidad Eafit
Colombia
ariosm@eafit.edu.co

Marcela Londoño
Universidad Eafit
Colombia
mlondonol@eafit.edu.co

Juan Alejandro Osorno
Universidad Eafit
Colombia
jaosornob@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 problem women experience in their daily life, this can include verbal and non-verbal abuse, and can take place everywhere, in Medellin “over 61,5% of women for certain neighborhoods manifest that they felt insecure to move around the city after 7pm” [10]. The importance of this problem is that the fear women experienced of sexual harassment, make them take a certain behaviour to prevent walking and passing by those places they’re more likely to be abuse, which affect their life opportunities and freedoms of mobility. Due to this, in order to make women feel more safe, this project and research is about using an algorithm that is able to calculate different paths depending on the distance and the sexual harassment of Medellin streets.

Key words

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

1. INTRODUCTION

Everyday sexism might not be noticed by perpetrators or bystanders, but it can wear women down and is linked to poorer physical and mental health. They often experience sexual objectification, such as street harassment and unwanted touching. Rules and laws against gender discrimination do not prevent people with sexist attitudes from treating others unfairly in everyday interactions, so we decided to help them raise their incentive and avoid being treated like that everyday. That is our motivation and what our program does. It helps people, especially women, to recognize their value and make the best decisions in their daily path, taking into account the distance and the risk of each street to provide the best path to womens, because this is our main reason in the project.

1.1. The problem

The problem is to find an algorithm that calculate the three different paths considering the distance and the sexual harasssment in Medellin streets, always trying to find paths that have a short distance and a small risk of sexual harassment, in order to produce a path safe for women in their daily mobility around the city. The impact this problem has on society is to create an algorithm that calculates and identifies safe routes to provide women security during their travel around the city, which makes women able to reach life opportunities they are missing because of their restricted mobility. It’s useful to solve this problem to gather information about the most common places women are victims of sexual harassment, this helps to find more solutions to solve this global problematic.

1.2 Solution

For this work, we chose the Dijsktra’s Algorithm to execute our solution to the problem explained in section 1.1. The reason for this selection is that we find the execution and implementation of this algorithm better and more useful to function with Medellin’s map, the algorithm also receives the origin and destination, it runs through all the weight graph (through the adjacent streets in the map) trying to not pass through a visited vertex, and it finishes when all the vertices are visited, then the algorithm returns the shortest and safest path from the starting point to the destination. In section 4.2.1 is a more extended version of the implementation for our solution, and in section 3.2.2 how the algorithm works.

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.

2.1 Preventing Sexual Harassment Through a Path Finding Algorithm using Nearby Search

This research made in Mumbai has the purpose of finding an algorithm that helps find a path considering the risk of sexual harassment. “The results from this case study depend on previous work on heatmaps, which predict places at high risk of sexual harassment incidents.”

The algorithm they used is Google’s Maps API, which is the combination of Nearby Search and Directions API. Also, they used the Bressenham’s line algorithm to compute the average of risk scores of steps for each route to determine the best one. They prioritize the safety route first, before considering the distance [1].

2.2 Route-The Safe: A Robust Model for Safest Route Prediction Using Crime and Accidental Data

This study in New York City focuses on finding an algorithm which considers the crime and accident risk of the routes, with the purpose to suggest people the safest route to the given destination. In order to make the prediction of the safety of a route they retrieved information from NYPD available data from the Arrest and Accidents Datasets. Making the proper calculations from analyzing the NYPD Datasets, they used Google API technology combined with K Means clustering algorithm and KNN Regressor algorithm to find the proposed solution.

The solution they presented recommends the safer route from one origin to a destination, the safer route means that it's the path with the lowest risk score and it's calculated based on accidents and crimes that happened on that route or near it (which was took from the NYPD Dataset), if more than one route has the lowest risk score the solution suggests the route with the shortest distance [2].

2.3 Empowering Women through a Privacy-Aware Location Based Application

A research made in Bangladesh by the Dept. of Computer Science and Engineering of Dhaka University [3], was created with the aim of empowering womens and also protecting their identities, preventing them from feeling shame or fear to inform these situations, taking into account all the social reality that this country lives related with the rights and perception of women. SafeStreet provides many tools that allow them to report in live time where they are suffering harassment, and also helping others to avoid those places where the harassment is high at determined hours.

In the same way, this provides the safest route that a woman can choose depending on all these factors and being friendly in their use with the users, making it easy to use all the tools that has. The research does not provide detailed information of how the algorithm works, however, they explained all the software processes followed to develop this App [3].

2.4 Safe street rangers: crowdsourcing approach for monitoring and reporting street safety

The faculty of engineering at the Chiang Mai University, allocated in Thailand; reports a mechanism that tries to monitor street safety. It allows the users around a certain zone to report different conditions about the streets and their status focusing on things like the brightness, traffic signs, solitariness and past accidents. The program system is based on a data server, a mobile and a web version of the app. Each report would be verified by the admin users to prove the value of the report.

This work could help us think outside our main goal and find more ways to deduce the factor that composes the problem we´re trying to solve. In fact, there are some limitations that we can attempt to expand with our own system and research [7].

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] .

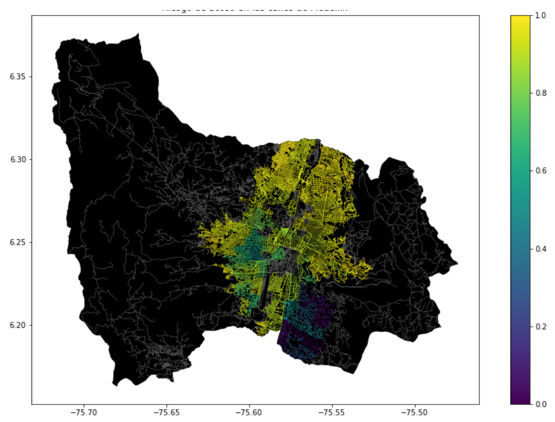


Figure 1. Risk of sexual harassment calculated as a linear combination of the fraction of households that feel unsafe

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

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

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

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 Bresenham's Line Algorithm

"Bresenham Line Algorithm is an optimistic & incremental scan conversion Line Drawing Algorithm which calculates all intermediate points over the interval between start and end points, implemented entirely with integer numbers and the integer arithmetic. It only uses addition and subtraction and avoids heavy operations like multiplication and division." [4]. The complexity of this algorithm is O(n) in the worst, average, and best case time, and it's O(1) in the space complexity.

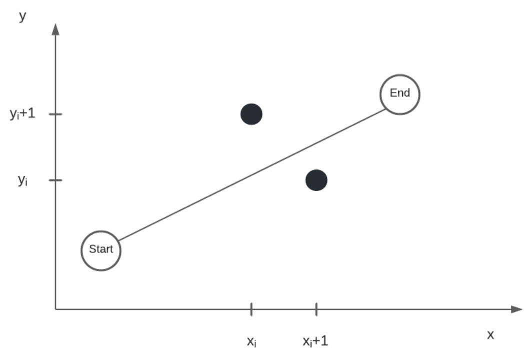


Figure 2. Representation of Bresenham's Line Algorithm.

3.2.2 Dijsktra's Algorithm

Dijkstra's Algorithm allows us to find the shortest path between any two vertices of a graph. The main base of this algorithm is that any subpath between two vertices A-D is also the shortest path between this subpath B-D. The algorithm searches for the next best solution hoping that in the end this will be the best solution of all possible solutions, because the algorithm overestimates the distance of each vertex from the start point [5].

All starts with a weighted graph, where you choose a starting vertex and an aim vertex, and the rest of vertices have infinity values. The algorithm starts to go to each vertex and updates its path length according to the shortest length between adjacent vertices.

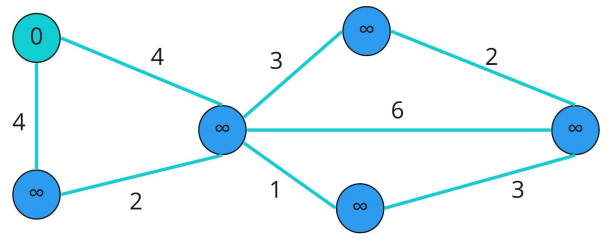


Figure 3. Representation of the starting phase of Dijkstra's Algorithm.

The algorithm avoids updating paths that were already visited. After each iteration, the algorithm picks the unvisited vertex with the least path length. All this process finishes when all the vertices have been visited.

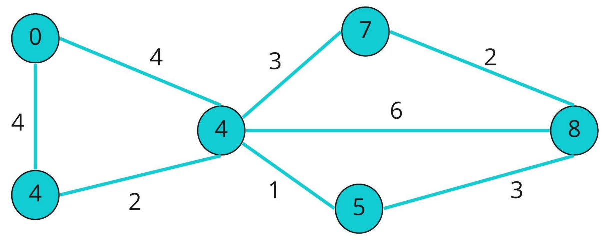


Figure 4. Representation of the final phase of Dijkstra's Algorithm where the shortest path is found.

The time complexity is O(E Log V), where E is the number of edges and V is the number of vertices. The space complexity is O(V) [6].

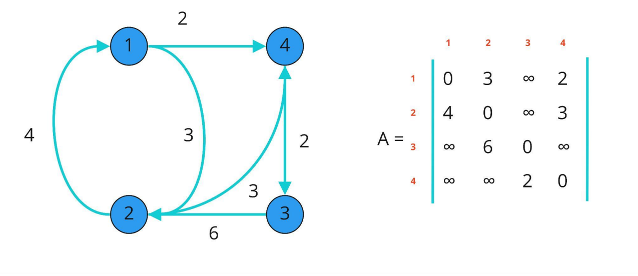
Dijkstra's Algorithm helps us to find the shortest path, the location in a map and is even used in social networking applications to create effective connections between users [6].

3.2.3 Floyd-Warshall Algorithm

Floyd-Warshall Algorithm is an algorithm that finds the shortest path between all the pairs of vertices in a weighted graph. This algorithm does not work where the sum of the edges in a cycle is negative (negative cycles). Only works with directed and undirected weighted cycles [8].

A weighted graph has in each edge a numerical value associated with it.

The algorithm starts with a matrix of $n * n$ dimensions, where n is the number of vertices. All the positions inside the matrix are indexed according to a row and a column position. The values in each position are the distances between i and j path, if they do not have a path, the cell is left as infinity.



$$A = \begin{array}{c|cccc|} & 1 & 2 & 3 & 4 \\ 1 & 0 & 3 & \infty & 2 \\ 2 & 4 & 0 & \infty & 3 \\ 3 & \infty & 6 & 0 & \infty \\ 4 & \infty & \infty & 2 & 0 \end{array}$$

Figure 5. Representation of creation first matrix Floyd-Warshall Algorithm.

Then, we create a second matrix with this first matrix created. We update the solution matrix by considering all vertices as an intermediate vertex. Picking all vertices one by one and updating their value as an intermediate vertex in the shortest path. As doing this, we are considering vertices $0, 1, 2,..., k - 1$ as intermediate vertices.

In each pair, there are two possible cases, 1) k is not an intermediate vertex in the path, so we keep the value as it was. 2) k is an intermediate vertex, so we update the value of A[i][j] as A[i][k] + A[k][j] if A[i][j] > A[i][k] + A[k][j].

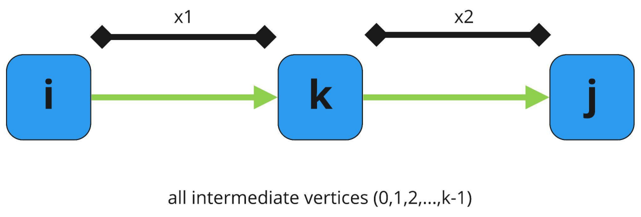


Figure 6: Representation of all the substructures created by the all-pairs shortest path.

The algorithm has three loops, each loop has a complexity constant. So, the time complexity is $O(n^3)$. The space complexity is $O(n^2)$[8][9].

3.2.4 Algorithm A*

This algorithm is a graph traversal and path search algorithm. What it does is help us find the quickest way to a goal overpassing some obstacles. Their mechanism consists in picking the node with the lowest "f" parameter, which is a value resultanting of the sum of other two parameters, 'g' and 'h'. 'g' is the movement cost from the starting point to a given cell, and 'h' is the movement cost from the current location to the destination point.

The complexity of this algorithm depends on the heuristic. In the worst case of an unbounded search space, the number of nodes expanded is exponential in the depth of the solution (the shortest path) d: O(bd), where b is the branching factor (the average number of successors per state) [11].

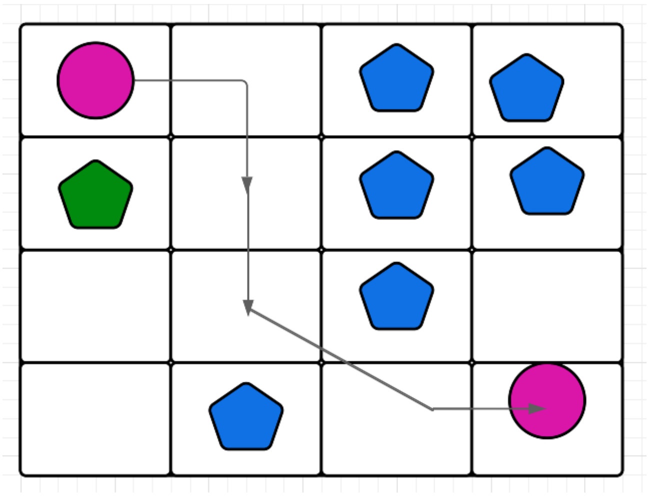


Figure 7: Visual description of the mechanic avoiding obstacles.

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 explained below, help us sort the information and represent it better in our graph, either the estimate of the amount of sexual harassment or the distance between the points.

[4] https://github.com/alejoriosm04/ST0245

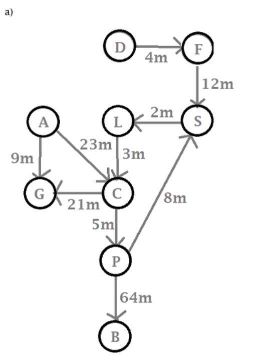
4.1.1 Adjacency list using a dictionary

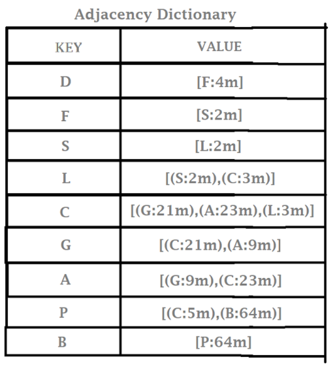
Since our graph contains a large number of nodes and connections between them, it could be a very messy and complex process to organize, an adjacency list data structure allows us to optimize our code. Plus to that, if we transform the list into a dictionary we make it easier to understand.

For the implementation of the adjacency list, we conserve a primary list, which is going to be two-dimensional. Inside that list are, in the first parameter, all the nodes in our graph, and for the second parameter, the nodes connected to it. We could see here that this list was going to be very large and difficult to access a certain position.

Replacing our initial list with a dictionary makes the problem more approachable. Instead of using our nodes as the first parameters on our list, we are going to use them as our keys. Those keys will let us access the values: a list with all the connections.

This allows us to compact our code and easily find the connections from a specific vertex or node. **The data structure is presented in Figure 8.**




Adjacency Dictionary

KEY	VALUE
D	[F:4m]
F	[S:2m]
S	[L:2m]
L	[(S:2m),(C:3m)]
C	[(G:21m),(A:23m),(L:3m)]
G	[(C:21m),(A:9m)]
A	[(G:9m),(C:23m)]
P	[(C:5m),(B:64m)]
B	[P:64m]

Figure 8: On the left we see an example of a graph and its representation as an adjacency list transformed as a dictionary.

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 Dijkstra's Algorithm

Our aim was to find the path with the shortest path and the lowest risk of sexual harassment, combining these two variables to find the best path. This algorithm allowed us to find this path on a weighted graph as was explained previously.

The algorithm receives three parameters; the graph as a data structure that represents the weighted graph with the paths, the starting point and the target point, both in coordinate format as a tuples.

We created a dictionary with the distances from the start to all the nodes. Then, the distance of the starting point was setted to zero. In the same way, a dictionary was created to store the previous node of each node. Finally, two lists were created, the first one will store the nodes that have been visited, and the second one has all the nodes that have not been visited, that is, all the keys of the dictionary, since the algorithm itself has not been started.

Subsequently, the algorithm actually starts working. While there are nodes that have not been visited, a cycle will iterate over the neighbors of the current node and will pop the node with the minimum distance value in the dictionary, if the neighbor is not in the visited list, (throughout the cycle, the current nodes will be removed from the unvisited list and added to the visited list) the distance from the start to the neighbor will be calculated taking into the account the two variables mentioned at the beginning, in this case, we created three different functions that calculate these variables and return a weight. The use of these types of weights are in the tests. Then, if this distance is less than the current distance, the distance of the neighbor will be set to the distance and the previous node of the neighbor will be set as the current node.

The main cycle will finish when the current node is the target or there are no more nodes unvisited. In summary, the dictionary with the previous nodes will store all the neighbors nodes with the most optimal distance between the start point and the target.

Finally, a list will save the complete path, adding the previous nodes starting in the target node and returning to the start point [12].

The algorithm is exemplified in Figure 9.

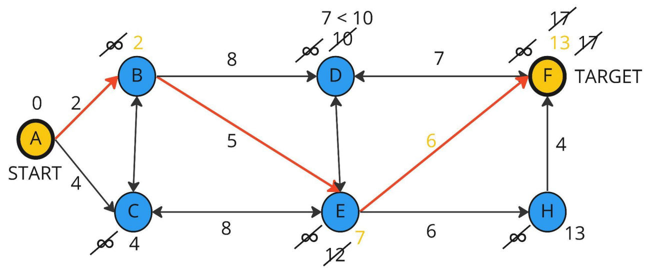


Figure 9: Calculation of a path that reduces both distance and risk of harassment.

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

The other two paths use the same algorithm as the first path, Dijkstra. However, the difference between all these paths is in the way in which the variable v is calculated. On the first path, we assign to each variable of v a weight and then add them, i.e, the distance and the harassment risk have an importance value, this will give us a total value for v, representing a total weight that will be noticed in the construction of the graph and the algorithm ($v = 30d + 500r$).

Consequently, with the second path, we wanted to analyze how this affects the final path if any of these variables has an importance value and are taken equally ($v = dr$). Finally, with the last path, we give the harassment risk more importance than any other formula used, creating an exponential formula where the growth depends totally on the harassment risk value ($v = d^{10r}$).

In summary, we wanted to see how the behavior of the algorithm is and how the final results are when we change these weights in the nodes. Later in the report, are the results of these, taking into account the final distance, the average harassment risk, and the execution time.

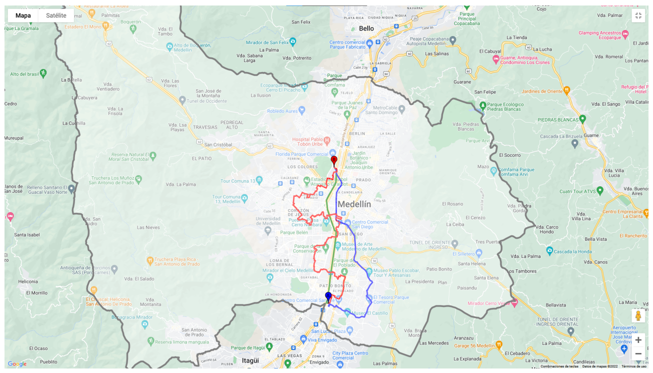


Figure 4: Map of the city of Medellín showing three pedestrian paths that reduce both the risk of sexual harassment and the distance in meters between the EAFIT University and the National University.

4.3 Algorithm complexity analysis

Dijkstra's algorithm was implemented with an adjacency list so its time complexity is O(V+E log V), where V is the number of nodes (represents the intersections) and E is the number of edges (represents the streets). In the worst case, the while loop would go past all the nodes for adding them to the priority queue, inside this loop, the algorithm passes through all of the node's neighbors meaning that the complexity of this part is O(V+E). And, adding or removing an element to a priority queue has a time complexity of O(log V), therefore, the time complexity of the algorithm in the worst case is O(V+E log V).

As the implementation of the algorithm we used an adjacency list, the memory complexity is O(V). In the worst case the memory complexity is O(V^2) when every node is connected to all the other nodes, however, it's impossible that a street in Medellin is connected to all the other streets, so the memory complexity used is O(V).

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

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

Data Structure	Complexity of memory
Adjacency List	O(V)

Table 2: Memory complexity of the adjacency list, where V is the number of nodes.

4.4 Algorithm design criteria

First of all, to improve the performance of the algorithm in terms of time and memory, we chose Adjacency Lists as a data structure to create the graph with its nodes and edges. This was the best option because with an Adjacency List we are able to access the information of each node in constant time $O(V)$, something required since many functions need to search for this information. Thus, if an Adjacency Matrix were used a great amount of memory and time would be needed to perform these actions in the worst cases $O(V * V)$.

On the other hand, the selected algorithm was Dijkstra because of its efficiency and speed in calculating the shortest path in a directed graph. However, to improve this efficiency, we added priority queues, to make sure that the algorithm is always exploring the nodes with the lowest value, avoiding unnecessary node exploring. In the same way, the algorithm checks if the current node pushed from the priority queue is the target to stop and break the loop, also avoiding unnecessary node exploring. In the development of the algorithm it was tried to implement Dijkstra recursively however, there is a lot of data, so the recursion leads to an exponential behavior that the memory can not afford.

Additionally, when the graph is created, we store all the necessary data of each node in a tuple as a key of the node in the Python dictionary. This improves the time accessing certain node data.

Finally, the map is displayed with an API of Google Maps. This option was selected, primarily, because of its facility implementation in the development environment. Secondly, this API provided us with an optimal graphing of the Medellin Polygon and each path created by the algorithm using official data of Google Maps. And finally, the performance of creating the HTML file with the JavaScript API is faster than other libraries of Python [13]. The only con of this option is that you need an official API Key provided by Google Cloud to use this product, otherwise, you will not be allowed to use this product.

In general terms, these options made possible the optimal development of the algorithm and obtaining these results.

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 University	National University	8165.85 m	0.62
EAFIT University	National University	13513.44 m	0.34
EAFIT University	National University	22153.3 m	0.73

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

5.2 Algorithm execution times

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

Calculation of v	Average run times (s)
$v = 30d + 500r$	0.712 s
$v = dr$	0.753 s
$v = d^{10r}$	0.901 s

Table 4: *Dijkstra Algorithm* execution times for each of the three calculator paths between EAFIT University and National University.

6. CONCLUSIONS

Analyzing the results of the algorithm tested with different origins and destinations, we can conclude that when choosing a path, this has either a low sexual harassment risk or a short distance, but it is difficult to find a path with the lowest value of these two parameters. The usefulness of this algorithm in the city is relevant, because it helps people (even more tourists) to be alert of any situation considering the path they are taking, since the shortest path is not equal to the path with the lowest risk.

When the harassment risk has a greater importance than the distance, generate a path with an exaggerated distance and a harassment risk insignificant. However, in the case when the distance and the harassment risk have equal importance, we obtained optimal results in both, distance and risk.

The time of execution of the algorithm is good in a real life situation, however it can be more efficient but it'll make a little difference. For a mobile or a web application, we would recommend all three paths, always showing the distance and the risk each path has.

In general, the project has many things to improve to create something more useful for people and the city. For example, the streets database of the city, the main aim of the project and the solution approach.

6.1 Future work

In the future we would like to continue working in the project to improve the graphs of the paths and making it more user-friendly, which involves web development, also, we would like to work in the calculation of the algorithm with a more accurate database of the sexual harassment risk in Medellin, taking into account more external factors of the city, since being such a subjective indicator it can be unpredictable, which involve statistics.

Finally, with much more time and freedom of work, this project could be improved in many ways and created an optimal final product.

ACKNOWLEDGEMENTS

First of all, this work was possible thanks to the help of the teacher assistants of Data Structures and Algorithms I, Isabel Mora, Julian Ramirez, Valeria Cardona, Gregorio Bermudez and Samuel Rico, who consistently helped with the implementation of the algorithms, giving explanations or advises, and were always willing to solve questions and concerns.

We are grateful for help with the implementation of the algorithm to Alejandro Restrepo for comments that greatly improved the code of this research.

We are grateful for help with technical support and inspiration in the implementation of the algorithm to Luis Torres for comments that greatly improved this manuscript.

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*.

Finally, the first three authors are grateful to their parents for supporting their education at EAFIT University.

REFERENCES

1. Ma, D., 2020. Preventing Sexual Harassment Through a Path Finding Algorithm Using Nearby Search. Available at: https://omdena.com/blog/path-finding-algorithm/.

2. Soni, S., Gauri Shankar, V., and Chaurasia, S., 2019. Route-The Safe: A Robust Model for Safest Route Prediction Using Crime and Accidental Data. Available at: https://www.researchgate.net/profile/Venkatesh-Gauri-Shankar/publication/338096313_Route-The_Safe_A_Robust_Model_for_Safest_Route_Prediction_Using_Crime_and_Accidental_Data/links/5dfddeb392851c83648deef2/Route-The-Safe-A-Robust-Model-for-Safest-Route-Prediction-Using-Crime-and-Accidental-Data.pdf.

3. Ali, M., Rishta, S., Ansari, L., Hashem, T. and Khan, A., 2015. *SafeStreet | Proceedings of the Seventh International Conference on Information and Communication Technologies and Development*. [online] ACM Other conferences. Available at: https://dl.acm.org/doi/pdf/10.1145/2737856.2737870.

4. OpenGenus IQ: Computing Expertise & Legacy. 2022. Bresenham Line Drawing Algorithm. [online] Available at: https://iq.opengenus.org/bresenham-line-drawining-algorithm/.

5. Programiz.com. n.d. *Dijkstra's Algorithm*. [online] Available at: https://www.programiz.com/dsa/dijkstra-algorithm.

6. GeeksforGeeks. n.d. *Dijsktra's algorithm*. [online] Available at: https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/.

7. Panurat Sutigoolabud, Peerawit Naprae, and Santi Phithakkitnukoon. 2019. *Safe street rangers: crowdsourcing approach for monitoring and reporting street safety.* [online] Available at: https://doi-org.ezproxy.eafit.edu.co/10.1145/3341162.3349317.

8. Programiz.com. n.d. *Floyd-Warshall Algorithm*. [online] Available at: https://www.programiz.com/dsa/floyd-warshall-algorithm.

9. GeeksforGeeks. n.d. *Floyd Warshall Algorithm | DP-16 - GeeksforGeeks*. [online] Available at: https://www.geeksforgeeks.org/floyd-warshall-algorithm-dp-16/.

10. Gutierrez, E., 2019. *El 90,1 por ciento de las mujeres no denuncia el acoso callejero*. [online] El Tiempo. Available at: https://www.eltiempo.com/colombia/medellin/el-90-1-por-ciento-de-las-mujeres-no-denuncia-el-acoso-callejero-en-medellin-355056.

11. GeeksforGeeks. (2022, 30 mayo). A* Search Algorithms. [online] Available at: https://www.geeksforgeeks.org/a-search-algorithm/.

12. ThinkX Academy, 2021. *[7.5] Dijkstra Shortest Path Algorithm in Python*. [video] Available at: https://youtu.be/OrJ004Wid4o.

13. Anon. 2015. Map tips: Speeding up page load times with the google maps javascript API. [online]. Available at: https://mapsplatform.googleblog.com/2015/09/map-tips-speeding-up-page-load-times.html.