ALGORITHMS TO FIND THE SHORTEST SAFE ROUTE TO PREVENT SEXUAL HARASSMENT

Luis Alejandro Baena Marín
Universidad Eafit
Colombia
labaenam@eafit.edu.co

Juan Camilo Bermúdez Colorado
Universidad Eafit
Colombia
jcbermudec@eafit.edu.co

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

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

ABSTRACT
Many people are concerned about street sexual harassment. This is a very important issue. When there is real danger, we think a lot about if we should go to a given place or if we shouldn't. Because of that, we'll try to develop an algorithm that gives people the most secure and shortest path to go to a given place. This carries lots of difficulties. For example, we must define very well the criteria to consider a certain place "secure". We must define where we are going to obtain our data to apply these criteria, and how we are going to model the criteria into a path finding. And, of course, how we are going to make a path finding. What is the algorithm you have proposed to solve the problem? What quantitative results have you obtained? What are the conclusions of this work? The abstract should be **at most 200 words**. (*In this semester, you should summarize here the execution times, and the results obtained with the three paths*).

Key words

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

1. INTRODUCTION

Sexual harassment is a problem that exists in all countries of the world, many actions are considered sexual harassment, which range from catcalling to rape. The main victims of this problem are women.

1.1. The problem

On the other hand, we all like to spend as little time as possible on transport. Also, People who are new to the city, have no idea about the safe routes. Though people rely on google maps for planning their routes; yet it only provides the shortest path & give no consideration for safety of the path

1.2 Solution

To find the shortest path with the least risk of harassment between an initial node and a final node given by a user, we first created a graph with the data from the file "calles_de_medellin_con_acoso.csv", some of the edges did not have weight, so we used the mean, and then we set the final weigth considering the risk and distance. Finally, using Dijkstra's algorithm, we managed to find the solution. We use Dijkstra's algorithm because there are not anegative weights, and also, since Dijkstra's algorithm uses BFS (Breadth-First Search), it has a better run time than other type of algorithms.

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 Incorporating a Safety Index into Pathfinding

It considers both traffic safety, defined as "the ratio of the rate of deceleration to avoid a crash to the maximum rate of deceleration available", and travel time. They managed to develop the methodology for the index by using roadside crash mechanisms but struggled with limitations such as not all crash types being considered. Other conditions such as vehicle type and pavement were considered. They applied it to the shortest path search algorithm. A review of it based on the findings of previous articles considered it to be a good enough approximation for road safety.

2.2 Preventing Sexual Harassment Through a Path

Bresenham's line algorithm is a line drawing algorithm that determines the points of an n-dimensional raster that should be selected in order to form a close approximation to a straight line between two points.

The general idea behind this algorithm is: given a starting endpoint of a line segment, the next grid point it traverses to get to the other endpoint is determined by evaluating where the line segment crosses relative to the midpoint (above or below) of the two possible grid points choices.

Zooming back out to our problem at hand here, we apply these grid coverage methods to compute the average of risk scores of steps for each route to determine the best.

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

The algorithm works this way: first the user enters the destination location. If there is only one route, that is the one we show to the user. On the other hand, if there is more than one route, we choose the safest route using the Knn regression model, and if there are several routes with the same score, the one with the shortest distance is chosen. That is the one that is shown to the user.

KNN regression is a non-parametric method that, in an intuitive manner, approximates the association between independent variables and the continuous outcome by averaging the observations in the same neighborhood.
The model suggests the safest route by selecting the route which has the lowest risk score. If more than one route has the lowest risk score it suggest the route which has the shortest distance.

2.4 Safety-aware routing for motorised tourists based on open data and VGI

In this article is also developed a safety index based on volunteered geographic information and governmental data, specially from police stations nearby. The primary data taken in account was of course crime data. They used it in LA and calculated the least dangerous path and the shortest one. The algorithm they used to find the paths with and without the index was the OPTICS algorithm. The algorithm orders points from the path as if they were linear. Then saves a relative distance for each point and calculates the density-based clusters in spatial data.

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] .

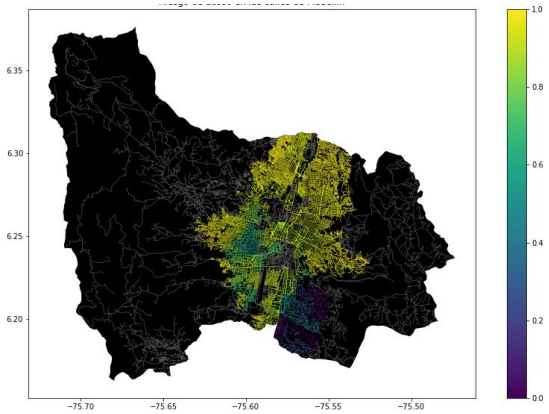


Figure 1. Risk of sexual harassment calculated as a linear combination of the fraction of households that feel unsafe and the fraction of households with income below one minimum wage, obtained from the 2017 Medellín Quality of Life Survey.

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. *(In this semester, examples of such algorithms are DFS, BFS, Dijkstra, A*, Bellman, Floyd, among others).*

3.2.1 Find the shortest path in a maze

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

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

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

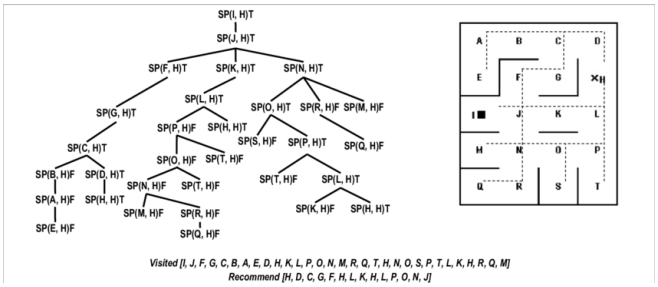
It explores all four possible paths and recursively validates if they lead to the destination, starting from the given cell. Then updates the minimum path length each time it reaches the destination cell. It backtracks when a path doesn't reach its destination or has explored all possible paths from the cell. Its time complexity will be high because all paths are traveled. The complexity of the algorith is O(M*N) where M and N are dimensions of the matrix.

1	1	1	1
0	1	1	0
0	0	1	**1**

3.2.2 Shortest Path in Maze using Backtracking

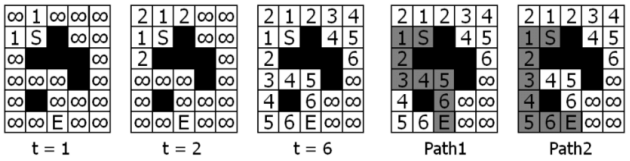
This algorithm also uses backtracking to explore all possibilities. It simply checks whether a move is valid or not. If it is valid, it keeps moving until stuck, otherwise backtrack to the former cell and explore other possible moves to the destination.

The complexity of the algorith is O(M*N) where M and N are dimensions of the matrix.



3.2.3 Shortest path in a maze – Lee Algorithm

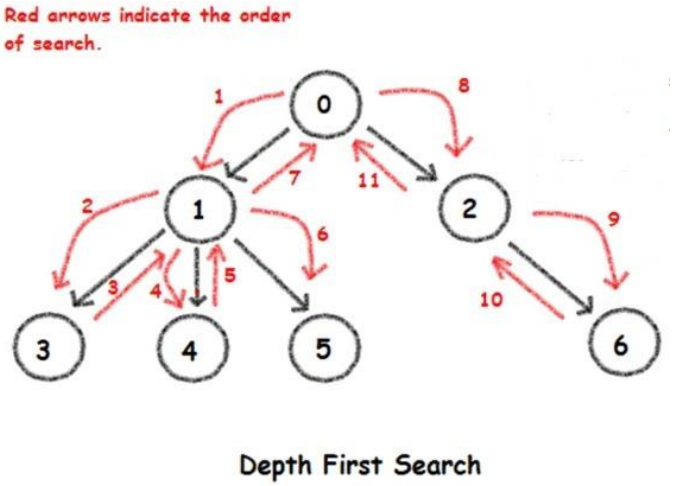
The Lee algorithm is one possible solution for maze routing problems based on Breadth–first search. It always gives an optimal solution, if one exists, but is slow and requires considerable memory. Following is the complete algorithm: Create an empty queue and enqueue the source cell having a distance 0 from the source (itself) and mark it as visited. The Loop till queue is empty. Dequeue the front node. If the popped node is the destination node, then return its distance. Otherwise, for each of four adjacent cells of the current cell, enqueue each valid cell with +1 distance and mark them as visited. And if all the queue nodes are processed, and the destination is not reached, then return false. The complexity of the algorith is O(M*N) where M and N are dimensions of the matrix.



3.2.4 DFS algorithm

Also called Depth First Search, the algorithm starts at the top node and goes as far as it can down a given branch (path), then backtracks until it finds an unexplored path, and then explores it. The algorithm does this until the entire graph has been explored.

The complexity for the DFS algorithm is O(V+E), here V is the number of vertices and E the edges.



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

We used a graph of the information provided in the file "calles_de_medellin_con_acoso.csv".

[4] https://github.com/jcbermudec/ST0245-001

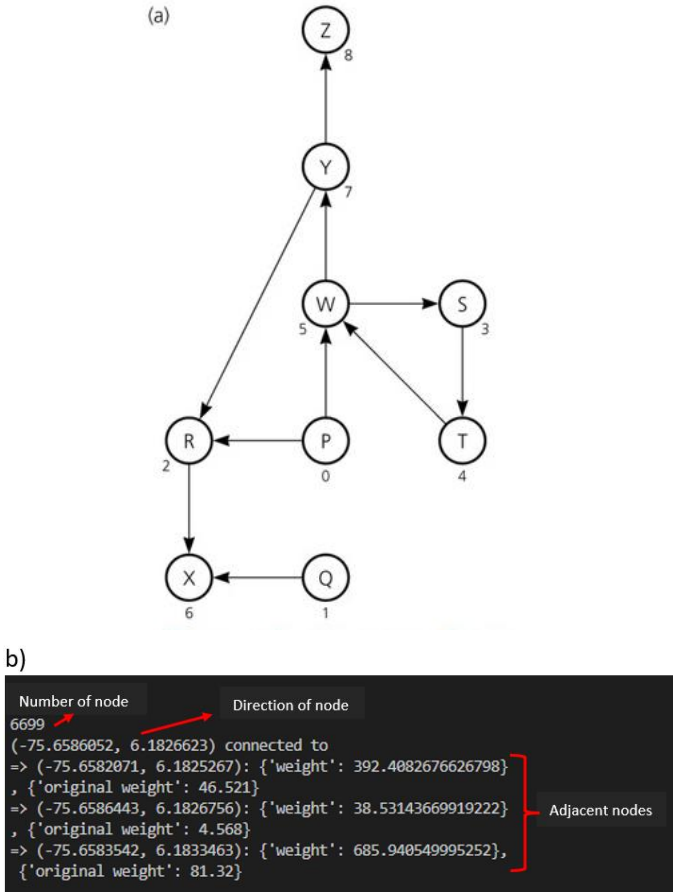


Figure 2: An example street map is presented in (a) and an example of one node representation as an adjacency list in (b).

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

Before creating the graph, we realized that some edges had no weight, so we put the mean of all the weights. To calculate the new weight (considering the risk of harassment), we used: new_weight = distance * (risk * 10) and then create the graph with the "networkx" library. We traversed the graph with dijsktra's algorithm which works as follows: Instantiate a dictionary that will eventually map vertices to their distance from the start vertex. Assign the start vertex a distance of 0 in a min heap. Assign every other vertex a distance of infinity in a min heap. Remove the vertex with the smallest distance from the min heap and set that to the current vertex. For the current vertex, consider all of its adjacent vertices and calculate the distance to them as (distance to the current vertex) + (edge weight of current vertex to adjacent vertex). If this new distance is less than the current distance, replace the current distance. Repeat 4 and 5 until the heap is empty. After the heap is empty, return the distances

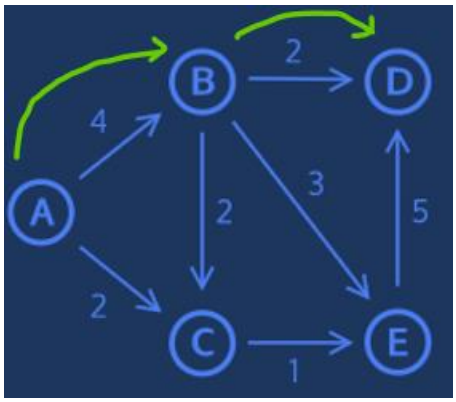


Figure 3: Example of calculating a path that reduces both distance and risk of harassment from A to D (please note that the weigth currently considers the risk and distance).

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

To obtain the other paths, we changed the formula for the new weight, in the last two more relevance was given to risk or distance.

The algorithm is exemplified in Figure 4.

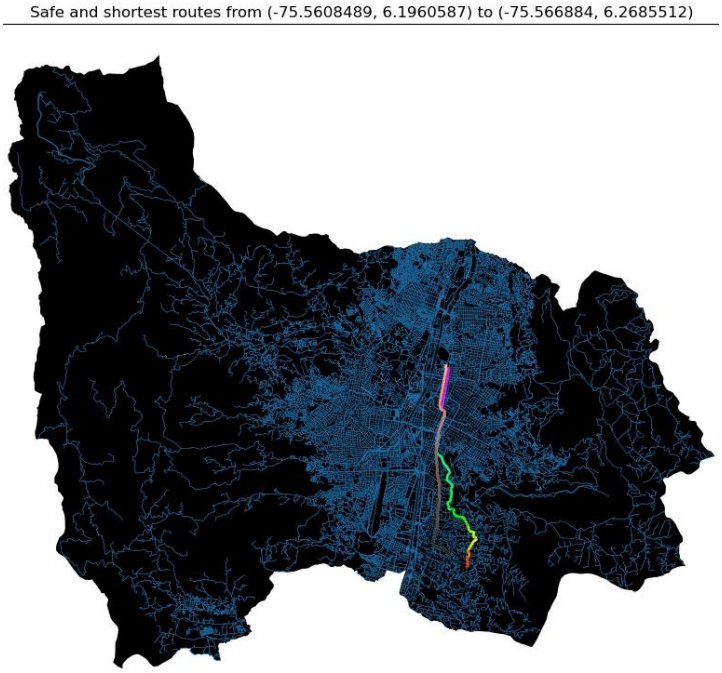
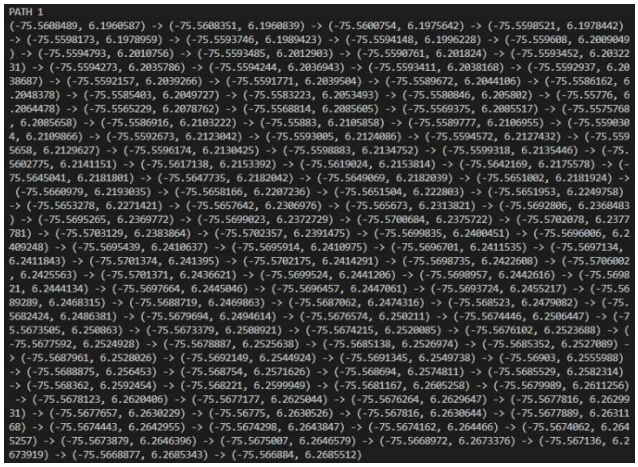
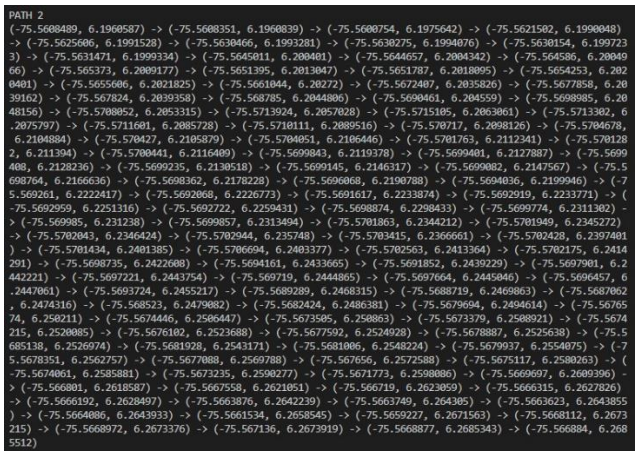
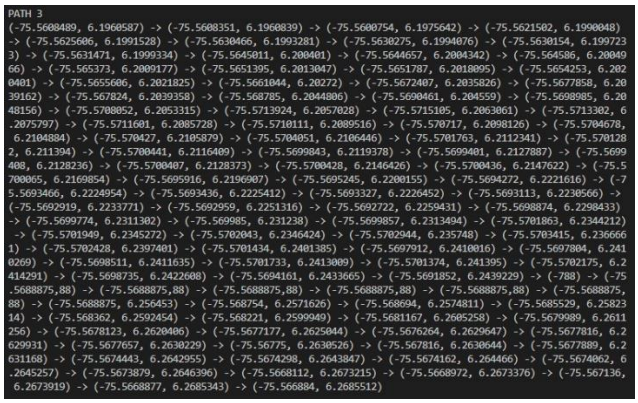


Figure 4: Map of the city of Medellín showing three pedestrian paths that reduce both the risk of sexual harassment and the distance in meters between the EAFIT University and the National University. (There are 3

different paths, but there are 2 that overlap because they are similar)







4.3 Algorithm complexity analysis

Just like breadth-first search and depth-first search, to search through an entire graph, in the worst case, we would go through all of the edges and all of the vertices resulting in a runtime of O(E + V). For Dijkstra's, we use a heap to keep track of all the distances. Searching through and updating a heap with V nodes takes O(log V) because in each layer of the heap, we reduce the number of nodes we are looking at by a factor of 2. In the worst case, we would update the heap every iteration. Since there are at most E + V iterations of Dijkstra's and it takes log V to update a min-heap in the worst case, then the runtime of Dijkstra's is O((E+V)*log V).

Algorithm	Time complexity
Dijkstra's algorithm	O((V+E)*log V)

Table 1: Time complexity of the name of your algorithm, where V represents the number of vertices and E represents the number of edges in the graph.

Data Structure	Complexity of memory
Graph (represented as an adjacency list)	Worst case: $O(V^2)$ Average case: O(V+E)
Dictionary	O(N)

Table 2: Memory complexity of a graph and a dictionary, where V represents the number of vertices and "N" is the number of elements in the dictionary.

4.4 Algorithm design criteria

We decided to implement a minimalist interface for the users to avoid any misunderstood. With this Menu the user can enter the coordenates of the nodes and the algorithm shows the best posible ways (3 paths).

We implemented Dijkstra's Algorithm due to its ability to find the shortest path from one node to every other node within the same graph data structure. This means, that rather than just finding the shortest path from the starting node to another specific node, the algorithm works to find the shortest path to every single reachable node.

The algorithm runs until all of the reachable nodes have been visited. Therefore, you would only need to run Dijkstra's algorithm once, and save the results to be used again and again without re-running the algorithm, which save a lot of time.

Finally, to create the graph we used a library called "networkx". And for drawing our paths, we implemented a library called "matplotlib" and "geopandas".

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	Unal	9832	0.572
Eafit	Unal	9401.977	0.6394
Eafit	Unal	9353.253	0.7045

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.

Calculate the execution time for the queries presented in Table 3.

Calculation of v	Average run times (s)
v = 1	0.10115 seconds
v = 2	0.13978 seconds
v = 3	0.09570 seconds

Table 4: *Algorithm* execution times *(Please write the name of the algorithm, e.g. DFS, BFS, A*)* for each of the three calculator paths between EAFIT and Universidad Nacional.

6. CONCLUSIONS

The second and third paths are quite similar, a fact that can be verified by seeing that they travel almost the same distance and have a similar risk, however, the first path travels more distance with considerably less risk than the other two paths. This is useful for the city mainly in the sense that it allows users to make a decision based on what they need and can prioritize more at the moment (safety or distance). Execution times are quite reasonable as none of the 3 paths take more than 0.14 seconds to draw. For a mobile or web application, I would recommend the third path, since it presents a balance between distance and risk with respect to the other two paths.

6.1 Future work

Perhaps it would be nice to plot the map and roads in a 3D environment, instead of a flat image like we did. It would also be interesting to carry out tests with linear regressions that seek to find the best optimization between distance and risk of harassment.

ACKNOWLEDGEMENTS

The first two authors of this research are financially supported by Sapiencia and Generación E, for which we thank both institutions.

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. He, Z. and Qin, X., 2016, Incorporating a Safety Index into Pathfinding, Transportation, Research Record 2659. 343-364.
2. Ma, D., 2020, Preventing Sexual Harassment Through a Path Finding Algorithm Using Nearby Search, Omdena.
3. Gauri, V. and Sandeep, C., 2019, Route-The Safe: A Robust Model for Safest Route Prediction Using Crime and Accidental Data, ResearchGate.
4. Keler, A. and Damascene, J., 2016, Safety-aware routing for motorised tourists based on open data and VGI, ResearchGate.
5. Techie Delight, 2022, Find the shortest path in a maze, Techie Delight.
6. Techie Delight, 2022, Shortest path in a maze – Lee Algorithm, Techie Delight.
7. Pencil Programmer, 2022, Shortest Path in Maze using Backtracking, Pencil Programmer.
8. Li, J. and Han, X., 2019, Revised Pulse Algorithm for Elementary Shortest Path Problem with Resource Constraints, Seventh International Symposium on Computing and Networking.