Preventing Street Harassment by Using Constrained Shortest Path Algorithms

Samuel Lopera
Universidad Eafit
Colombia
sloperat@eafit.edu.co

Santiago Lopez
Universidad Eafit
Colombia
Slopezc12@eafit.edu.co

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

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

ABSTRACT

Street harassment or catcalling is a problem that has been affecting our society for a very long time. It primarily consists of unwanted sexualized comments, gestures and actions forced on a stranger in a public place without their consent. This is problematic because it affects a person's sense of freedom and security in the streets. This is a problem that mainly affects women but it's not exclusive to them. It is also closely related to sexism because street harassment is usually the consequence of a sexist culture.

Keywords: Constrained shortest path, street sexual harassment, secure-path identification, crime prevention.

1. INTRODUCTION

In Colombia, women must face constant sexual harassment when they are in public. This makes women feel unsafe, it reduces their sense of freedom, and it affects their health and well-being. More specifically in Medellin, when women were asked about how they feel in public, 60% of them said that they felt that the city was unsafe for them, 50% said that they are afraid of public parks and other public spaces, and the list goes on. [1] This motivates us to bring a solution to help women feel safer in Medellin.

1.1. Problem To help reduce street harassment in Medellin, we are facing the problem of using an algorithm to find the shortest path possible from one point to another, without exceeding a certain risk of harassment, and finding the path with the least amount of harassment, without exceeding a certain distance.

1.2 Solution

The solution for our problem is tackled from the field of data structures, more specifically, directed graphs (or digraphs) implemented as hash tables. Using those we managed to implement pathfinding algorithms such as Dijkstra's algorithm, modifying them to take into account the harassment risk as a second constraint to the problem (the first being distance, as this is a shortest path problem), the nature of our election being divided between two reasons: firstly, the time complexity, some graph pathfinding algorithms can be slightly or marginally faster than recursion based shortest path algorithm; secondly, the nature of our dataset, the data provided for the project has the origin and destination provided as tuples, but, since tuples are immutable data structures (in python), we chose to interpret them as strings for a hash table implementation, where each origin relates to a list of destination

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 Safest Routing Algorithm for Evacuation Simulation in Case of Fire

In this project, the team developed an algorithm that could find the safest route in an evacuation in a fire. In the words of the authors "The algorithm shows a positive impact on the evacuation time and overall, on the safety during an evacuation simulation." [2]

2.2 Urban navigation beyond shortest route

The case of safe paths This work is very similar to ours. They utilized crime data to provide same urban navigation. Then, the goal is to find a short and low-risk path from one point to another. Their algorithm outputs a small set of paths that provide tradeoffs between distance and safety. [3] This one is interesting, because it is really similar to our project. .

2.3 Vehicle Routing Problem with elementary shortest path-based column generation

The team involved in this project considered some of the shortest path algorithms (Ex: Dijkstra's Algorithm, A* Algorithm, Floyd-Warshal's Algorithm, etc.) that exists as of today have an extremely high time complexity which makes them impractical for some existing problems. The problem they are trying to solve is related to the vehicle routing problem and they try to check which algorithm is more optimal for their problem. [4]

2.4 Shortest path problem in static navigation situations

This project is related to graph navigation, nonetheless it is still relevant for the problem at hand, considering graph related algorithms relate to quite a lot of problems, including the shortest route problem. [5]

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 whether 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].

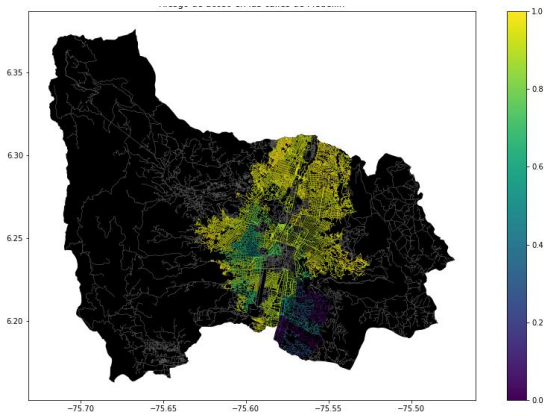


Figure 1. Risk of sexual harassment calculated as a lineal combination of the fraction of households that feel insecure and the fraction of households with income below one minimum wage, obtained from Life Quality Survey of Medellín, in 2017.

3.2.1 Dijkstra's Algorithm

It's an algorithm for finding the shortest path between two nodes in a weighted graph. It evaluates all the distance between nodes, and at the end it gives the least cost path from the start node to the goal node. It also finds the least cost path from the starting node to all the other nodes in the graph [6].

3.2.2 Bellman–Ford Algorithm

It gives the same result as Dijkstra's Algorithm because it also gives you the shortest path from one node to all other nodes in the graph. The difference between both algorithms is that Bellman-Ford's algorithm works with negative edge weights. [7]

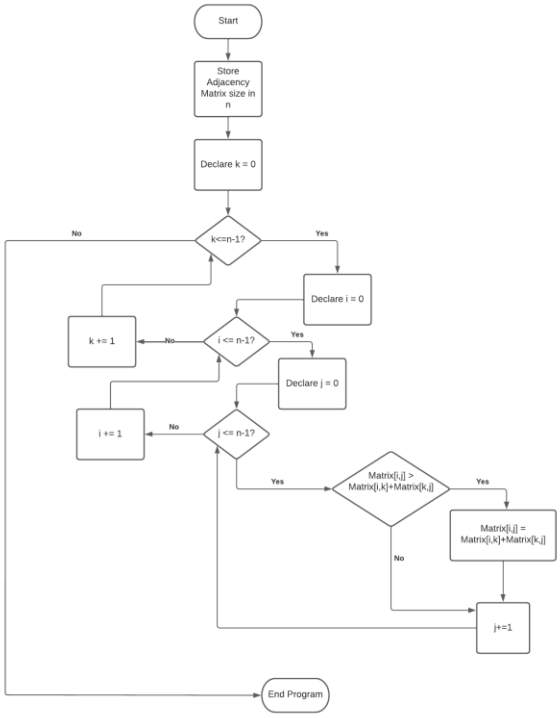
3.2.3 Floyd-Warshall Algorithm

The Floyd-Warshall algorithm is designed to find the shortest route from any node to any node in weighed directed graphs allowing negative weights, it iterates over the adjacency matrix of the graph showing the weights of each of the direction of one edge to another, if there an edge is not directed to another its considered "infinite", it iterates k times over the size of the adjacency matrix (which has a size of nxn) it iterates a second time i times over the size of the matrix, and a last j times over the size of the matrix, when it's on the last cycle it checks if the weight in the position (i, j) is bigger than the weights of the position (i,k) and (k,j) added; if the condition is true, it assigns the lowest weight of them; when it has finished iterating it returns a matrix where any position (i,j) is the shortest distance from the node i to the node j. It has a worst-case complexity of $O(2^{64})$, and for inputs bigger than 2^{64} tends to overflow. [8]

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

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

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



3.2.4 A* Algorithm

The A* algorithm is designed to be a complete algorithm, if there is a solution it will always find it; it analyzes considering the heuristic value of a node, and after analyzing all the adjacent nodes it returns the shortest possible route. It has a time complexity of $O(b^d)$ where b is the branching factor (Average number of successors per state) and d is the shortest route possible. [9]

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

In our project we decided to use hash tables and Graphs for motives already explained in the "Solution" subsection

4.1.1 Hash Tables:

In the words of an article from the Association for computing machinery (ACM) a hash table is defined as "fundamental data structures that optimally answer membership queries." [10] Where data is organized by a set of "keys" that relate to certain data stored within the data structure; for example, in a book you can organize the chapters in two main ways, by chapter number or by chapter title. Organizing by chapter number would be more akin to an array, organized by a proper numeric index; in the other hand, organizing by chapter title is closer to a hash table, where you know the information stored from the title, it being the key to access what is stored within. Luckily for the team, python has its own dictionary implementation without the need for any external libraries, so that is the one that will be used.

Hash Table is presented in Figure 2.

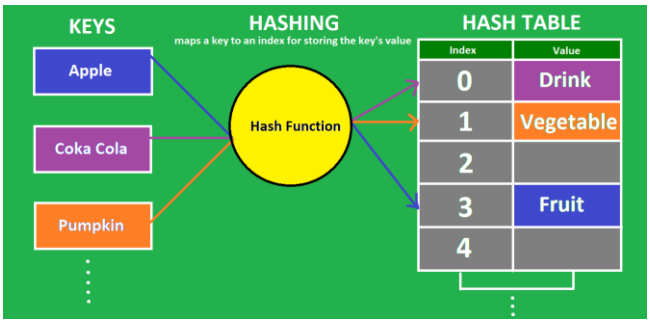


Figure 2: An example of a hash table is presented here, showing how it associates String values as indexes

4.1.2 Graphs

In a formal sense, Graphs are defined as a series of interconnected points, where a single point can be connected to multiple different points, including itself; speaking from a computer science perspective we can present Graphs in three main ways, as an adjacency matrix where the intersection represent which point is connected with each, for example, the point [1,2] in the matrix would represent the connection from the point 1 to the point 2, it is to note that the relations aren't always bidirectional, in a lot of cases two given points can be connected in only one direction, in the matrix, the connection is represented as a value; usually 0 or infinity if there is no connection, and any number different from 0 or infinity if they are connected, on that note, a value of 1 is used typically when there is no information about the connection, usually called weight.

Graphs can also be represented as lists or hash tables (where they are functionally the same), where each index contains a list with all the connections of a given node, the main difference being the implementation

[4]https://github.com/SamuelLoperaT/ST0245-001/tree/master/codigo

An example of a hash table implemented graph is presented in figure 3

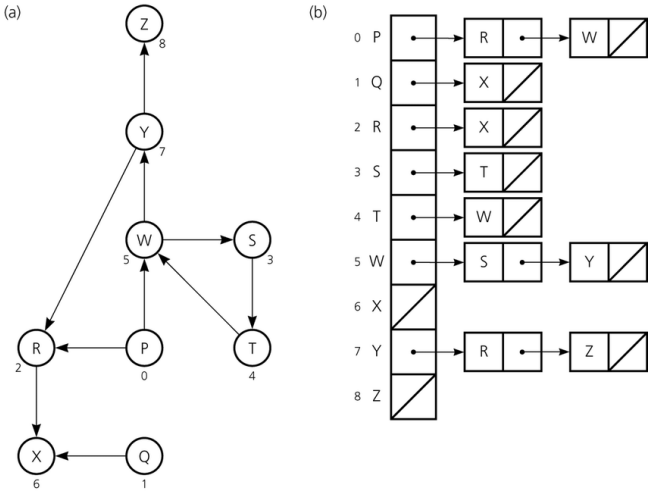


Figure 3: An example of a street map is presented in (a) and its representation as a hash table in (b).

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 algorithm

We used an slightly modified version of Dijkstra's algorithm for this project; it allows us to verify multiple paths, and prevent them of going over a certain "risk" threshold, Dijkstra's algorithm checks the relative distance from one node to all of it's neighbors, selecting the ones with the least distance with the origin node, then iterating to that node, it repeats itself until it reaches the destination, the only tweak we needed to make was to keep adding the harassment risks so it does not go up a certain threshold for it to give us the shortest constrained path.

Algorithm is exemplified in Figure 4.

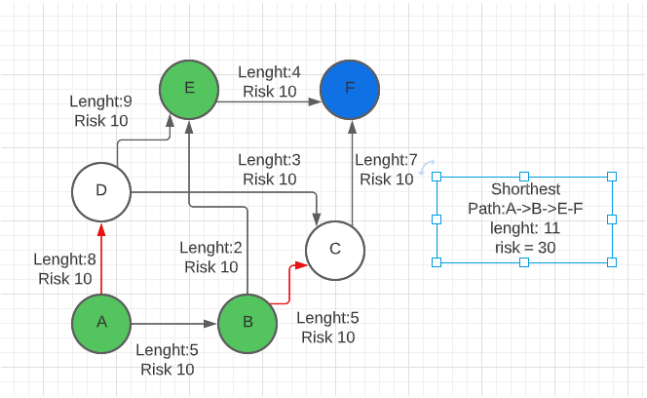


Figure 4: the algorithm iterates between nodes to reach its destination, discarding the ones with the most relative length, while it creates the path the algorithm verifies the risk sum does not exceed the given threshold so it can backtrack to check other lengthier paths with less risk

REFERENCES

1. Alcaldía de Medellín. Ciudades Y Espacios Públicos Seguros Para Las Mujeres Y Las Niñas, 2018. URL:https://www.medellin.gov.co/sicgem_files/adff8f26-2f7e-4308-af10-4b9351f35e6c.pdf

2. Shikhalev, D. Khabibulin, R. and Ulrich KemlohWagoum A. 2014. Development of a Safest Routing Algorithm for Evacuation Simulation in Case of Fire.Proceedings of the 6th International Conference onAgents and Artificial Intelligence, 685-690.

3. Galbrun, E. Pelechrinis, K. Terzi, E. 2016. Urban navigation beyond shortest route: The case of safe paths .Information Systems, Volume 57, 160-171.

4. Alain Chabrier,Vehicle Routing Problem with elementary shortest path-based column generation, Madrid,2005

5. Mariusz Dramski, Shortest path problem in static navigation situations in Metody Informatyki Stosowanej, Szczecin , 2011, 173-180

6. Jaakko. 2016. Dijkstra's algorithm explained. URL: https://www.youtube.com/watch?v=CL1byLngb5Q

7. Sambol, M. 2015. Bellman-Ford in 4 minutes — Theory. URL: https://www.youtube.com/watch?v=9PHkk0UavIM

8. Stefan Hougardy, The Floyd–Warshall algorithm on graphs with negative cycles,Bonn, 2010

9.Multiple Authors, A* search algorithm in Wikipedia: the free encyclopedia, https://en.wikipedia.org/wiki/A*_search_algorithm

10. **Charalampos Papamanthou, Roberto Tamassia, Nikos Tirandopoulos,2009, Authenticated Hash Tables on the ACM digital library** https://dl.acm.org/doi/abs/10.1145/1455770.1455826?casa_token=YIygAbboIWoAAAAA:ZD6R-0bdHxulcr0P1w0mYCY1pCHNgHYNumk59OWrHxmn6tPUFdW4ZrVgEbuq4fcnZ7sv-of28IeiPJQ