SHORTEST PATH FINDING ALGORITHM TO PREVENT STREET HARASSMENT

Santiago Neusa Ruiz
Universidad Eafit
Colombia
sneusar@eafit.edu.co

Manuela Castaño Franco
Universidad Eafit
Colombia
mcastanof1@eafit.edu.co

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

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

ABSTRACT
The streets are a vital space for the development of society, but the interactions that take place there are not always great. Sexual harassment is a fact, and it is our duty to fight it. This paper presents the creation of an algorithm aimed at solving the need to find routes that are short in distance, involving the least amount of street harassment possible. The project is contextualized in the present, while providing some background (similar algorithms) in order to show previously developed alternatives besides our support for the fight against sexual street harassment.

Key words

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

1. INTRODUCTION
Sexual harassment is a problem that has been present in our society for decades. We tend to think of solutions such as increasing education levels or increasing penalties or punishments for those who commit such an act, but science has a different approach. Different factors have led to develop a solution from technology, since many daily processes are involved by this, it can be a great tool to fight this problem. We propose an algorithm that takes into account the distance and the level of harassment in the streets of Medellín, in order to find the best route that a person can take when going from a point A to a point B. The proposed method does not leave aside the ideas presented at the beginning of this section, but complements them.

1.1. The problem
The problem we are facing is the creation of an algorithm that identifies the three shortest and safest paths, in order to reduce the crime and sexual harassment risk pedestrians face on a daily basis. Thereby, the likelihood of these encounters will decrease; this will let pedestrians, especially vulnerable groups like women and children, feel safe while walking towards their destination.

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 Urban Navigation Beyond Shortest Route: The Case Of Safe Paths

This study is labeled as an intelligent urban navigation work and it was focused on the cities of Chicago and Philadelphia, generating a bicriteria solution to the safest and shortest path problem. The two factors taken into consideration were the length and risk of the route. For the construction of the algorithm, data from OpenStreetMap (OSM) was exported and publicly available crime information was recollected.

The algorithm is recursive, and was based mostly on the Dijkstra solution. This algorithm proved a great performance, as shown in the results through efficiency and efficacy. Finally, the program displays not only one but rather a small subset of the best path options for the pedestrian to choose from. [1]

2.2 Proposing A Multi-Criteria Path Optimization Method In Order To Provide A Ubiquitous Pedestrian Wayfinding Service

This study presents a more complete method of calculating paths tailored to pedestrians. Put to test in Tehran, Iran, this method focused on context and user awareness to give the best path option. These two factors are then broken down into four criteria: length, safety, difficulty and attraction. The last three also have, inside each one of them, diverse categories that take into account the user's age, gender, infrastructure preference, as well the street's width and slope. The above, with the intention to deliver the best result personalized for each user.

This method made use of the Dijkstra algorithm. The four factors go through a mathematical model, where they are added and generate one final path option. [2]

2.3 SafeJourney: A Pedestrian Map Using Safety Annotation For Route Determination
This paper showcases the creation and implementation of a website to calculate the best route for students at Universiti Teknologi PETRONAS, a private university in Malaysia. The scope of the project was small, only focused on the commute students have to do inside the campus, e.g. from one block to another and so on.

The solution was mainly based on the Dijkstra algorithm. The website was built using XHTML, PHP and MySQL to display the best path highlighted on the university's campus map. It also shows step by step instructions through text and images. [3]

2.4 Shortest Path Algorithms for Pedestrian Navigation Systems

This work presents a penalty-based algorithm to calculate the shortest and most accessible path for people with reduced mobility, emphasizing mostly on wheelchair users.

The algorithm is based on the k shortest paths model (KSP), while also contemplating factors like the road's pavement state, its width, and whether it is a ramp or not. The 10 shortest paths are calculated and then the most accessible one is chosen to be delivered to the user. The method was tested in the city of Thessaloniki, in Greece, proving great performance in most of the conducted experiments. [4]

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 pminus the normalized CL. Figure 1 presents the calculated risk of bullying. The map is available on GitHub[3] .

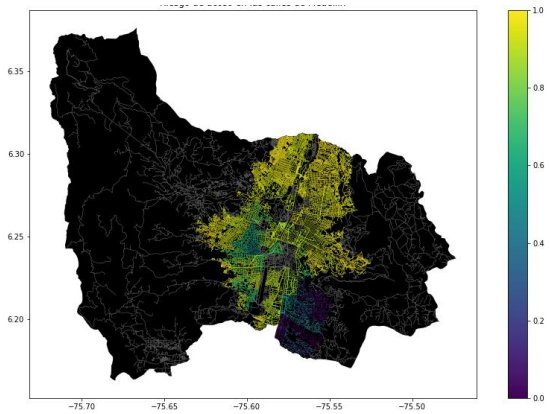

Figure 1. Risk of sexual harassment calculated as a linear combination of the fraction of households that feel unsafe and the fraction of households with income below one minimum wage, obtained from the 2017 Medellín Quality of Life Survey.

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 Depth First Search

The Depth First Search (DFS) is an algorithm for deeply searching or exploring graphs or trees. [5] The execution of the algorithm begins at the root node, examining each branch before backtracking. It uses a stack data structure to get the subsequent vertex, and to start a search, whenever a dead-end appears in any iteration. [6]

Because depth-first search constantly grows until the deepest node in the current branch of the search tree, we implement it using the LIFO queue, often known as a stack. According to Norvig and Russell [7]:

> "A LIFO queue means that the most recently generated node is chosen for expansion. This must be the deepest unexpanded node because it is one deeper than its parent — which, in turn, was the deepest unexpanded node when it was selected."

One notorious problem we may encounter with DFS is the possibility of having the target node next to the current node, but since DFS goes to the end of each branch, we cannot reach it until that branch is fully explored.

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

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

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

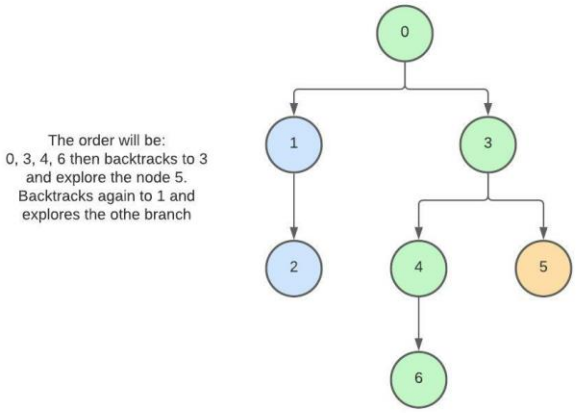

Figure 2. DFS Algorithm.

3.2.2 Breadth First Search

Breadth-first search (BFS) is an algorithm that is used for searching or exploring graphs or trees [8]. This algorithm selects a single node (initial or source point) in a graph and then visits all the nodes adjacent to the selected node. BFS accesses these nodes one by one. Once the algorithm visits and marks the starting node, it moves towards the nearest unvisited nodes and analyses them [9]. These iterations continue until all the nodes of the graph have been successfully visited and marked, or until the goal is reached.

BFS implements the FIFO queue (first in, first out) because when queried, returns the oldest element, based on the order they were inserted [8].

The BFS algorithm helps to solve the problem left by DFS.

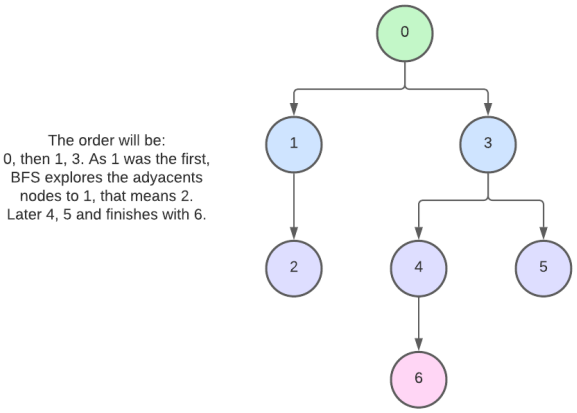

Figure 3. BFS Algorithm.

3.2.3 Dijkstra's Shortest Path Algorithm

Dijkstra's algorithm is one way to find the shortest route across a weighted graph from a starting node to a target node. This procedure builds a tree of all the shortest routes from the source, or first vertex, to every other point in the graph. By calculating the travel time to each linked node, Dijkstra's algorithm determines the shortest path. It then records the edge's distance in a list, placing the shortest at the top, and begins exploring other nodes in the order of that list. Just until the destination node is reached, this process stops [10].

This method differs from the others in that it provides the shortest distance, whereas DFS and BFS just provide an answer to whether a node is connected to another.

One significant problem with this algorithm is the fact that even if the shortest path is the better option, sometimes that's not the way you can get to the target node. This algorithm does not have any orientation variable [11].

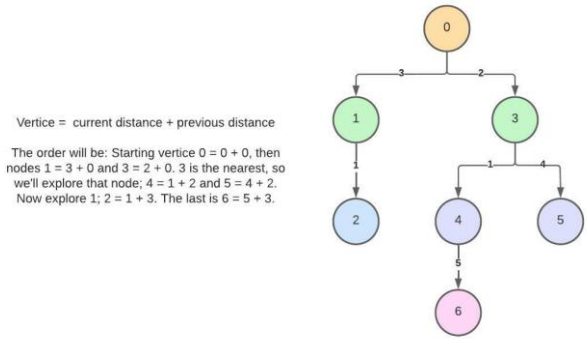

Figure 4. Dijkstra's algorithm.

3.2.4 A* (A Star algorithm)

A* (pronounced A Star) is a popular pathfinding and graph traversal technique. In order to make more optimum decisions, the A* algorithm adds heuristic to a standard graph-searching algorithm, essentially planning ahead at each step so a more optimal decision is made. Like Dijkstra, A* constructs a lowest-cost path tree from the start node to the destination node, but it differs from Dijkstra in the way it utilizes a function called f(n) for each node to estimate the overall cost of a path by adding the edge cost and a heuristic variable. In this instance, that outcome will be at the front of our queue [12].

The heuristic variable helps to avoid the problem Dijkstra has. With 2 variables in game the judgment is better.

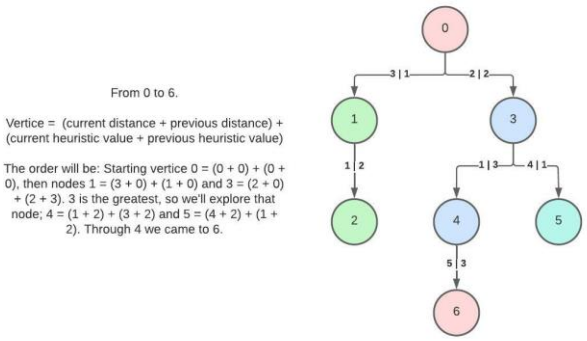

Figure 5. A Algorithm.*

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. Galbrun E., Pelechrinis K., and Terzi E. Urban navigation beyond shortest route: The case of safe paths. *ScienceDirect, 57.* 160-171. Retrieved August 15, 2022 from https://www.sciencedirect.com/science/article/pii/S0306437915001854

2. Sahelgozin, M., Sadeghi-Niaraki, A., and Dareshiri, S., Proposing a multi-criteria path optimization method in order to provide a ubiquitous pedestrian wayfinding service. in *International Conference on Sensors & Models in Remote Sensing & Photogrammetry*, (Kish Island, Iran, 2015), International Society for Photogrammetry and Remote Sensing. 639–644. Retrieved August 15, 2022 from https://www.int-arch-photogramm-remote-sens-spatial-inf-sci.net/XL-1-W5/639/2015/

3. K. H. Yew, T. T. Ha and S. D. S. J. Paua, SafeJourney: A pedestrian map using safety annotation for route determination. in *2010 International Symposium on Information Technology*, (2010), IEE, 1376-1381. Retrieved August 15, 2022 from https://www.researchgate.net/publication/224171685_SafeJourney_A_pedestrian_map_using_safety_annotation_for_route_determination

4. Koritsoglou, K., Tsoumanis, G., Patras, V., and Fudos, I, Shortest Path Algorithms for Pedestrian Navigation Systems 2022. *Information 13*, 6, 269. Retrieved August 15, 2022 from https://www.mdpi.com/2078-2489/13/6/269.

5. En.wikipedia.org. 2022. Depth-first search - Wikipedia. Retrieved August 13, 2022 from https://en.wikipedia.org/wiki/Depth-first_search

6. Walker, A., 2022. BFS Vs. DFS: Know the Difference with Example. Guru99. Retrieved August 13, 2022 from https://www.guru99.com/difference-between-bfs-and-dfs.html

7. Norving, P. and Russell, S. Artificial Intelligence, A Modern Approach, Thirth Edition. Pearson Education, Inc., New Jersey, 2010.

8. Khan Academy. 2022. The breadth-first search algorithm (BFS). Khan Academy. Retrieved August 13, 2022 from https://www.khanacademy.org/computing/computer-science/algorithms/breadth-first-search/a/the-breadth-first-search-algorithm

9. Sehgal, A., 2022. Breadth First Search | BFS Algorithm - Scaler Topics. Retrieved August 13, 2022 from https://www.scaler.com/topics/data-structures/breadth-first-search/

10. Stack Abuse. 2022. Graphs in Python - Theory and Implementation. Retrieved August 14, 2022 from https://stackabuse.com/courses/graphs-in-python-theory-and-implementation/lessons/dijkstras-algorithm

11. Bullinaria, J. Lecture Notes for Data Structures and Algorithms. School of Computer Science University of Birmingham, Birmingham, 2019.

12. Simplilearn. 2022. A* Algorithm Concepts and Implementation. Retrieved August 14, 2022 from https://www.simplilearn.com/tutorials/artificial-intelligence-tutorial/a-star-algorithm