BUSQUEDA HACIA LA VISUALIZACION DE LA MEJOR RUTA

Mateo Ramírez Rubio
Universidad Eafit
Colombia
mramirezr4@eafit.edu.co

- Jorge Gutiérrez Toro
Universidad Eafit
Colombia
jdgutierr2@eafit.edu.co

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

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

RESUMEN
El principal problema que se nos plantea en el proyecto es acerca de la insuficiencia de caminos cortos con una cantidad considerablemente baja de acoso sexual. Debido a esto se hace de bastante importancia, al ser algo tan requerido del día a día y contando con tan bajas posibilidades. Algunos problemas relacionados a este son casos donde se presentan índices de asaltos y robos donde se pueden implementar en las rutas de acoso.

Se ha utilizado un algoritmo computacional, se han obtenido diversas rutas con multitudes de resultados.

En este trabajo/Proyecto que se está desarrollando se ha ido trabajando y utilizando diferentes funciones y opciones de código para poder ejemplificar y dar respuesta a diferentes problemas que han ido presentado a lo largo del proyecto, se han invertido varias horas para dar consigo los resultados más efectivos hacia el camino de menor riesgo yendo a su vez con un camino de una distancia considerablemente más corta; en tanto en este trabajo se han albergado diferentes métodos y procesos para llegar a los mejores resultados.

1. INTRODUCCIÓN

La principal razón por la que estamos dando respuesta de un camino más corto con un riesgo menor al acoso sexual es debido a las situaciones que solemos vivir hoy en día (como casos de violaciones y secuestros que se suelen ver en nuestra ciudad y país, en zonas y barrios en específico, además no solamente vivimos esta situación en nuestro país si no en todo el mundo, por ende, hemos dispuesto a darle una solución a este problema.

1.1. Problema

El problema consiste principalmente en la dificultad de encontrar caminos seguros y con una distancia relativamente corta para llegar a un destino, esto afecta a la sociedad principalmente a los ciudadanos del día a día que se movilizan sin ningún vehículo en concreto, debido a estas razones es muy importante darle la solución adecuada para que todas aquellas personas que son afectadas por este problema o aquellas que puedan ser afectadas en un futuro puedan evitar este problema.

1.2 Solución

Para solucionar el problema anteriormente presentado se escogió el algoritmo de Dijkstra, ya que este es un algoritmo eficiente de complejidad $O(V^2)$ que sirve para encontrar el camino más corto desde un vértice origen hasta todos los demás vértices del grafo. Para esta solución se hace que el algoritmo arroje solo la ruta más corta desde un origen dado hasta un destino.

1.3 Estructura del artículo

A continuación, en la Sección 2, presentamos trabajos relacionados con el problema. Posteriormente, en la Sección 3, presentamos los conjuntos de datos y los métodos utilizados en esta investigación. En la Sección 4, presentamos el diseño del algoritmo. Después, en la Sección 5, presentamos los resultados. Finalmente, en la Sección 6, discutimos los resultados y proponemos algunas direcciones de trabajo futuro.

2. TRABAJOS RELACIONADOS

A continuación, explicamos cuatro trabajos relacionados con la búsqueda de caminos para prevenir el acoso sexual callejero y la delincuencia en general.

• **Be-safe travel**, una aplicación geográfica basada en la web para explorar una ruta segura en un área.
Route Planners System o sistema de planificación de rutas como Google Maps solo considera la distancia más corta en el cálculo de la ruta óptima. La selección de la ruta óptima en este estudio no solo considera la distancia más corta, sino que también involucra otros factores, a saber, el nivel de seguridad. Esta investigación considera la necesidad de una aplicación que recomiende la vía más segura para ser transitada por los pasajeros del vehículo mientras transitan por una zona. Esta investigación propone Be-Safe Travel, una aplicación basada en la web que utiliza la API de Google a la que pueden acceder las personas a las que les gusta conducir en un área, pero que aún no conocen los caminos que están a salvo de la delincuencia.
• **Sketch Factor**, la aplicación que permite a los usuarios informar sobre sus experiencias para crear un resumen de ciertos vecindarios para otros usuarios.
La aplicación permite crear reportes de experiencias sospechosas que han sufrido los usuarios, con el fin de crear zonas o áreas peligrosas para que sean evitadas por los usuarios.
• **OMDENA**, previene el acoso sexual a través de un algoritmo de búsqueda de ruta que facilita al usuario la búsqueda de una ruta corta y a la misma vez una ruta más segura.

Dependiendo de la distancia o la ruta que se seleccione esta va aumentando o disminuyendo en índice de acoso.
• **Aryan Guptaa, Bhavye Khetan**, los delitos aumentan día a día; por lo tanto, la seguridad se está convirtiendo en una preocupación importante para las personas de hoy. Incluso mientras viaja, las personas deben ser conscientes y elegir la ruta más segura para viajar. Las personas que son nuevas en la ciudad no tienen idea de las rutas seguras. Aunque las personas confían en los mapas de Google para planificar sus rutas; sin embargo, solo proporciona el camino más corto y no tiene en cuenta la seguridad del camino. Aunque existen otras aplicaciones de planificación de rutas que proporcionan la ruta más segura, estas no tienen en cuenta todos los factores que explican la seguridad de la ruta. Aparte de otras aplicaciones de navegación, este documento describe un método innovador para encontrar la ruta más segura con el mayor valor de seguridad. También nos damos cuenta de que la ruta más segura puede resultar excesivamente larga y lenta, por lo que, para que sea práctica, la incorporamos de manera efectiva junto con el parámetro de distancia, haciéndola útil en el mundo real.

3. MATERIALES Y MÉTODOS

En esta sección, explicamos cómo se recogieron y procesaron los datos y, después, diferentes alternativas de algoritmos del camino más corto restringido para abordar el acoso sexual callejero.

3.1 Recogida y tratamiento de datos

El mapa de Medellín se obtuvo de Open Street Maps (OSM) [1]y se descargó utilizando la API[2] OSMnx de Python. La (i) longitud de cada segmento, en metros; (2) la indicación de si el segmento es de un solo sentido o no, y (3) las representaciones binarias conocidas de las geometrías se obtuvieron de los metadatos proporcionados por OSM.

Para este proyecto, se calculó la combinación lineal (CL) que captura la máxima varianza entre (i) la fracción de hogares que se sienten inseguros y (ii) la fracción de hogares con ingresos inferiores a un salario mínimo. Estos datos se obtuvieron de la encuesta de calidad de vida de Medellín, de 2017. La CL se normalizó, utilizando el máximo y el mínimo, para obtener valores entre 0 y 1. La CL se obtuvo mediante el análisis de componentes principales. El riesgo de acoso se define como uno menos la CL normalizada. La Figura 1 presenta el riesgo de acoso calculado. El mapa está disponible en GitHub[3].

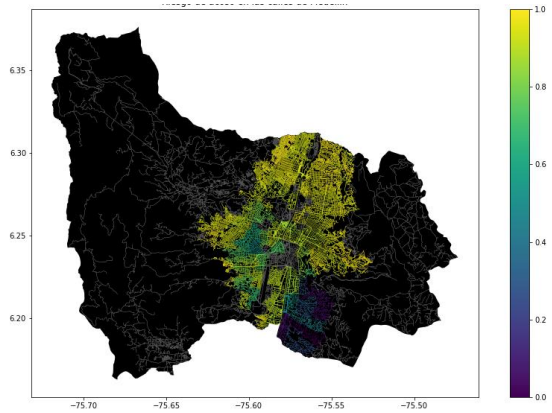

Figura 1. Riesgo de acoso sexual calculado como una combinación lineal de la fracción de hogares que se sienten inseguros y la fracción de hogares con ingresos inferiores a un salario mínimo, obtenida de la Encuesta de Calidad de Vida de Medellín, de 2017.

3.2 Alternativas de camino más corto con restricciones

A continuación, presentamos diferentes algoritmos utilizados para el camino más corto restringido. *(En este semestre, ejemplos de dichos algoritmos son DFS, BFS, una versión modificada de Dijkstra, una versión modificada de A*, entre otros).*

• Búsqueda bidireccional

La idea de la búsqueda bidireccional es ejecutar dos búsquedas simultáneas: una hacia delante desde el estado inicial y la otra hacia atrás desde el objetivo (Meta), parando cuando las dos búsquedas se encuentren en el centro. La motivación es para reducir u optimizar el tiempo que se tarda en realizar las búsquedas de amplitud y profundidad y por eso se creó esta búsqueda con la premisa de que b d/2 + bd/2 mucho menor que b d . Donde b es el número máximo de ramificaciones y d la profundidad que tiene el árbol.

• Bellman Ford

El algoritmo de Bellman-Ford determina la ruta más corta desde un nodo origen hacia los demás nodos para ello es requerido como entrada un grafo cuyas aristas posean pesos. La diferencia de este algoritmo con los demás es que los pesos pueden tener valores negativos ya que Bellman-Ford me permite detectar la existencia de un ciclo negativo.

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

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

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

• Floyd\u2013Warshall

El algoritmo de Floyd\u2013Warshall se usa para encontrar las rutas más cortas entre todos los pares de puntos en un gráfico, donde cada borde del gráfico tiene un peso que es positivo o negativo. La mayor ventaja de usar este algoritmo es que todas las distancias más cortas entre cualquiera de los puntos se pueden calcular en O(V3), donde V es el número de puntos en un gráfico.

• Pencil Programmer, Shortest path in maze usin Backtring.

El algoritmo de Pencil Programmer se usa para encontrar fácilmente el camino más corto en el laberinto usando el algoritmo de retroceso. La idea es seguir moviéndose a través de una ruta válida hasta que se atasque, de lo contrario, retroceda hasta la última celda atravesada y explore otras rutas posibles hacia el destino. Para cada celda, los siguientes 4 movimientos son posibles.

4. DISEÑO E IMPLEMENTACIÓN DEL ALGORITMO

A continuación, explicamos las estructuras de datos y los algoritmos utilizados en este trabajo. Las implementaciones de las estructuras de datos y los algoritmos están disponibles en GitHub[4].

4.1 Estructuras de datos

La estructura de datos implementada para realizar el algoritmo escogido es la lista de adyacencia utilizando un diccionario. Primero introducimos cada vértice (origen de una calle) como llave en un diccionario y luego iteramos sobre este introduciéndole a cada vértice su respectivo destino (cuando un origen tiene varios destinos, estos se anexan a la misma llave), distancia y riesgo de acoso.

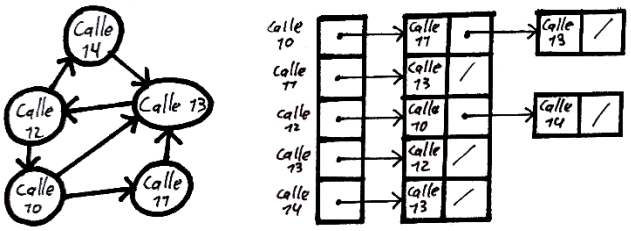

Figura 2: Un ejemplo de mapa de calles se presenta en la parte izquierda y su representación como lista de adyacencia en la posición derecha.

4.2 Algoritmos

En este trabajo, proponemos un algoritmo para la solución de los problemas del camino más corto restringido. El primer problema para resolver es encontrar la ruta más corta, en metros, para ir de un punto A a un punto B, sin sobrepasar un riesgo ponderado de acoso. El segundo problema para solucionar, similar al anterior, es hallar el camino con el menor riesgo promedio ponderado de acoso, sin sobrepasar una distancia, en metros.

El algoritmo escogido para solucionar dichos problemas es el algoritmo de Dijkstra.

Este algoritmo consta de explorar entre todos los caminos posibles entre un origen dado y un destino, tomando en cuenta el peso de cada camino, el algoritmo coge el menor peso que en este caso sería la distancia entre cada calle o el riesgo ponderado de aquella calle[5].

La complejidad del algoritmo de Dijkstra es: $O(V^2)$, donde V son los vértices del grafo dado.

Debemos de tener en cuenta que el peso de las aristas del grafo que le pasaremos al algoritmo no debe ser negativo (si ese es el caso se podría optar por una opción que acepta pesos negativos como lo es el algoritmo de Bellman Ford**).**

Otro par de requisitos importantes que debe cumplir el grafo dado para hacer uso del algoritmo de Dijkstra son:

- El grafo dado puede ser tanto dirigido como no dirigido.
- El grafo dado debe ser ponderado.

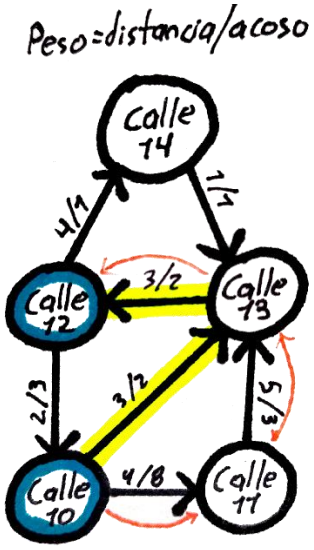

Figura 3: Resolución del problema del camino más corto restringido con el algoritmo de Dijkstra.

4.4 Análisis de la complejidad de los algoritmos

Como anteriormente mencionamos, la complejidad del algoritmo de Dijkstra es de $O(V^2)$ esto se debe a que en el algoritmo hay dos ciclos anidados, y también por hacer uso

[4] https://github.com/MateoRamirezRubio1/ST0245-002

[5]https://upload.wikimedia.org/wikipedia/commons/thumb/5/57/Dijkstra_Animation.gif/270px-Dijkstra_Animation.gif

de matriz de adyacencia como creación del grafo. Esta complejidad se puede mejorar realizando el grafo no por matiz de adyacencia sino con lista de adyacencia, pero esto cambiaria un poco el algoritmo por dentro, dando así con esto una nueva complejidad de O(ElogV), donde V es el número de vértices y E el número total de aristas.

Algo curioso de la complejidad del algoritmo utilizando lista de adyacencia es que también se puede denotar como O(VElog V) donde V es el número de vértices E el número total de aristas asociadas a un solo nodo. Esto se debe a que, como ejemplo, cambiando E por N, la complejidad misma de E es E = O(VN) dando esto O(VNlog V) = O(VElog V). Sin embargo, la forma O(ElogV) es una estimación más ajustada.

Algoritmo	Complejidad temporal
Dijkstra (matriz adyacencia)	$O(V^2)$
Dijkstra (lista adyacencia)	O(ElogV)

Tabla 1: Complejidad temporal del algoritmo de Dijkstra, donde V es el número de vértices y E el número total de aristas.

Estructura de datos	Complejidad de la memoria
Tabla Hash	O(V)

Tabla 2: Complejidad de memoria del nombre de la estructura de datos que utiliza su algoritmo, donde V es el número de elementos almacenados en la tabla.

4.5 Criterios de diseño del algoritmo

La mejor opción fue la implementación del algoritmo de Dijkstra por medio de la librería NetworkX, haciendo esta el trabajo más sencillo y optimizado con menos líneas de código. Este algoritmo fue el más apropiado para la ocasión ya que es el que tiene una menor complejidad y también considerando que ningún peso de las aristas del grafo es negativo.

Igualmente, aunque también es una buena opción y es bastante similar al algoritmo de Dijkstra, es el algoritmo de Bellman Ford.

Finalmente nos decidimos por el algoritmo de Dijkstra ya que este a comparación con el de Bellman Ford le gana en complejidad de tiempo (siendo esto prioritario en el proyecto), puesto que la complejidad de este último nombrado es de $E = O(V^2)$ siendo la complejidad total del tiempo $O(V^3)$ donde V son los vértices y E las aristas.

También como a punto a favor de confianza para el algoritmo de Dijkstra, será implementado por Google en Google Maps[6] .

5. RESULTADOS

En esta sección, presentamos algunos resultados cuantitativos sobre el camino más corto y el camino con menor riesgo.

5.1.1 Resultados del camino más corto

A continuación, presentamos los resultados obtenidos para el camino más corto, sin superar un riesgo medio ponderado de acoso *r,* en la Tabla 3.

Origen	**Destino**	**Distancia más corta**	**Sin exceder *r***
Universidad EAFIT	Universidad de Medellín	6142.57	0.64
Universidad de Antioquia	Universidad Nacional	815.44	0.76
Universidad Nacional	Universidad Luis Amigó	1469.18	0.76

Tabla 3. Distancias más cortas sin superar un riesgo de acoso medio ponderado *r*.

5.1.2 Resultados de menor riesgo de acoso

A continuación, presentamos los resultados obtenidos para el trayecto con menor riesgo de acoso medio ponderado, sin superar una distancia *d,* en la Tabla 4.

Origen	**Destino**	**Acoso más bajo**	**Sin exceder *d***
Universidad EAFIT	Universidad de Medellín	48.39	54.844
Universidad de Antioquia	Universidad Nacional	6.83	54.36
Universidad Nacional	Universidad Luis Amigó	7.58	97.95

Tabla 3. Menor riesgo de acoso ponderado sin superar una distancia *d* (en metros).

[6] https://dijkstranewsco.wordpress.com/2018/05/17/google-maps-aplicara-el-algoritmo-de-dijkstra-para-completar-su-mapa-con-vias-terciarias/

5.2 Tiempos de ejecución del algoritmo

En la Tabla 4, explicamos la relación de los tiempos medios de ejecución de las consultas presentadas en la Tabla 3.

	Tiempos medios de ejecución (s)
Universidad EAFIT a Universidad de Medellín	01.20 s
De la Universidad de Antioquia a la Universidad Nacional	01.13 s
De la Universidad Nacional a la Universidad Luis Amigó	01 s

Tabla 4: Tiempos de ejecución del algoritmo de Dijkstra para las consultas presentadas en la Tabla 3.

6. CONCLUSIONES

Es interesante darnos cuenta de como cambia drásticamente la mayoría de los trayectos entre la opción del camino más corto y el camino con menor riesgo de acoso, siendo esta una buena función para la ciudad que se podría aprovechar con una buena implementación donde el usuario deberá tomar una decisión importante dependiendo de su necesidad en el momento.

Hay diferentes formas de hacer uso del algoritmo de Dijkstra para mejorar sus tiempos de ejecución. En este proyecto se implementó dicho algoritmo mediante la Liberia NetworkX, ayudando ésta bastante a la optimización de código.

6.1 Trabajos futuros

Hay varias ideas para mejorar el proyecto en un futuro, algunas de estas pueden ser realizar una aplicación móvil como también una página web, implementar un sistema de GPS para después de dado el camino si se quiere más corto o con menor riesgo de acoso, guie al usuario en su trayecto y muestre como se puede encontrar el tráfico en el momento, como también el tiempo total que se tardaría en llegar al destino.

Igualmente, se podría actualizar los datos tomados para la implementación del algoritmo y ayudándose de la ciencia de datos implementar un modelo predictivo conectando las noticias del día a día relacionadas con el acoso y tal vez construcciones de vías nuevas o desviaciones o cierres de estas, con los datos obtenidos tanto nuevos como los "antiguos" y así que se mantengan estos datos actualizados automáticamente sin acción humana para una mejor precisión a la hora de la ejecución para obtener las mejores rutas según las preferencias del usuario.

AGRADECIMIENTOS

Esta investigación ha sido apoyada por el curso de Estructuras de Datos y Algoritmos I de la Universidad EAFIT a quien agradecemos estas oportunidades de aprendizaje mutuo para un mejor desarrollo como profesional y como persona.

Los autores agradecen al profesor Juan Carlos Duque, de la Universidad EAFIT, por facilitar los datos de la Encuesta de Calidad de Vida de Medellín, de 2017, procesados en un *Shapefile*.

REFERENCIAS

1. 1.Charles E. Noon y F. Benjamin Zhan. Researchgate. *https://www.researchgate.net/profile/Charles-Noon/publication/220412981_Shortest_Path_Algorithms_An_Evaluation_Using_Real_Road_Networks/links/0c96053c7aeeede964000000/Shortest-Path-Algorithms-An-Evaluation-Using-Real-Road-Networks.pdf*
2. Hackerearth. *https://www.hackerearth.com/practice/algorithms/graphs/shortest-path-algorithms/tutorial/*
3. Geeks for Geeks. *https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/*
4. Geeks for Geeks. *https://www.geeksforgeeks.org/generate-graph-using-dictionary-python/*
5. S. E. Drayfus. *https://apps.dtic.mil/sti/pdfs/AD0693567.pdf*
6. NetworkX. *https://networkx.org/documentation/stable/reference/algorithms/shortest_paths.html*