Algoritmos Para La Prevención Del Acoso Callejero

Brayan Zuluaga Giraldo
Universidad Eafit
Colombia
bdzuluagag@eafit.edu.co

Samuel Rendón Trujillo
Universidad Eafit
Colombia
srendont@gmail.com

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

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

RESUMEN
El problema a tratar en el presente proyecto es el acoso callejero en la ciudad de Medellín y su periferia, así como las posibles formas de evitar este por medio del uso de software. Esto se debe a que la ciudad cuenta con unos grandes índices de acoso callejero, problema el cual es una clara vulneración a la integridad personal de los individuos afectados. De esto, nos deja constancia la gran cantidad de denuncias realizadas en su amplia mayoría por el colectivo femenino, las cuales expresan su incomodidad frente a esta situación.

Palabras clave

Camino más corto restringido, acoso sexual callejero, identificación de rutas seguras, prevención del crimen.

1. INTRODUCCIÓN

En la ciudad de medellín se han evidenciado una gran cantidad de casos de acoso callejero, lo que ha motivado a la implementación de herramientas de software con el objetivo de reducir el riesgo de acoso bajo ciertas rutas planteadas, así pues, teniendo en cuenta las preferencias del usuario, tanto el riesgo de acoso de un camino propuesto, como la longitud de dicho camino al encontrar un riesgo preferido.

1.1. Problema

El problema base del proyecto se centra en el acoso callejero en la ciudad de Medellín, el impacto de este se encuentra, principalmente, en la gran vulneración de los derechos de las personas, así como la afectación de la salud mental y física de las víctimas, que termina por aumentar el índice de inseguridad en muchos sectores de la ciudad, creando una normalización al acoso, que prolonga de esa forma la problemática.

1.2 Solución

Para el problema propuesto, se ha puesto en marcha el desarrollo de un programa que, teniendo en cuenta los datos recolectados en la ciudad de Medellín sobre el nivel de acoso en las diferentes calles, se encuentre, según dos puntos especificados por el usuario, una ruta que satisfaga las necesidades de cada individuo, así pues, teniendo en cuenta factores como la longitud máxima de la ruta y el acoso promedio máximo que no puede superar la ruta a encontrar, desplegando la ruta gráficamente en el mapa de medellín. Así pues, por medio del algoritmo 'Dijkstra', se busca, en un grafo compuesto por las calles de Medellín, una ruta que esté dentro de las necesidades del usuario.

En la versión actual. hay variedad de aspectos por mejorar, los principales, relacionados con la complejidad en tiempo del programa, al operar con una tan inmensa cantidad de datos, requiere de la implementación de una estructura de datos más eficaz, principalmente, en la creación del grafo correspondiente al camino que se quiere buscar, además, por ahora, las búsquedas pueden ser o de la ruta más corta, o de la ruta más segura, no tomando ambos factores en cuenta al mismo tiempo. Todos estos aspectos serán mejorados para la futura versión del programa. El sistema actual es funcional con las coordenadas y posee las funcionalidades solicitadas para la presente entrega.

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.

2.1 Buscador de rutas para prevenir el acoso callejero usando búsqueda cercana

El problema planteado se basa en cómo prevenir el acoso callejero, por medio de una plataforma que analiza el riesgo de acoso en rutas desde la posición inicial hasta la posición final. Por medio de cuadrículas ubicadas encima del mapa, se sugiere al usuario primero, la ruta con menos acoso, incluso si la distancia a recorrer es muy larga, luego, se sugieren las demás rutas, menos seguras, pero más rápidas.

2.2 Plataforma móvil para prevenir casos de acoso sexual callejero

El problema planteado es el acoso callejero, en el caso ejemplo en la nación de Ecuador. Basándose en casos registrados de acoso a lo largo del tiempo, se reconocen las zonas de riesgo, para proporcionar la información previa a los usuarios que realicen consultas al momento de desplazarse de un lugar a otro. Se implementa además el uso de notificaciones al momento de acercarse a una zona de riesgo.

2.3 Route-The Safe: Un modelo sólido para la predicción de rutas más seguras utilizando datos sobre delitos y accidentes.

El problema planteado por este modelo de predicción es encontrar la ruta más segura para el usuario en un recorrido determinado por él mismo, ya que existen otras aplicaciones que simplemente se limitan a encontrar la ruta más corta sin tener en cuenta el nivel de seguridad para el usuario. La aplicación usa datos actualizados de los delitos y crímenes de la ciudad de New York y determina el nivel de riesgo a través de un algoritmo de Machine Learning. Además tiene planteado a futuro que el mismo usuario pueda ingresar ciertos factores de riesgo que él mismo se encuentre y reportarlo a la app.

2.4 Conciencia de seguridad para enrutamiento de turistas motorizados basado en open data y VGI.

El trabajo se basa en determinar una ruta que evita relativamente áreas peligrosas dentro de una ciudad. Para determinar qué tan peligrosa es cierta región, esto deriva en la utilización de datos geográficos voluntarios (VGI).

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].

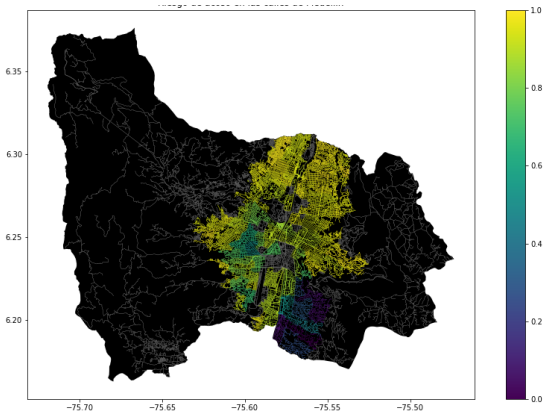


Figura 1. Riesgo de acoso sexual calculado como una combinación lineal de la fracción de hogares que se sienten inseguros y la fracción de hogares con ingresos inferiores a un salario mínimo, obtenida de la Encuesta de Calidad de Vida de Medellín, de 2017.

3.2 Alternativas de camino más corto con restricciones

A continuación, presentamos diferentes algoritmos utilizados para el camino más corto restringido.

3.2.1 Algoritmo de Dijkstra

El algoritmo se basa en la búsqueda del camino más corto en estructuras de datos como un grafo, partiendo de un nodo base, se recorren los nodos adyacentes, comparando los valores de las aristas, realizando así un recorrido comparando los demás posibles caminos, hasta llegar al nodo objetivo partiendo de decisiones basadas en la suma de las aristas, acabando el algoritmo cuando se alcance el nodo objetivo, habiendo encontrado el camino más corto posible.

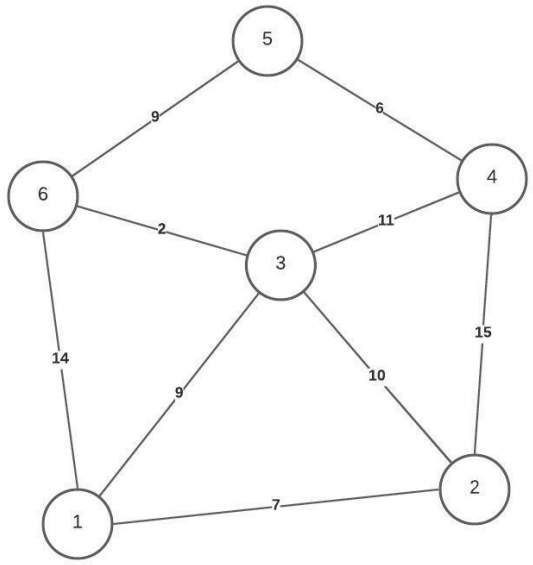
[1] https://www.openstreetmap.org/

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

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

https://github.com/Rendxnn/ST0245-002

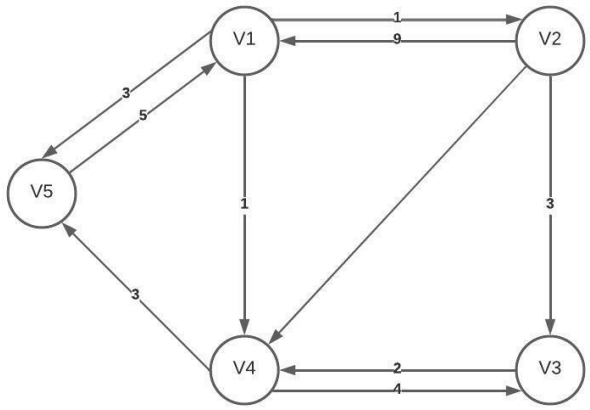
Complejidad Dijkstra: $O(|V|^2 + |E|)$



3.2.2 Algoritmo de Floyd-Warshall

El algoritmo de Floyd-Warshall encuentra el camino más corto, pero no solo entre dos nodos establecidos, sino que crea una tabla de los caminos más cortos entre los dos nodos. Así pues, este crea una matriz del 'peso' o la distancia entre los nodos, comparando con cada columna. En cada posición de la matriz se ubican los valores de los pesos entre los nodos, y cuando dos nodos no están conectados (o no poseen una flecha que los conecte en su representación gráfica) se establece que el peso entre estos es infinito (∞).

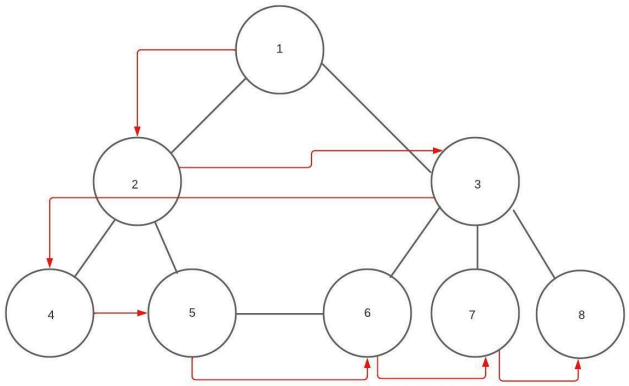
Complejidad Floyd-Warshall: $O(n^3)$



3.2.3 Algoritmo Breadth-first search (BFS)

Este algoritmo BFS se usa mucho por ejemplo al momento de hallar la ruta más corta cuando el peso entre todos los nodos es 1, cuando se requiere llegar con un movimiento de caballo de un punto a otro con el menor número de pasos. El algoritmo va creando un árbol a medida que va recorriendo el grafo.

Complejidad de BFS: O(V + E)



3.2.4 Algoritmo Depth First Search (DFS)

Este algoritmo DFS detecta ciclos en un grafo si es conexo, halla el número de componentes conexos, entre otras utilidades. DFS va creando un árbol a medida que va recorriendo el grafo, pero a diferencia de otros como BFS, este lo hace a profundidad. Para esto puede trabajar como Stack o de manera recursiva.

Complejidad de DFS: O(V + E)

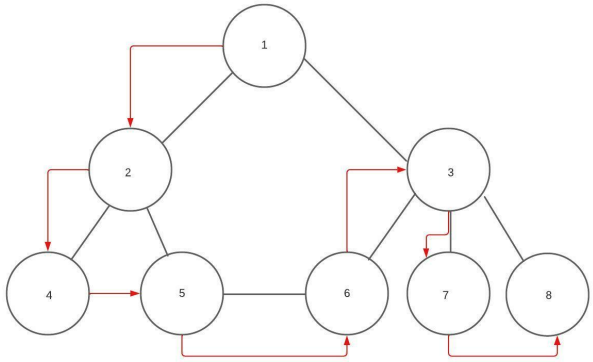


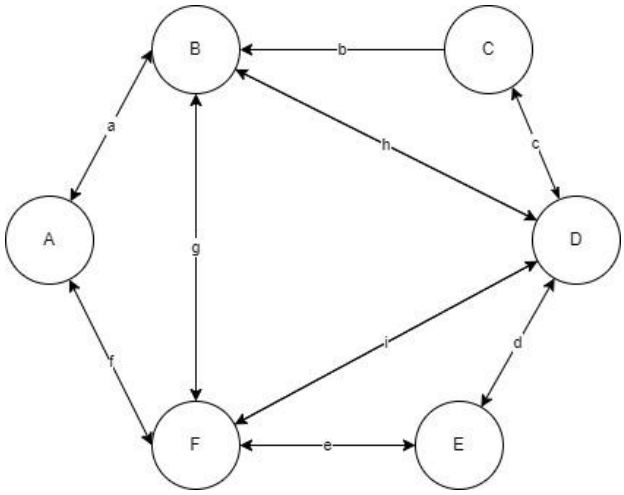
Tabla 3. Distancias más cortas sin superar un riesgo de acoso medio ponderado *r*.

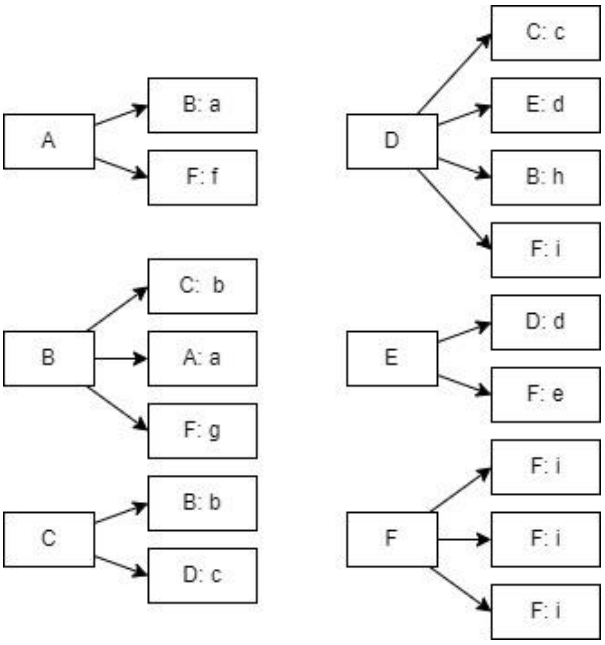
DISEÑO E IMPLEMENTACIÓN DEL ALGORITMO

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.

4.1 Estructuras de Datos

En el presente proyecto, se ha implementado una lista de adyacencia utilizando un diccionario de Python, haciendo así, que cada coordenada se vea vinculada con las coordenadas a las que esta tiene acceso directo por medio de calles, creando una especie de grafo, donde, por medio de un diccionario de diccionarios, el 'peso' de cada arista del grafo, (siendo las coordenadas los vértices), sea, si bien la longitud de la calle, o el riesgo de acoso en la versión actual de desarrollo. Se adjunta figura para la representación de la estructura de datos empleada.





4.2 Algoritmos

En este trabajo, proponemos algoritmos para el problema del camino más corto restringido. El primer algoritmo calcula el camino más corto sin superar un riesgo medio ponderado de acoso *r*. El segundo algoritmo calcula el camino con el menor riesgo medio ponderado de acoso sin superar una distancia *d*.

4.2.1 Primer algoritmo: Algoritmo Dijkstra.

Este algoritmo, con la particularidad de encontrar la distancia más corta entre 2 vértices en un determinado grafo, esto sin superar un riesgo medio ponderado de acoso r. El algoritmo va visitando cada uno de los nodos hasta que no haya ninguno sin recorrer. En primera instancia se selecciona el vértice con la menor distancia desde la fuente y lo visita, este va actualizando la distancia con cada nodo consecuente que va encontrando. A continuación, va repitiendo el mismo proceso hasta que haya visitado todos los vértices. Este algoritmo requiere de un STP (un árbol de ruta más corta) partiendo desde la base.

Dijkstra como sabemos, usa como propiedad básica la suma de los positivos, por ende una desventaja de este algoritmo es al momento de encontrar aristas negativas llegando a una respuesta incorrecta.

El algoritmo se ejemplifica en la Figura 3.

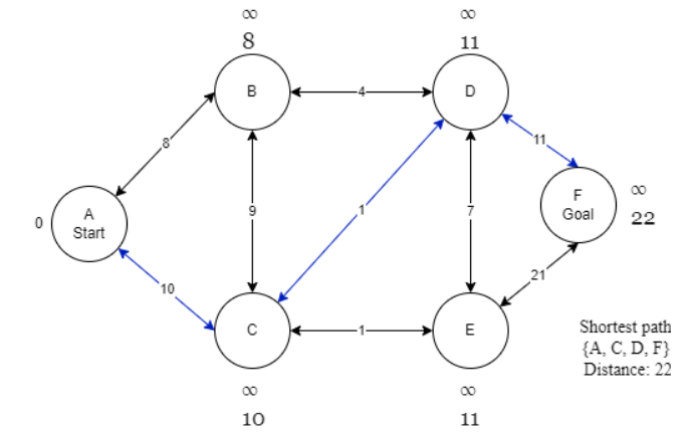
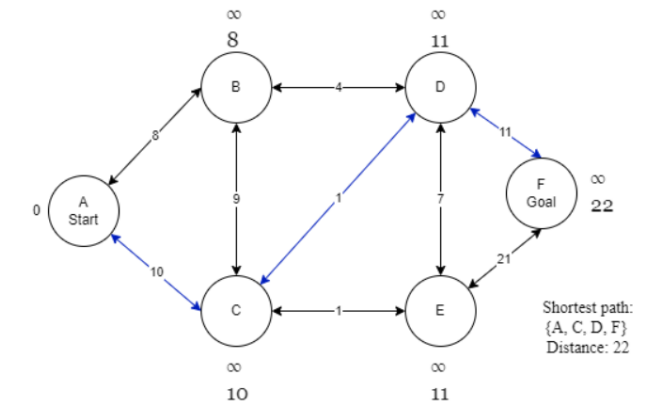


Figura 4: Resolución del problema del camino más corto (seguro) restringido con algoritmo Dijkstra, tomando como peso de aristas el riesgo de acoso de la calle representada.

4.2.2 Segundo algoritmo

Al igual que para la resolución de la situación planteada en el punto 4.2.1, el algoritmo utilizado para encontrar el camino más seguro es el algoritmo Dijkstra, tomando, en este caso, como peso de las aristas del grafo construido, el riesgo de acoso registrado en la calle representada como arista del grafo, realizando el mismo proceso ya expuesto. Tras analizar el archivo, se encontró que muchas de las calles registradas no poseen un valor asignado de riesgo de acoso, figurando en el archivo como "nan", así pues, se implementó el código de forma que los valores vacíos "nan" fuesen reemplazados con 0 o con unos 1 durante la lectura del archivo, para que posterior a la lectura de este, y durante la creación del grafo (en forma de diccionario de python) los valores encontrados que fuesen 0 o 1 fuesen reemplazados por el valor promedio del acoso. (El valor promedio de acoso fue encontrado en el archivo replaceAverage.py, luego almacenado como una variable), logrando así, asignar un valor más realista de riesgo de acoso a las calles (aristas) que no lo tuviesen asignado. El algoritmo se ejemplifica en la Figura 4.



4.4 Análisis de la complejidad de los algoritmos

Analizando el algoritmo implementado, al recorrer la lista de los nodos no visitados (unseenNodes en Dijkstra.py) y recorrer el grafo creado (diccionario de diccionarios de Python), se establece que la complejidad ha de ser $O(V^2)$, se sabe que la búsqueda de cualquier elemento sobre una tabla de hash o diccionario de Python es O(n), por tanto, en el peor de los casos, si se pretende realizar la búsqueda de una coordenada almacenada que se encuentre al final del diccionario, la complejidad de dicha búsqueda será O(V), siendo V la cantidad total de datos almacenados, y teniendo en cuenta que el proceso se realiza con ciclos enlazados, el peor de los casos sería O(V*V), expresado de mejor manera como $O(V^2)$.

Algoritmo	**Complejidad Temporal**
Algoritmo Dijkstra	$O(V^2)$

Tabla 1: Complejidad temporal del algoritmo Dijkstra, tomando V como la cantidad de nodos que se recorren a la hora de correr al algoritmo.

Analizando la estructura de datos, se encuentra que al emplear una tabla de hash, cuya complejidad es conocido que es O(n), siendo n la cantidad de datos almacenados, así pues, la complejidad en memoria de las coordenadas almacenadas, siendo las coordenadas los vértices del grafo, almacenados como las claves del diccionario, y siendo los valores unos diccionarios que almacenan como clave coordenadas a las cuales tiene conexión directa la clave original, y como valor el peso de la arista (sea la longitud de la calle, el riesgo de acoso, o la multiplicación de estos para encontrar el camino con un balance entre distancia y seguridad.

Estructura de datos	**Complejidad de la memoria**
Tabla de hash - diccionario	O(V)

Tabla 2: Complejidad de memoria de la tabla de hash, donde V es la cantidad de coordenadas almacenadas.

4.5 Criterios del diseño del algoritmo

El algoritmo fue diseñado de esta manera debido a su compatibilidad con la estructura de datos seleccionada, así pues, aunque la complejidad obtenida es mejorable, se mantiene como una complejidad trabajable. Además, otros algoritmos que fuesen considerados (como A*) requerían de conocimientos y codificaciones muy avanzadas, que implican inteligencia artificial y demás conocimientos que aún no se poseen. Así pues, Dijkstra fue el más acertado en cuanto a la compatibilidad con la estructura de datos implementada, aplicabilidad en cuanto a conocimientos, viabilidad de implementación y correcto funcionamiento con respecto al problema planteado.

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	7228.1 m	0.84
Universidad de Antioquia	Universidad Nacional	815.4 m	0.83
Universidad Nacional	Universidad Luis Amigó	1478.6 m	0.85

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	0.87	5,000
Universidad de Antioquia	Universidad Nacional	0.83	7,000
Universidad Nacional	Universidad Luis Amigó	0.85	6,500

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

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 la Universidad de Medellín	30.1 s
De la Universidad de Antioquia a la Universidad Nacional	32.5 s
De la Universidad Nacional a la Universidad Luis Amigó	34.25 s

Tabla 4: Tiempos de ejecución del algoritmo Dijkstra para las consultas presentadas, desde el momento en que se ingresan las coordenadas inicial y final, hasta que termina de correr y se puede acceder al mapa generado.

6. Conclusiones

Tras el estudio de la situación, la comprensión del problema, la estructuración de los datos y la codificación de la solución, se concluye que, comparando los caminos sugeridos según la necesidad ingresada en el sistema, los caminos graficados resultan ser significativamente diferentes, así como los valores de la distancia a recorrer o el riesgo de acoso promedio en situaciones donde se busca, si bien, la ruta más corta, la más segura, o un balance entre ambas opciones. El desarrollo del presente sistema resulta útil para la ciudad, especialmente para ciudadanos que sean potencialmente más vulnerables a situaciones de acoso callejero, que, aunque los tiempos de ejecución y la complejidad del programa son mejorables en gran medida, su uso si puede resultar viable en muchas situaciones reales.

6.1 Trabajos futuros

Para el futuro del trabajo, se propone la optimización del programa ya realizado, así como la mejora en la estructura de los datos, la velocidad del algoritmo (ya sea buscando una implementación del algoritmo seleccionado aún más óptima o buscando un algoritmo más avanzado), la precisión del manejo de datos no registrados, más opciones para las necesidades del usuario que lo emplee, así como demás funciones y mejoras visuales, implementación de inteligencia artificial en una versión futura del algoritmo (preferiblemente con la implementación de A*), entre otras mejoras que harían del sistema una vía real por la que optasen los ciudadanos en la vida cotidiana.

AGRADECIMIENTOS

Se extiende un agradecimiento a Miguel Cock, Juan Sebastián Camacho y Víctor Daniel Arango (Ingeniería de sistemas, segundo semestre, Eafit) por los comentarios y sugerencias para la optimización y formas de resolución de problemas presentados durante la resolución de la situación inicial.

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. Mirino, A. Risald y Suyoto. Best Routes Selection Using Dijkstra And Floyd-Warshall Algorithm. Recuperado el 21 de Febrero de 2022: https://ieeexplore.ieee.org/abstract/document/8265662

2. [1] OMDENA, Preventing Sexual Harassment Through a Path Finding Algorithm Using Nearby Search,

Recuperado el 22 de Febrero de 2022 de: https://omdena.com/blog/path-finding-algorithm/

3. Altamirano, K. PLATAFORMA MOVIL PARA PREVENIR CASOS DE ACOSO SEXUAL CALLEJERO, Recuperado el 22 de Febrero de 2022 de: http://www.dspace.espol.edu.ec/xmlui/handle/123456789/36935?show=full

4. Route-The Safe: A Robust Model for Safest Route Prediction Using Crime and Accidental Data, Venkatesh Gauri Shankar and Chaurasia Sandeep. Recuperado el 22 de febrero de 2022 de:

https://github.com/mauriciotoro/ST0245-Eafit/blob/master/proyecto/Trabajos-relacionados/Route-The-Safe.pdf

5. Safety-aware routing for motorized tourists based on open data and VGI, Andreas Keler and Jean Damascene Mazimpaka. Recuperado el 22 de febrero de 2022 de:

https://github.com/mauriciotoro/ST0245-Eafit/blob/master/proyecto/Trabajos-relacionados/Safety-aware_routing_for_motorised_tourists_based_.pdf

6. ALGORITMO DE BÚSQUEDA: DEPTH FIRST SEARCH (PARTE 1). Algorithms and More. Recuperado el 22 de febrero de 2022 de:

https://jariasf.wordpress.com/2012/03/02/algoritmo-de-busqueda-depth-first-search-parte-1/

7. ALGORITMO DE BÚSQUEDA: BREADTH FIRST SEARCH. Algorithms and More. Recuperado el 22 de febrero de 2022 de:

https://jariasf.wordpress.com/2012/02/27/algoritmo-de-busqueda-breadth-first-search/

8. Herrera Jimenez, S. Salcedo Parra, O y Gallego Torres, A Efficiency Graphs Algorithmic's applications oriented to GMPLS. Recuperado el 16/05/2022 de: http://www.scielo.org.co/pdf/rfing/v23n36/v23n36a09.pdf