ALGORITMO PARA CALCULAR LA RUTA MÁS SEGURA Y ÓPTIMA

Samuel Andrés Areiza
Universidad Eafit
Colombia
saareizat@eafit.edu.co

Sara Maria Cano
Universidad Eafit
Colombia
smcanom@eafit.edu.co

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

Jaime Andres Riascos
Universidad Eafit
Colombia
jariascoss@eafit.edu.co

RESUMEN

En los últimos años, han aumentado los casos de acoso sexual y crímenes en las calles de Medellín. Por lo tanto, es fundamental enfrentar esta problemática desarrollando e implementando un algoritmo que encuentre el mejor camino entre dos puntos de la ciudad, disminuyendo la distancia a recorrer y buscando el trayecto más seguro, con el fin de disminuir la inseguridad callejera. En este proyecto se implementó el algoritmo de Dijkstra para calcular tres caminos tomando en cuenta la distancia en metros y el riego de acoso callejero. Esto es fundamental ya que en un camino es posible que sea menor la distancia, pero no el riesgo, mientras que en otro caso ocurre al contrario y finalmente en el camino restante ambas variables son optimizadas en igual medida. Este algoritmo tiene un tiempo de ejecución de 1 segundo en promedio y una complejidad de O((V+E) log|V|).

Palabras clave

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

1. INTRODUCCIÓN

Según Juliana Martínez Londoño (2020), Secretaria de la Secretaría de la Mujer de Medellín, “La sensación de seguridad también se ve determinada por otros aspectos como ver en las noticias que las mujeres son asaltadas en la noche, y no solo asaltadas sino que al tiempo que las robaron, las violaron.”. En consecuencia, las mujeres se sienten inseguras al andar solas por su propia ciudad. Este es un gran problema porque evita que las mujeres se apropien del espacio público y se puedan desplazar con tranquilidad.

A pesar de todo, las mujeres no se pueden quedar paralizadas por el miedo y, por esta razón, ellas han aprendido métodos de prevención como: las redes de cuidado entre amigas, identificar por donde caminar y por donde no, cambiarse acera al pasar por un sector o vestirse de cierta forma, entre otras. Sin embargo, estos métodos no siempre son efectivos y muchas mujeres siguen siendo víctimas de acoso. [1]

1.1. Problema

Hoy en día, aplicaciones como Google Maps o Waze se han vuelto populares al momento de calcular la ruta más corta para desplazarse en la ciudad. Estas plataformas pueden identificar el estado del tráfico en tiempo real, sin embargo, aún no tienen la capacidad de identificar qué zonas son seguras o no. Es entonces que, en algunas ocasiones, estas aplicaciones generan rutas que son óptimas debido a que la distancia es la más corta posible, pero pasan por lugares inseguros.

Por lo tanto, surge la cuestión que deseamos resolver, cómo desarrollar un algoritmo que calcule tres rutas diferentes con el menor valor de una variable v, que representa el riesgo de acoso callejero (r) y la distancia en metros (d).

1.2 Solución

El algoritmo seleccionado para solucionar esta problemática es el algoritmo de Dijkstra ya que este es el más efectivo para recorrer grafos y encontrar el camino más corto con menor peso en grafos dirigidos, como es el caso del mapa de Medellín. El algoritmo calcula originalmente el camino más corto desde un vértice inicial hacia todos los demás vértices. Por esta razón, hemos modificado el algoritmo para que calcule el camino más corto desde un vértice inicial a otro vértice, que representa el destino. Además, modificamos el algoritmo para que retorne un arreglo con las coordenadas de cada vértice que hace parte del camino seleccionado.

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 HarassMap: El uso de datos de origen colectivo para mapear el acoso sexual en Egipto

El presente trabajo presenta una herramienta llamada HarassMap, la cual busca la prevención del acoso sexual en Egipto, mediante la recopilación de información de riesgo acerca de las regiones de este país.

Esta aplicación toma como base principal la tecnología crowdsourcing, donde es la misma sociedad quien va reportando la información acerca de cuáles son las regiones peligrosas; posteriormente, esta información es utilizada

para cartografiar los incidentes y establecer las regiones que sean más o menos seguras, para así, notificar a sus usuarios cual es el recorrido más recomendado con énfasis en la seguridad de este. [2]

2.2 Predicción de una ruta segura para las mujeres utilizando Google Maps

Este trabajo, realizado por el Departamento de Ciencias Computacionales e Ingeniería del Instituto Internacional de Investigación e Estudios Manav Rachna, propone un algoritmo que permite al usuario elegir entre la ruta más corta o la más segura para llegar de un punto "a" a un punto "b". El camino más corto y seguro es calculado a partir de registros de crimen en el área.

Además, este algoritmo recurre a Google Maps cuando el usuario prefiere consultar el camino más corto, sin tomar en cuenta la seguridad. Este algoritmo también compara los caminos más seguros según los datos de crimen y los más cortos según Google Maps, con el fin de encontrar un solo camino que sea el más corto y más seguro. Finalmente, si el usuario desea encontrar el camino más seguro, sin que este sea el más corto, el programa se basa únicamente en el registro de crímenes. Por último, para calcular la ruta más corta, el programa utiliza el algoritmo de Dijkstra. [3]

2.3 MehfoozAurat: Transformar los teléfonos inteligentes en Dispositivos de seguridad contra el acoso de mujeres.

Este articulo presenta un análisis de la herramienta MehfoozAurat y el impacto de esta en la sociedad pakistaní. Esta aplicación surge bajo el marco de la inseguridad que sufren las mujeres trabajadoras de clase media o baja en Pakistán, las cuales no tienen oportunidad o facilidad de usar transporte privado para su movilidad.

Esta aplicación tiene distintos mecanismos de ayuda para la movilidad, pero se hará principal enfoque en la elección de rutas seguras para los usuarios. Este se inició mediante reportes judiciales para elegir que rutas eran más y menos seguras, así, en el mapa de la aplicación se colocaban de color verde o rojo, respectivamente. Posteriormente, se iba complementando la información con base en reportes que los mismos usuarios fueran dejando en la aplicación después de recorrer ciertas zonas de las ciudades. Mas especificaciones del algoritmo no se encuentran en el artículo. [4]

2.4 SafeStreet: Empoderar a las mujeres contra el acoso callejero mediante una aplicación basada en la privacidad de la ubicación

SafeStreet es una aplicación móvil desarrollada por el Departamento de Ciencias Computacionales e Ingeniería de la Universidad de Daca para proteger a las mujeres del acoso sexual en público. Esta aplicación permite a las mujeres encontrar el camino más seguro para llegar a su destinación en cualquier hora del día. El usuario ingresa su posición actual, su destinación final y la hora en la que desea realizar este recorrido y la aplicación le muestra el camino más seguro y el más inseguro.

Adicionalmente, la aplicación le advierte al usuario en tiempo real sobre posibles riesgos cerca. El algoritmo de búsqueda para encontrar el camino más seguro se basa en los reportes anónimos de acoso de los otros usuarios. [5]

3. MATERIALES Y MÉTODOS

En esta sección, explicamos cómo se recogieron y procesaron los datos y, después, diferentes alternativas de algoritmos de caminos que reducen tanto la distancia como el riesgo de 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. El mapa incluye (1) la 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 obtenidas de los metadatos proporcionados por OSM.

Para este proyecto, se calculó una 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].

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

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

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

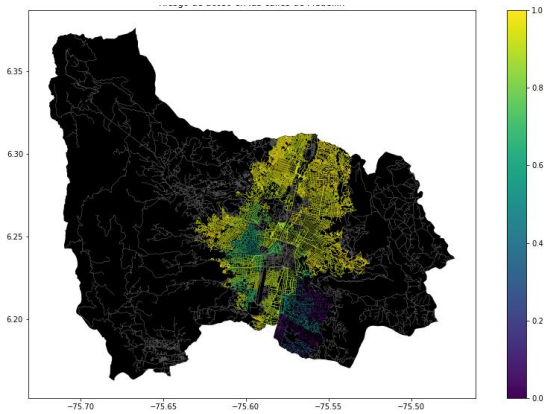


Figura 1. Riesgo de acoso sexual calculado como una combinación lineal de la fracción de hogares que se sienten inseguros y la fracción de hogares con ingresos inferiores a un salario mínimo, obtenidas de la Encuesta de Calidad de Vida de Medellín, de 2017.

3.2 Alternativas de caminos que reducen el riesgo de acoso sexual callejero y distancia

A continuación, presentamos diferentes algoritmos utilizados para un camino que reduce tanto el acoso sexual callejero como la distancia. *(En este semestre, ejemplos de dichos algoritmos son DFS, BFS, Dijkstra, A*, Bellman, Floyd, entre otros).*

3.2.1 Algoritmo de Lee

Este algoritmo, basado en el Breadth-First Search (BFS), busca cual es el camino más corto, dada una matriz rectangular, entre un punto de origen A y un punto de destino B. Para esto solo se permiten cuatro movimientos, subir, bajar, dirigirse a la izquierda o derecha.

Consiste en recorrer todos los caminos posibles para llegar del origen al destino y marcar cada posición de la matriz que ya se haya recorrido, así puede descartar, en el proceso recursivo, aquellos caminos que recorren un mismo trayecto más de una vez; de tal manera que, al final solo compara cual es el camino menos corto que si haya llegado al destino. Este tiene complejidad O(m*n), donde m y n son las dimensiones de la matriz. [6]

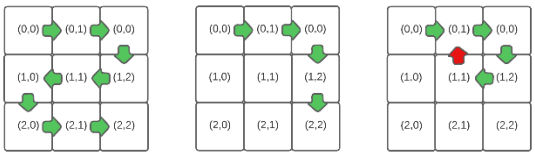


Figura 1. Algoritmo de Lee

3.2.2 Algoritmo de Dijkstra

El algoritmo de Dijkstra es utilizado para encontrar el camino más corto en un grafo dado G(V,E), donde V es el número de vértices y E es el número de aristas. Además, cada vértice tiene un costo asignado. El objetivo de este algoritmo es calcular el camino para recorrer el grafo desde un punto "a" a un punto "b" con el menor costo.

El primer paso llevado a cabo por el algoritmo es comparar los costos de cada vértice adyacente al vértice inicial y elegir el vértice con el menor costo. Luego, suma los costos del vértice inicial con el vértice seleccionado, calcula el costo de los vértices adyacentes al vértice seleccionado en el paso 1 y elige el de menor costo. El algoritmo repite el paso anterior hasta llegar al último vértice, obteniendo así el camino con el menor costo. La complejidad de este algoritmo es O(|E| + |V|log|V|). [7][8]

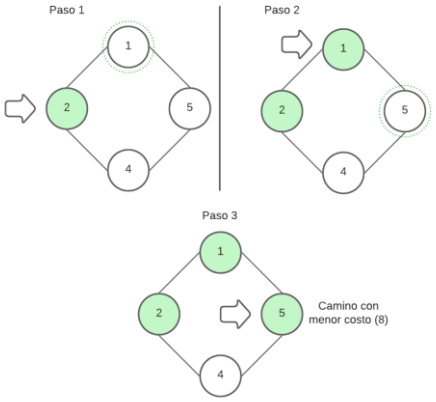


Figura 2. Algoritmo de Dijkstra

3.2.3 Algoritmo de Búsqueda en Profundidad (DFS)

Este algoritmo es utilizado para encontrar si hay una forma de ir desde un punto de origen A a un punto de destino B, generalmente en grafos, pero no necesariamente encuentra el camino más corto. Este es relativamente simple e intuitivo, se basa en primero cuestionarse si es posible ir a un nodo adyacente al que se encuentra el recorrido en ese instante y cada que pasa por un vértice marca el anterior como ya recorrido, en caso de haber más de una posibilidad elige un camino al azar y guarda en una pila el vértice del que partió.

Sigue de esta manera hasta que se encuentra en una situación donde ya no puede dirigirse a otro vértice, entonces retrocede hasta la última bifurcación que haya tomado el camino. Repitiendo este proceso hasta llegar al destino, retornando

un camino posible, pero no el óptimo. En este caso, la complejidad O(V*E) donde V es la cantidad de vértices y E la cantidad de aristas. [9]

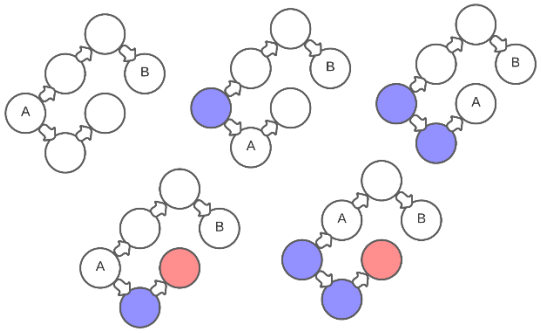


Figura 3. Algoritmo DFS

3.2.4 Algoritmo geométrico A-Star

El algoritmo A-Star (A*) es un algoritmo de búsqueda heurística con el fin de planear trayectorias. Este algoritmo utiliza una función *f(n)* que representa el costo final entre cada celda de una cuadrícula, *f(n)* está definida de la siguiente manera:

$$f(n) = g(n) + h(n)$$

Donde *g(n)* representa el costo desde de la celda inicial a la celda actual y *h(n)* representa el costo estimado de la celda actual a la celda objetivo. Existen dos métodos para calcular *h(n)*: Distancia Euclidiana *(h_E)* o Distancia Manhattan *(h_M)*.

$$h_E = \sqrt{(x_a - x_f)^2 + (y_a - y_f)^2}$$

$$h_M = |x_a - x_f| + |y_a - y_f|$$

(x_a, y_a): coordenadas de la celda actual
(x_f, y_f): coordenadas de la celda objetivo

El algoritmo solo puede buscar en cuatro direcciones: arriba, abajo, derecha e izquierda, utilizando la Distancia Manhattan. Mientras que con la distancia Euclidiana puede buscar la distancia en todos los nodos adyacentes. La función *h(n)* permite priorizar las celdas para las cuales *f(n)* retorna el menor valor. Por lo tanto, este algoritmo no necesariamente recorre todas las celdas. Pero en el peor de los casos, es decir cuando recorra todas las celdas, el algoritmo tiene una complejidad de O(n), donde n es el número de celdas. [10]

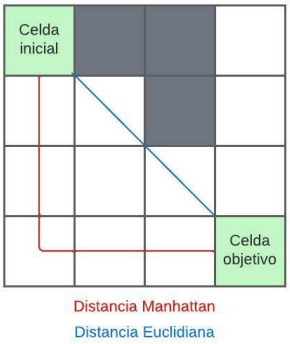


Figura 4. Algoritmo geométrico A-Star

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].

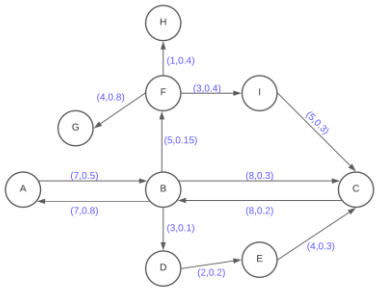
4.1 Estructuras de datos

Los datos sobre el acoso callejero en la ciudad de Medellín fueron almacenados en un dataFrame mediante la librería Pandas, la cual permite una facilidad a la hora de manejar los datos y elaborar la estructura de datos para representar el mapa de la ciudad.

El mapa es un grafo cuya representación ha sido mediante una lista de adyacencia a partir de diccionarios, tras la creación de un diccionario donde las llaves son todos los posibles orígenes y en cada valor se almacena un diccionario con los posibles destinos para cada origen, donde se almacenan tuplas que contenían la información sobre la longitud y el índice de riesgo para cada relación origen~destino. La estructura de los datos se presenta en la Figura 2.

4 https://github.com/saracanom/ProyectoAlgoritmoSeguroOptimo

a)



b)

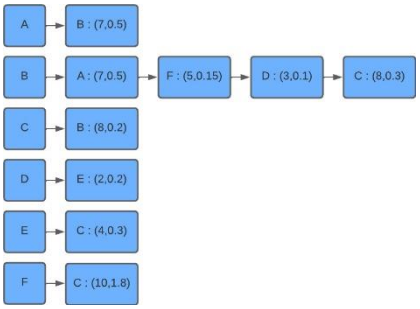


Figura 2: Un ejemplo de mapa de calles se presenta en (a) y su representación como lista de adyacencia en (b).

4.2 Algoritmos

En este trabajo, proponemos un algoritmo para un camino que minimiza tanto la distancia como el riesgo de acoso sexual callejero.

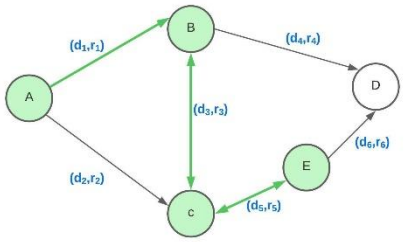
4.2.1 Algoritmo para un camino que reduce tanto la distancia como el riesgo de acoso sexual callejero

Al momento de recorrer este grafo para retornar el camino más optimo entre un punto A y un punto B, se ha empleado el algoritmo de Dijkstra. Por lo tanto, se ha diseñado una función que recibe tres parámetros: el grafo previamente creado, los vértices de origen y destino.

Para emplear este algoritmo inicialmente se crea un diccionario donde se almacenará el peso mínimo asociado a cada vértice, estos pesos son obtenidos con el producto entre el valor de la longitud y el índice de riesgo asociado para cada relación origen ~ destino, los cuales son inicializados en infinito y "el padre" de cada uno, es decir, en qué vértice se estuvo antes de alojarse en el vértice actual del recorrido, el cual será inicializado como un String vacío; posterior a esto, se actualizan estos valores para el vértice de origen, con peso 0 y padre "None".

El algoritmo de Dijkstra usa una cola de prioridades la cual es inicializada con la tupla (0, Origin_Vertex) y es rellenada con todos los vecinos del vértice actual, actualizando sus respectivos pesos mínimos y "su padre" según corresponda. Además, cada que ejecuta su método.pop() retorna el valor mínimo en la cola, en este caso, el siguiente paso con menor peso asociado a su trayecto, lo cual permite ejecutarlo hasta que el vértice actual sea el vértice destino.

En este caso se empieza a rellenar una lista con todos los padres desde el vértice destino hasta llegar al vértice origen, para finalmente ingresar el peso actual de dicho recorrido, lo cual permite no solo retornar el valor del peso mínimo desde el punto A hasta el punto B, sino también el recorrido puntual llevado a cabo. El cual posteriormente será graficado con ayuda de la librería gmplot. El algoritmo se ejemplifica en la Figura 3.



Vértice	Paso 1	Paso 2	Paso 3	Paso 4
A	(0,A)	*	*	*
B	∞	(d_1*r_1,A)	*	*
C	∞	(d_2*r_2,A)	((d_1*r_1)+(d_3*r_3),B)	*
D	∞	∞	((d_1*r_1)+(d_4*r_4),B)	((d_1*r_1)+(d_4*r_4),B)
E	∞	∞	∞	((d_1*r_1)+(d_3*r_3)+(d_5*r_5),C)

*Retorna: [A,B,C,E, (d_1*r_1)+(d_3*r_3)+(d_5*r_5)]*

Figura 3: Cálculo de un camino que reduce tanto la distancia como el riesgo de acoso.

4.2.2 Cálculo de otros dos caminos para reducir tanto la distancia como el riesgo de acoso sexual callejero

Para poder encontrar el camino con menor distancia y más seguro, se calcularon otros dos caminos. Uno de ellos se calcula a partir de la suma de la variable distancia y el riesgo de acoso. El otro camino se calcula al elevar la distancia al riesgo de acoso. Ambos caminos se calculan utilizando el algoritmo de Dijkstra, lo único que cambia es la manera de combinar las variables distancia y riesgo. El algoritmo se ejemplifica en la Figura 4.

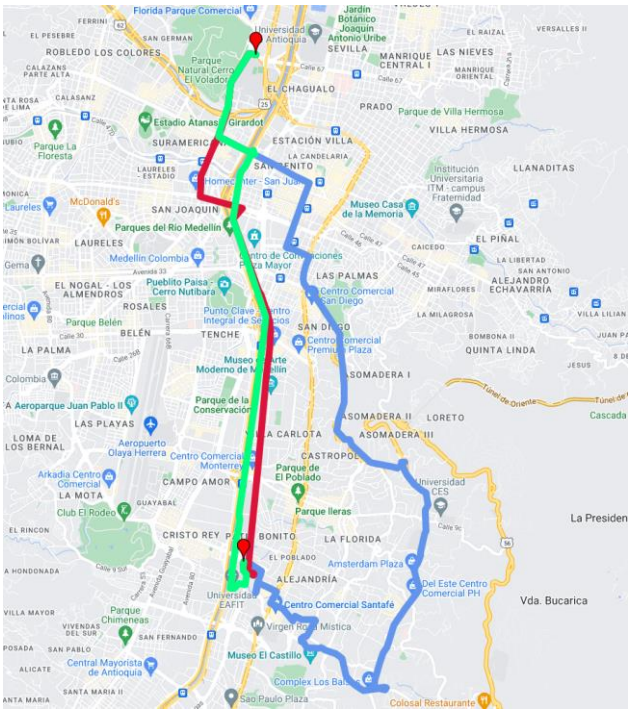


Figura 4: Mapa de la ciudad de Medellín donde se presentan tres caminos para peatones que reducen tanto el riesgo de acoso sexual como la distancia en metros entre la Universidad EAFIT y la Universidad Nacional.

4.3 Análisis de la complejidad del algoritmo

Sea V el número de intersecciones (nodos) y E el número de calles (aristas). En el peor de los casos, el algoritmo tendrá que pasar por todos los vértices para añadirlas a la cola de prioridad y por cada vértice que añade, con un ciclo debe pasar por todos los vértices adyacentes para elegir aquel con menor peso. Hasta esta parte, la complejidad es de O(V+E). Como estamos haciendo uso de una cola de prioridades, añadir un elemento a esta tiene una complejidad de O(log |V|) [11]. Finalmente, la complejidad del algoritmo es de O((V+E) log |V|))

Algoritmo	Complejidad temporal
Algoritmo de Dijkstra	O((V+E) log |V|))

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

Estructura de datos	Complejidad de la memoria
Lista de Adyacencia	$O(V^2)$

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

4.4 Criterios de diseño del algoritmo

El criterio más importante es el bajo tiempo de ejecución y simplicidad del algoritmo. Fue por esta razón que elegimos el algoritmo de Dijkstra implementado con cola de prioridades ya que es de los algoritmos de optimización de recorridos en grafos más conocidos debido a su reducida complejidad y facilidad para comprender e implementar.

A medida que se iba codificando el algoritmo se evidenció la necesidad de ir asignando y guardando el nodo anterior, llamado "nodo padre", que mejor optimizaba el recorrido. Para esto inicialmente se implementó un arreglo donde se iban añadiendo todos los padres desde el vértice destino hasta el vértice origen, es decir cada segmento del recorrido. Sin embargo, posteriormente se utilizó una pila (deque()), debido a que esta hace más eficiente el retornar el recorrido optimo y reduce el tiempo de ejecución.

Otro criterio importante era graficar claramente el mapa de Medellín con los tres caminos seguros. Gracias a la librería de GmPlot, se pudo cumplir con este criterio ya que esta librería ubica las coordenadas en Google Maps y las une por medio de una línea. Por lo tanto, es fácil para el usuario visualizar la ruta óptima.

5. RESULTADOS

En esta sección, presentamos algunos resultados cuantitativos sobre los tres caminos que reducen tanto la distancia como el riesgo de acoso sexual callejero.

5.1 Resultados del camino que reduce tanto la distancia como el riesgo de acoso sexual callejero

A continuación, presentamos los resultados obtenidos de *tres caminos que reducen tanto la distancia como el acoso,* en la Tabla 3.

Origen	Destino	Distancia	Riesgo
Eafit	Unal	16642m	0.35
Eafit	Unal	8574m	0.69
Eafit	Unal	9061.75m	0.58

Tabla 3. Distancia en metros y riesgo de acoso sexual callejero (entre 0 y 1) para ir desde la Universidad EAFIT hasta la Universidad Nacional caminando.

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.

Cálculo de v	Tiempos medios de ejecución (s)
v = r*d	1.01033 s
v = r+d	1.01030 s
v = d^r	1.00640 s

Tabla 4: Tiempos de ejecución del nombre del Algoritmo de Dijkstra, donde d es la distancia en metros y r el riesgo de acoso para cada uno de los tres caminos calculadores entre EAFIT y Universidad Nacional.

6. CONCLUSIONES

Se obtuvieron tres caminos distintos: d^r, con el cual se encontró un camino que optimizaba en gran medida el riesgo promedio del recorrido, pero no la distancia, llegando a casi duplicar la distancia y reducir a la mitad el riesgo promedio con respecto a los otros dos. Caso contrario con el parámetro r + d, siendo el algoritmo que optimizaba en mayor magnitud la distancia del camino, pero no le daba relevancia al riesgo. Por último, r*d, ha sido el caso que mejor balance presentaba con respecto a ambos parámetros.

Cualquiera de los algoritmos ha presentado un tiempo de ejecución bastante bajo, lo cual permite que la implementación de este en una aplicación móvil o web sea bastante útil en una situación cotidiana. Además, cualquiera brindaría un camino que el usuario debería conocer, sin embargo, se le recomendaría al parámetro d^r, debido a que este presentó la tasa de riesgo más bajo de los tres, detalle muy importante para el trabajo, pues esta es la principal problemática que se trata de solucionar.

6.1 Trabajos futuros

Se buscaría implementar el algoritmo en una aplicación que cualquier ciudadano en Medellín tuviera fácil acceso. También, extender esto para poder usarse en el transporte intermunicipal del área metropolitana o incluso disponerlo en otras ciudades de Colombia. Con lo cual se permite ampliar el trabajo en áreas estadísticas, Machine Learning, desarrollo web, entre otras.

AGRADECIMIENTOS

Este proyecto fue posible gracias a la colaboración de los monitores: Isabel Mora, Gregorio Bermudez, Samuel Rico y Valeria Cardona. Les agradecemos por su tiempo y dedicación.

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 archivo *Shapefile*.

REFERENCIAS

1. Puentes, A. L. 2020. “¿Llegaste bien?” Una reflexión sobre cómo es vivir el espacio público de la ciudad cuando se es mujer. Recuperado el 21 de agosto de 2022, de El Colombiano: https://www.elcolombiano.com/antioquia/como-es-vivir-el-espacio-publico-de-medellin-cuando-se-es-mujer-CB13654035

2. Young, C. 2014. HarassMap: Using Crowdsourced Data to Map Sexual Harassment in Egypt. Technology Innovation Management Review, 4(3): 7-13. DOI: http://doi.org/10.22215/timreview/770

3. Bura, D., Singh, M. and Nandal, P. Predicting Secure and Safe Route for Women using Google Maps. En *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon),* (Faridabad, India, 2019), Institute of Electrical and Electronic Engineers, 103-108. DOI: 10.1109/COMITCon.2019.8862173

4. Muhammad Yasir Sarosh, Muhammad Abdullah Yousaf, Mair Muteeb Javed, and Suleman Shahid. 2016. MehfoozAurat: Transforming Smart Phones into Women Safety Devices Against Harassment. En *Proceedings of the Eighth International Conference on Information and Communication Technologies and Development* (ICTD '16). Association for Computing Machinery, New York, NY, USA, Article 61, 1–4. DOI: https://doi.org/10.1145/2909609.2909645

5. Mohammed Eunus Ali, Shabnam Basera Rishta, Lazima Ansari, Tanzima Hashem, and Ahamad Imtiaz Khan. 2015. SafeStreet: empowering women against street harassment using a privacy-aware location based application. En *Proceedings of the Seventh International Conference on Information and Communication Technologies and Development* (ICTD '15). Association for Computing Machinery, New York, NY, USA, Article 24, 1–4. DOI: https://doi.org/10.1145/2737856.2737870

6. N.D. Shortest path in a maze – Lee Algorithm. In Techie Delight Website. 2021. Given a maze in the form of the binary rectangular matrix, find the shortest path’s length in a maze from a given source to a given destination. DOI: https://www.techiedelight.com/lee-algorithm-shortest-path-in-a-maze/

7. Noto, M. and Sato, H. A method for the shortest path search by extended Dijkstra algorithm. En *Smc 2000 conference proceedings. 2000 ieee international conference on systems, man and cybernetics. 'cybernetics evolving to systems, humans, organizations, and their complex interactions'* (Nashville, TN, USA, 2000), Institute of Electrical and Electronic Engineers , 2316-2320 vol.3. DOI: 10.1109/ICSMC.2000.886462.

8. Barbehenn, M. A note on the complexity of Dijkstra's algorithm for graphs with weighted vertices. En *IEEE Transactions on Computers*, 47 (2). 263-. Feb. 1998. DOI: 10.1109/12.663776.

9. Andrade, E. Núñez, F, J. and Tomás, V, T. Análisis de algoritmos de búsqueda en espacio de estados. N.D. En Universidad Autónoma del Estado de Hidalgo: Escuela Superior de Huejutla. Huejutla de Reyes, Hidalgo, México, C.P. 43000. DOI: https://www.researchgate.net/publication/322348310_Analisis_de_algoritmos_de_busqueda_en_espacio_de_estados

10. Tang, G., Tang, C., Claramunt, C., Hu, X. and Zhou, P. 2021. Geometric A-Star Algorithm: An Improved A-Star Algorithm for AGV Path Planning in a Port Environment. *IEEE Access, 9.* 59196-59210 DOI: 10.1109/ACCESS.2021.3070054

11. Datta, S. 2021. Time Complexity of Inserting into a Heap. Recuperado el 6 de noviembre de 2022, de Baeldung: https://www.baeldung.com/cs/heap-insertion-complexity