ALGORITMO PARA LA CIRCULACIÓN SEGURA EN LAS CALLES DE MEDELLÍN

Cristian Camilo Cárdenas Mogollón
Universidad EAFIT
Colombia
cccardenam@eafit.edu.co

Manuela Caro Villada
Universidad EAFIT
Colombia
mcarov@eafit.edu.co

https://github.com/ManuCarov/Proyecto.git

RESUMEN

El acoso callejero en las calles de Medellín representa un gran peligro en la población, principalmente para el público femenino, donde el riesgo de algún daño físico y psicológico es inminente. Para este complejo, decidimos tomarnos la tarea de crear un programa enfocado en la seguridad de los peatones. Con el algoritmo A*, recreamos el mapa de Medellín, donde el usuario ingresa su punto de origen y el punto de destino, que sería el lugar donde desee llegar, y el programa suelta 3 posibles rutas que puede utilizar, tomando como prioridad la seguridad y el menor tiempo posible.

Se hizo un ejemplo de una ruta que guíe de la Universidad EAFIT a la Universidad Nacional, sus resultados fueron…..

Podemos observar que tenemos un código completamente funcional, que toma como objetivo la seguridad del peatón. Dando una solución a la circulación más segura por las calles de Medellín, que ha dado un conflicto por mucho tiempo y no se le había buscado una solución.

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

1. INTRODUCCIÓN

Con el acoso callejero creando peligros en el día a día, es de gran importancia priorizar la integridad de los ciudadanos, para así, otorgar una movilización segura y accesible para todos; y así, reducir el riesgo de crímenes en Medellín.

1.1. Problema

Se necesita un algoritmo que pueda calcular tres posibles caminos, en el que se tome en cuenta la reducción de tiempo y seguridad en el trayecto.

1.2 Solución

Para solucionar esta problemática y generar una forma segura de transitar por las calles. Se desea crear una aplicación que forme tres rutas en las que se ponga como prioridad el porcentaje de riesgo y la reducción de tiempo. Con esto, planeamos utilizar el algoritmo A*, que es el más eficaz para encontrar el camino más corto entre los grafos.

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 SafeRoute: aprendiendo a circular por las calles de forma segura en un entorno urbano

Estudios recientes muestran que el 85% de las mujeres han cambiado su ruta de viaje para evitar el acoso y las agresiones. A pesar de esto, las herramientas de mapeo actuales no brindan información a los usuarios para que se hagan cargo de su seguridad personal. SafeRoute es una solución novedosa al problema de navegar por las ciudades y evitar el acoso callejero y la delincuencia. A diferencia de otras aplicaciones de navegación por calles, SafeRoute presenta un nuevo tipo de generación de rutas a través de “deep reinforcement learning”. Esto nos permite optimizar con éxito para la búsqueda de rutas de criterios múltiples e incorporar el aprendizaje de representación dentro de nuestro marco.

SafeRoute aprende a elegir calles favorables para crear un camino corto y seguro con una función de recompensa que incorpora seguridad y eficiencia. Dado el acceso a informes de delitos recientes en muchas ciudades urbanas, entrenamos nuestro modelo para experimentos en Boston, Nueva York, y San Francisco. Probamos nuestro modelo en áreas de estas ciudades, específicamente en las regiones pobladas del centro donde caminan los turistas y aquellos que no están familiarizados con las calles. Evaluamos SafeRoute y mejoramos con éxito los métodos más avanzados hasta en un 17 % en la distancia promedio local de los delitos, mientras disminuimos la longitud de la ruta hasta en un 7 %.

2.2 Route-The Safe: Un modelo robusto para una predicción de ruta más segura usando datos de crimenes y accidentes

La seguridad y la protección se convirtieron en la máxima prioridad de las personas debido al creciente número de delitos en las ciudades. Incluso el uso de mapas de Google mientras viaja, puede poner en peligro la vida y situaciones peligrosas. Muchas mujeres toman caminos diferentes a los recomendados por Google Maps y otras aplicaciones similares debido a problemas de seguridad. Las personas que son locales de las ciudades puede saber qué rutas son seguras para viajar, pero las personas que son turistas o nuevas en el ciudad confían en sus conductores o invierten mucho tiempo en investigar sobre la seguridad del área. Para lidiar con un problema tan creciente relacionado con la seguridad, las personas necesitan soluciones mejores y más eficientes.

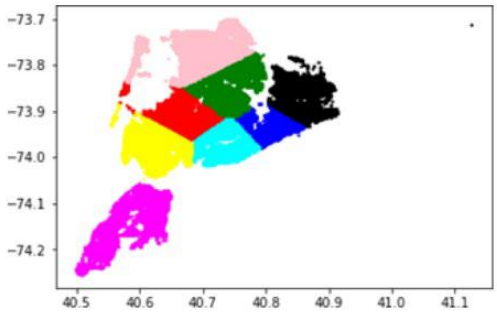
La necesidad de una solución que sugiera una ruta segura se está volviendo mucho más importante de lo que era. Con la ayuda de tal solución, las personas se sentirán más seguras que antes mientras viajen.

Se realizó un procesamiento previo de los datos para obtener resultados mejores y más precisos. en preprocesamiento de

datos se realizaron los siguientes pasos: el conjunto de datos de crimen y accidente el conjunto de datos se importó del sitio web de Nypd, de ambos conjuntos de datos, las columnas que no se requerían se eliminaron, las filas de ambos conjuntos de datos que tenían nan o null se eliminaron los valores, se analizaron y descartaron los valores atípicos, se cambió el nombre de las columnas para mejor comprensión, creó una nueva columna en el conjunto de datos de accidentes denominada (AS) como que se muestra en Algo 1, eliminó los delitos del conjunto de datos de delitos que no fueron de utilidad en el model, creó una nueva columna en el conjunto de datos de delitos denominada (CS) como se muestra en Algo 1, asignó las puntuaciones de delincuencia entre 1 y 15, creó una nueva columna en ambos conjuntos de datos nombrado como 'Índice', asignado 'a' a todas las filas del conjunto de datos de accidentes y 'c' a todas las filas de conjunto de datos de delincuencia, creó dos nuevos conjuntos de datos que consisten solo en Manhattan Brough de conjunto de datos de accidentes y delitos , concatenó los conjuntos de datos de delitos y accidentes de Manhattan.

Ahora, el siguiente paso es el diseño del modelo: el agrupamiento anidado de K Means se realizó en función de latitud y longitud. Agrupamiento de K Means realizado por primera vez: K Means El agrupamiento se realizó después de determinar el número de agrupaciones que se formarán utilizando el método del codo K Significa agrupamiento realizado por segunda vez: K Significa agrupamiento se hizo en uno de los grupos ya formados del agrupamiento anterior después de determinar el número de conglomerados a formar usando el método del codo, centroides de grupos recién formados utilizando el agrupamiento K Means.

El resultado de la agrupación de K Means en toda la ciudad de Nueva York se realiza en función de la latitud y la longitud.



Los regresores KNN que predijeron la puntuación de accidentes y delitos se analizan utilizando la puntuación R2. R-Squared también llamado coeficiente de determinación calcula estadísticamente que tan cerca están los datos de la línea de regresión ajustada. Entonces, si el valor de R-Squared es alto, el modelo se ajustará mejor a sus datos.

2.3 Siempre Seguras.

Angelina Espejel y Ortiz Espinoza son las creadoras de la aplicación “Siempre Seguras”, que nació con el objetivo de proteger a las mujeres del acoso callejero, generando un mapeo para identificar las zonas donde existe mayor incidencia de acoso; y así, darle visibilidad a la realidad de muchas mujeres que corren un peligro constante.

Angelina creó su propio algoritmo, donde recaudó los primeros datos de información. Su objetivo fue la aplicación de Twitter, donde es regular que las personas descarguen sus emociones de experiencias vividas recientemente; así se podrá recaudar y analizar los tuits de mujeres que sufrieron algún tipo de acoso. Se localiza el origen de estos tuits y se generará un mapa de lugares recurrentes a estos peligros.

Con la información recolectada de la aplicación, se ha descubierto que las mujeres sufren mayor acoso en transporte público y en las mañanas, cuando salen a hacer ejercicio. Con el constante peligro que puede significar esto, estas mujeres se han reunido con autoridades estatales para exponer su problemática y convencer a desarrollar y mejorar la aplicación. Actualmente, la aplicación se encuentra en fase beta, solo disponible en México.

2.4 Hollaback

En 2005, Nueva York. Una mujer, Thao Nguyen, denuncio a la policía un problema de acoso, donde un hombre se estaba masturbando delante de ella; la policía no hizo nada al respecto, pero ella pudo tomar justicia de su propia mano y publicar una foto del hombre en sus redes sociales. Eso bastó para inspirar a personas en crear un foro para denunciar este tipo de acoso y buscar erradicarlo, visibilizarlo y crear una red de apoyo para las víctimas.

De un foro a una aplicación, nació Hollaback. El funcionamiento de esta plataforma es sencillo: el registro normal, a través de cuentas externas como Google o Facebook; con esto ingresas a la página principal de la aplicación, que es un mapa, donde aparecen puntos rojos y verdes. Cada punto es una persona que ha vivido una experiencia por esa zona, los puntos rojos son ciudadanos que les sucediera algún tipo de acoso y cuentan su experiencia (en los que muchas adjuntan fotos de sus acosadores); Y los puntos verdes son personas que lograron presenciar algún tipo de acoso hacía otra persona.

Con esto, se ha ayudado a identificar y protegerse de muchos hombres, estar prevenidos en algunos sectores y darle voz a las víctimas que tuvieron un abuso moral (o incluso físico) a levantar la voz.

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].

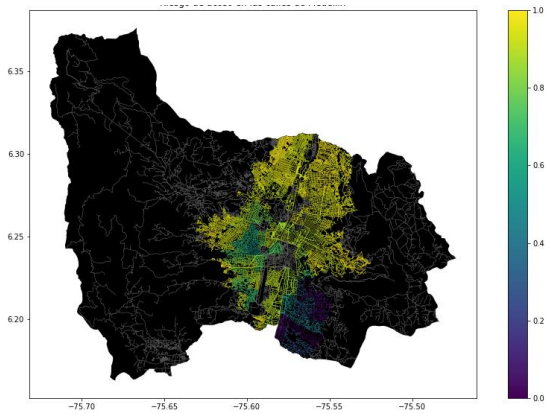


Figura 1. Riesgo de acoso sexual calculado como una combinación lineal de la fracción de hogares que se sienten inseguros y la fracción de hogares con ingresos inferiores a un salario mínimo, obtenidas de la Encuesta de Calidad de Vida de Medellín, de 2017.

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.

3.2.1 Algoritmo Best-first search

El Algoritmo “Best-first Search” es una estrategia de control sistemático que combina “breadth-first” y “depth-first search”. La principal diferencia entre “Best -first Search” y las técnicas de “Brute Force Search” es que hacemos uso de una evaluación o función heurística para ordenar los objetos SearchNode en la cola. De esta manera, nosotros elegir el SearchNode que parece ser el mejor, antes que cualquier otro, independientemente de su posición en el árbol o gráfico. Intenta expandir el nodo que está más cerca de la meta, sobre la base de que esto es probable para llegar a una solución rápidamente. Por lo tanto, evalúa los nodos usando solo la heurística función: f(n) = h(n).

En sus características tenemos que:

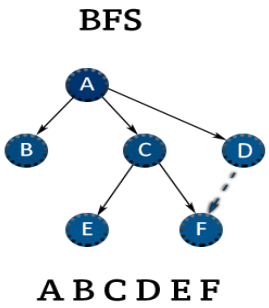
- Al igual que la “depth-first” y la “breadth-first search”, la “best-first search” utiliza dos listas. Abierto: para realizar un seguimiento de la frontera de la búsqueda. Cerrado: para registrar estados ya visitado.
- Ordena los estados abiertos de acuerdo con alguna estimación heurística de su cercanía a un objetivo.
- En cada iteración a través del ciclo, considere el estado más prometedor en la siguiente lista abierta.
- Al visitar un estado secundario, si ya está abierto o cerrado, el algoritmo comprueba si se llega al hijo

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

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

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

por un camino más corto, esta vez en comparación con el último tiempo que llegó a este.

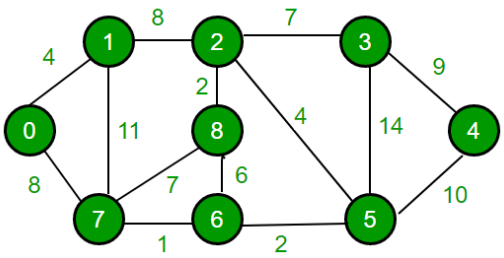


3.2.2 Algoritmo de Dijkstra

• Si los pesos de mis aristas son de valor 1, entonces bastará con usar el algoritmo de BFS.
• Si los pesos de mis aristas son negativos no puedo usar el algoritmo de dijsktra, para pesos negativos tenemos otro algoritmo llamado Algoritmo de Bellmand-Ford.

El algoritmo trabaja de la siguiente forma:
Primero marcamos todos los vértices como no utilizados. El algoritmo parte de un vértice origen que será ingresado, a partir de ese vértices evaluaremos sus adyacentes, como dijkstra usa una técnica greedy – La técnica greedy utiliza el principio de que para que un camino sea óptimo, todos los caminos que contiene también deben ser óptimos- entre todos los vértices adyacentes, buscamos el que esté más cerca de nuestro punto origen, lo tomamos como punto intermedio y vemos si podemos llegar más rápido a través de este vértice a los demás. Después escogemos al siguiente más cercano (con las distancias ya actualizadas) y repetimos el proceso. Esto lo hacemos hasta que el vértice no utilizado más cercano sea nuestro destino. Al proceso de actualizar las distancias tomando como punto intermedio al nuevo vértice se le conoce como relajación (relaxation).

Dijkstra es muy similar a BFS, si recordamos BFS usaba una Cola para el recorrido para el caso de Dijkstra usaremos una Cola de Prioridad o Heap, este Heap debe tener la propiedad de Min-Heap es decir cada vez que extraiga un elemento del Heap me debe devolver el de menor valor, en nuestro caso dicho valor será el peso acumulado en los nodos.

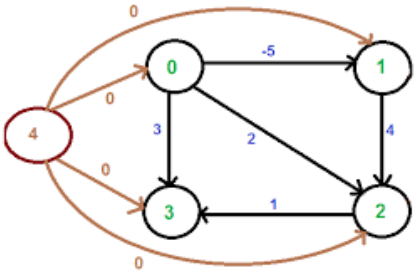


3.2.3 Algoritmo de Johnson

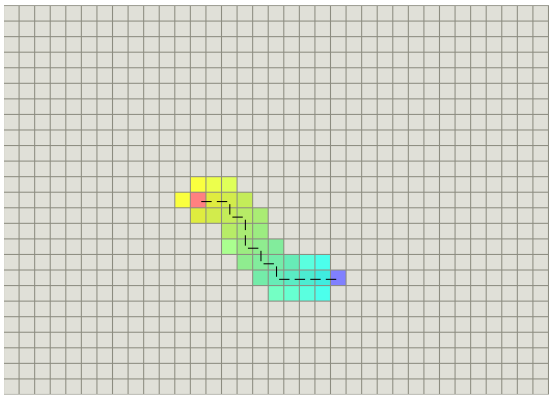
El algoritmo de Johnson consiste en los siguientes pasos:

1. Primero se añade un nuevo nodo q al grafo, conectado a cada uno de los nodos del grafo por una arista de peso cero.
2. En segundo lugar, se utiliza el algoritmo de Bellman-Ford, empezando por el nuevo vértice q, para determinar para cada vértice v el peso mínimo h(v) del camino de q a v. Si en este paso se detecta un ciclo negativo, el algoritmo concluye.
3. Seguidamente, a las aristas del grafo original se les cambia el peso usando los valores calculados por el algoritmo de Bellman-Ford: una arista de u a v con tamaño w(u, v), da el nuevo tamaño w(u, v) + h(u) – h(v)
4. Por último, para cada nodo s se usa el algoritmo de Dijkstra para determinar el camino más corto entre s y los otros nodos, usando el grafo con pesos modificados.

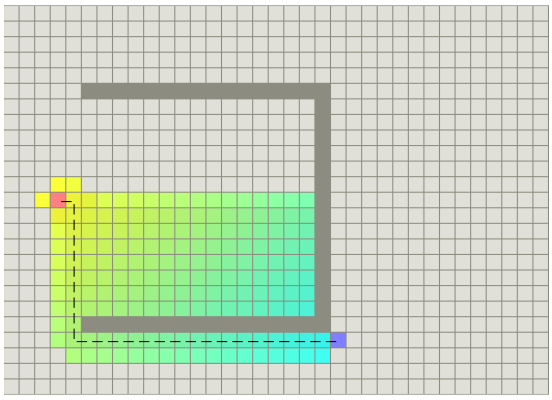
En el grafo con pesos modificados, todos los caminos entre un par de nodos s y t tienen la misma cantidad h(s) – h(t) añadida a cada uno de ellos, así que un camino que sea el más corto en el grafo original también es el camino más corto en el grafo modificado y viceversa. Sin embargo, debido al modo en el que los valores h(v) son computados, todos los pesos modificados de las aristas son no negativos, asegurando entonces la optimalidad de los caminos encontrados por el algoritmo de Dijkstra. Las distancias en el grafo original pueden ser calculadas a partir de las distancias calculadas por el algoritmo de Dijkstra en el grafo modificado invirtiendo la transformación realizada en el grafo.



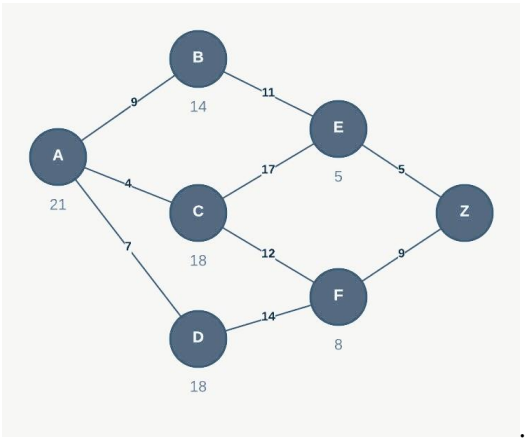
3.2.4 Algoritmo A*



En el ejemplo con un obstáculo cóncavo, A* encuentra un camino tan bueno como el que encontró el Algoritmo de Dijkstra:



El secreto de su éxito es que combina la información que usa el algoritmo de Dijkstra (que favorece los vértices que están cerca del punto de partida) y la información que usa Greedy Best-First-Search (que favorece los vértices que están cerca del objetivo). En la terminología estándar utilizada cuando se habla de A*, g(n)representa el costo exacto de la ruta desde el punto de partida hasta cualquier vértice ny h(n)representa el costo heurístico estimado desde el vértice nhasta la meta. En los diagramas anteriores, el amarillo (h) representa los vértices lejos de la meta y el verde azulado (g) representa vértices alejados del punto de partida. A* equilibra los dos a medida que avanza desde el punto de partida hasta la meta. Cada vez que pasa por el bucle principal, examina el vértice nque tiene el valor más bajo f(n) = g(n) + h(n)

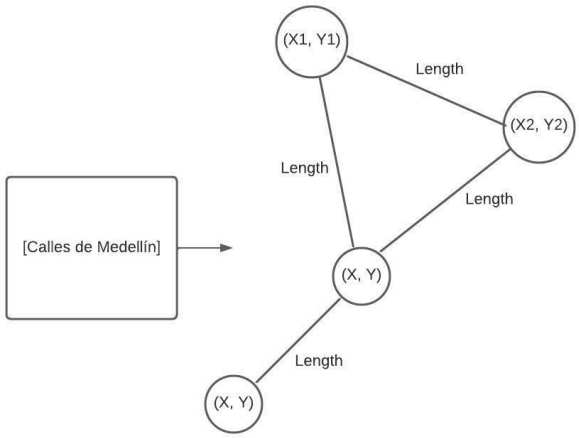


4. DISEÑO E IMPLEMENTACIÓN DEL ALGORITMO

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

Para guardas de una manera más fácil las calles en Medellín. Creamos una lista en la que se almacena toda la información para, posteriormente, llevarla a un grafo. Se organiza de tal manera, siendo los vértices las coordenadas de inicio y final de las calles, y las aristas son la distancia que hay entre estos dos puntos.



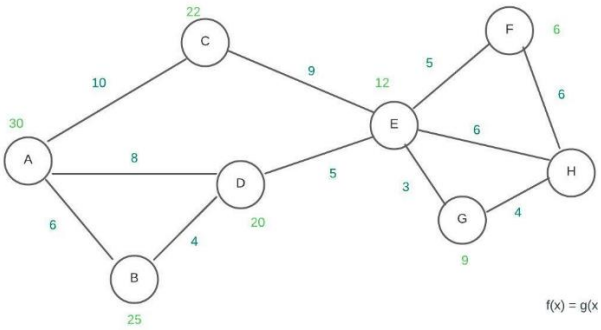
4.2 Algoritmos

[4] https://github.com/ManuCarov/Proyecto.git

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 A*

El algoritmo A-Star se le conoce como una variación más optimizada del Dijkstra. Sirve para encontrar rutas más cortas por medio de los grafos.



¿Cómo funciona?: El algoritmo se maneja en términos de costo, definiendo cuál camino cuesta menos en recorrerlo.

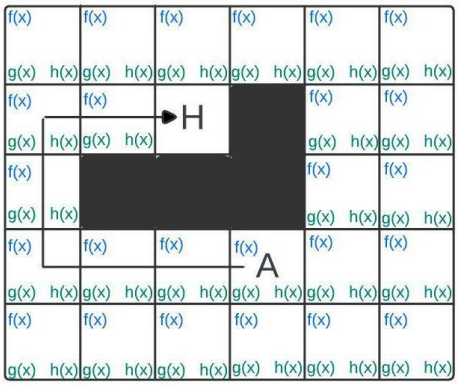
La manera de concretar cada ruta sería por la fórmula:

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

Donde g(n) sería el valor de la arista, camino de un nodo al siguiente nodo.

Y h(n) sería el valor de distancia entre el nodo de destino y *n* nodo.

Al sumar estos dos valores de las múltiples rutas, se selecciona la ruta que tenga el valor de f(n) y es la ruta seleccionada.



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

Para los dos nuevos caminos generados por el código, en el algoritmo A* se cambia la heurística, que en este caso sería el riesgo de acoso en las calles de Medellín, por datos extremos. Si aumentamos la heurística, se muestra el que sería el camino más peligroso (línea roja) y si se disminuye la heurística, se generaría el camino más seguro (línea azul)

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

Para la complejidad temporal, se analizó que el peor de los casos del A* sería convertirse el algoritmo Dijkstra.

Hallar la complejidad de la memoria fue definir las iteraciones de cada línea, dándoles un valor de 1, n y n^2, dependiendo de la estructura. Al final se saca el resultado mayor y este define la complejidad de Big(O).

Algoritmo	Complejidad temporal
A*	O(E + VlogV)

Tabla 1: Complejidad temporal del algoritmo A*, donde V son las calles y E las intersecciones

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

Tabla 2: Complejidad de memoria del Grafo, donde V es la máxima iteración de una variable.

4.4 Criterios de diseño del algoritmo

Se escogieron los grafos como nuestra estructura de datos, ya que es un complemento para el algoritmo A*. Con este, podemos obtener caminos óptimos para el trayecto, donde escogemos como prioridad la seguridad vial.

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	7321.27(m)	0.56
Eafit	Unal	9342.39(m)	0.78
Eafit	Unal	25777.88(m)	0.15

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.

Tiempos medios de ejecución (s)
2.513915 s
4.636644 s
1.809330 s

Tabla 4: Tiempos de ejecución del nombre del *algoritmo (Por favor, escriba el nombre del algoritmo, por ejemplo, DFS, BFS, A*)* para cada uno de los tres caminos calculadores entre EAFIT y Universidad Nacional.

6. CONCLUSIONES

Ahora, con una solución visual al problema que planteamos desde el inicio de este informe. Podemos observar la generación de 3 rutas, desde el punto A al punto B, relativamente distintas. Donde, ahora, brindamos una ayuda al ciudadano peatonal, expuesto día a día a los peligros de las calles, exclusivamente mujeres, Ahora que tienen la opción de caminar en un ambiente más seguro y verificar cuales son las calles más peligros para no estar cerca de algún riesgo.

Tenemos una gran mejora para los colombianos; que con un trabajo profesional más avanzado, puede ser una idea muy exitosa y solicitada; ya que nuestro código necesita ayuda para mejorar la optimización y así mostrar los resultados de una formar más rápida, para la comodidad del usuario. De igual manera, fue una experiencia muy favorecedora.

6.1 Trabajos futuros

Nos gustaría mejorar en la optimización a la hora de crear los 3 caminos, para dar una mejor eficiencia a un usuario. Además de una mejor forma de visualizar los datos en un mapa dentro del mismo programa de Python.

AGRADECIMIENTOS

Esta investigación ha sido parcialmente apoyada por Generación E, Donante.

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

Observatorio violencia, 2017. «Hollaback!»: Una App contra el acoso callejero. https://observatorioviolencia.org/hollaback-una-app-contra-el-acoso-callejero/

Forbes México, 2021. Siempre Seguras: la app para mapear el acoso sexual callejero en México. https://www.forbes.com.mx/tecnologia-siempre-seguras-app-mapear-acoso-sexual-callejero-mexico/

College of IT – University of Babylon, 2014. Best-First Search Algorithm & Greedy Search Algorithm. https://www.uomus.edu.iq/img/lectures21/MUCLecture_2021_123356.pdf

Algorithms and More, 2012. Camino mas corto: algoritmo de Dijkstra. https://jariasf.wordpress.com/2012/03/19/camino-mas-corto-algoritmo-de-dijkstra/

Wikipedia, 2022. Johnson's algorithm. https://en.wikipedia.org/wiki/Johnson%27s_algorithm

Red Blob Games, 2022. Pathfinding with A* http://theory.stanford.edu/~amitp/GameProgramming/

Cornell University, 2018. SafeRoute: Learning to Navigate Streets Safely in an Urban Environment. https://arxiv.org/abs/1811.01147

Algoritmo A*. (s. f.). Grapheverywhere. Recuperado 4 de octubre de 2022, de https://www.grapheverywhere.com/algoritmo-a/