

MEDELLÍN SEGURA: ALTERNATIVA DE SOLUCIÓN AL ACOSO SEXUAL CALLEJERO.

Juan Camilo Villa Correa Universidad Eafit Colombia jcvillac@eafit.edu.co	Kristian Restrepo Osorio Universidad Eafit Colombia krestrepoo@eafit.edu.c o	Emanuel Patiño Vera Universidad Eafit Colombia epatinov@eafit.edu.co	Andrea Serna Universidad Eafit Colombia asernac1@eafit.edu.co	Mauricio Toro Universidad Eafit Colombia mtorobe@eafit.edu.co
---	---	---	--	--

RESUMEN

El problema a abordar en este proyecto es el acoso sexual callejero, el cual es un tema muy importante en un país como el nuestro donde hay tasas tan altas de acoso, por esto, es esencial que se generen alternativas eficientes para tratar estos problemas sociales. Así, buscar una solución al temor ocasionado en muchas de las personas de distintos géneros, intentando alcanzar la seguridad en las mismas, y que puedan vivir “tranquilas”, sin sentirse expuestas ante la ignorancia de quienes producen este problema, no solo desde la parte sexual, como lo estamos tratando en este proyecto, sino también con otros problemas relacionados como el machismo, los otros distintos tipos de acoso, la inseguridad, etc.

Palabras clave

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

1. INTRODUCCIÓN

Lo que nos motivó a realizar esta alternativa de solución, es buscar el vivir en una sociedad mejor, donde las personas sientan seguridad al realizar cualquiera de sus cometidos diarios, sin pensar en que pueden encontrarse con alguien quien los puede acosar sexualmente y producir un sentimiento de riesgo y miedo. Es importante para nosotros buscar que las personas no se expongan ante dicho peligro, y además de esto, que puedan llegar de la manera más rápida a su lugar de destino, por lo tanto, buscamos generar 3 caminos por donde las personas puedan evitar ese tipo de peligro.

Desde hace ya varios años atrás, sabemos lo que es en nuestro país este tipo de problema, y más específico en nuestra región, Antioquia, donde se han evidenciado muchos casos de acoso sexual, principalmente ante las mujeres. Es frustrante escuchar tantos casos sobre este problema y que no se haga nada al respecto, así que es una motivación buscar dar un paso por el bien común y por la seguridad de quienes queremos.

1.1. Problema

El problema que abordaremos consiste en calcular tres caminos diferentes que reduzcan tanto la distancia como el riesgo de acoso, esto siendo vital para ayudar a las personas a buscar el camino más seguro. El impacto que este problema tiene en la sociedad principalmente radica en la inseguridad de las personas para desplazarse de un lugar a otro y, por tanto, además de ser útil resolver este problema por medio de la tecnología, también se ha vuelto una necesidad realizarlo ya que no es justo para nadie el “vivir con miedo”, y si existen herramientas con la capacidad de crear un tipo de solución a problemas como estos, es una obligación hacer uso de ellas para buscar el mejoramiento de la sociedad, que tanto las mujeres, como quienes son acosados sexualmente, tenga una alternativa para evitar pasar por momentos traumáticos y evitar ciertos peligros más que se generan a partir de esto.

1.2. Solución

Para solucionar este problema, utilizaremos el Algoritmo de Dijkstra, el cual procesará los datos almacenados en forma de grafo, y nos brindará una serie de coordenadas que representan la ruta más corta y segura entre un origen y un destino particular. De forma breve, se explica en que consiste el algoritmo de Dijkstra en el siguiente párrafo tomado de un artículo publicado por la UDB:

Dijkstra es un algoritmo eficiente que sirve para encontrar el camino de coste mínimo desde un nodo origen a todos los demás nodos del grafo. Este algoritmo es un típico ejemplo de algoritmo ávido, que resuelve los problemas en sucesivos pasos, seleccionando en cada paso la solución más óptima con el objeto de resolver el problema.

El fundamento sobre el que se basa este algoritmo es el principio de optimizar: si el camino más corto entre los vértices “u” y “v” pasa por el vértice “w”, entonces la parte del camino que va de “w” a “v” debe ser el camino más corto entre todos los caminos que van de “w” a “v”. [11]

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 La Ruta -Segura

La aplicación **La Ruta - Segura** se centra en lidiar con el creciente problema relacionado con la seguridad. Muchas de las aplicaciones que se utilizan para calcular la ruta más corta entre dos puntos, como lo es el caso de Google Maps, pueden llegar a poner en peligro la vida de las personas al viajar, ya que no toma en cuenta la seguridad de las rutas. Es por esto que, **La Ruta - Segura** se enfoca en la necesidad de una solución que sugiere una ruta segura y que, gracias a la ayuda de tal solución, las personas puedan sentirse más seguras mientras viajan.[1]

Así, la aplicación puede:

- Predecir una ruta segura haciendo uso de los datos de delincuencia y accidentes, sin dejar de lado la distancia entre el punto de origen y el destino.
- Divide la ciudad de Nueva York en pequeñas regiones de riesgo aplicando un agrupamiento anidado de los datos, brindando mejores predicciones ya que también se toman en cuenta las áreas de riesgo más pequeñas.
- Calcula la puntuación de riesgo basándose en la puntuación de riesgo de los clústeres más cercanos.

Como se puede ver, la aplicación se desarrolla tomando datos de la ciudad de Nueva York en Estados Unidos, esos datos son brindados por las autoridades locales y se basan en índices de criminalidad de la ciudad. Se debe tener cuidado al utilizar aplicaciones de este tipo, ya que también tienden a presentar algunas desventajas como:

- Se utilizan sólo datos de delitos que son de naturaleza altamente subjetiva.

- Los datos no se actualizan.
- Los enfoques utilizados ignoran las áreas delictivas más pequeñas.

El algoritmo toma datos de NYC DataAbierta, los cuales indican los tipos de crímenes cometidos y su recurrencia en lugares de la ciudad

Después, se siguen algunos pasos:

- Inicializar la API de Google Maps y configurar su función Gmaps, la cual es necesaria para realizar diferentes operaciones.
- Se procesan los datos cargados de NYC DataAbierta, una base de datos.

El siguiente paso es el etiquetado de puntuación de accidentes de un punto (AS) y puntuación de delincuencia de un punto (CS). Se tiene mucho cuidado al asignar valores (CS) a diferentes tipos de delitos ofensas. La asignación de valores (CS) a diferentes delitos es una suposición basada en la clase a la que pertenece el delito y el tipo de pena de prisión entregado al sospechoso. El conjunto de datos de arrestos se utiliza en este documento para averiguar qué tipo de crimen ocurrió en qué lugar. En la siguiente tabla se muestran los valores de diferentes delitos. La puntuación de incidentes de un punto (AS) se calcula como se muestra:

DELITO	PUNTUACION CRIMEN
Violación	15
Crímenes sexuales	15
Homicidio-vehículo negligente	14
Asesinato y Homicidio involuntario	14
Homicidio-negligente	14
Delitos relacionados con niños	13

- $AS = PK * 2 + CK * 2 + MK * 2 + PI + CI + MI$
- Los valores asignados en esta ecuación corresponden a suposiciones después de un análisis exhaustivo [1].

Resultados:

El programa muestra la ruta más corta y segura (con menos puntuación de crimen) para ir de un punto A, a un punto B, como se observa a continuación:



2.2 Recomendador de ruta segura

Es una aplicación que busca resolver el problema de aplicaciones como Google Maps, las cuales solo se centran en encontrar la ruta más corta o rápida entre dos puntos, pero dejan de lado la seguridad de esas mismas rutas. *Safe Path Recommender* es una aplicación web que, basada en la investigación, encuentra la ruta más segura considerando el índice de peligro de los caminos posibles.[2]

Esta aplicación es desarrollada por un grupo de estudiantes en Delhi, capital nacional de La India, la cual ha sido considerada como un centro de crímenes como asesinatos, secuestros, robos, entre otros. La gente camina por las calles, pero no están seguras, regresan del trabajo, fiestas, cenas nocturnas, parques, etc. Ni siquiera los niños están seguros al volver de la escuela.

Para la aplicación, se utilizó información de los sitios "delhistats.com" and "datagov.in"(Open Government Data (OGD) Platform India). De los dos conjuntos de datos, uno contiene datos de Crímenes en India, mientras que el otro contiene datos de Crímenes en Delhi.

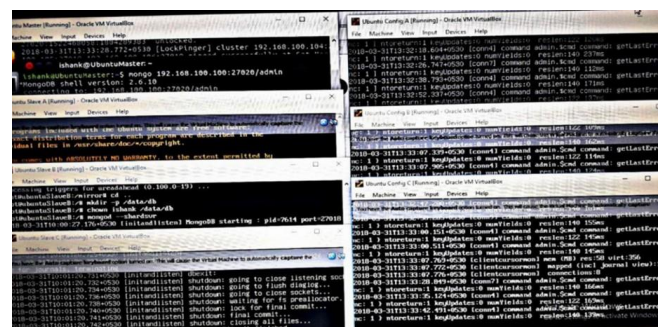
Aquí algunos detalles sobre el desarrollo de la app:

- Para la investigación se implementó una aplicación web en Cloud 9, proporcionada por AWS- amazon web services.
- Se dividió el Backend en base de datos distribuida en 4 nodos, los cuales representan zonas diferentes (Este, Oeste, Norte, Sur). Siendo "Norte" el "Master node" y los demás los "Slave nodes".

Se utilizó la base de datos MongoDB, la cual es un programa de código abierto que se utiliza para documentación, esta usa documentos json con esquemas.

Se utilizó Google Maps Api de geocodificación inversa usando las longitudes y latitudes del conjunto de datos para ser más precisos. En los datos sobre "Crimen en la India" que se implementaron más API como:

- API de JavaScript de Google Maps
- API de inserción de Google Maps
- API de indicaciones de Google Maps
- API de geolocalización de Google Maps
- Servicio web de la API de Google Places:



Estas son algunas de las imágenes compartidas por sus desarrolladores en las que se muestra la configuración de los servidores y cada uno de los nodos.[2]

Resultados:

- La aplicación web muestra diferentes rutas clasificadas por colores de acuerdo con el nivel de riesgo en cada una de ellas.
- El algoritmo muestra una calificación de 0-4 de acuerdo al nivel delictivo del camino.
- Una calificación de peligro de '0' significa que el lugar es relativamente seguro (con menos antecedentes penales en el pasado), mientras que un índice de 4 significa que el lugar tiene altos antecedentes penales en el pasado.



Aplicación web que muestra una ruta segura entre "Saket, Nueva Delhi, India" y "Karkardooma, Nueva Delhi, India" con los diferentes marcadores (Calavera, Triángulo de signo de exclamación, Smiley, Green Ticks, etc.)[2]

2.3 Encontrando rutas seguras

El objetivo de esta aplicación desarrollada por estudiantes universitarios de México, es proporcionar rutas seguras con un sistema de información móvil para aumentar el nivel de confianza de los ciudadanos que utilizan la infraestructura urbana de la ciudad. Para esto, el enfoque combina datos estadísticos obtenidos de información oficial, con la percepción de las personas a diario y reflejada por datos de colaboración abierta. Por lo tanto, se toman en cuenta eventos para un tiempo en particular, y se realizan actualizaciones continuas para proporcionar eventos recientes que ocurren en un momento específico, lugares que son reportados por una comunidad de redes sociales. El método basado en crowdsourcing considera una gran base de datos de tweets, que fueron recopilados por la aplicación móvil, mientras que los datos oficiales fueron dados por las autoridades locales.[3]

- Para la realización de esta app se desarrollan 5 etapas:
- Recuperación de fuentes de datos sobre delitos
- El depósito de datos sobre delitos

- El procesamiento de los datos
- La agrupación de los datos
- Generación de rutas seguras

Al basarse en fuentes de información no oficiales como lo son usuarios de Twitter, se desarrolló un algoritmo que permite dar un valor a cada delito de acuerdo a su naturaleza.

Información oficial								
Event type	Year: 2012				Year: 2013			
	Q ₁	Q ₂	Q ₃	Q ₄	Q ₁	Q ₂	Q ₃	Q ₄
Robbery of passenger in public transport	159	175	154	111	102	134	115	86
Robbery of passer without violence	47	80	54	79	73	46	49	57

Input: $N = n_1, n_2, \dots, n_k$ Set of nodes
 n_i start node
 n_j finish node
Result: $D = d_1, d_2, \dots, d_k$ Set of distance values (weighted crime rate)

(1) Assign distance values $d(n_i) = 0, d(n_k) = \infty \forall n_i \neq n_k$
(2) Let $U = N - n_i$
(3) Let current node $n_i = n_i$
(4) while $exists(n_i)$ do
 foreach $neighbor(n_i)$ do
(6) Let $be_n = neighbor(n_i)$
(7) if $d(n_i) > length(n_i, n_j) + d(n_j)$ then
(8) $d(n_j) = length(n_i, n_j) + d(n_i)$
(9) $U = U - n_j$
(10) if $n_j \notin U$ or $min(length(n_i, n_j)) = \infty \forall n_i, n_j \in U$ then
(11) return $d(n_i) \forall n_i \in N$
(12) else
(13) $n_i = d(n_i) = min(d(n_j)) \forall n_j \in U$

Crímenes reportados por los usuarios de twitter						
SSPDFVIAL	Mexico City	07.14.2010	369,115	154.65	Yes	http://ssp.df.gob.mx/
PolloVial	Mexico City	01.31.2013	667	71.91	No	No website
Trafico889	Mexico City	05.14.2009	137,099	90.54	No	http://siempre889.com/trafico/
Alertux	Mexico City	10.16.2012	179,574	35.59	No	http://www.alertux.com/
072AvialCDMX	Mexico City	10.20.2010	83,535	134.71	Yes	http://www.agu.df.gob.mx/
RedVial	Mexico City	03.09.2010	63,702	44.81	No	http://rvial.mx/

Resultados:

La aplicación móvil se implementó en Android 4.0, y las pruebas se realizaron en dispositivos móviles. El repositorio de eventos reportados, está compuesto por 5.441 sucesos, que fueron recopilados y procesados a partir de los tweets y correlacionado con informes oficiales. Basando en este conjunto de datos, se genera la tabla que indica el número de veces que cada atributo aparece, dados eventos como robo y crimen. Aquellos valores con menos frecuencia, nos indican cuando una ruta es segura. Todos los valores se asignan a los correspondientes nodos en la red. Además de la probabilidad de que un evento ocurra en algún lugar, también se calcula la fecha específica y se almacena en una tabla de información.[3]

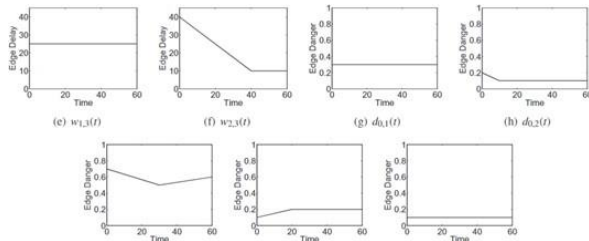
2.4 Ruta segura más corta

Debido al creciente interés en los sistemas de transporte avanzados, este trabajo, busca obtener la información sobre el camino más corto en una gran red dinámica en tiempo real y, por lo tanto, el problema de la ruta más corta dependiente del tiempo. Además de esto, se busca dar solución al problema de la seguridad de dichas rutas.[4]

El algoritmo del programa **Ruta segura más corta**, se basa en tres pasos o condiciones principales que determinan la seguridad de las rutas:

- Caso 1: existe tanto una ruta segura con un tiempo de viaje relativamente largo, como una ruta insegura con un tiempo de viaje relativamente corto al mismo tiempo. En este caso devuelve la ruta segura.
- Caso 2: todas las rutas son seguras. En este caso devuelve el más corto.
- Caso 3: no existe un camino seguro para el recorrido dado. En este caso, devuelve un camino inseguro, pero más corto. La cual se podría decir que no es una opción tan buena.

Vemos que, en cada uno de estos casos se habla acerca de la vía más corta y su seguridad, pero ¿Cómo se determinan dichos atributos? Bien, para esto se desarrolla un algoritmo teniendo en cuenta múltiples factores como en todos los casos anteriormente descritos. A continuación, se presentan algunas de las imágenes compartidas por los autores del trabajo sobre su algoritmo utilizado.[4]



Todas estas, son gráficas que comparan el nivel de riesgo en función del tiempo de la ruta.

Resultados

Después de la realización de varias pruebas, el programa logró ser implementado usando C++, en una memoria CPU/1G de 3,0 GHz PC con XP. Además, se puso a prueba el algoritmo en un conjunto de datos reales con 16326 nodos y 26528 intersecciones (los nodos representan el inicio, el final y las intersecciones de las carreteras, mientras que los bordes representan la carretera). A partir de esto se generó una base de datos dependiente del tiempo y subgrafos que corresponden a las subáreas del experimento.

También se generaron funciones de factor de peligro lineales por partes continuas, estableciendo parámetros como el promedio de factores de peligro y la longitud del camino.[4]

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 API² 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 GitHub³.

¹ <https://www.openstreetmap.org/>

² <https://osmnx.readthedocs.io/>

³ <https://github.com/mauriciotoro/ST0245Eafit/tree/master/proyecto/Datasets/>

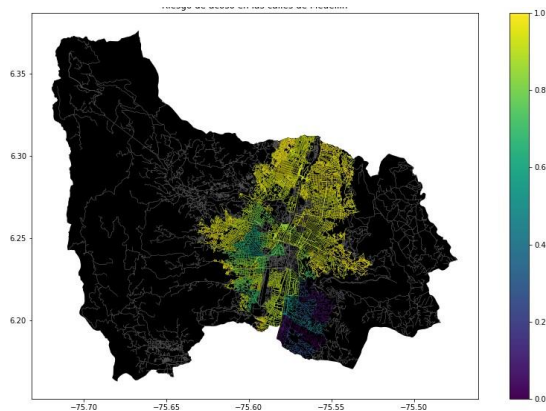


Figura 1. Riesgo de acoso sexual calculado como una combinación lineal de la fracción de hogares que se sienten inseguros y la fracción de hogares con ingresos inferiores a un salario mínimo, obtenidas de la Encuesta de Calidad de Vida de Medellín, de 2017.

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.

3.2.1 Algoritmo Dijkstra

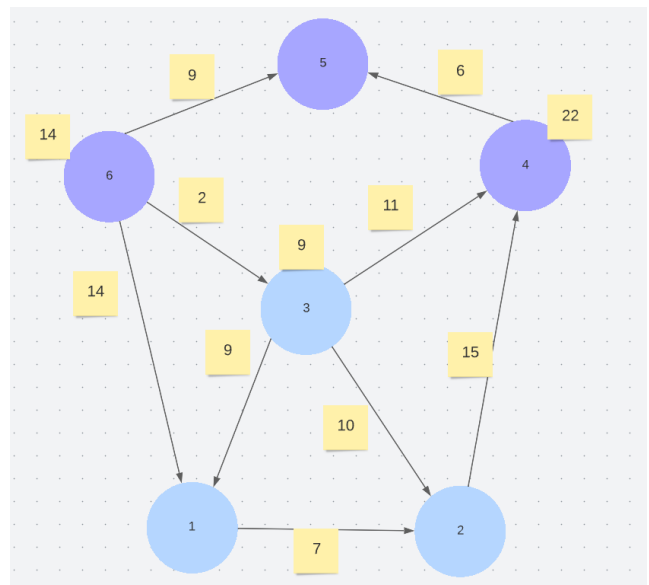
También llamado algoritmo de caminos mínimos es un algoritmo para la determinación del camino más corto dado un vértice origen al resto de los vértices en un grafo con pesos en cada arista. Este algoritmo fue descubierto por Edsger Dijkstra, un científico de la computación de los Países Bajos. La idea subyacente en este algoritmo consiste en ir explorando todos los caminos más cortos que parten del vértice origen y que llevan a todos los demás vértices; cuando se obtiene el camino más corto desde el vértice origen hasta el resto de los vértices que componen el grafo, el algoritmo se detiene.[6]

Pasos del algoritmo: primero Establezca todas las distancias de los vértices = infinito excepto el vértice de origen establezca la distancia de origen=0.[5]

En segundo lugar, empuje el vértice de origen en una cola de prioridad mínima en la forma (distancia, vértice), ya que la comparación en la cola de prioridad mínima será según las distancias de los vértices. [5]

En tercer lugar, haga estallar el vértice con la distancia mínima de la cola de prioridad (al principio, el vértice reventado = fuente).[5]

En cuarto lugar, Actualiza las distancias de los vértices conectados al vértice emergente en el caso de "distancia del vértice actual + peso del borde < distancia del siguiente vértice", luego empuje el vértice con la nueva distancia a la cola de prioridad. En quinto lugar, si el vértice reventado se visita antes, simplemente continúe sin usarlo. Aplique el mismo algoritmo nuevamente hasta que la cola de prioridad esté vacía.[5]



3.2.2 Algoritmo de Bellman Ford

El algoritmo de Bellman Ford se usa para encontrar las rutas más cortas desde el vértice de origen hasta todos los demás vértices en un gráfico ponderado. El algoritmo de Dijkstra resuelve este mismo problema en un tiempo menor, pero requiere que los pesos de las aristas no sean negativos, salvo que el grafo sea dirigido y sin ciclos. Por lo que el Algoritmo Bellman-Ford normalmente se utiliza cuando hay aristas con peso negativo. Este algoritmo fue desarrollado por Richard Bellman, Samuel End y Lester Ford.[7]

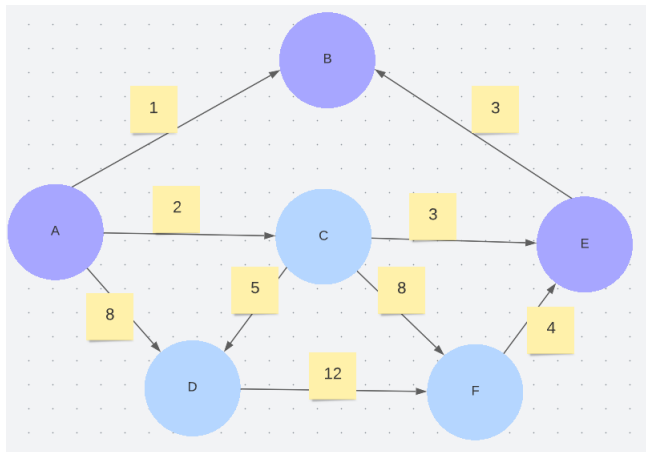
Pasos del algoritmo:

En primer lugar, el bucle exterior atraviesa desde 0: $n-1$.

En segundo lugar, recorra todos los bordes, verifique si la distancia del siguiente nodo > distancia del nodo actual + peso del borde, en este caso, actualice la distancia del siguiente nodo a "distancia del nodo actual + peso del borde". [5]

Este algoritmo depende del principio de relajación en el que la distancia más corta para todos los vértices se reemplaza gradualmente por valores más precisos hasta llegar finalmente a la solución óptima. [5]

Al principio, todos los vértices tienen una distancia de "Infinito", pero solo la distancia del vértice de origen = 0, luego actualice todos los vértices conectados con las nuevas distancias (distancia de vértice de origen + pesos de borde), luego aplique el mismo concepto para los nuevos vértices con nuevas distancias y así sucesivamente. [5]



3.2.3 Algoritmo de Floyd-Warshall

Al igual que los demás algoritmos su función es encontrar las rutas más cortas en un grafo ponderado dirigido con borde positivo o negativo. [8]

Pasos del algoritmo:

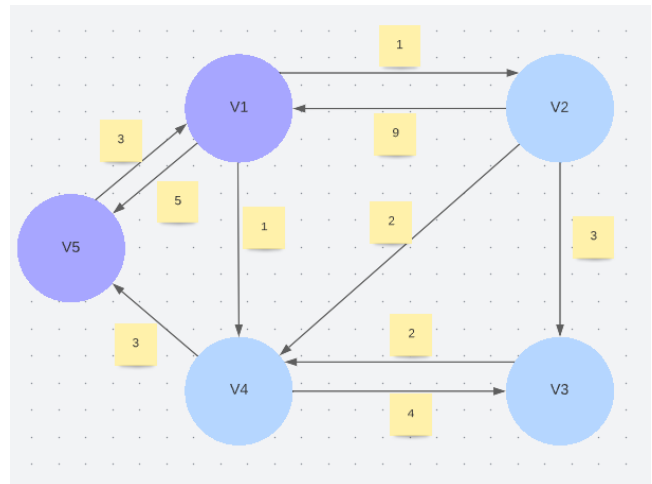
Formar las matrices iniciales C y D, donde C es la matriz de adyacencia, y D es una matriz del mismo tamaño cargada con valores iniciales $D_{ij} = i$. Se toma $k=1$. [10]

Se selecciona la fila y la columna k de la matriz C y entonces, para i y j, con $i \neq k$, $j \neq k$ e $i \neq j$, hacemos:

Si $(C_{ik} + C_{kj}) < C_{ij} \rightarrow D_{ij} = D_{kj}$ y $C_{ij} = C_{ik} + C_{kj}$. En caso contrario, dejamos las matrices como están. [10]

Si $k \leq n$, aumentamos k en una unidad y repetimos el paso anterior, en caso contrario páramos las iteraciones. [10]

La matriz final C contiene los costes óptimos para ir de un vértice a otro, mientras que la matriz D contiene los penúltimos vértices de los caminos óptimos que unen dos vértices, lo cual permite reconstruir cualquier camino óptimo para ir de un vértice a otro. [10]



3.2.4 El algoritmo de Johnson

El problema es encontrar los caminos más cortos entre cada par de vértices en un grafo dirigido ponderado dado y los pesos pueden ser negativos. Usando el algoritmo de Johnson, podemos encontrar los caminos más cortos de todos los pares en tiempo $O(V^2 \log V + VE)$. El algoritmo de Johnson usa Dijkstra y Bellman-Ford como subrutinas. [9]

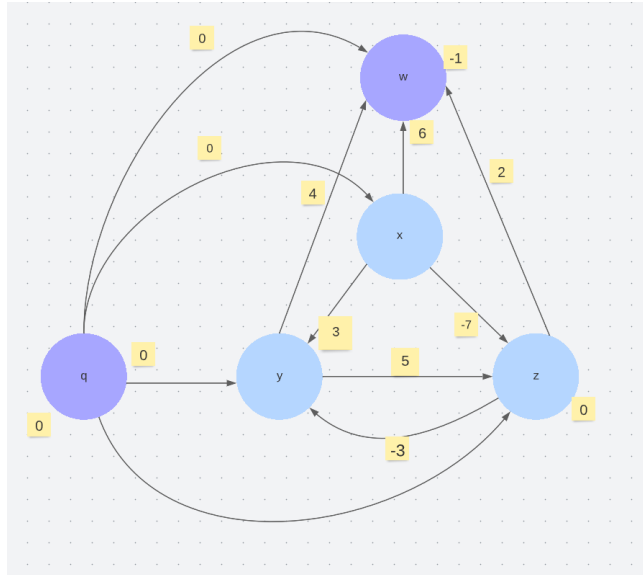
Pasos del algoritmo:

Sea G el grafo dado. Agregue un nuevo vértice "s" al grafo, agregue aristas desde el nuevo vértice a todos los vértices de G. Sea G' el grafo modificado. [9]

Se ejecuta el algoritmo Bellman-Ford en G' con s como fuente. Sean las distancias calculadas por Bellman-Ford $h[0], h[1], \dots, h[V-1]$. Si encontramos un ciclo de peso negativo, entonces regresa. Tenga en cuenta que el ciclo de peso negativo no puede ser creado por nuevos vértices ya que no hay borde en s. Todas las aristas son del s. [9]

Vuelva a ponderar los bordes del gráfico original. Para cada borde (u, v) , asigne el nuevo peso como “peso original + $h[u] - h[v]$ ”. [9]

Elimine los vértices agregados y ejecute el algoritmo de Dijkstra para cada vértice. [9]



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^[1].

4.1 Estructuras de datos

Para leer y almacenar la información contenida en el archivo CSV sobre las calles de Medellín, se utilizó un grafo, el cual se encuentra representado por un diccionario que contiene otros diccionarios en su interior, esto se debe a que existen vías denominadas “oneway”, las cuales nos demuestran que un origen también puede ser un destino. En dicho diccionario, las claves son las coordenadas y los valores contienen tanto el riesgo como la distancia representados por medio de la expresión $\text{distancia} \cdot \text{riesgo}$ que nos permite obtener el peso de cada nodo de la lista de adyacencia.

La estructura de los datos se presenta en la Figura 2.

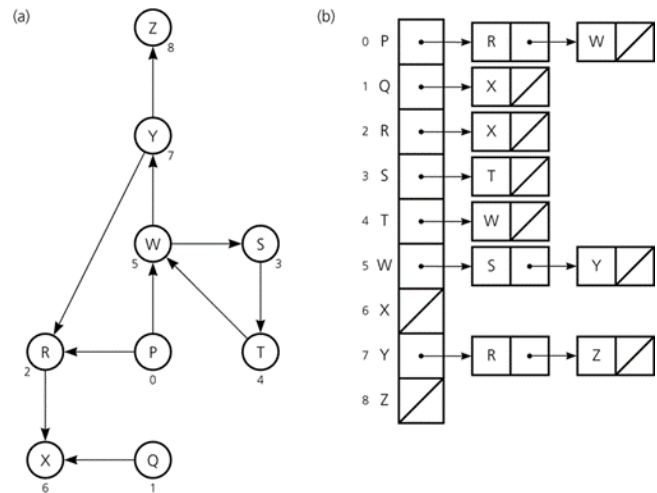


Figura 2: Un ejemplo de mapa de calles se presenta en (a) y su representación como lista de adyacencia en (b). En esta lista, observamos cada uno de los nodos o calles adyacentes que pueden existir en el grafo ya diseñado como se mencionó anteriormente.

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 de Dijkstra

Para el diseño del algoritmo, nuestro equipo utilizó el algoritmo de Dijkstra, para implantarlo en primer lugar realizamos la creación de un grafo, por medio de un diccionario de diccionarios el cual iba tener sus respectivos nodos y sus aristas, las cuales definían su peso mediante la multiplicación de la longitud y el riesgo, entonces entre más grande fuera la longitud o el riesgo más peso iba tener la arista. Una vez teniendo el grafo implantamos Dijkstra para encontrar el camino más corto y con menos riesgo entre un punto y otro, para Dijkstra implementamos una función la cual recibía el grafo y la coordenada de inicio (desde donde iba a comenzar nuestro camino) el primer paso que realizamos en la función fue crear un diccionario el cual iba almacenar todos los orígenes y estos serán los nodos que tendremos que ir visitando, luego asignamos a la distancia inicial un valor de 0 porque estamos en el nodo inicial, luego de ello creamos un diccionario donde se almacenarán todos los nodos previos, los cuales fueron la mejor opción con respecto a su peso, creamos dos listas, una en la se iban a ir almacenando los nodos visitados y la otra lista almacenaba los nodos no visitados que en principio eran todos, pero que en cada iteración cada nodo que eliminábamos de la lista de nodos no visitados iba siendo agregado a lista de visitados.

Creamos un nodo actual el cual será el nodo con la menor distancia. Del nodo actual iteraremos en cada uno de sus “vecinos”, creamos una variable distancia, la cual va a ser la suma de la distancia mínima inmediatamente anterior, mas el peso que le costaría llegar a ese nodo vecino, luego si esa distancia es menor a la que tiene actualmente el nodo, entonces asignaremos a esa distancia menor al “vecino” luego avanzamos al siguiente nodo y almacenamos ese nodo en el diccionario de nodos previos, todos los pasos anteriores se repetirán hasta que todos los nodos hallan sido visitados. Finalmente, la función retornará las distancias y todos los **nodos previos**, en otras palabras esos nodos previos serán como todos los caminos con sus menores distancias y riesgos de acoso a **cualquier destino** pero no al destino final, sino a cualquier destino, puesto que para calcular el camino más corto a **un destino en específico** creamos otra función, esa nueva función recibirá el inicio(nuestro punto de partida), el destino(nuestro punto de llegada) y todos los nodos previos que había retornado la otra función, los cuales tienen el camino más corto del inicio a cualquier lugar, creamos una lista vacía en cual se irán almacenando todos los nodos necesarios para llegar al destino, creamos un nodo actual que será nuestro destino, creamos un ciclo el cual se detendrá cuando ese nodo actual sea igual al final y en cada iteración del ciclo ira almacenando cada nodo(origen) por el que paso hasta poder llegar al inicio finalmente almacenamos el inicio y posteriormente invertimos la lista, ya que había quedado al “revés”. Finalmente, lo que retorna esa función es una lista con todos los orígenes(nodos) necesarios para llegar desde el inicio hasta el destino, siendo este el camino con menos distancia y menos riesgo de acoso, solucionado así el problema planteado en este proyecto.

El algoritmo se ejemplifica en la Figura 3.

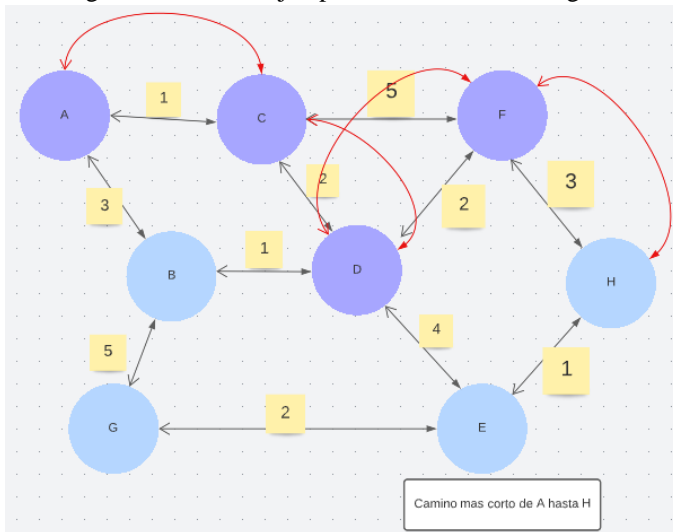


Figura 3: Cálculo de un camino que reduce tanto la distancia como el riesgo de acoso.

REFERENCIAS

1. Sandeep, C., Gauri Shankar, V., Soni, S. 2019. Route-The Safe: A Robust Model for Safest Route Prediction Using Crime and Accidental Data. Recuperado agosto 13, 2022 de https://www.researchgate.net/publication/338096313_Route-The_Safe_A_Robust_Model_for_Safest_Route_Prediction_Using_Crime_and_Accidental_Data
2. Agarwal, I. 2021. Safe Path Recommender based on Crime Statistics using Distributed Database. Recuperado agosto 13, 2022 de https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3833474
3. Felix Mata, M., Torres, M., Guzmán, G. Quitero, R. 2016. A Mobile Information System Based on Crowd-Sensed and Official Crime Data for Finding Safe Routes: A Case Study of Mexico City. Recuperado agosto 13, 2022 de https://www.researchgate.net/publication/298331916_A_Mobile_Information_System_Based_on_Crowd-Sensed_and_Official_Crime_Data_for_Finding_Safe_Routes_A_Case_Study_of_Mexico_City
4. Jigang, W., Jin, S., Ji, H. Srikanthan, T. 2011. Algorithm for Time-dependent Shortest Safe Path on Transportation Networks. Recuperado agosto 13, 2022 de <https://www.sciencedirect.com/science/article/pii/S1877050911001591>
5. Algoritmos de ruta más corta. Recuperado el 14 de agosto de 2022 de hackerearth: <https://www.hackerearth.com/practice/algorithms/graph/s/shortest-path-algorithms/tutorial/>
6. Algoritmo de Dijkstra. Recuperado el 14 de agosto de 2022 de Wikipedia: https://es.wikipedia.org/wiki/Algoritmo_de_Dijkstra#:~:text=Teorema%3A%20El%20algoritmo%20de%20Dijkstra.%F0%9D%91%BB_%F0%9D%92%85%F0%9D%92%8C%2B%7Cv%7C.
7. Algoritmo de Bellman-Ford. Recuperado el 14 de agosto de 2022 de Wikipedia: https://es.wikipedia.org/wiki/Algoritmo_de_Bellman-Ford

8. Algoritmo de Floyd-Warshall. Recuperado el 14 de agosto de 2022 de Wikipedia: https://en.wikipedia.org/wiki/Floyd%E2%80%93Warshall_algorithm
9. Algoritmo de Johnson para caminos más cortos de todos los pares, 2022. Recuperado el 14 de agosto de 2022 de GeeksforGeeks: <https://www.geeksforgeeks.org/johnsons-algorithm/>
10. Algoritmo de Floyd-Warshall. Recuperado el 14 de agosto de 2022 Wikipedia: https://es.wikipedia.org/wiki/Algoritmo_de_Floyd-Warshall#:~:text=En%20inform%C3%A1tica%2C%20el%20algoritmo%20de,v%C3%A9rtices%20en%20una%20C3%BAnica%20ejecuci%C3%B3n.
11. UDB, Facultad de Ingeniería. Algoritmos para la ruta más corta en un Grafo. Recuperado el 16 de octubre de 2022 de https://www.udb.edu.sv/udb_files/recursos_guias/informatica-ingenieria/programacion-iv/2019/ii/guia-10.pdf