Facultad de Ciencias Económicas y Estadística

Universidad Nacional de Rosario

Anteproyecto

[Acá va el título de la tesina]

Director: Iván Millanes
Codirector: Diego Marfetán Molina
Alumna: Valentina Salvarezza

Índice

1 Introducción

En la actualidad, cuando una persona necesita desplazarse de un punto a otro, es común recurrir a aplicaciones como Google Maps. Estas herramientas transformaron la manera en que se planifican los desplazamientos, proporcionando información precisa sobre las rutas más convenientes, ya sea caminando, en automóvil, en bicicleta o utilizando otros medios de transporte. Además de indicar el recorrido a seguir, muchas de estas aplicaciones estiman el tiempo de llegada y consideran factores como el tráfico o las condiciones meteorológicas para ofrecer mejores alternativas.

Sin embargo, surgen algunas preguntas fundamentales: ¿cómo funcionan realmente estas aplicaciones? ¿Cómo son capaces de calcular la ruta más corta o más rápida en función de las preferencias del usuario? Las respuestas se encuentran en la teoría de grafos, una rama de las matemáticas y la informática que permite modelar relaciones y estructuras en diversas áreas, como redes de comunicación, análisis de redes sociales y optimización de sistemas logísticos. En el contexto de la planificación de rutas, el problema se modela mediante un grafo ponderado, donde los nodos representan intersecciones y las aristas los caminos que los conectan. A través de este modelo es posible calcular las rutas más eficientes, evaluando cada trayecto en función de costos asociados, como la distancia o el tiempo de viaje, e identificando la opción óptima que minimice dichos costos.

Para resolver este problema se emplea el algoritmo de Dijkstra, desarrollado por Edsger W. Dijkstra en 1956 y publicado en 1959. Este algoritmo, ampliamente utilizado en teoría de grafos, permite encontrar el camino más corto entre dos nodos cuando los pesos de las aristas son no negativos. Su eficiencia y robustez lo convirtieron en una herramienta clave en diversas aplicaciones, como sistemas de navegación, redes informáticas y planificación logística. Uno de los servidores que implementa este algoritmo es OSRM (Open Source Routing Machine), un enrutador de código abierto diseñado para procesar datos provenientes de OpenStreetMap (OSM). OSRM, al integrarse con OSM, calcula rutas optimizadas en función de distintos criterios, como el medio de transporte o restricciones en el camino, y su capacidad para manejar grandes volúmenes de datos y ofrecer respuestas rápidas lo convierte en una solución ampliamente utilizada en aplicaciones de movilidad, logística y optimización de rutas.

En este trabajo se desarrollará una aplicación web mediante Shiny en RStudio que permitirá ilustrar de manera más completa el funcionamiento de estas aplicaciones de navegación. La aplicación trabajará con datos de la ciudad de Rosario disponibles en OpenStreetMap, y ofrecerá una interfaz intuitiva donde el usuario podrá seleccionar intersecciones como puntos de origen y destino. A partir de esta selección, se generará el recorrido más corto o el más rápido, dependiendo de la preferencia del usuario, y se adaptará el cálculo según el medio de transporte elegido. Comprender cómo se estructuran y procesan estos datos no solo permite analizar en profundidad el funcionamiento de estas herramientas, sino también explorar posibles aplicaciones en movilidad urbana, planificación del transporte público y optimización logística.

2 Objetivos

2.1 Objetivo General

El objetivo principal de esta tesina es desarrollar una herramienta para la optimización de rutas en la ciudad de Rosario, basada en la aplicación del algoritmo de Dijkstra. La implementación se realizará a través de una Shiny App en RStudio, que permitirá a los usuarios calcular recorridos eficientes para vehículos considerando criterios de distancia y tiempo de viaje.

2.2 Objetivos Específicos:

- Analizar los datos geográficos representados por nodos (intersecciones) y aristas (calles y caminos), que constituyen un tipo de dato no convencional, nunca antes tratado en la carrera, para evaluar su adecuación y optimización en el cálculo de rutas.
- Desarrollar e implementar el algoritmo de Dijkstra y su extensión Multi Level Dijkstra (MLD) como métodos de optimización para el cálculo de rutas en grafos ponderados por distancia y tiempo de viaje. Esto permitirá generar dos tipos de recorridos: el más corto y el más rápido, de acuerdo con la preferencia del usuario.
- Crear una Shiny App en RStudio para visualizar las rutas optimizadas para vehículos, utilizando un servidor de OSRM desplegado localmente. Este servidor es crucial para realizar el cálculo de las rutas, ya que procesa los datos geográficos de Rosario y genera los recorridos más cortos y rápidos, los cuales se mostrarán a través de una interfaz interactiva.

3 Marco Teórico

3.1 Teoría de Grafos

La teoría de grafos es una rama fundamental de las matemáticas y la informática que estudia las relaciones entre objetos representados como nodos (también llamados vértices) y las aristas (conexiones entre los nodos). En el contexto de la optimización de rutas, los grafos se emplean para modelar sistemas de transporte, redes viales y de comunicación. A continuación, se describen los principales tipos de grafos que se utilizan en este campo:

- Grafo simple: Un grafo simple es aquel en el que no existen aristas repetidas ni ciclos en sus conexiones. En el caso de redes viales, este tipo de grafo puede representar una ciudad donde cada intersección es un nodo, y cada calle es una arista, sin tener en cuenta la dirección del tráfico.
- Grafo dirigido: Un grafo dirigido tiene aristas que poseen una dirección específica, es decir, la relación entre dos nodos es unidireccional. Este tipo de grafo es fundamental para representar redes de tráfico en las cuales la dirección de las calles es importante, como en las intersecciones de una ciudad donde los caminos tienen una dirección definida.
- Grafo ponderado: En un grafo ponderado, cada arista tiene un valor asociado que puede representar el costo de transitar por ese camino. Este valor puede ser, por ejemplo, la distancia, el tiempo de viaje o el costo monetario de recorrer una calle. Los grafos ponderados son esenciales para el cálculo de rutas más eficientes, ya que permiten evaluar distintos criterios de optimización (como el camino más corto o el más rápido).
- Grafo no dirigido: A diferencia del grafo dirigido, las aristas en un grafo no dirigido no tienen una dirección específica. Este tipo de grafo puede ser útil para modelar situaciones en las que las conexiones entre los puntos no dependen de una dirección particular (por ejemplo, calles sin restricciones de un solo sentido).

Para la construcción de rutas urbanas, generalmente se utilizan grafos ponderados y dirigidos, ya que las rutas tienen costos asociados (como el tiempo o la distancia) y las direcciones de tránsito deben ser respetadas en la mayoría de las redes viales.

3.2 Algoritmo de Dijkstra

El algoritmo de Dijkstra, propuesto por Edsger W. Dijkstra en 1956, es uno de los métodos más conocidos y utilizados para encontrar el camino más corto entre un nodo de origen y los demás nodos de un grafo ponderado. En un grafo ponderado, cada arista tiene un peso asociado, que puede representar distancia, tiempo o cualquier otro tipo de costo. Este algoritmo es ampliamente empleado en sistemas de navegación y optimización de rutas debido a su eficiencia para calcular el camino más corto en grafos con aristas de peso no negativo.

El funcionamiento del algoritmo es el siguiente:

- Inicialización: Se asigna un valor de costo de 0 al nodo origen y un valor infinito a todos los demás nodos.
- Selección del nodo más cercano: En cada paso, se selecciona el nodo con el costo acumulado más bajo desde el origen. Este nodo es marcado como visitado, y su costo se hace permanente.
- Actualización de vecinos: A continuación, se revisan los nodos vecinos del nodo visitado, y si el costo de llegar a ellos a través del nodo visitado es menor que el costo previamente asignado, se actualiza dicho costo.
- Repetición: Este proceso se repite hasta que todos los nodos tengan un costo definitivo o se haya encontrado el camino más corto al nodo destino.

La principal ventaja del algoritmo de Dijkstra es su eficiencia en encontrar el camino más corto en grafos grandes.

4 Multi-Level Dijkstra (MLD)

El Multi-Level Dijkstra (MLD) es una extensión del algoritmo de Dijkstra diseñada para mejorar la eficiencia en el cálculo de rutas en redes grandes, como las que pueden encontrarse en una ciudad extensa. Esta extensión se utiliza principalmente para reducir el tiempo de procesamiento al calcular rutas en grandes bases de datos geográficos.

En el MLD, el grafo se divide en niveles jerárquicos o estratos, donde los nodos de cada nivel son agrupados de acuerdo con su proximidad o características similares. El proceso de cálculo de rutas se realiza de manera multi-nivel, lo que significa que en lugar de considerar todos los nodos a la vez, el algoritmo realiza un primer recorrido en los niveles superiores del grafo, utilizando rutas predefinidas o "macro-rutas" que conectan los puntos principales de la red. Luego, se realiza un refinamiento local en los niveles inferiores para calcular la ruta óptima entre los nodos más cercanos.

Ventajas de MLD:

- Reducción de la complejidad computacional: Al dividir el grafo en niveles, el MLD permite calcular rutas mucho más rápidamente, lo cual es esencial en sistemas de navegación en tiempo real.
- Optimización en redes grandes: Es especialmente útil en ciudades grandes o en redes de transporte que involucran miles de intersecciones y caminos.

En el contexto de la planificación de rutas urbanas, el MLD es ideal para gestionar grandes volúmenes de datos geográficos, como los de OpenStreetMap (OSM), y ofrecer resultados rápidos y precisos al usuario, lo que lo convierte en una opción preferida para aplicaciones de movilidad urbana.

5 OSRM (Open Source Routing Machine)

Open Source Routing Machine (OSRM) es una herramienta de enrutamiento de código abierto que utiliza los datos geográficos de OpenStreetMap (OSM) para calcular rutas óptimas. OSRM implementa el algoritmo de Dijkstra y otros métodos de optimización de rutas, lo que le permite ofrecer respuestas rápidas y eficientes, incluso con grandes volúmenes de datos. La integración de OSRM con OSM es fundamental para generar rutas basadas en la topología de la red vial de una ciudad, como Rosario.

El proceso de despliegue de OSRM implica varios pasos importantes:

1. Obtención de datos geográficos: Se descargan los datos de OpenStreetMap para la ciudad de Rosario, que contienen la información detallada sobre calles, intersecciones, rutas y otras infraestructuras de transporte. Estos datos incluyen información sobre las distancias, el tipo de carretera y las restricciones de tráfico.
2. Preprocesamiento de datos: Antes de poder utilizar OSRM para calcular rutas, es necesario realizar un proceso de preprocesamiento de los datos. Este paso convierte los datos de OSM en un formato que OSRM pueda manejar eficientemente. Durante este proceso, se calcula la red de carreteras y se optimizan las rutas predefinidas para permitir cálculos rápidos en tiempo real.
3. Construcción del servidor local: Para realizar los cálculos de rutas, se debe desplegar un servidor local de OSRM en la máquina que gestionará las solicitudes de enrutamiento. Este servidor utilizará los datos preprocesados y estará listo para responder a las solicitudes de los usuarios.
4. Despliegue en la nube: Si es necesario hacer que el sistema sea accesible de forma remota o para manejar una gran cantidad de usuarios, se puede optar por desplegar el servidor en la nube. Esto permite que los usuarios de diferentes dispositivos o ubicaciones puedan acceder a los servicios de enrutamiento sin necesidad de estar conectados a un servidor local.

El uso de OSRM en este proyecto es clave para calcular rutas eficientes en la ciudad de Rosario. Al integrar OSRM con una Shiny App en RStudio, se puede ofrecer una interfaz interactiva que permite a los usuarios seleccionar puntos de origen y destino, y obtener rutas optimizadas basadas en diferentes criterios (más corta, más rápida, etc.). Esto no solo facilita la visualización de las rutas, sino que también mejora la experiencia de usuario al ofrecer resultados rápidos y precisos.