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.
- 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 Metodología

3.1 Teoría de Grafos

La teoría de grafos es una rama central de las matemáticas y la informática que se dedica al estudio de las relaciones entre objetos representados por nodos o vértices, y las aristas, que son las conexiones entre ellos. Un grafo G se define formalmente como $G = (V, E)$, donde V es un conjunto no vacío de vértices y E es un conjunto de aristas que conectan pares de vértices. Estos modelos resultan fundamentales en la optimización de rutas, dado que permiten representar intersecciones como nodos y calles como aristas, facilitando el análisis de recorridos.

- **Grafo simple:** Las aristas no tienen dirección, por lo que el recorrido puede hacerse en ambos sentidos. Este tipo de grafo se utiliza cuando se desea representar conexiones bidireccionales, como calles donde se puede circular en ambos sentidos. Un caso de aplicación se da cuando el desplazamiento es a pie, ya que caminando es posible combinar ambos sentidos de la calle para encontrar el camino más corto en términos de cantidad de aristas.
- **Grafo dirigido:** en este caso, las aristas tienen una orientación específica, representada como un par ordenado (v, w). Esto implica que la conexión desde v hacia w no necesariamente permite el recorrido inverso, lo que lo hace adecuado para modelar calles con sentido único.
- **Grafo ponderado:** además de ser dirigido o no dirigido, cada arista lleva asociado un peso $p(e)$ donde e corresponde al peso de la arista, que puede representar distancia, tiempo de viaje u otro costo. La longitud de un camino se mide como la suma de los pesos de sus aristas, permitiendo modelar tanto el recorrido más corto como el más rápido.

En síntesis, la elección del tipo de grafo determina cómo se representan las restricciones y características de la red vial. En el presente trabajo, se emplean principalmente grafos dirigidos y ponderados, donde la ponderación se realiza por distancia o por tiempo, según el criterio de optimización considerado.

3.2 Algoritmo de Dijkstra

El algoritmo de Dijkstra, propuesto por Edsger W. Dijkstra en 1956, es un método ampliamente utilizado para determinar el camino de costo mínimo entre un nodo de origen y los demás nodos de un grafo ponderado. En estos grafos, cada arista tiene un peso que puede representar distancia, tiempo u otro tipo de costo, y se asume que estos pesos son no negativos. Este algoritmo es fundamental en sistemas de navegación y optimización de rutas, ya que permite calcular caminos mínimos de manera rápida y precisa.

La idea central del algoritmo consiste en avanzar desde un nodo de origen hacia los nodos adyacentes siguiendo rutas de menor costo acumulado, hasta alcanzar los destinos deseados.

Para ello, a cada nodo v se le asigna un valor $D(v)$ que representa el costo mínimo acumulado desde el nodo inicial hasta v. Inicialmente, estos valores se definen como $D(origen) = 0$ y $D(v) = \infty$ para el resto de los nodos, indicando que aún no se conoce un camino hacia ellos.

Durante la ejecución del algoritmo, los nodos se dividen en dos categorías:

- Etiquetas temporales, que pueden actualizarse si se encuentra un camino más corto.
- Etiquetas permanentes, que indican que se ha determinado el menor costo posible hacia ese nodo.

El procedimiento del algoritmo se desarrolla de la siguiente manera:

1. Se define un conjunto T con todos los nodos no visitados, inicialmente todos los nodos del grafo.
2. Se selecciona el nodo $u \in T$ con el menor valor de $D(u)$ y se marca como visitado, convirtiendo su etiqueta en permanente.
3. Se revisan todos los nodos vecinos t de u. Si el costo de llegar a t pasando por uuu es menor que el costo previamente asignado $D(t)$, se actualiza su valor de acuerdo a la regla: $D(t) = minD(t), D(u) + p(u, t)$ donde $p(u, t)$ representa el peso de la arista que conecta u con t.
4. El proceso se repite hasta que todos los nodos tengan etiquetas permanentes o se haya determinado el camino mínimo hacia el nodo destino.

El algoritmo garantiza que, al asignar una etiqueta permanente a un nodo, se ha determinado el costo mínimo desde el origen hacia dicho nodo. Esto se debe a que, durante su ejecución, evalúa todas las rutas posibles para llegar a cada nodo, asegurando la solución óptima. Sin embargo, este enfoque resulta computacionalmente costoso cuando se aplica al cálculo de recorridos en ciudades grandes, como Rosario. Por este motivo, en contextos de mayor escala se utilizan extensiones del algoritmo, como Multi Level Dijkstra

3.3 Multi-Level Dijkstra (MLD)

El Multi-Level Dijkstra (MLD) es una extensión del algoritmo de Dijkstra diseñada para optimizar la eficiencia en el cálculo de rutas en redes viales grandes, como las que se encuentran en ciudades extensas. Su objetivo principal es reducir la complejidad computacional y el tiempo de procesamiento al calcular rutas sobre grandes bases de datos geográficos.

El enfoque del MLD se basa en una estructura jerárquica de niveles:

- **Nivel 0:** El grafo original se divide en pequeñas celdas que representan barrios o vecindarios, incluyendo todas las calles e intersecciones internas.

- **Nivel 1:** Varias celdas de nivel 0 se agrupan en unidades más grandes, como distritos, reduciendo la complejidad al analizar recorridos entre distintas zonas de la ciudad.
- **Nivel 2 y superiores:** Los distritos se combinan en macrozonas que abarcan áreas extensas, por ejemplo, Norte, Sur, Este y Oeste.

Durante la fase de preprocesamiento, el algoritmo identifica los nodos frontera, es decir, las intersecciones que conectan diferentes celdas, y calcula previamente los caminos más cortos entre ellos. Con esta información se construye un grafo reducido (overlay graph) que actúa como un conjunto de atajos entre celdas, permitiendo que el cálculo de rutas se concentre únicamente en las áreas relevantes.

El procedimiento de consulta funciona de manera multi-nivel:

- Si el origen y destino están dentro de la misma celda de nivel 0, la búsqueda se realiza directamente sobre el subgrafo local.
- Si se encuentran en celdas distintas, se utiliza primero el grafo reducido de niveles superiores para "saltar" entre regiones y luego se desciende a niveles más detallados hasta reconstruir el trayecto completo calle por calle.

Este enfoque permite mantener la precisión en la ruta óptima, mientras se reduce significativamente la cantidad de nodos a explorar, lo que resulta crucial para aplicaciones de movilidad urbana que manejan grandes volúmenes de datos geográficos, como los provenientes de OpenStreetMap (OSM).

En resumen, el MLD combina la exhaustividad del algoritmo de Dijkstra con una estructura jerárquica eficiente, optimizando tanto el tiempo de cálculo como la gestión de rutas en redes urbanas extensas.

3.4 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 en redes viales. OSRM implementa el algoritmo de Dijkstra y sus extensiones, como Multi-Level Dijkstra (MLD), lo que le permite generar rutas rápidas y precisas incluso con grandes volúmenes de datos urbanos. Su principal aplicación es el cálculo de rutas en sistemas de navegación, planificación urbana y análisis de movilidad.

3.4.1 Obtención de los datos

Los datos geográficos de la ciudad de Rosario se obtuvieron de OpenStreetMap (OSM) mediante la plataforma BBbike, que permite descargar áreas específicas en formato .osm.pbf. En este caso, se descargaron específicamente los datos correspondientes a toda la ciudad de Rosario, asegurando que únicamente se incluyan los nodos (intersecciones) y aristas (calles) relevantes para el cálculo de rutas urbanas, evitando el procesamiento de información innecesaria.

3.4.2 Cálculo de rutas

Para poder calcular tanto el camino más corto como el camino más rápido, se construyeron dos servidores OSRM locales. Esto se debe a que OSRM utiliza un archivo de configuración llamado car.lua, que define cómo se calculan las rutas según el criterio seleccionado:

- Servidor de camino más corto: se modifica car.lua para priorizar la distancia (weight_name = 'distance').
- Servidor de camino más rápido: se modifica car.lua para priorizar el tiempo de viaje (weight_name = 'duration')

El uso de servidores separados permite calcular cada tipo de recorrido de manera independiente, garantizando que la aplicación pueda entregar resultados coherentes y optimizados según el criterio deseado.

3.4.3 Despliegue en la nube

Dado que la Shiny App se despliega en shinyapps.io, los servidores locales deben estar también accesibles de forma remota para que la aplicación pueda comunicarse con ellos. Por este motivo, los servidores OSRM se migraron a la nube utilizando DonWeb, asegurando que la Shiny App pueda conectarse a los servicios de enrutamiento sin depender de la red local.

Con este despliegue, tanto la aplicación web como los servidores de OSRM están disponibles en la nube, lo que permite que los usuarios puedan calcular rutas optimizadas desde cualquier dispositivo y ubicación, garantizando disponibilidad continua y compatibilidad con la aplicación Shiny.