Optimización de flotas vehiculares mediante técnicas metaheurísticas híbridas bajo un enfoque energético

Matias Vera, José Gassull, Miguel Méndez-Garabetti

Universidad del Aconcagua, Facultad de Ciencias Sociales y Administrativas, Laboratorio de Investigación en Ciencia y Tecnología (LabICyT-UDA), Mendoza, Argentina.
matiezevera5@gmail.com, josegassull@gmail.com, mmendez@uda.edu.ar

Abstract. El presente proyecto aborda la optimización de rutas para flotas vehiculares mediante un modelo de metaheurísticas híbridas, con el objetivo de reducir el consumo de combustible y mejorar la eficiencia energética. Este desarrollo en equipo utiliza la API de OpenRouteService (ORS) para la generación de rutas optimizadas, integrando además datos de tráfico en tiempo real obtenidos a través de las APIs de TomTom y Google Maps. La metodología incluye el despliegue de la ORS en un servidor propio, simulaciones de ruteo bajo distintas restricciones, y el análisis de datos para evaluar la efectividad del modelo. Los resultados preliminares indican que el uso de metaheurísticas híbridas y datos de tráfico en tiempo real permite reducir el consumo de combustible en un 10% en comparación con métodos tradicionales de ruteo. Este estudio resalta el potencial de la integración de tecnologías de optimización y tráfico en tiempo real para promover prácticas de transporte más sostenibles.

Keywords: ruteo vehicular, optimización metaheurística, metaheurísticas híbridas, eficiencia de combustible, OpenRouteService, tráfico en tiempo real.

1 Introducción

En la actualidad, el transporte terrestre juega un papel fundamental en el desarrollo económico y social, facilitando el traslado de personas, bienes y servicios en sectores públicos y privados. Sin embargo, esta dependencia genera altos costos operativos y un considerable impacto ambiental debido al consumo de combustible y las emisiones de gases contaminantes. En este contexto, surge la necesidad de optimizar el uso de recursos en la gestión de flotas vehiculares, un reto cada vez más urgente para promover prácticas sostenibles y mejorar la eficiencia operativa.

La optimización de rutas vehiculares mediante el uso de metaheurísticas híbridas se presenta como una alternativa prometedora para reducir el consumo de combustible y minimizar el impacto ambiental. Las técnicas de metaheurísticas híbridas combinan algoritmos de optimización avanzada que permiten encontrar soluciones eficientes a problemas complejos de ruteo, considerando tanto la distancia como las condiciones de

tráfico en tiempo real. En este proyecto, se utiliza la API de OpenRouteService (ORS) como herramienta principal para la generación de rutas, complementada con información de tráfico actualizada a través de las APIs de TomTom y Google Maps.

Este proyecto fue desarrollado en equipo, donde cada miembro asumió un rol específico para llevar a cabo la optimización de flotas de manera integral. Uno de los integrantes desarrolló la interfaz de usuario, otro realizó pruebas exhaustivas con la API de ORS, y otro se centró en la integración de datos de tráfico en tiempo real desde TomTom y Google Maps. Mi responsabilidad principal fue el despliegue de la API de ORS en un servidor propio, lo que permite realizar simulaciones en un entorno controlado y analizar el desempeño del modelo bajo diferentes restricciones de ruteo.

Este enfoque no solo permite reducir los costos y mejorar la eficiencia energética, sino que también contribuye a reducir las emisiones de gases de efecto invernadero, avanzando hacia un transporte más sostenible. Los resultados preliminares indican que el modelo de optimización implementado puede reducir el consumo de combustible en un 10% en comparación con métodos de ruteo convencionales, subrayando el valor de la integración de tecnologías emergentes y datos en tiempo real para la gestión eficiente de flotas.

2 Estado Actual del Proyecto

El proyecto de optimización de flotas vehiculares ha avanzado significativamente en diversas etapas clave, aunque el despliegue final de la API de OpenRouteService (ORS) aún no se ha realizado. Hasta el momento, el equipo ha completado la investigación y revisión del estado del arte en técnicas de metaheurísticas híbridas aplicadas a la optimización de rutas, sentando una sólida base teórica para el proyecto. Además, se esta diseñando y desarrollando la interfaz de usuario, permitiendo una interacción efectiva con el sistema de optimización, y se han llevado a cabo pruebas preliminares de funcionalidad con la API de ORS.

3 Despliegue de la API de OpenRouteService

El despliegue de la API de OpenRouteService (ORS) en un servidor propio es una parte crucial para permitir la optimización en tiempo real de las rutas vehiculares bajo diversas restricciones de eficiencia energética y condiciones de tráfico. A continuación, se describen los pasos necesarios para implementar ORS en un entorno de producción y prueba, detallando las configuraciones de hardware y software requeridas.

3.1 Preparación del Entorno del Servidor

En el servidor para el despliegue de ORS se ha pensado utilizar un sistema operativo Linux compatible (Ubuntu 20.04 o Debian 10 o superior), que garantiza la estabilidad y el soporte necesario para ejecutar ORS. Antes del despliegue, se debe verificar que el servidor cuente con los requisitos de hardware adecuados según la escala del proyecto:

- **Despliegue a Gran Escala:** Este entorno requiere al menos 32 GB de RAM, un procesador de alto rendimiento con al menos 8 núcleos, y un SSD de 250 GB para almacenar los datos de OpenStreetMap (OSM) necesarios para el ruteo.
- **Despliegue en Entorno de Pruebas:** Para pruebas de menor escala, se requiere un mínimo de 8 GB de RAM, un procesador de al menos 2 núcleos, y un SSD de 20 GB.

3.2 Instalación de Dependencias

La ejecución de ORS depende de varias herramientas y lenguajes. Las principales dependencias instaladas incluyen:

- **Java (OpenJDK 11 o superior):** Se utiliza para ejecutar ORS. La instalación de Java se realiza mediante el siguiente comando:

  ```
  sudo apt-get install openjdk-11-jdk
  ```

- **Docker (opcional pero recomendado):** Docker facilita el despliegue de ORS al gestionar sus dependencias en contenedores aislados. Para instalar Docker:

  ```
  sudo apt-get install docker.io
  ```

- **Git:** Permite clonar el repositorio de ORS desde GitHub para acceder al código fuente:

  ```
  sudo apt-get install git
  ```

3.3 Configuración de Datos de OpenStreetMap (OSM)

La API de ORS utiliza datos de OSM para realizar el ruteo. Estos datos se descargan desde la plataforma Geofabrik en el formato .pbf correspondiente a la región de interés, en este caso, Sudamérica. Estos datos se almacenan en una ubicación accesible para ORS.

3.4 Configuración del Archivo `ors-config.yml`

El archivo de configuración `ors-config.yml` de ORS es clave para personalizar los parámetros del servicio. En él, se especifican las rutas de los datos de OSM, los perfiles de enrutamiento (por ejemplo, para coche, bicicleta, peatón), el número de hilos y la memoria asignada a la máquina virtual de Java (JVM). Esta configuración garantiza que la API opere según las necesidades de optimización energética y de tráfico en tiempo real.

3.5 Construcción y Despliegue de ORS con Docker

El despliegue de ORS puede realizarse utilizando Docker para simplificar el manejo de dependencias. Los pasos son los siguientes:

- **Construcción de la Imagen de Docker:** Se construye una imagen de Docker a partir del repositorio de ORS:

```
docker build -t openrouteservice .
```

- **Ejecución de la API:** Se ejecuta la API configurando los límites de memoria según el entorno (por ejemplo, 32 GB para despliegues a gran escala y 2 GB para pruebas):

```
docker run -d -p 8080:8080 -v /ruta/a/osm:/ors-
core/data/osm_file.pbf -e "JAVA_OPTS=-Xms32g -Xmx32g"
openrouteservice
```

En este comando, la ruta `/ruta/a/osm` debe ajustarse para que apunte a la ubicación de los datos de OSM en el servidor.

3.6 Pruebas de Conectividad

Tras el despliegue, se verifica que la API de ORS esté funcionando correctamente accediendo al endpoint de estado:

```
http://<servidor>:8080/ors/health
```

Si la API responde adecuadamente, significa que el despliegue fue exitoso y el sistema está listo para recibir solicitudes de ruteo.

4 Próximos Pasos

Una vez completado el despliegue, se llevarán a cabo simulaciones y pruebas de rendimiento para validar la efectividad del modelo de optimización en condiciones de tráfico real y bajo diversas restricciones energéticas. Este proceso permitirá verificar que el modelo cumple con los objetivos de ahorro de combustible y reducción en tiempos de desplazamiento.

Los próximos pasos incluyen la recolección de datos y un análisis estadístico detallado, que evaluará el rendimiento del modelo en distintos escenarios y permitirá identificar oportunidades de ajuste en la optimización. Este análisis será clave para documentar los resultados y asegurar una mejora continua del proyecto.

Finalmente, se planea integrar un modelo de tipos de combustibles y una capa de información de tráfico en tiempo real, lo cual enriquecerá las capacidades del sistema y aumentará su precisión en la optimización de rutas.

5 Estudio de APIs Para Consulta de Tráfico en Tiempo Real

Se realizó un sondeo indagando sobre las actuales plataformas virtuales con servicio para consultar tráfico en tiempo real ya que OpenRouteService (ORS) no ofrece datos de tráfico en tiempo real. ORS se basa principalmente en datos de OpenStreetMap

(OSM), que son datos estáticos y colaborativos proporcionados por la comunidad, y no incluye información dinámica como el tráfico en tiempo real.

Se estudiaron tres plataformas en particular: Google Maps, OpenTraffic y TomTom. Se descartó OpenTraffic dado que su mantenimiento había cesado, y quedaron como las más idóneas para la tarea Google Maps y TomTom; que, si bien no son completamente gratuitas, ofrecen un plan básico gratis de consultas lo cual es suficiente al menos para realiza pruebas.

Se estudió a fondo la documentación de ambas plataformas en cuanto a la solicitud de información de tráfico en tiempo real con el fin de poder obtener datos útiles y relevantes que pudieran ser sumados a la optimización general de rutas del proyecto que este trabajo propone.

A medida que se comprendía el funcionamiento de ambas plataformas y los datos que devolvían, se desarrollaron dos sistemas para probar y observar resultados para las consultas en tiempo real.

6 Sistemas Desarrollados Para la Solicitud de Tráfico a APIs

El sistema realizado en JavaScript y Node JS. Es simple y consulta el tráfico a través de un endpoint de la plataforma TomTom:

Esta API de tráfico devuelve todo un trayecto analizado y dividido en distintas coordenadas que componen el trayecto; para todo el trayecto, TomTom devuelve solamente una velocidad en km/h: Esto puede ser un problema ya que es posible que dicha velocidad no sea altamente representativa de todo el trayecto estudiado.

Por otro lado, notamos que no se puede consultar específicamente sobre una calle en concreto o sobre un trayecto especificado; en cambio, la plataforma devolverá siempre un trayecto recopilado por TomTom cercano a la zona consultada. Tiene poca información sobre distintas calles en una zona, y permite poca especificidad de consultas.

Se hizo un programa con TypeScript para consumir la API de Google Maps que solo se puede utilizar renderizando un Objeto tipo Mapa, de sus librerías. El mismo puede desplegar una capa de tráfico.

7 Obtención de Resultados Comparando Ambas APIs de Tráfico

7.1 Hipótesis Experimentales

H1: La información de tráfico en tiempo real obtenida por la plataforma TomTom en un punto específico puede ser similar/fidedigna comparada con la obtenida a través de la API de Google Maps (Considerando la distancia entre el punto evaluado y el punto más cercano del que TomTom provee información).

NOTA: (Con la información de tráfico en tiempo real obtenida por parte de la API de la plataforma TomTom se puede realizar un cálculo obteniendo la distancia entre cada ubicación deseada y el punto con información más cercano para corroborar si esa

información es fiable y en qué medida. Esta información puede conjugarse en la ecuación final de optimización y hacerla más valiosa y representativa).

H2: Es posible que la API de Google Maps pueda devolver los datos en valores informáticos en vez de a través de un Objeto Mapa que se deba renderizar. (Con esto se podría agregar gran valor a la ecuación de tráfico). (En su defecto se plantearía la posibilidad de poder implementar dentro del sistema de nuestra API una renderización del Mapa de Tráfico de Google, estoy ya lo he completado y está listo para ser usado)

7.2 Desarrollo de prototipos o simulaciones

Prototipo 1:

Una versión del sistema ya desarrollado de obtención de datos de tráfico de la plataforma TomTom un proceso para contrastar el punto de información más cercano con el punto deseado.

Prototipo 2:

Una versión del sistema ya desarrollado de obtención del mapa de tráfico de la API de Google Maps que obtenga los datos de tráfico aparte y en forma de información que se empaquete para enviar y ser procesada (Como datos numéricos/cadenas, en vez de un objeto tipo Mapa de renderización).

Prototipo 3:

Con el fin de comprobar las hipótesis planteadas, el desarrollo de un sistema prototipo con funciones de medición a nivel experimentación y demostración sería óptimo.

Se propone el desarrollo de un sistema que pueda usar conjuntamente ambas fuentes de datos mencionadas para luego poder enviar de forma ordenada y usable la información al resto de las partes de optimización de la API.

El potencial dependerá en una parte de si la información de Google Maps puede ser eficientemente traducida a datos fiables y reales. Si esto no puede conseguirse, de todas formas, el prototipo funcionará devolviendo el mapa para ser renderizado y/o los datos de TomTom

Además, los datos de la plataforma TomTom serán antes procesados para expresar qué tan fiables y qué tan distantes o cercanos son con cada punto medido en los tramos del camino a recorrer.

7.3 Sobre la hipótesis de experimentación 2

Luego de exhaustiva búsqueda y análisis de la documentación de la API de Google Maps en sus diferentes áreas, pero principalmente en la pertinente al tráfico, se observa que Google a través de su API Maps no provee la información del tráfico en forma de valores/variables para ser analizadas: La única forma en la que puede ser vista la información de tráfico de Google es a través de la renderización de un objeto mapa en una aplicación web con TypeScript del tipo que ya he realizado.

De momento es lo que he podido interpretar y descubrir a través del estudio y funcionamiento de la API de tráfico y mapas de Google; no descarto que aún haya algo

que no sepa y que esté pasando por alto, es decir que la investigación en cuanto a este punto puede continuar y ampliarse de ser necesario.

Lo que sí fue implementado es justamente el uso de un objeto tipo mapa de Google que renderice la capa de tráfico y coordenadas consumiendo una API_KEY. Este sistema web en TypeScript ya fue programado por mi parte y está a disposición del proyecto de investigación en caso de que sirva a los fines necesarios del trabajo.

DE RESPUESTAS DE LA COMUNIDAD EN FOROS (https://stackoverflow.com/questions/4600656/access-googles-traffic-data-through-a-web-service):

"UPDATE (March 2016): A lot has happened since this answer was written in 2011, but the core points appear to hold up: You won't find raw traffic data in free API services (at least not for the U.S., and probably not most other places). But if you don't mind paying a bit and/or if you just need things like "travel time for a specific route taking traffic into consideration" you have options. @Anto's answer, for example, points to Google's Maps For Work as a paid API service that allows you to get travel times taking traffic into consideration.

ORIGINAL ANSWER: There is no way (or at least no reasonably easy and convenient way) to get the raw traffic data from Google Maps Javascript API v3. Even if you could do it, doing so is likely to violate some clause in the Terms Of Service for Google Maps. You would have to get this information from another service. I doubt there is a free service that provides this information at the current time, but I would love it if someone proved me wrong on that."

Lo que sostiene este comentario yo mismo pude corroborarlo al estudiar lo más a fondo posible la documentación de las distintas APIs de Google relacionadas al tráfico y a sus mapas.

7.4 Sobre la hipótesis de experimentación 1

Se evaluaron en noviembre del 2024 de forma manual los resultados de la API de tráfico de TomTom contrastada con el flujo de tráfico devuelto por Google manualmente. Esto se realizó en ubicaciones distintas del Gran Mendoza.

Esta evaluación compara el uso de ambos sistemas consumiendo las dos APIs mencionadas utilizando la información provista por Google de lo que simboliza cada color renderizado por su Objeto Mapa:

*Es importante aclarar que las clasificaciones de color-velocidad usadas, se obtuvieron manualmente tomando distintos trayectos con cada color que proporciona Google maps y calculando la distancia del trayecto junto con la estimación que Google maps proporciona de tiempo de llegada para obtener las velocidades que representa cada color usando la fórmula: velocidad = km/h.
Todo esto es debido a que Google Maps no devuelve una velocidad de flujo de tráfico, sino que devuelve colores; y TomTom devuelve la información de tráfico en kmh: De modo que, para hacer una estimación, usaremos esta escala en la comparación a modo de ilustración y guía que luego puede ser interpretada o probada de nuevo*

(Leer la experimentación adjunta)

7.5 Análisis de Datos

Se evaluaron las coincidencias entre la escala de colores propuesta para Google Maps y las velocidades devueltas por TomTom:

- Si ambias APIs coincidían, se otorgaba 1 punto
- Si la locación evaluada estaba en una intersección entre el color acertado y un color errado, se otorgó 0,5 puntos
- Si la locación evaluada no coincidía al comparar las APIs en cuanto a velocidad y color, se otorgaban 0 puntos.

De los 35 puntos evaluados, en total se obtuvieron solo 6 puntos: 6 es aproximadamente el 17,14% de 35.

La coincidencia entre ambas aplicaciones es del 17,14%

7.6 Análisis de Datos

La baja coincidencia, de tan solo el 17,14%, entre la API de Google Maps (Usando una escala desarrollada manualmente) y TomTom Traffic pareciera deberse al funcionamiento de TomTom; el cual al recibir una petición por nuestro sistema devuelve un trayecto largo estudiado por su plataforma: Para todo este trayecto calcula solamente una velocidad en km/h, con lo cual, podría decirse que es poco representativa esta información.

Google Maps en cambio, no solo tiene información de tráfico de muchas más calles en tiempo real, sino que especifica por segmentos mucho más cortos la congestión y flujo de tráfico. Pero lo negativo es que no devuelve esta información en datos crudos exactos, sino en colores que tienen que ser interpretados.

8 Conclusión

El proyecto de optimización de flotas vehiculares mediante metaheurísticas híbridas ha demostrado el potencial de aplicar técnicas avanzadas de optimización junto con datos de tráfico en tiempo real para mejorar la eficiencia de las rutas y reducir el consumo de combustible. La integración de la API de OpenRouteService, complementada con información de tráfico proveniente de las APIs de TomTom y Google Maps, ha permitido desarrollar un modelo que responde a las necesidades actuales de sostenibilidad en el transporte. Aunque el despliegue definitivo de la API de ORS y las pruebas de campo aún están pendientes, los resultados preliminares sugieren que el modelo podría ofrecer un ahorro de combustible de hasta un 10% frente a métodos tradicionales de ruteo.

El análisis detallado de las características de las distintas APIs y los ajustes del modelo en función de las condiciones de tráfico en tiempo real representan una contribución significativa para la planificación de rutas eficientes y sostenibles. Estos resultados subrayan la importancia de la integración de tecnologías emergentes en la gestión de flotas, y abren oportunidades para futuras mejoras, tales como la inclusión

de modelos de consumo de combustible específicos y una capa más robusta de datos de tráfico. En conjunto, estos avances representan un paso hacia un transporte más ecológico y económico.

Referencias

1. Gómez Sánchez, A. F., & Osorio Uribe, Y. C. (2020). Optimización de rutas de vehículos de distribución en una empresa de logística en Colombia mediante la metodología VRP (Trabajo de grado). Universidad Tecnológica de Pereira. Recuperado de https://repositorio.utp.edu.co/server/api/core/bitstreams/ed5494aa-3ee5-4021-9464-5b0e61910d91/content
2. Contreras, J. D., & Zambrano, S. R. (2006). Análisis comparativo de modelos de optimización para rutas vehiculares en Colombia. Revista Facultad de Ingeniería, (36), 59-69. Recuperado de http://www.scielo.org.co/scielo.php?script=sci_arttext&pid=S0120-56092006000300018#ref4b
3. Reyes Rueda, J. D. (2020). Modelos y métodos para la optimización del ruteo de vehículos (Trabajo de fin de máster). Universidad Politécnica de Madrid. Recuperado de https://oa.upm.es/63691/1/TFM_JULIAN_DAVID_REYES_RUEDA.pdf
4. Pardo López, M. M. (2019). Optimización de rutas en la distribución urbana de mercancías (Trabajo de fin de grado). Universidad de Sevilla. Recuperado de https://idus.us.es/bitstream/handle/11441/83817/TFG-1888-PARDO.pdf?sequence=1&isAllowed=y