Algoritmos avanzados de búsqueda y optimización

Informe EV (2da entrega)

Integrantes: Ignacio Aguerrondo, Joaquín Batista, Matías Jordá y Juan Sebastián Paroli

Universidad Católica del Uruguay

Docentes: Michel Pedrera, Pío Dos Santos

2do Semestre, 2025

1. Introducción

El aumento del uso de vehículos eléctricos (EV) plantea nuevos desafíos en la planificación de rutas eficientes. A diferencia de los vehículos convencionales, los EV están sujetos a restricciones de autonomía debido a la capacidad limitada de sus baterías y la disponibilidad de estaciones de carga. En este contexto, el problema de encontrar el camino óptimo entre dos puntos no depende únicamente de la distancia más corta, sino también de la gestión inteligente de la energía y la ubicación de los cargadores intermedios.

Este proyecto tiene como objetivo desarrollar un sistema que calcule la ruta más corta viable para un vehículo eléctrico en un mapa de Uruguay o Montevideo, considerando la autonomía del vehículo y las estaciones de carga disponibles. Para ello se implementan y comparan distintos algoritmos de búsqueda, específicamente Dijkstra y A*, evaluando su desempeño en escenarios con y sin restricciones energéticas.

La propuesta busca contribuir al diseño de herramientas de apoyo a la movilidad eléctrica, optimizando el uso de los cargadores y favoreciendo la planificación eficiente de viajes sostenibles.

2. Fundamento teórico

2.1 Algoritmos de búsqueda en grafos

Los algoritmos de búsqueda son métodos computacionales que permiten explorar un conjunto de estados o nodos conectados mediante aristas (grafos), con el fin de encontrar una secuencia óptima desde un estado inicial hasta un estado objetivo. En el caso del problema del camino más corto, los nodos representan ubicaciones y las aristas las conexiones viales con sus respectivos costos (distancia o tiempo).

2.1.1 Dijkstra

Dijkstra es un algoritmo clásico de búsqueda que calcula el camino más corto desde un nodo origen hasta todos los demás en un grafo ponderado con pesos no negativos. Utiliza una cola de prioridad para expandir el nodo con el costo acumulado más bajo. Su ventaja principal es que garantiza encontrar siempre el costo mínimo, aunque su desempeño puede verse afectado en grafos muy grandes si no se optimiza la estructura de datos o si el grafo es demasiado denso.

2.1.2 A* (A-star)

A* extiende el algoritmo de Dijkstra incorporando una heurística que estima el costo restante desde un nodo actual hasta el destino. Su función de evaluación se define como:

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

donde $g(n)$ es el costo acumulado desde el inicio hasta el nodo n, y $h(n)$ es una estimación heurística del costo desde n hasta el objetivo.
Una heurística admisible (por ejemplo la distancia euclidiana) garantiza que A* encuentre el camino óptimo con menos expansiones de nodos que Dijkstra.

2.2 Restricciones energéticas en vehículos eléctricos

En el contexto de EV, el problema del camino más corto se transforma en un problema de búsqueda con recursos limitados. La autonomía de la batería restringe la distancia máxima que puede recorrer el vehículo sin recargar, por lo que el modelo debe incorporar:

- estados adicionales que representen el nivel de energía disponible, y
- nodos especiales donde existan cargadores.

De esta forma, el grafo se amplía para incluir:

- transiciones entre nodos que consumen energía proporcional a la distancia recorrida;
- transiciones hacia nodos de carga que restauran parcial o totalmente la energía del vehículo;
- penalizaciones por desvíos muy grandes respecto de la ruta ideal.

Este tipo de modelado es común en problemas de Electric Vehicle Routing Problem (EVRP) y puede resolverse mediante adaptaciones de algoritmos de búsqueda o combinando búsqueda con técnicas de optimización.

2.3 Heurísticas y adaptación del modelo

Para que A* funcione de manera eficiente en este escenario, la heurística no puede depender solo de la distancia física al destino. Debe considerar también el nivel de batería y la cercanía a cargadores.

Una forma genérica de expresar esto es mediante una heurística compuesta:

$$f(n) = g(n) + \alpha \cdot dist(n, destino) + \beta \cdot penalty\_energía(n)$$

donde:

- dist(n, destino) es la distancia estimada hasta el objetivo,
- penalty_energía(n) es un término que aumenta si el nivel de batería es bajo y no hay cargadores cercanos,
- α y β son coeficientes que quedan a determinar en la etapa final.

En esta entrega dejamos planteada la forma de la heurística, pero el ajuste fino de α y β y la validación experimental se hará en la entrega final.

3. Estado actual de la implementación

Para esta entrega intermedia ya contamos con un repositorio funcional en GitHub:

Repositorio: https://github.com/JoaquinBatser/ev-routing

El repositorio está organizado en módulos que separan claramente la lógica de:

- algoritmos (algorithms/), donde se encuentran las implementaciones base de Dijkstra y A*
- grafo y carga de datos (graph/), donde se prepara la estructura sobre la que corren los algoritmos;
- visualización (visualization/), pensada para generar animaciones o imágenes del recorrido;
- un script principal (main.py) que es donde se corre el programa.

Además, se incorporó un archivo de datos:

- cargadores.json, que contiene las posiciones de estaciones de carga que el sistema utilizará como nodos especiales. (https://portal.ute.com.uy/clientes/movilidad-electrica/carga-de-vehiculos)

En esta etapa el sistema ya puede:

1. cargar un grafo vial obtenido a partir de OpenStreetMap (vía *osmnx* u otras herramientas);
2. identificar los nodos donde hay cargadores (según el JSON);
3. ejecutar Dijkstra o A* entre un origen y un destino prefijados;
4. producir una salida que luego puede visualizarse.

Lo que no está cerrado todavía (y se deja explícito para la entrega final) es:

- el modelo completo de consumo energético,

- la expansión de estados “(nodo, batería)”,

- y la comparación cuantitativa entre las dos búsquedas.

4. Datos y modelado del grafo

Durante el desarrollo surgió una dificultad práctica: cargar todo el mapa de Uruguay como un único grafo implica una cantidad muy grande de nodos y aristas, lo que a su vez produce tiempos de carga elevados y un consumo de memoria alto.

Por ese motivo, para esta etapa se adoptó la estrategia de trabajar con un subconjunto geográfico (por ejemplo Montevideo o una zona recortada), que permite:

- probar y depurar los algoritmos,
- integrar los cargadores reales del archivo cargadores.json,
- y preparar el código para luego generalizar.

El modelo de datos que estamos usando es entonces:

1. Grafo vial base: derivado de OSM, donde cada intersección es un nodo y cada tramo de calle es una arista ponderada por distancia (o tiempo).
2. Cargadores como nodos especiales: a partir de cargadores.json, se cruzan las ubicaciones de cargadores con los nodos del grafo o se crean nodos adicionales que representan estas estaciones.
3. Atributos previstos para la siguiente entrega: capacidad de batería, consumo por kilómetro, tiempo de carga.

5. Criterios de evaluación y próximos pasos

De acuerdo con la propuesta del proyecto final del curso, en la etapa de cierre vamos a evaluar el sistema según los siguientes criterios:

1. Correctitud de la ruta: que el algoritmo encuentre un camino desde el origen al destino.

2. Viabilidad energética: que el camino propuesto respete la autonomía del vehículo, recurriendo a los cargadores cuando sea necesario.

3. Desempeño del algoritmo: cantidad de nodos expandidos y tiempo de cómputo, comparando Dijkstra vs A* en el mismo grafo.

4. Uso de cargadores: si el camino propuesto efectivamente aprovecha las estaciones disponibles del archivo cargadores.json.

5. Calidad de la heurística: en qué medida la heurística compuesta reduce el espacio de búsqueda sin perder soluciones válidas.

Próximos pasos (entrega final):

- incorporar explícitamente el estado de carga de la batería en los nodos del grafo o en la función de expansión;

- completar la definición de la heurística con valores concretos de α y β;

- ejecutar experimentos con más de un origen–destino para mostrar que el enfoque generaliza;

- documentar en el informe final la integración con el código del repositorio.

6. Bibliografía

- Russell, S. & Norvig, P. (2021). *Artificial Intelligence: A Modern Approach* (4th ed.). Pearson.

- Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to Algorithms* (3rd ed.). MIT Press.

- Artmeier, A., Haselmayr, J., Leucker, M., & Sachenbacher, M. (2010). "The Shortest Path Problem Revisited: Optimal Routing for Electric Vehicles." *KI - Künstliche Intelligenz*, 24(1), 1–5.

- Schneider, M., Stenger, A., & Goeke, D. (2014). "The Electric Vehicle-Routing Problem with Time Windows and Recharging Stations." *Transportation Science*, 48(4), 500–520.

- Fiorino, S. (2023). *Maps Pathfinding – Comparando Algoritmos: A* vs Dijkstra, en el mapa de la ciudad.* GitHub. https://github.com/santifiorino/maps-pathfinding

- Propuesta de Proyecto Final del Curso, "Algoritmos avanzados de búsqueda y optimización", UCU, 2025.