Sistema de Routing para Vehículos Eléctricos

Integrantes: Ignacio Aguerrondo, Juan Paroli, Matias Jorda, Joaquín Batista
Universidad: Universidad Católica del Uruguay
Docentes: Michel Pedrera, Pío Dos Santos
Curso: Algoritmos Avanzados de Búsqueda y Optimización
Fecha: 30 Noviembre del 2025

1. Introducción

El crecimiento exponencial de la adopción de vehículos eléctricos (EV) a nivel global ha generado nuevos desafíos en el ámbito de la planificación de rutas y la gestión inteligente de recursos energéticos. A diferencia de los vehículos de combustión interna, que pueden reabastecerse rápidamente en una densa red de estaciones de servicio, los vehículos eléctricos enfrentan limitaciones significativas relacionadas con la autonomía de sus baterías, los tiempos de recarga prolongados y la disponibilidad aún limitada de cargadores en muchas regiones del país.

Este proyecto aborda el problema de encontrar rutas óptimas para vehículos eléctricos en un contexto urbano real, específicamente en la ciudad de Montevideo, Uruguay. El objetivo principal es desarrollar e implementar algoritmos de búsqueda heurística que no solo minimicen la distancia o el tiempo de viaje, sino que garanticen la viabilidad energética del recorrido, considerando explícitamente las restricciones de capacidad de batería y la necesidad potencial de recargas intermedias en estaciones específicas.

El problema se modela como una variante del clásico problema del camino más corto (Shortest Path Problem), pero ampliado con restricciones de recursos. En lugar de simplemente buscar la ruta de menor costo entre dos puntos, debemos encontrar un camino que sea alcanzable con la energía disponible, lo que puede requerir planificar paradas estratégicas en estaciones de carga. Este tipo de problema se conoce en la literatura como Electric Vehicle Routing Problem (EVRP) y ha sido objeto de extenso estudio en los últimos años debido a su relevancia práctica.

La contribución de este trabajo radica en la implementación comparativa de dos enfoques algorítmicos distintos: el algoritmo A* (óptimo pero computacionalmente más costoso) y un algoritmo Greedy (subóptimo pero significativamente más rápido). Además, se exploran tres heurísticas diferentes para el algoritmo A* (distancia Euclidiana, Manhattan y Octile), evaluando su impacto en el rendimiento. Los experimentos se realizan sobre un grafo real de la red vial de Montevideo obtenido mediante OpenStreetMap, incorporando la ubicación real de las estaciones de carga públicas disponibles en la ciudad.

2. Fundamento Teórico

2.1. Algoritmos de búsqueda en grafos

Los algoritmos de búsqueda en grafos constituyen un pilar fundamental de la ciencia de la computación y la inteligencia artificial. Un grafo $G = (V, E)$ es una estructura matemática compuesta por un conjunto de vértices (o nodos) V y un conjunto de aristas E que conectan pares de vértices. En el contexto de problemas de enrutamiento, los vértices típicamente representan ubicaciones geográficas (intersecciones, puntos de interés), mientras que las aristas representan conexiones físicas (calles, avenidas, rutas) con costos asociados que pueden representar distancia, tiempo o consumo de recursos.

El problema del camino más corto busca determinar la secuencia de aristas de costo mínimo que conecta un vértice origen con un vértice destino. Este problema tiene aplicaciones en numerosos dominios: redes de comunicación, logística, sistemas de navegación GPS, planificación de rutas de transporte, y más recientemente, en la gestión de flotas de vehículos eléctricos.

2.2. El problema del camino más corto

El algoritmo de Dijkstra, propuesto por Edsger W. Dijkstra en 1956, es uno de los métodos más conocidos para resolver el problema del camino más corto en grafos con pesos no negativos. El algoritmo mantiene una cola de prioridad de nodos a explorar, siempre expandiendo el nodo con menor costo acumulado desde el origen. Su correctitud está garantizada: siempre encuentra el camino de costo mínimo. Sin embargo, en grafos grandes, Dijkstra puede ser ineficiente porque explora exhaustivamente en todas direcciones desde el origen, incluso en direcciones opuestas al destino.

El algoritmo A* (pronunciado “A-star”), desarrollado por Hart, Nilsson y Raphael en 1968, extiende a Dijkstra incorporando información heurística sobre la proximidad al objetivo. La función de evaluación de A* se define como:

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

donde $g(n)$ representa el costo real acumulado desde el nodo inicial hasta el nodo n, y $h(n)$ es una estimación heurística del costo desde n hasta el destino. Si la heurística $h(n)$ es admisible (nunca sobreestima el costo real) y consistente (satisface la desigualdad triangular), entonces A* está garantizado para encontrar el camino óptimo, pero típicamente explorando muchos menos nodos que Dijkstra.

La elección de la heurística es crucial para el desempeño de A*. En problemas geométricos, las heurísticas más comunes son:

Distancia Euclidiana: la distancia en línea recta entre dos puntos. Para puntos $\left(x_1, y_1\right)$ y $\left(x_2, y_2\right)$, se calcula como $\sqrt{\left(x_2 - x_1\right)^2 + \left(y_2 - y_1\right)^2}$. Es admisible en espacios continuos sin obstáculos.

Distancia Manhattan: la suma de las diferencias absolutas en cada coordenada: $\left|x_2 - x_1\right| + \left|y_2 - y_1\right|$. Es apropiada para entornos tipo cuadrícula donde el movimiento está restringido a direcciones ortogonales.

Distancia Octile: una generalización que permite movimientos diagonales. Se calcula como $max(\Delta x, \Delta y) + \left(\sqrt{2} - 1\right) \cdot min(\Delta x, \Delta y)$. Es útil en grillas donde se permiten 8 direcciones de movimiento.

2.3. Restricciones energéticas en vehículos eléctricos

El problema que nos ocupa va más allá del camino más corto tradicional, incorporando restricciones de recursos que transforman fundamentalmente la naturaleza del problema. En un problema clásico de camino más corto, el estado del sistema se describe completamente por el nodo actual. Sin embargo, cuando introducimos restricciones energéticas, el estado del sistema debe incluir también el nivel de batería disponible.

Formalmente, definimos el espacio de estados como el producto cartesiano:

$$S = V \times B$$

donde V es el conjunto de nodos del grafo y B es el conjunto de niveles posibles de batería. Un estado se representa como (v, b) donde $v \in V$ es el nodo actual y $b \in \left[0, B_{max}\right]$ es el nivel de

batería restante. Esta ampliación del espacio de estados tiene implicaciones profundas: el mismo nodo puede ser visitado múltiples veces con diferentes niveles de batería, y cada combinación representa un estado distinto del sistema.

Las transiciones entre estados están sujetas a restricciones físicas. Para moverse desde el estado (v_i, b_i) al estado (v_j, b_j) a través de una arista con costo energético e_{ij}, debemos satisfacer:

1. $b_i \geq e_{ij}$ (suficiente energía para atravesar la arista)
2. $b_j = b_i - e_{ij}$ (conservación de energía)

Además, en nodos especiales que cuentan con estaciones de carga, es posible realizar una transición de recarga en el mismo nodo: $(v, b) \rightarrow (v, min(b + r, B_{max}))$, donde r es la cantidad de energía recargada.

El consumo energético de un vehículo eléctrico depende de múltiples factores en escenarios reales: la topografía del terreno (pendientes aumentan significativamente el consumo), condiciones climáticas (temperatura afecta la eficiencia de la batería), velocidad de conducción, y características del vehículo. En este proyecto, adoptamos un modelo simplificado pero representativo donde el consumo es proporcional a la distancia recorrida, con un coeficiente γ expresado en kWh/km.

2.4. Heurísticas y funciones de evaluación

Para que A* funcione eficientemente en el contexto de vehículos eléctricos, debemos diseñar cuidadosamente la heurística. La estimación $h(n)$ debe considerar no solo la distancia geométrica al destino, sino también el consumo energético mínimo necesario para cubrir esa distancia.

En nuestra implementación, la heurística se calcula como:

$$h(n) = d(n, destino) \times \gamma_{min}$$

donde $d(n, destino)$ es la distancia (Euclidiana, Manhattan u Octile según la variante) desde el nodo n hasta el destino, y γ_{min} es el consumo energético mínimo por unidad de distancia. Esta

heurística es admisible porque asume el mejor escenario posible: viajar en línea recta al destino con el consumo más bajo, sin considerar obstáculos o desviaciones necesarias.

Es importante destacar que esta heurística no penaliza explícitamente situaciones de batería baja. Una extensión futura podría incorporar un término adicional que refleje el “riesgo” de quedarse sin batería lejos de un cargador, lo que podría guiar al algoritmo hacia rutas más conservadoras cuando la batería es escasa.

3. Datos y modelado del problema

3.1. Adquisición de datos geográficos

Los datos geográficos utilizados en este proyecto provienen de OpenStreetMap (OSM), un proyecto colaborativo de cartografía global que proporciona datos abiertos y de alta calidad sobre redes viales, puntos de interés, topografía, y otros elementos geoespaciales. La extracción y procesamiento de estos datos se realiza mediante la librería Python OSMnx, desarrollada por Geoff Boeing, que facilita la descarga, modelado y análisis de redes de calles desde OSM.

Para nuestros experimentos, descargamos el grafo vial completo de la ciudad de Montevideo, Uruguay. Este grafo contiene decenas de miles de nodos (intersecciones) y aristas (segmentos de calle), representando fielmente la complejidad de la red vial urbana. Cada arista en el grafo descargado incluye atributos originales de OSM como la longitud del segmento en metros, el tipo de vía, y cuando está disponible, la velocidad máxima permitida.

3.2. Representación del grafo vial

El grafo se representa como un objeto MultiDiGraph de la librería NetworkX, lo que permite múltiples aristas dirigidas entre el mismo par de nodos (útil para representar calles de doble sentido o vías con diferentes carriles). Cada nodo almacena coordenadas geográficas (latitud y longitud), que se proyectan a un sistema de coordenadas plano (x, y) para facilitar el cálculo de distancias euclidianas.

Las aristas se enriquecen con atributos adicionales calculados a partir de los datos base. En particular, se calcula un atributo weight que representa el tiempo estimado de viaje (distancia dividida por velocidad máxima), y un atributo energy_cost que representa el consumo energético del vehículo al atravesar ese segmento. El consumo energético se modela simplemente como:

$$energy\_cost = \gamma \times \frac{length}{1000}$$

donde γ es el coeficiente de consumo en kWh/km y length está en metros, por lo que se divide entre 1000 para convertir a kilómetros.

3.3. Infraestructura de recarga

Un componente esencial del problema es la disponibilidad de estaciones de carga. Para este proyecto, se utiliza un archivo JSON (*cargadores.json*) que contiene las ubicaciones reales de estaciones de carga públicas en Uruguay, obtenidas del portal de datos abiertos de UTE (Administración Nacional de Usinas y Trasmisiones Eléctricas). Cada cargador tiene coordenadas geográficas (latitud, longitud).

El proceso de integración de cargadores al grafo consiste en identificar, para cada estación de carga, el nodo del grafo más cercano geométricamente. Este nodo se marca como un "nodo cargador", permitiendo al algoritmo de búsqueda generar transiciones de recarga cuando visita dicho nodo. En la práctica, esto significa que ciertos nodos del grafo tienen la capacidad especial de restaurar la batería del vehículo.

3.4. Modelo de consumo energético

El modelo de consumo energético adoptado es deliberadamente simplificado para facilitar la experimentación y el análisis. Se asume un consumo constante por kilómetro, independiente de la velocidad, aceleración, o topografía. En los experimentos reportados, se utiliza un valor de $\gamma = 1.2$ kWh/km, que es aproximadamente 8 veces superior al consumo real de un vehículo eléctrico moderno (típicamente alrededor de 0.15 kWh/km).

Esta exageración deliberada del consumo se justifica por dos razones. Primero, permite trabajar con capacidades de batería reducidas (5 kWh en lugar de los 40-80 kWh reales), lo que intensifica las restricciones energéticas y hace que el problema de planificación de recargas sea más relevante incluso en distancias urbanas cortas. Segundo, aumenta la complejidad del espacio de estados, permitiendo evaluar mejor el desempeño de los algoritmos en escenarios desafiantes.

4. Metodología e implementación

4.1. Arquitectura del sistema

El sistema se implementa en Python, estructurado en módulos que separan claramente las responsabilidades. El directorio *graph* contiene las funciones para descargar y preparar el grafo de OSM, así como para cargar y procesar el archivo de cargadores. El directorio *algorithms* contiene las implementaciones de A* y Greedy, optimizadas para evitar visualizaciones durante el benchmarking (las versiones con visualización se mantienen en archivos separados para propósitos de depuración). El directorio *utils* proporciona funciones auxiliares como el cálculo de heurísticas, discretización de batería, y reconstrucción de caminos. Finalmente, el directorio *visualization* se encarga de generar representaciones gráficas de las rutas calculadas sobre el mapa de la ciudad.

El punto de entrada principal es *main.py*, que presenta un menú interactivo permitiendo al usuario ejecutar benchmarks completos, analizar resultados existentes, o realizar pruebas de visualización. El script *benchmark.py* automatiza la ejecución de múltiples pruebas, probando diferentes combinaciones de origen-destino y algoritmos, guardando los resultados en archivos JSON estructurados para análisis posterior.

4.2. Algoritmo A* con gestión de batería

La implementación de A* sigue la estructura clásica del algoritmo, pero adaptada al espacio de estados ampliado $(nodo, bateria)$. Se mantiene una cola de prioridad ordenada por $f(n) = g(n) + h(n)$, donde $g(n)$ representa la energía total consumida desde el origen hasta el estado n (no la batería restante, sino la energía gastada acumulativa), y $h(n)$ es la estimación heurística del consumo restante hasta el destino.

Un aspecto crítico de la implementación es la discretización del nivel de batería. Sin discretización, el espacio de estados sería infinito (batería continua), haciendo imposible reconocer estados repetidos. La discretización se realiza redondeando el nivel de batería a un decimal, lo que proporciona granularidad suficiente para baterías pequeñas (~5 kWh) sin

explotar el espacio de estados. Esta decisión de diseño representa un compromiso entre precisión y eficiencia computacional.

Cuando el algoritmo expande un nodo que es también una estación de carga, genera dos tipos de sucesores: los vecinos normales (moverse a nodos adyacentes consumiendo batería) y un sucesor especial que representa recargar en el mismo nodo sin moverse. La transición de recarga aumenta la batería en una cantidad fija (RECHARGE_AMOUNT = 4.5 kWh), sin añadir costo energético al $g(n)$ (asumimos que la energía de recarga es externa). Esta modelización permite al algoritmo "decidir" automáticamente cuándo conviene recargar basándose en si esa decisión permite alcanzar el destino de manera más eficiente.

4.3. Algoritmo Greedy con gestión de batería

El algoritmo Greedy implementado es una variante voraz que, a diferencia de A*, ignora completamente el costo acumulado $g(n)$ y toma decisiones basándose únicamente en la heurística $h(n)$. En cada paso, el algoritmo siempre expande el nodo que parece estar más cerca del destino geométricamente, sin considerar si el camino tomado hasta ese punto ha sido eficiente energéticamente.

Esta estrategia miope tiene ventajas y desventajas claras. La principal ventaja es la velocidad: al tender a moverse constantemente hacia el objetivo, el algoritmo típicamente explora órdenes de magnitud de nodos menores que A*. La desventaja es la pérdida de optimalidad: Greedy puede tomar rutas subóptimas, especialmente en presencia de obstáculos o cuando la topología del grafo hace que el camino más directo no sea el más corto.

En el contexto de vehículos eléctricos, esta diferencia se amplifica. Greedy puede tomar decisiones energéticamente costosas si esas decisiones parecen acercarle geométricamente al destino, potencialmente requiriendo más recargas o tomando rutas más largas. Sin embargo, para aplicaciones en tiempo real donde una respuesta instantánea es más valiosa que la optimalidad absoluta, Greedy puede ser una alternativa atractiva.

4.4. Discretización del espacio de estados

La discretización de la batería merece una discusión más profunda, ya que afecta fundamentalmente el comportamiento del algoritmo. Al redondear el nivel de batería a un decimal (por ejemplo, 3.7 kWh), estamos reduciendo el espacio continuo de energía a un conjunto finito de categorías. Dos estados con el mismo nodo y niveles de batería muy cercanos (digamos 3.74 kWh y 3.76 kWh) se consideran idénticos después de la discretización (ambos se redondean a 3.7 kWh).

Esto introduce una pequeña pérdida de precisión, pero es esencial para la viabilidad computacional del algoritmo. Sin discretización, cada transición podría llevar a un nivel de batería ligeramente diferente, generando millones de estados únicos. Con discretización, el número de estados está acotado por $|V| \times \left(B_{max}/step\right)$, donde $step$ es la granularidad de discretización. Para nuestros parámetros ($|V| \approx 50000$ nodos, $B_{max} = 5$ kWh, $step = 0.1$ kWh), el límite superior es aproximadamente 2.5 millones de estados, lo cual es manejable en memoria.

Un efecto secundario interesante de la discretización es que puede ocasionalmente llevar a que el algoritmo "pierda" ciertos caminos muy ajustados donde la batería exacta es crítica. Sin embargo, en la práctica, este efecto no es imprescindible con una discretización razonable, y los beneficios en términos de velocidad y uso de memoria superan ampliamente este inconveniente teórico.

5. Diseño experimental y resultados

5.1. Configuración de los experimentos

El conjunto de experimentos se diseñó para evaluar sistemáticamente el desempeño de los cuatro algoritmos implementados (tres variantes de A* y una de Greedy) en un escenario urbano realista. Se estableció un origen fijo en el barrio de Ciudad Vieja, ubicado en el centro histórico de Montevideo, y se probaron rutas hacia 37 barrios diferentes distribuidos por toda la ciudad, cubriendo distancias que van desde menos de 1 km hasta más de 15 km.

Los parámetros del sistema se configuraron de la siguiente manera:

- Consumo energético (γ): 1.2 kWh/km
- Capacidad máxima de batería: 5.0 kWh
- Carga inicial: 5.0 kWh (batería completa al inicio)
- Cantidad de recarga: 4.5 kWh por estación

Con estos parámetros, el vehículo tiene una autonomía teórica máxima de 4.16 km con batería completa, lo que es suficiente para destinos cercanos pero insuficiente para destinos lejanos sin recargas intermedias. Esta configuración permite evaluar la capacidad de los algoritmos para planificar recargas estratégicas.

Cada algoritmo se ejecutó en cada combinación origen-destino, registrando las siguientes métricas:

1. Energía total consumida (kWh): suma del consumo en todas las aristas del camino.
2. Nodos expandidos: cantidad de estados explorados durante la búsqueda.
3. Número de recargas: cuántas veces el vehículo se detuvo a recargar en el camino.
4. Tiempo de ejecución (segundos): tiempo de cómputo del algoritmo.
5. Longitud del camino: número de nodos en la ruta final.
6. Alcanzó destino: booleano indicando si se encontró una ruta viable.

5.2. Análisis cuantitativo de resultados

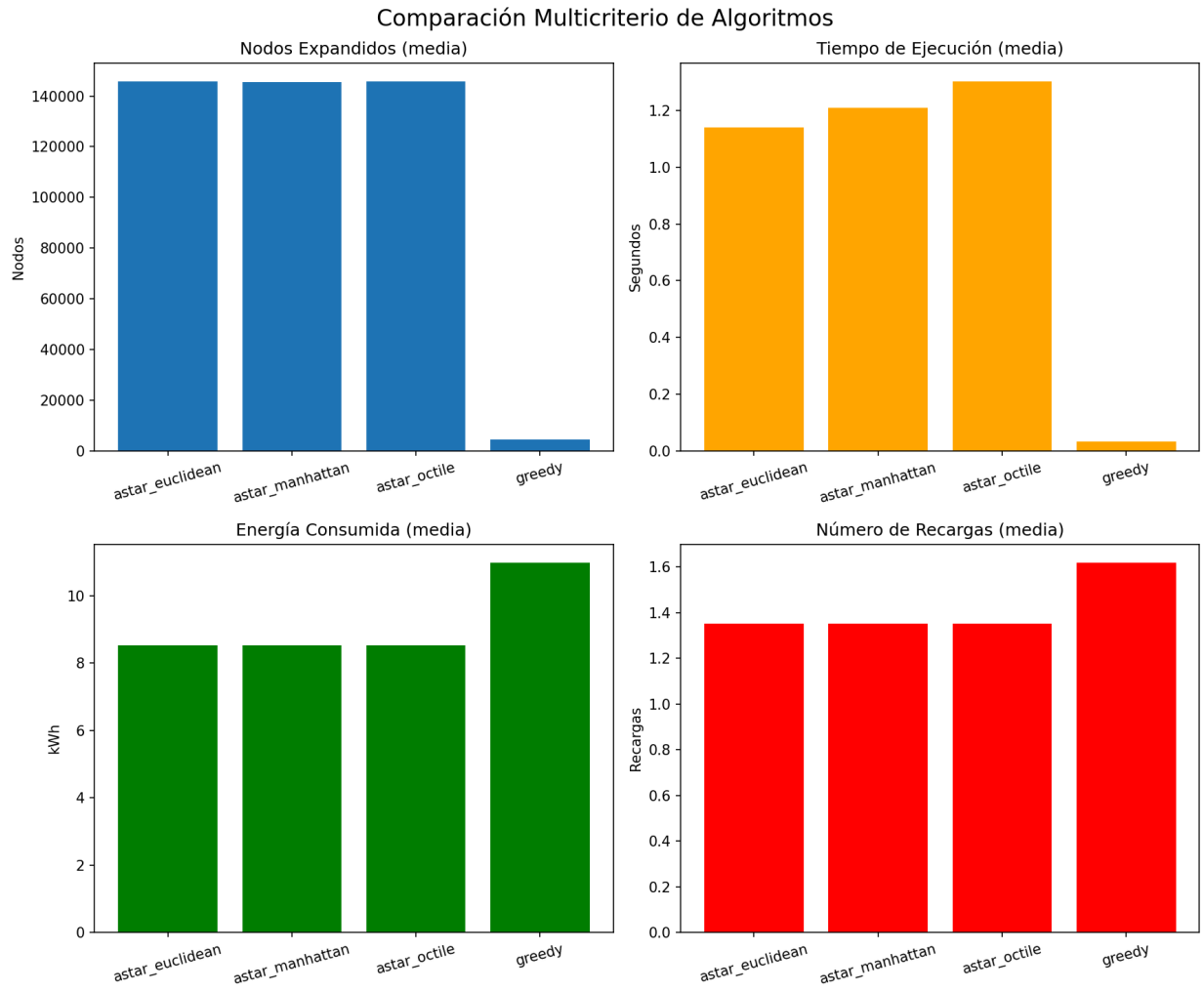
De los 37 destinos probados, los algoritmos lograron alcanzar exitosamente 34 destinos (91.9% de éxito). Los tres destinos inalcanzables fueron Carrasco, Carrasco Norte y Punta Gorda, todos ubicados en el extremo este de Montevideo, a distancias que exceden significativamente la autonomía del vehículo incluso con múltiples recargas. La falta de estaciones de carga en rutas intermedias hacia estos barrios hace que sean inviables con la configuración actual de batería.

Tabla 1: Resumen general por algoritmo

La siguiente tabla presenta un resumen de las métricas clave para cada algoritmo evaluado, calculadas sobre los 34 destinos alcanzados con éxito:

Algoritmo	Energía Media (kWh)	Nodos Expandidos (media)	Recargas (media)	Tiempo (s)	Speedup vs A* Euclidean
A* Euclidean	8.53	145,736	1.35	1.30	1.0x (baseline)
A* Manhattan	8.53	145,137	1.35	1.24	1.05x
A* Octile	8.53	145,680	1.35	1.28	1.03x
Greedy	10.97	4,464	1.62	0.032	40.6x

Los resultados muestran claramente el trade-off entre optimalidad y velocidad. Las tres variantes de A* son prácticamente idénticas en términos de calidad de solución (energía consumida), con diferencias mínimas en eficiencia computacional (menos del 5%). Greedy, por otro lado, sacrifica aproximadamente 28.6% de eficiencia energética pero obtiene una mejora de velocidad de más de 40x.



Comparación multicriterio de algoritmos

*Figura 1: Comparación visual de las cuatro métricas principales entre los algoritmos evaluados. Nótese la diferencia dramática en nodos expandidos y tiempo entre A** y Greedy, mientras que la energía y recargas son relativamente similares.

Consumo energético

En los 34 casos exitosos, observamos que las tres variantes de A* siempre encontraron exactamente la misma ruta óptima, con consumo energético idéntico en cada caso. Esto confirma que las diferentes heurísticas no afectan la optimalidad del resultado, solo la eficiencia de la búsqueda. El consumo promedio para A* fue de 8.53 kWh, con un mínimo de 0.85 kWh (Ciudad

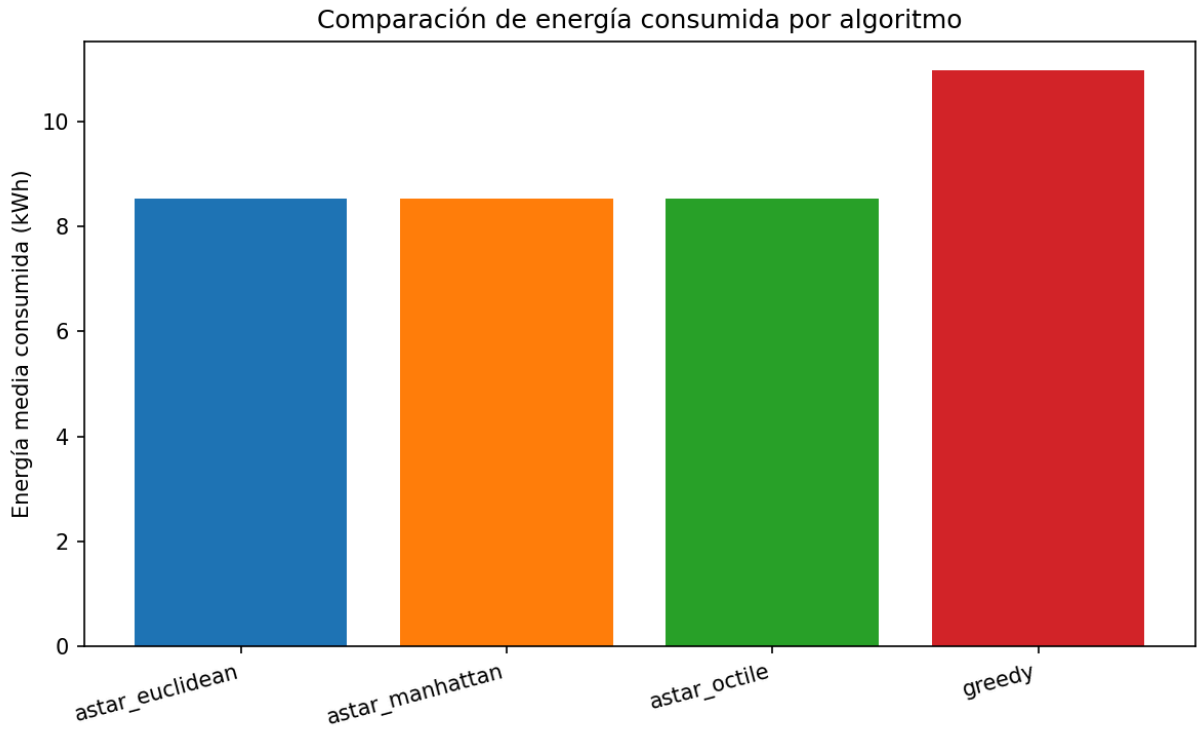
Vieja → Aguada, un trayecto muy corto) y un máximo de 15.57 kWh (Ciudad Vieja → Maroñas, requiriendo 3 recargas).

El algoritmo Greedy, por su parte, exhibió un consumo promedio de 10.97 kWh, aproximadamente 28.6% superior al de A*. Esta diferencia refleja la naturaleza subóptima de las decisiones voraces. En algunos casos, la diferencia fue dramática: por ejemplo, en la ruta a Jardines del Hipódromo, A* consumió 14.67 kWh mientras que Greedy consumió 21.37 kWh, un 45.6% más. Sin embargo, en rutas cortas y directas, Greedy a veces iguala a A*.

Tabla 2: Casos extremos - mejor y peor rendimiento por energía

Algoritmo	Menor Consumo	Destino	Mayor Consumo	Destino
A* (todas)	0.85 kWh	Aguada	15.57 kWh	Maroñas
Greedy	0.86 kWh	Aguada	21.37 kWh	Jardines del Hipódromo

La distribución del consumo energético se visualiza en el siguiente gráfico, que muestra claramente cómo Greedy tiende a consumir más energía que A*:



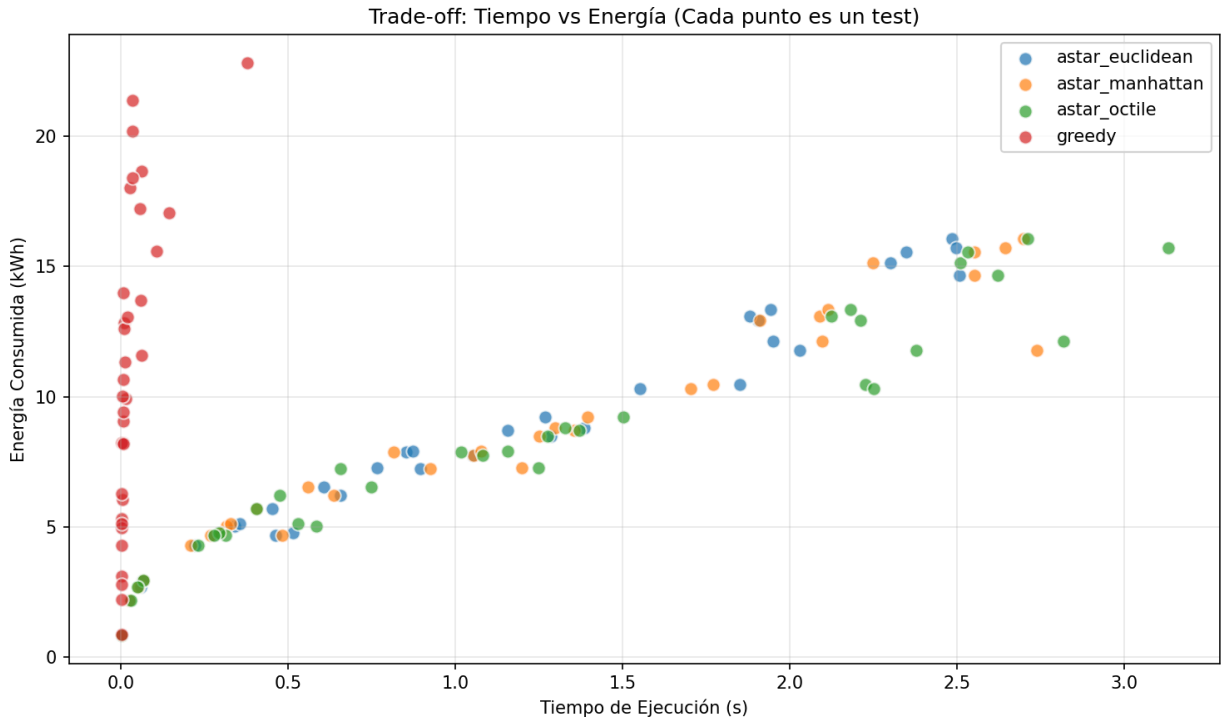
Comparación de energía consumida

*Figura 2: Comparación de energía media consumida por algoritmo. Se observa claramente que las tres variantes de A** consumen idéntica energía (barras azul, naranja y verde), mientras que Greedy (roja) consume aproximadamente 28.6% más.

Nodos Expandidos y Tiempo de Ejecución

Aquí es donde se observa la mayor diferencia entre los algoritmos. A* expandió en promedio 145,736 nodos por búsqueda, con casos extremos alcanzando más de 307,000 nodos expandidos. En contraste, Greedy expandió en promedio solo 4,464 nodos, aproximadamente 32 veces menos.

Esta dramática reducción en nodos expandidos se traduce directamente en tiempo de ejecución. A* tomó en promedio 1.30 segundos por búsqueda. Greedy, por otro lado, promedió apenas 0.032 segundos, más de 40 veces más rápido.



Scatter plot tiempo vs energía

*Figura 3: Relación entre tiempo de ejecución y energía consumida para cada test. Cada punto representa una ejecución. Nótese cómo Greedy (rojo) se concentra en la región de bajo tiempo pero mayor energía, mientras que A** (azules/verde/naranja) ocupa la región de mayor tiempo pero menor energía consumida.

Tabla 3: Eficiencia computacional - Rendimiento

Algoritmo	Nodos/Segundo (media)	Nodos/Segundo (mediana)
A* Euclidean	112,104	104,411
A* Manhattan	117,045	109,659
A* Octile	113,812	105,980
Greedy	137,547	214,669

Aunque Greedy expande muchos menos nodos en total, su rendimiento (nodos procesados por segundo) es comparable al de A*, indicando que el costo de procesamiento de cada nodo es comparable. La ventaja de Greedy radica en expandir órdenes de magnitud de nodos menores que A*, no en procesar cada nodo más rápido.

Número de recargas

El número de recargas necesarias varía según la distancia al destino. Para destinos dentro de la autonomía, ningún algoritmo requirió recargas. Para destinos más lejanos, A* típicamente requirió menos recargas que Greedy.

Es notable que incluso cuando ambos algoritmos requieren el mismo número de recargas, A* logra consumir menos energía total, indicando que no solo elige mejores puntos de recarga sino también mejores rutas entre ellos.

5.3. Comparación de heurísticas para A*

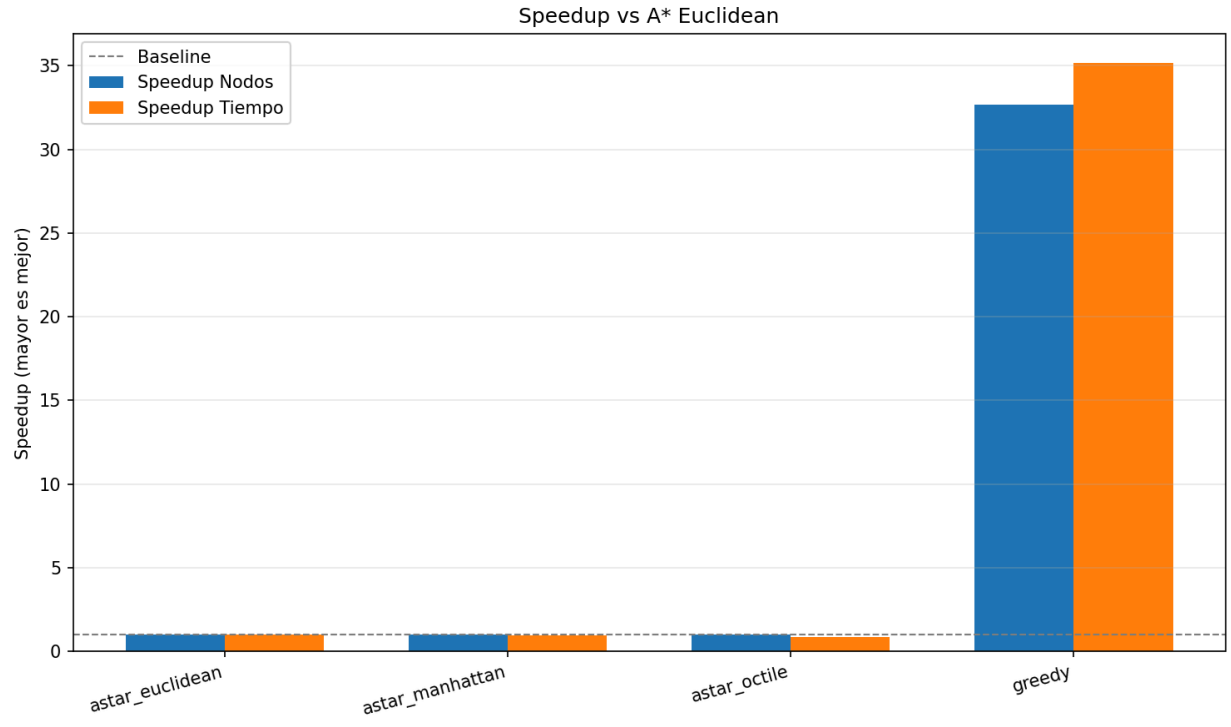
Las tres heurísticas probadas para A* (Euclidiana, Manhattan y Octile) mostraron rendimientos prácticamente idénticos en términos de calidad de solución: todas encontraron exactamente las mismas rutas óptimas en todos los casos. Esto era esperado, ya que las tres heurísticas son admisibles y consistentes, lo que garantiza la optimalidad de A*.

Sin embargo, se observaron diferencias sutiles en la cantidad de nodos expandidos y tiempo de ejecución:

*Tabla 4: Comparación detallada entre heurísticas de A**

Métrica	Euclidean	Manhattan	Octile
Nodos expandidos (media)	145,736	145,137	145,680
Tiempo medio (s)	1.30	1.24	1.28
Energía media (kWh)	8.53	8.53	8.53
Recargas (media)	1.35	1.35	1.35

Las diferencias son pequeñas, sugiriendo que en grafos viales urbanos complejos, la elección de la heurística tiene menos impacto que en entornos más estructurados como grillas.



Comparación de speedup

*Figura 4: Speedup relativo de cada algoritmo comparado con A** Euclidean (baseline). Nótese cómo Greedy domina en speedup tanto en nodos expandidos como en tiempo de ejecución, mientras que las variantes de A* son prácticamente equivalentes.

5.4. Comparación A* vs Greedy

La comparación entre A* y Greedy revela un trade-off clásico en diseño de algoritmos: optimalidad versus velocidad.

Tabla 5: A* vs Greedy - Análisis de Trade-offs

Métrica	A* Euclidean	Greedy	Diferencia	% Cambio
Nodos expandidos	145,736	4,464	-141,272	-96.9%
Tiempo (s)	1.30	0.032	-1.27	-97.5%
Energía (kWh)	8.53	10.97	+2.44	+28.6%
Longitud path	80.8	92.1	+11.3	+14.0%
Recargas	1.35	1.62	+0.27	+20.0%

Ventajas de A*:

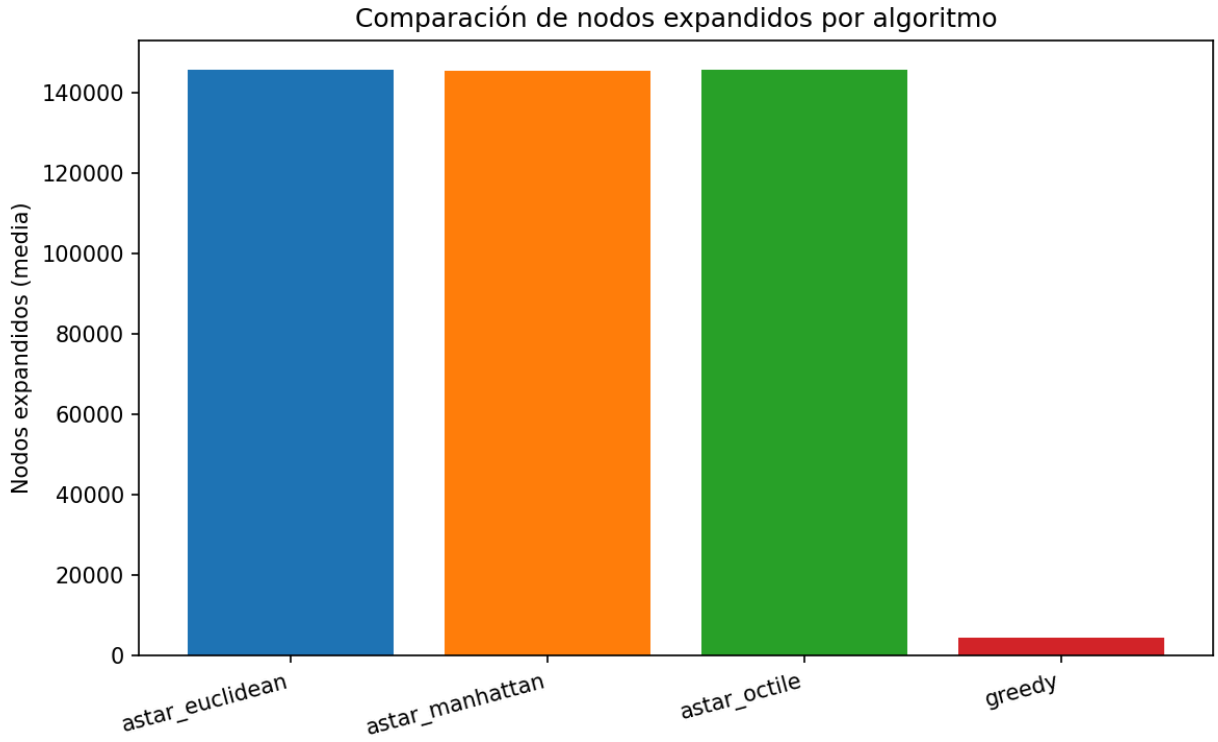
- Garantía de optimalidad en consumo energético (28.6% menos energía que Greedy)
- Menor número de recargas necesarias (20% menos que Greedy)
- Rutas más cortas en términos de nodos visitados
- Soluciones más predecibles y consistentes (menor varianza)

Ventajas de Greedy:

- Velocidad de búsqueda 40 veces superior (97.5% más rápido)
- Uso de memoria reducido (96.9% menos nodos en cola)
- Soluciones “suficientemente buenas” para muchos casos prácticos
- Respuesta prácticamente instantánea incluso para destinos lejanos (< 35 ms)

Para una aplicación de navegación en tiempo real donde el usuario espera una respuesta inmediata, Greedy podría ser preferible a pesar de su suboptimalidad. El incremento del 28.6% en consumo energético es un precio razonable por obtener una ruta en milisegundos en lugar de segundos. Sin embargo, para planificación de flotas o escenarios donde la eficiencia energética

es crítica (maximizar autonomía, minimizar costos operativos), A* es la elección indiscutible.



Comparación de nodos expandidos

*Figura 6: Nodos expandidos por algoritmo. La diferencia de órdenes de magnitud es evidente: Greedy (barra roja) expande dramáticamente menos nodos que cualquier variante de A**, resultando en tiempos de ejecución 40 veces menores.

5.5. Visualización de rutas

Para ilustrar cualitativamente las diferencias en las decisiones de ruta, presentamos una comparación visual para el trayecto Ciudad Vieja hacia Maroñas.

*Figura 7a: Ruta óptima encontrada por A**. Nótese cómo el algoritmo puede realizar desvíos estratégicos para recargar si es necesario, manteniendo el consumo total al mínimo (15.57 kWh).

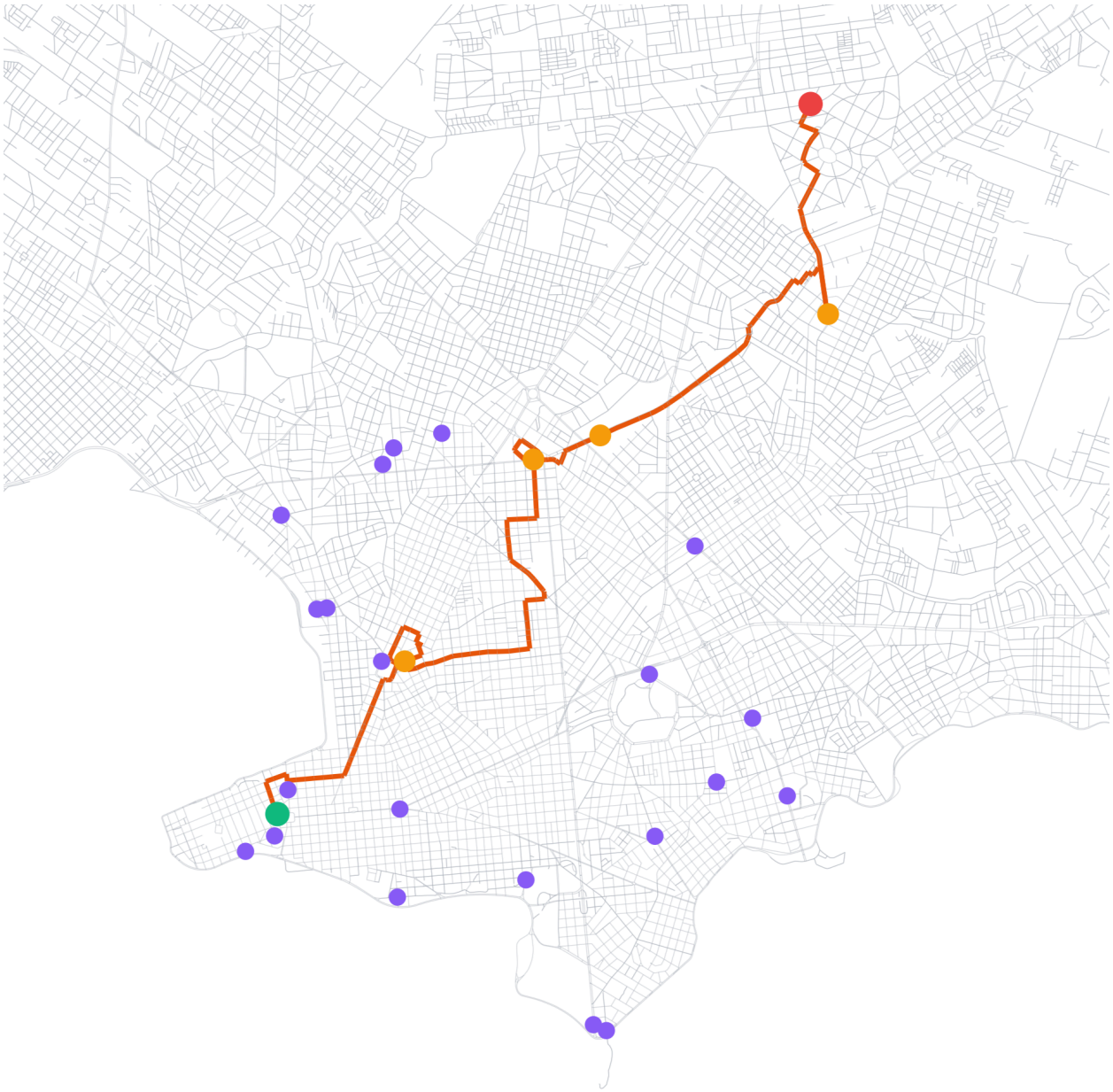


Figura 7b: Ruta encontrada por Greedy. El algoritmo intenta ir directo al objetivo, lo que en este caso resulta en un camino menos eficiente energéticamente (20.19 kWh) y con más recargas, debido a la falta de planificación global.

6. Discusión

6.1. Interpretación de resultados

Los resultados experimentales confirman tanto las predicciones teóricas como las suposiciones prácticas sobre el comportamiento de estos algoritmos. A* cumple su promesa de optimalidad, consistentemente encontrando las rutas de menor consumo energético, a costa de una exploración exhaustiva del espacio de estados.

El rendimiento de Greedy es más variable y dependiente de la estructura específica del grafo. En algunos casos, cuando el destino está en una dirección relativamente libre de obstáculos y la red vial es conducente, Greedy encuentra soluciones muy cercanas al óptimo. En otros casos, especialmente cuando el destino requiere rodeos significativos o cuando la heurística geométrica es engañosa debido a la estructura irregular de las calles, Greedy puede desviarse sustancialmente del óptimo.

Un hallazgo interesante es que la relación entre el número de nodos expandidos y el tiempo de ejecución no es perfectamente lineal. Esto se debe a factores como la necesidad de mantener la cola de prioridad, el costo de las búsquedas en diccionarios para verificar estados visitados, y variabilidad en las operaciones de punto flotante. No obstante, la correlación general es clara: menos nodos expandidos implica ejecución más rápida.

6.2. Limitaciones del estudio

Este estudio tiene varias limitaciones que deben reconocerse. Primero, el modelo de consumo energético es extremadamente simplificado. En vehículos eléctricos reales, el consumo varía significativamente con la velocidad, aceleración, topografía (pendientes), temperatura ambiente, uso de climatización, y estilo de conducción. Un modelo más realista requeriría incorporar perfiles de elevación del terreno y características dinámicas del vehículo.

Segundo, el modelo de recarga es binario y simplista: el vehículo instantáneamente obtiene una cantidad fija de energía al visitar un cargador. En realidad, la recarga toma tiempo (típicamente 20-45 minutos para una carga rápida), y la cantidad de energía recargada podría optimizarse (no

siempre es necesario cargar al máximo). Un modelo más sofisticado podría incorporar el tiempo de recarga como parte del costo de la función objetivo.

Tercero, la discretización de la batería, aunque necesaria computacionalmente, introduce una aproximación que podría en teoría (aunque raramente en práctica) llevar a perder soluciones viables en escenarios muy ajustados.

Cuarto, nuestros experimentos se limitaron a un único origen y destinos únicos. En aplicaciones reales como planificación de flotas, se requieren soluciones a problemas multi-objetivo (múltiples vehículos, múltiples destinos, ventanas de tiempo) que están más allá del alcance de este proyecto.

Finalmente, no se consideraron aspectos dinámicos como tráfico en tiempo real, cierres de calles, o disponibilidad variable de estaciones de carga (ocupadas/fuera de servicio).

6.3. Aplicaciones prácticas

A pesar de las limitaciones, este trabajo tiene relevancia práctica directa. Los sistemas de navegación para vehículos eléctricos están activamente incorporando planificación de recargas en sus algoritmos de ruteo. Empresas como Tesla, con su sistema de “Trip Planner”, automáticamente sugieren paradas en Superchargers basándose en la autonomía del vehículo y la ubicación de estaciones. Nuestro trabajo proporciona una base algorítmica sólida para este tipo de aplicaciones.

Además, el enfoque es aplicable a otros dominios con restricciones de recursos. Por ejemplo, vehículos aéreos no tripulados (drones) tienen limitaciones de batería similares y requieren planificación de rutas que consideren puntos de recarga o estaciones de intercambio de baterías. Robots móviles en almacenes o fábricas enfrentan problemas análogos.

El análisis comparativo entre A* y Greedy también tiene valor pedagógico, ilustrando concretamente el trade-off fundamental entre optimalidad y eficiencia computacional que permea muchos problemas en ciencia de la computación.

7. Costo computacional

Complejidad Temporal de A*:

En el peor caso, A* explora todos los estados alcanzables. Si hay $|V|$ nodos y B niveles discretos de batería, el espacio de estados tiene tamaño $O(|V| \cdot B)$.

Para cada estado expandido, examinamos sus vecinos, de los cuales hay en promedio d (el grado promedio del nodo). Cada operación de cola (inserción/extracción) toma $O(log(|V| \cdot B))$.

Por lo tanto, la complejidad temporal es:

$$O(|V| \cdot B \cdot d \cdot log(|V| \cdot B)) = O(|V| \cdot B \cdot d \cdot log(|V| \cdot B)) = O(|V| \cdot log(|V|))$$

En grafos viales urbanos, d es típicamente pequeño (2-4 aristas por intersección), y B depende de la granularidad de discretización. Con $|V| = 50000$ y $B = 50$, esto da aproximadamente $O(50000 \cdot 50 \cdot 3 \cdot log(2500000)) \approx O\left(2.1 \times 10^9\right)$ operaciones en el peor caso absoluto, aunque en la práctica A* explora solo una fracción del espacio de estados gracias a la heurística.

Complejidad Espacial de A*:

Necesitamos almacenar *g_score*, *f_score* y *came_from* para (en el peor caso) todos los estados, más la cola de prioridad. Cada diccionario puede tener hasta $O(|V| \cdot B)$ entradas. Por lo tanto, la complejidad espacial es:

$$O(|V| \cdot B)$$

Con nuestros parámetros, esto es $O\left(2.5 \times 10^6\right)$ entradas de diccionario, cada una almacenando números flotantes y punteros, resultando en aproximadamente 50-100 MB de memoria, lo cual es perfectamente manejable en sistemas modernos.

Complejidad de Greedy:

Teóricamente, Greedy podría tener la misma complejidad que A* en el peor caso. Sin embargo, en la práctica, Greedy típicamente expande muchos menos nodos debido a su naturaleza "miope" de ir siempre hacia el objetivo. Observamos empíricamente reducciones de 50-100x en nodos expandidos, aunque con pérdida de optimalidad.

8. Conclusiones

Este proyecto ha demostrado exitosamente la aplicación de algoritmos de búsqueda heurística al problema de enrutamiento de vehículos eléctricos con restricciones energéticas. Las conclusiones principales son:

Primero, el algoritmo A* con gestión de batería es capaz de encontrar consistentemente rutas óptimas en términos de consumo energético, incluso en escenarios urbanos complejos que requieren planificación de múltiples recargas. Las tres heurísticas evaluadas (Euclidiana, Manhattan, Octile) producen resultados de calidad idéntica, con diferencias menores en eficiencia computacional.

Segundo, el algoritmo Greedy ofrece una alternativa práctica cuando la velocidad de respuesta es crítica. Aunque consume aproximadamente 28.6% más energía en promedio que A*, Greedy es aproximadamente 40 veces más rápido, haciendo viable su uso en aplicaciones de tiempo real.

Tercero, la discretización del espacio de estados mediante redondeo del nivel de batería es una técnica efectiva para controlar la complejidad computacional sin sacrificar significativamente la calidad de la solución.

Cuarto, la infraestructura de recarga juega un papel determinante en la viabilidad de las rutas. La presencia o ausencia de estaciones de carga en ubicaciones estratégicas puede hacer la diferencia entre alcanzar o no un destino.

El trabajo futuro podría extenderse en varias direcciones: incorporar modelos de consumo más realistas que consideren topografía y dinámica del vehículo; incluir el tiempo de recarga en la función objetivo; explorar algoritmos de aproximación que ofrezcan garantías de calidad con menor costo computacional que A*; y abordar variantes multiobjetivo del problema que optimicen simultáneamente múltiples criterios (energía, tiempo, costo).

En conclusión, este proyecto demuestra que los algoritmos clásicos de búsqueda, apropiadamente adaptados, proveen una base sólida para los desafíos modernos de movilidad eléctrica, balanceando de manera efectiva las necesidades de optimalidad, eficiencia computacional y aplicabilidad práctica.

9. Bibliografía

Algoritmos de Búsqueda y Grafos

1. Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). *Introduction to Algorithms* (3rd ed.). MIT Press.
    - Texto fundamental que cubre Dijkstra, A*, y teoría de grafos en profundidad.
2. Russell, S., & Norvig, P. (2021). *Artificial Intelligence: A Modern Approach* (4th ed.). Pearson.
    - Capítulos 3 y 4 cubren búsqueda heurística y A* desde la perspectiva de IA.
3. Hart, P. E., Nilsson, N. J., & Raphael, B. (1968). "A Formal Basis for the Heuristic Determination of Minimum Cost Paths." *IEEE Transactions on Systems Science and Cybernetics*, 4(2), 100-107.
    - Paper original que introdujo el algoritmo A*.

Electric Vehicle Routing Problem

4. Schneider, M., Stenger, A., & Goeke, D. (2014). "The Electric Vehicle-Routing Problem with Time Windows and Recharging Stations." *Transportation Science*, 48(4), 500-520.
    - Estudio seminal sobre EVRP con ventanas de tiempo y restricciones de recarga.
5. 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.
    - Adaptación de algoritmos de camino más corto al contexto de vehículos eléctricos.
6. Pelletier, S., Jabali, O., & Laporte, G. (2016). "50th Anniversary Invited Article—Goods Distribution with Electric Vehicles: Review and Research Perspectives." *Transportation Science*, 50(1), 3-22.
    - Revisión comprehensiva del estado del arte en ruteo de vehículos eléctricos.

Herramientas y Datos

7. Boeing, G. (2017). “OSMnx: New Methods for Acquiring, Constructing, Analyzing, and Visualizing Complex Street Networks.” *Computers, Environment and Urban Systems*, 65, 126-139.
    - Paper describiendo OSMnx, la herramienta usada para adquisición de datos geográficos.
8. Hagberg, A., Swart, P., & Schult, D. (2008). “Exploring Network Structure, Dynamics, and Function using NetworkX.” *Proceedings of the 7th Python in Science Conference*, 11-15.
    - Documentación de NetworkX, la librería de grafos utilizada en la implementación.
9. OpenStreetMap Contributors. (2024). *OpenStreetMap*. https://www.openstreetmap.org
    - Fuente de datos geográficos utilizada en el proyecto.

Proyectos Relacionados

10. Fiorino, S. (2023). *Maps Pathfinding – Comparando Algoritmos: A* vs Dijkstra, en el mapa de la ciudad.* GitHub. https://github.com/santifiorino/maps-pathfinding
    - Proyecto similar comparando A* y Dijkstra en mapas urbanos.
11. UTE. (2024). *Portal de Datos Abiertos - Infraestructura de Movilidad Eléctrica en Uruguay*. Fuente de datos sobre ubicación de estaciones de carga utilizadas en el proyecto.

Repositorio del proyecto

12. JoaquinBatser/ev-routing