OPTIMIZACIÓN DE RUTAS PARA VEHÍCULOS ELÉCTRICOS

Isabela Muriel Roldán
Universidad Eafit
Colombia
imurielr@eafit.edu.co

Mateo Flórez Restrepo
Universidad Eafit
Colombia
mflorezr@eafit.edu.co

Mauricio Toro
Universidad Eafit
Colombia
mtorobe@eafit.edu.co

RESUMEN
El objetivo de este informe es encontrar la manera de encontrar la ruta óptima en un mapa, de manera que se le pueda decir a un camión que necesite llegar a varios clientes el camino más corto y por donde gastará menos energía. Es importante solucionar este problema debido a que en el futuro la mayoría de los vehículos serán eléctricos y esto ayudará a decirle al usuario cual vía es la más apropiada para utilizar, y no solo para vehículos eléctricos, debido a que esto también puede favorecer a los demás vehículos.

Existen varios problemas relacionados a lo que se trata solucionar, entre ellos el problema de enrutamiento de vehículos o el del agente viajero, estos también serán explicados en este informe.

Palabras clave

Algoritmo, grafo, nodo, tiempos de ejecución, ruta óptima, energía, carros eléctricos, enrutamiento

Palabras de la clasificación de la ACM

Theory of computation → Design and analysis of algorithms → Graph algorithms analysis → Shortest paths

1. INTRODUCCIÓN

El gran avance que ha tenido la tecnología en los últimos años ha generado el deseo de crear objetos inteligentes que le ayuden a las personas en su día a día, esto incluye a los automóviles, que además de querer hacerlos inteligentes, se desea reducir el gasto de combustible para evitar el nivel de contaminación que este genera. Debido a este deseo de cuidar el medio ambiente se han creado carros eléctricos, que utilizan energía como medio de combustible, el problema es que estos no pueden recorrer grandes distancias debido a que se pueden quedar sin energía y no encontrar un lugar donde se pueda recargar.

El objetivo de este proyecto es crear un algoritmo que le ayude a los camiones a encontrar la ruta óptima para llegar a una serie de clientes ubicados en un mapa.

2. PROBLEMA

Sabemos que actualmente la tecnología está tratando todo por ser amigable con el medio ambiente y disminuir todo aquello que perjudique a la sociedad y su entorno. Los vehículos eléctricos han sido un gran ejemplo de esto, una propuesta para reducir la dependencia del petróleo y los gases de efecto invernadero. Sin embargo, el uso de estos vehículos tiene límites en cuanto al rango de conducción y tarda mucho la carga de su batería. El problema en si es crear un algoritmo de recorrido en dos dimensiones que encuentre las rutas óptimas para que un conjunto de vehículos eléctricos visite cierta cantidad de clientes considerando un mínimo gasto neto de tiempo.

3. TRABAJOS RELACIONADOS

3.1 El problema de enrutamiento de vehículos (VRP)

Es un problema de optimización combinatoria y de programación de entero qué pregunta "¿Cuál es el conjunto óptimo de rutas para una flota de vehículos que debe satisfacer las demandas de un conjunto dado de clientes?" Las implementaciones que más se usan para resolver este problema se basan en las heurísticas debido a que para grandes instancias del problema, que como sucede en ejemplos reales, producen buenos resultados. Existen muchas variaciones de este problema según las necesidades de optimización y requeridas para cada situación (Recorrida y entrega, ventajas de tiempo, capacidad, viajes múltiples, vehículo abierto). [1]

Posibles soluciones:

Heurísticas: Las heurísticas son algoritmos con los que se encuentran soluciones aproximadas a las soluciones óptimas, en un tiempo computacional razonable. Una de las heurísticas eficientes para la solución al problema es: La heurística del vecino más cercano.

- La Heurística del Vecino más Cercano: Este algoritmo construye las rutas secuencialmente, seleccionando de manera iterativa los puntos con menor distancia para insertarlos a la ruta. [2]

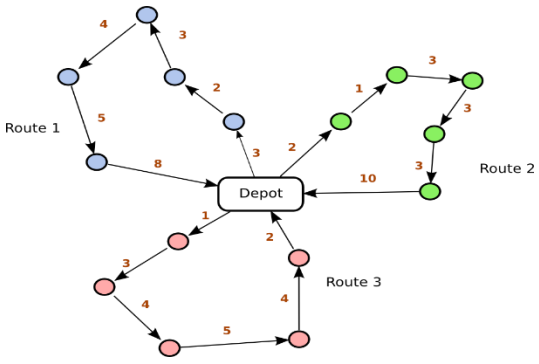


Figura 1: Problema de enrutamiento de vehículos(VRP).

3.2 Problema del agente viajero (TSP)

Este problema no es tan diferente al problema anterior, es un problema de optimización combinatoria que consiste en encontrar un recorrido completo que conecte todos los vértices pasando solo una vez por ellos y luego volver al punto de partida considerando el recorrido o distancia más corto en el mínimo de tiempo. [3]

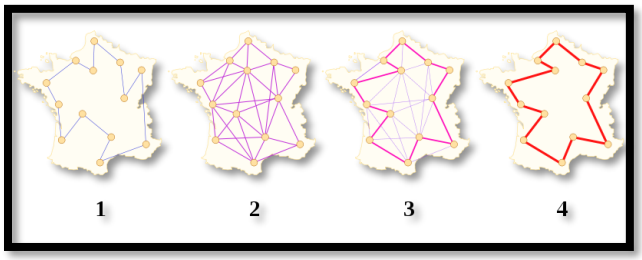


Figura 2: Conjunto de pasos para una solución a TSP

3.3 Problema de las colonias de hormigas

El algoritmo basado en las colonias de hormigas es utilizado para resolver varios problemas que requieren elegir la solución más corta entre varias posibles soluciones. Lo que hace este algoritmo es construir las posibles soluciones e ir escogiendo cual es la más óptima, para de esta manera, actualizar el peso o la distancia que hay entre cada nodo. [4]

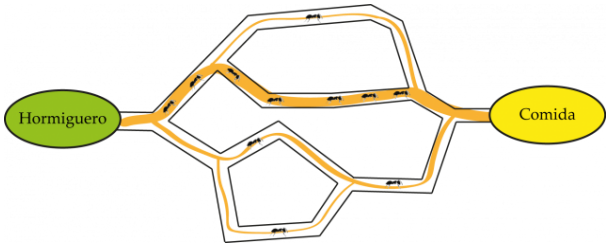


Figura 3: Ejemplo de elección del camino más corto usando algoritmo de la colonia de hormigas.

3.4 Algoritmos genéticos

Los algoritmos genéticos pueden ser utilizados para resolver problemas de búsqueda y optimización, estos están basados en el proceso genético de los organismos vivos debido a que lo que el algoritmo hace es crear soluciones e ir evolucionando de manera que las soluciones dadas sean las mejores y más óptimas.

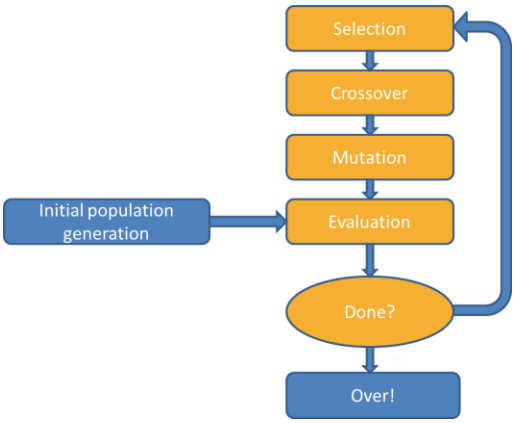
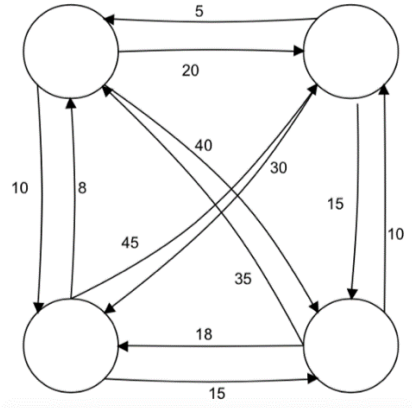


Figura 4: Explicación del funcionamiento de los algoritmos genéticos

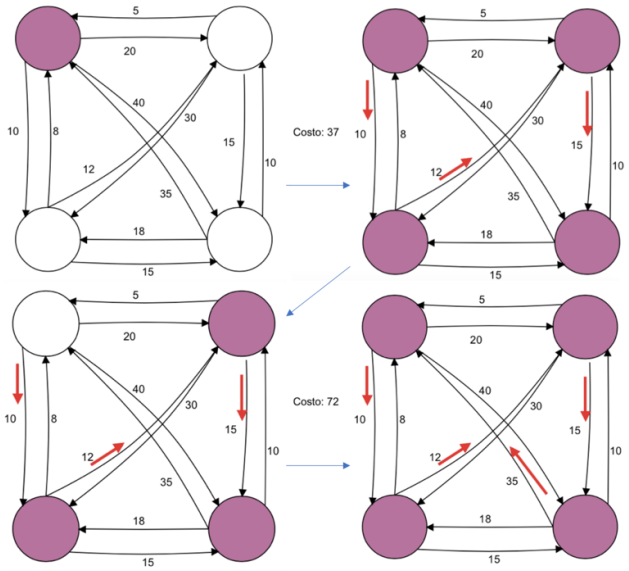
4. AGENTE VIAJERO, VECINO MÁS CERCANO

4.1 Estructura de datos



Gráfica 1: Grafo completo para representar los puntos del mapa y las distancias entre ellos

4.2 Operaciones de la estructura de datos



Gráfica 2: Representación del proceso de búsqueda de una ruta óptima por medio del vecino más cercano, a medida que se recorren los vértices se marcan como visitados.

4.3 Criterios de diseño de la estructura de datos

La razón por la que decidimos utilizar el problema del agente viajero y solucionarlo por medio del vecino más cercano es que está es una solución que a pesar de no dar la solución óptima, resuelve el problema con una complejidad mucho menor que otras soluciones como fuerza bruta, que si da la ruta más corta pero el tiempo de ejecución es muy largo.

Lo que hace el algoritmo es recorrer el grafo, siempre buscando el vértice más cercano al actual, y los va marcando a medida que los recorre, esto asegura que no se repetirán. Además, a diferencia de la solución por medio de fuerza bruta o backtracking, el algoritmo no recorre un vértice más de una vez, lo que asegura que este será mucho más eficaz que los otros mencionados anteriormente.

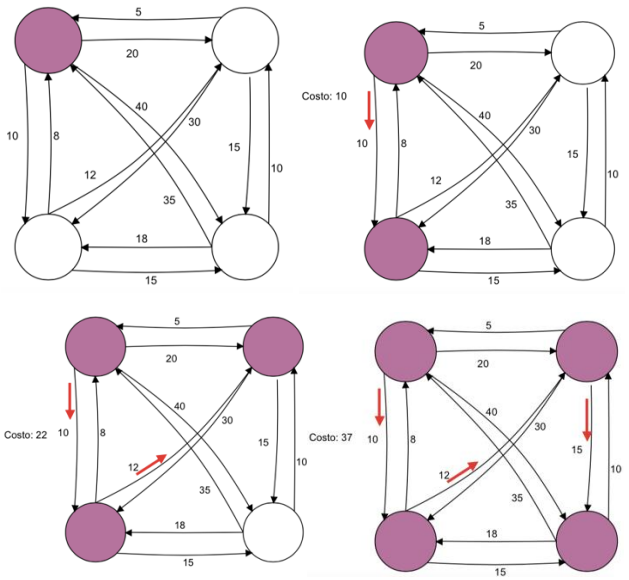
4.4 Análisis de complejidad

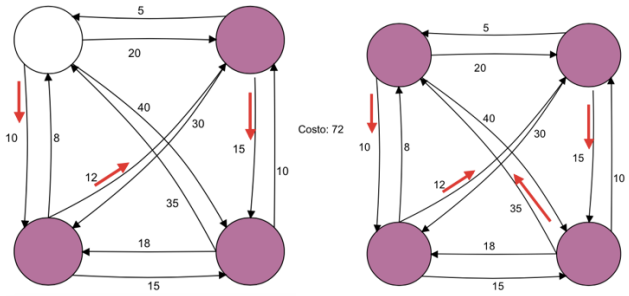
Método	Complejidad
Leer archivo	**O(n)**
Crear el grafo	**O(n^2)**
Encontrar la ruta más corta	**O(nlog(n))**

Tabla 1: Tabla para reportar la complejidad

4.5 Algoritmo

En la gráfica 3 se muestra paso a paso el recorrido que hace el algoritmo para encontrar una ruta eficiente.





Gráfica 3: Representación del recorrido por un grafo

4.6 Calculo de la complejidad del algoritmo

Sub problema	Complejidad
Leer cada archivo separando las variables	**O(L)**
Crear el grafo de listas enlazadas	**O(V*E)**
Calcular la distancia entre dos nodos	**O(X,Y)**
Encontrar la ruta	**O(V*min(E))**
Complejidad total	**O(L+(X,Y)+V^2*E*min(E))**

Tabla 2: Complejidad de cada uno de los sub-problemas que componen el algoritmo. Siendo L el número de información que hay en cada línea del archivo, V el número de vértices o nodos, E el número de aristas o distancias, y X, Y las coordenadas de ubicación de cada nodo en el plano.

4.7 Criterios de diseño del algoritmo

Para crear este algoritmo tuvimos en cuenta la eficiencia de las variables que teníamos definidas hasta el momento, además del tiempo y la memoria en ejecución, y una pequeña solución directa sin considerar aun algunos detalles, para tener una pequeña guía de lo que sería una solución real implementado todos los datos que nos son brindados. Hasta el momento nuestro avance se basa solo en encontrar la ruta más optima teniendo un grafo en un plano con n cantidad de nodos que se clasifican en estaciones, clientes y el depósito, nuestro punto de partida y de llegada.

Creamos entonces una versión que aprendimos del vecino más cercano, donde tenemos en cuenta el costo de cada ruta, y la distancia mínima entre nodos. Ya que este algoritmo es un algoritmo greedy, es decir, vamos a una solución directa evitando implementaciones de recursión. Así tendremos una complejidad más pequeña y más eficacia al encontrar la ruta más óptima. Siendo este un algoritmo con una posible solución, que cumple con los criterios definidos en el problema planteado; sin embargo,

aun debemos tener en cuenta aparte de la ruta, el tiempo y la energía que son los detalles importantes en nuestro proyecto, y que iremos implementando a medida que llegamos a la solución final real.

4.8 Tiempos de ejecución

Conjunto de datos	Mejor tiempo	Peor tiempo	Tiempo promedio
tc2c320s24cf0.txt	131 ms	338 ms	177 ms
tc2c320s24cf1.txt	139 ms	365 ms	187 ms
tc2c320s24cf4.txt	137 ms	299 ms	183 ms
tc2c320s24ct0.txt	134 ms	273 ms	175 ms
tc2c320s24ct1.txt	135 ms	356 ms	183 ms
tc2c320s24ct4.txt	140 ms	480 ms	193ms
tc2c320s38cf0.txt	157 ms	368 ms	208 ms
tc2c320s38cf1.txt	149 ms	328 ms	199 ms
tc2c320s38cf4.txt	159 ms	406 ms	210 ms
tc2c320s38ct0.txt	148 ms	505 ms	210 ms
tc2c320s38ct1.txt	153 ms	329 ms	202 ms
tc2c320s38ct4.txt	159 ms	341 ms	209 ms

Tabla 3: Tiempos de ejecución del algoritmo con diferentes conjuntos de datos

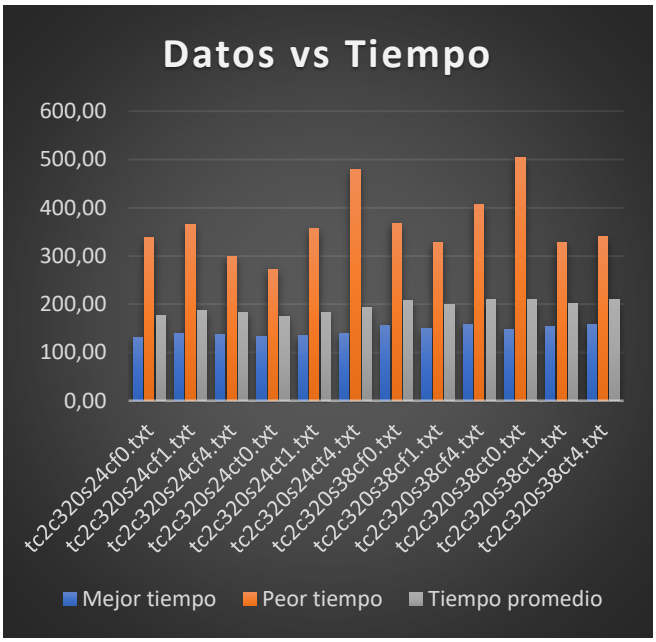
4.9 Memoria

Conjunto de datos	Memoria
tc2c320s24cf0.txt	8.5 MB
tc2c320s24cf1.txt	8.4 MB
tc2c320s24cf4.txt	8.5 MB
tc2c320s24ct0.txt	8.8 MB
tc2c320s24ct1.txt	8.4 MB
tc2c320s24ct4.txt	8.7 MB
tc2c320s38cf0.txt	8.7 MB
tc2c320s38cf1.txt	8.9 MB
tc2c320s38cf4.txt	7.5 MB
tc2c320s38ct0.txt	6.9 MB
tc2c320s38ct1.txt	7.9 MB
tc2c320s38ct4.txt	7.4 MB

Tabla 4: Consumo de memoria del algoritmo con diferentes conjuntos de datos

4.10 Análisis de los resultados

Teóricamente sabemos que el algoritmo que implementamos basado en "el vecino más cercano" para este problema, tiene una complejidad de O(nlog (n)) para el mejor de los casos y el caso promedio, y que para el peor de los casos equivale a O(n^2) lo que tiene coherencia con los resultados de la tabla 3 en donde podemos apreciar lo que se demora para cada conjunto de datos. Siendo el mejor de todos los casos 131 ms lo que se demora en encontrar la ruta y 505ms en el peor de todos los casos. De lo que podemos concluir que es un buen algoritmo para los casos promedios de cada conjunto de datos, aunque en algunos casos tome valores más grandes sigue siendo rápido para la solución a nuestro problema comparado con otros algoritmos. La memoria para cada conjunto de archivos es muy parecida, y no se consume tanta al manejar tantos conjuntos de datos, el limite según la tabla 4 esta entre 7.4 y 8.9 MB ya que no todos trabajan con la misma cantidad de datos.



Grafica 4. Representación grafica de los tiempos ejecución del algoritmo con diferentes conjuntos de datos

REFERENCIAS

1. Wikipedia. Problema de enrutamiento de vehículos. Recuperado Diciembre 12, 2017. https://es.wikipedia.org/wiki/Problema_de_enrutamiento_de_veh%C3%ADcul

2. Ramírez, L.E, una solución al problema de ruteo de vehículos abierto (OVRP), implementando la heurística del vecino más cercano. Recuperado Junio 20, 2016, de la facultad de ingeniería de la Universidad Distrital Francisco José de Caldas: http://repository.udistrital.edu.co/bitstream/11349/2985/1/RamirezRodriguezLuisErnesto.pdf

3. Sancho, F. Algoritmos de hormigas y el problema del viajante. Recuperado Noviembre 17, 2016, del Dpto. de Ciencias de la Computación e Inteligencia Artificial de la Universidad de Sevilla: http://www.cs.us.es/~fsancho/?e=71

4. Caparrini, F. and Work, W. 2016. Algoritmos de hormigas y el problema del viajante. *Cs.us.es*. http://www.cs.us.es/~fsancho/?e=71.

5. Robles Algarín, C. 2010. *Optimización por colonia de hormigas: aplicaciones y tendencias*.