RUTEO DE VEHÍCULOS ELÉCTRICO USANDO GRAFOS

Sebastián Giraldo Gómez
Universidad Eafit
Colombia
sgiraldog@eafit.edu.co

Alejandro Cano Múnera
Universidad Eafit
Colombia
acanom@eafit.edu.co

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

RESUMEN
El problema de encontrar un algoritmo que realice un ruteo optimo con un conjunto de datos en un grafo ha sido uno de los mayores retos planteado en la modernidad. La solución efectiva de este problema es muy importante ya que con esta se lograría un gran ahorro en tiempo, dinero, y una gran reducción en la emisión de gases, logrando así un mayor cuidado del medio ambiente y una mayor eficiencia en las empresas.
Los problemas más relevantes relacionados a este tienen que ver generalmente con encontrar la manera mas efectiva de recorrer un grafo para un fin determinado, ya sea encontrar un circuito Hamiltoniano, Euleriano, entre otros, En un tiempo relativamente corto.
La solución que planteamos fue basada en el vecino más próximo haciendo algunas modificaciones, los resultados son bastante buenos dado que se encontró una solución optima en un tiempo relativamente bueno.

Palabras clave
Ruta, grafos, vehículos eléctricos, vecino más cercano.

Palabras clave de la clasificación de la ACM
Theory of computation→ Design and analysis of algorithms → Graph algorithms analysis → Shortest paths

1. INTRODUCCIÓN →

A lo largo de la historia moderna se han diseñado numerosos algoritmos para recorrer un grafo, aunque algunos algoritmos son más eficientes que otros, no se ha encontrado uno que en un tiempo medianamente corto logre satisfacer las necesidades planteadas, ya que en el tiempo de ejecución son muy demorados. Generalmente se obtiene una respuesta buena y eficiente, pero no se comprueba que es la mejor. En este documento se plantea y se desarrolla un algoritmo que encuentra la ruta óptima para que un conjunto de camiones eléctricos visite un conjunto de clientes, para mejorar el servicio de una empresa.

2. PROBLEMA

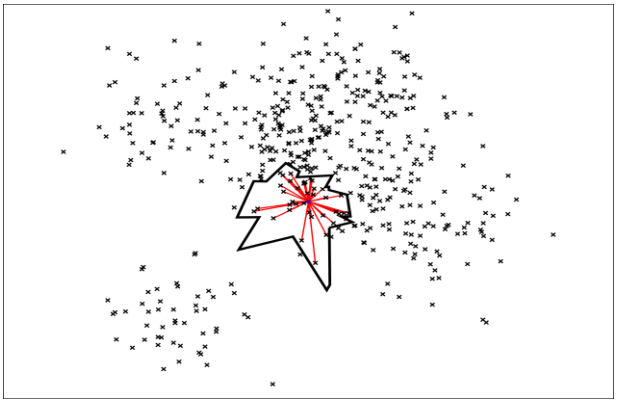
El problema por resolver consiste en diseñar un algoritmo para encontrar las rutas óptimas para que un conjunto de camiones eléctricos visite un conjunto de clientes. La pregunta es la siguiente: Dado una lista de clientes ubicados en un mapa vial bidimensional, un depósito de inicio y fin, y unas restricciones de tiempo y energía ¿cuáles son las rutas para un número libre de camiones eléctricos, para visitar todos los clientes, minimizando el tiempo total, que es la suma del tiempo del recorrido más el tiempo que toman las recargas de batería? [1]. Es importante resolver este problema ya que de esta manera estaríamos ahorrando tiempo, dinero y aportaríamos un mayor cuidado al medio ambiente por la reducción de las emisiones de gases, en la medida de que las compañías implementen algoritmo en sus rutas.

3. TRABAJOS RELACIONADOS

Este problema ha tenido varios intentos y formas de solucionarlo, con diferentes estrategias y formas heurísticas, pero aún no se ha encontrado la solución más óptima a este problema.

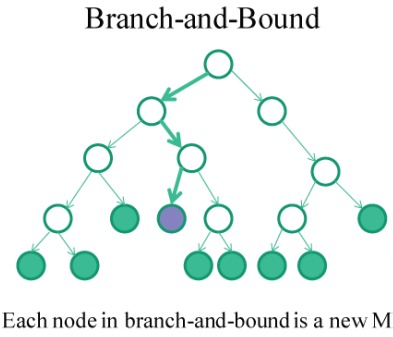
3.1 Vecino más próximo

El Algoritmo del vecino más próximo [2] (NN por sus siglas en inglés) o también llamado algoritmo voraz (greedy) permite al viajante elegir la ciudad no visitada más cercana como próximo movimiento. Este algoritmo retorna rápidamente una ruta corta. Para N ciudades aleatoriamente distribuidas en un plano, el algoritmo en promedio retorna un camino de un 25% más largo que el menor camino posible.20 Sin embargo, existen muchos casos donde la distribución de las ciudades dada hace que el algoritmo NN devuelva el peor camino [3].



Grafica 1. Ruta utilizando vecino más cercano

3.2 Método de branch and bound. El método de branch and bound (ramificación y poda), nos proporciona una solución óptima del problema del agente viajero, calculando mediante el algoritmo simplex la solución del modelo. A medida que aumente el tamaño de la red el método puede tardar gran cantidad de tiempo en resolverse, sin embargo, para redes de mediano tamaño es una excelente alternativa.



Grafica 2. Método de branch and bound.

3.3 Algoritmo Simplex

El Método Simplex [4] es un método iterativo que permite ir mejorando la solución en cada paso. La razón matemática de esta mejora radica en que el método consiste en caminar del vértice de un poliedro a un vértice vecino de manera que aumente o disminuya (según el contexto de la función objetivo, sea maximizar o minimizar), dado que el número de vértices que presenta un poliedro solución es finito siempre se hallará solución.

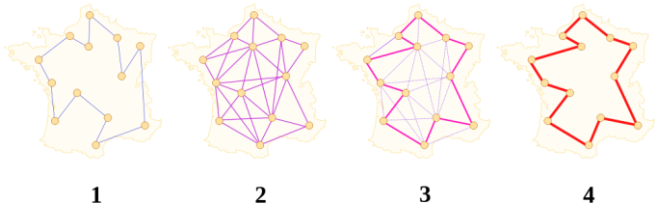
3.4 Método de fuerza bruta

El método de la fuerza bruta no implica la aplicación de ningún algoritmo sistemático, tan solo consiste en explorar todos los recorridos posibles. Es el menos óptimo y el más demorado de todos, debido a que al probar todas las posibilidades sin importar condiciones, ejecuta un sin número de acciones que ralentizan el proceso y lo pueden hacer casi infinito.

3.5 Optimización por colonia de hormigas

El investigador de Inteligencia artificial, Marco Dorigo describió en 1997 un método que genera heurísticamente "buenas soluciones" para el TSP usando una simulación de una colonia de hormigas llamada Ant Colony System (ACS).

ACS envía un gran número de hormigas (agentes virtuales) para explorar las posibles rutas en el mapa. Cada hormiga elige probabilísticamente la próxima ciudad a visitar basada en una heurística, combinando la distancia a la ciudad y la cantidad de feromonas depositadas en la arista hacia la ciudad. La hormiga exploradora, deposita feromonas en cada arista que ella cruce, hasta que complete todo el camino. En este punto la hormiga que completó el camino más corto deposita feromonas virtuales a lo largo de toda la ruta recorrida (actualización del camino global). La cantidad de feromonas depositadas es inversamente proporcional a la longitud del camino: el camino más corto, tiene más cantidad de feromonas.



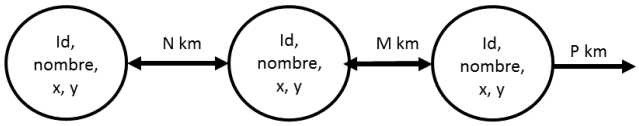
Grafica 3. Recorrido usando optimización de colonia de hormigas.

4. VECINO MAS CERCANO MODIFICADO

A continuación, explicamos la estructura de datos y el algoritmo.

4.1 Estructura de datos

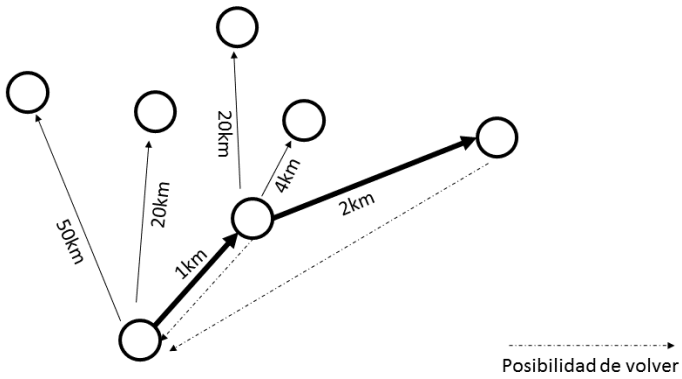
La estructura de datos que se maneja es un grafo basado en matrices de adyacencia, se eligió este modelo ya que de un nodo a se puede ir a todos los nodos b pertenecientes al grafo.



Gráfica 4: Nodos entrelazados entre si gráficamente, cada nodo contiene un identificador, un nombre, una posición en X y una posición en Y; se conoce el peso de los arcos entre todos los nodos.

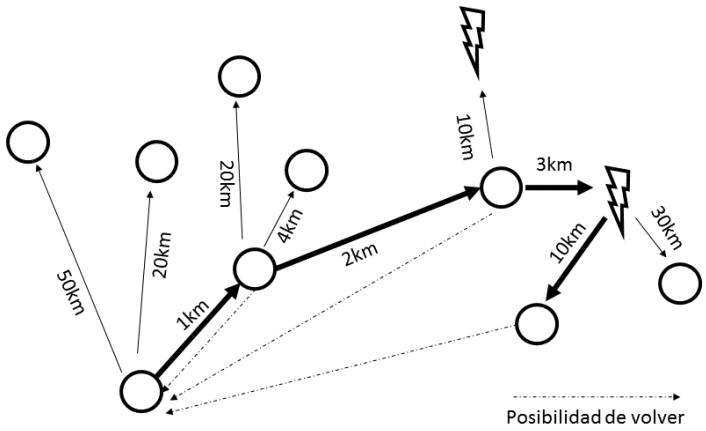
4.2 Operaciones de la estructura de datos

Buscar vecino mas cercano con tiempo y batería:



Gráfica 5: Imagen donde se muestra la elección del vecino más cercano.

Buscar vecino más cercano con tiempo y sin batería:



Gráfica 6: Imagen donde se muestra el recorrido que se realiza cuando se tiene tiempo y no batería en un vehículo.

4.3 Criterios de diseño de la estructura de datos

Después de analizar diferentes soluciones al problema, nosotros concluimos que una solución basada en un grafo dirigido usando una matriz de adyacencia era la mejor opción dado que un nodo A esta conectado con todos los nodos B del grafo, y de esta manera podríamos almacenar todas las distancias entre nodos para posteriormente acceder a ellas con una complejidad de O(1).

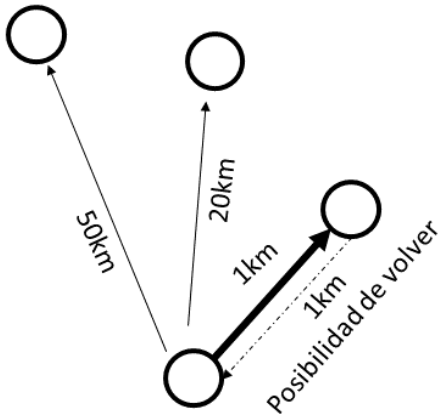
4.4 Análisis de Complejidad

Tabla de complejidades		
Sub problema	complejidad	significado
Leer Txt	O(n)	n: numero de lineas
Crear Matriz de distancias	$O(n^2)$	n: numero de nodos
Vecino mas cercano	O(n)	
Imprimir solucion	O(n*m)	n: numero de rutas m: nodos en dicha
Total	$O(n^2)$	n: numero de nodos

Tabla 1: Tabla para reportar la complejidad

4.5 Algoritmo

El algoritmo se basa principalmente en el vecino más cercano, sin embargo, tiene algunas modificaciones tales como que se puede recorrer con varios vehículos ya que existen algunas limitaciones como el tiempo y la batería. Además, siempre se mantiene control de volver al nodo inicial desde cualquiera visitado con los recursos disponibles.



Gráfica 7: Vecino más cercano

4.6 Cálculo de la complejidad del algoritmo

Tabla de complejidades		
Sub problema	complejidad	significado
Crear matriz de distancias	$O(n^2)$	n: numero de nodos
Vecino mas cercano	O(n)	
Distancia entre dos nodos	O(1)	
Agregar nodo a ruta	O(1)	
Agregar ruta	O(1)	
Total	$O(n^2)$	n: numero de nodos

Tabla 2: complejidad de cada uno de los subproblemas que componen el algoritmo.

4.7 Criterios de diseño del algoritmo

Después de analizar diferentes soluciones al problema, nosotros concluimos que una solución basada en el vecino más cercano es una buena solución al problema de encontrar un óptimo ruteo para vehículos eléctricos en un mapa dado. Mientras se garantice que desde el depósito (Nodo inicial) se pueda ir a todos los nodos y volver al mismo, teniendo en cuenta la batería y el tiempo del recorrido, el algoritmo funciona eficientemente. Uno de los factores claves es que podemos acceder a la distancia entre dos nodos en complejidad de O(1).

4.8 Tiempos de Ejecución

Tabla de consumo de tiempo		
DataSet	Descripcion	Tiempo (ms)
1	345 nodos	205,9
2	345 nodos	190,6
3	345 nodos	187,4
4	345 nodos	203,2
5	345 nodos	218,1
6	345 nodos	179,2
7	359 nodos	191,2
8	359 nodos	187,1
9	359 nodos	183,3
10	359 nodos	220,4
11	359 nodos	199,6
12	359 nodos	183,2
Total	12 archivos	2349,2

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

4.9 Memoria

Tabla de consumo de memoria		
DataSet	Descripcion	Memoria (Mb)
1	345 nodos	2500
2	345 nodos	2000
3	345 nodos	2000
4	345 nodos	2000
5	345 nodos	2000
6	345 nodos	3000
7	359 nodos	2000
8	359 nodos	2000
9	359 nodos	2000
10	359 nodos	3000
11	359 nodos	3000
12	359 nodos	3000
Total	12 archivos	28500

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

4.10 Análisis de los resultados

Tabla de consumo de memoria y tiempo			
DataSet	Descripcion	Memoria (Mb)	Tiempo (ms)
1	345 nodos	2500	205,9
2	345 nodos	2000	190,6
3	345 nodos	2000	187,4
4	345 nodos	2000	203,2
5	345 nodos	2000	218,1
6	345 nodos	3000	179,2
7	359 nodos	2000	191,2
8	359 nodos	2000	187,1
9	359 nodos	2000	183,3
10	359 nodos	3000	220,4
11	359 nodos	3000	199,6
12	359 nodos	3000	183,2
Total	12 archivos	28500	2349,2

Tabla 5: Análisis de los resultados obtenidos con la implementación del algoritmo

5. CONCLUSIONES

Para concluir, luego de analizar algunas estructuras de datos y ver la solución planteada vemos que la solución obtenida es optima y en tiempo bueno: La ruta obtenida en la solución final, satisface los parámetros dados. Por otra parte, pudimos deducir que el cambio de factores tales como: el tiempo de carga, el nodo seleccionado para visitar, entre otros pueden cambiar la solución. Este proyecto tiene muchos enfoques ya que con pocas modificaciones se puede proyectar a empresas del medio local para hacer sus rutas comerciales óptimas.

5.1 Trabajos futuros

Lo que nos gustaría mejorar más en un futuro es la optimización del código e implementar métodos de búsqueda más rápidos, para así obtener menor consumo de memoria y de tiempo, y posiblemente una ruta más óptima.

AGRADECIMIENTOS

Esta investigación fue soportada parcialmente por el Icetex – Programa Ser Pilo Paga 3

REFERENCIAS

[1] Montoya A., Guérethttps C. [En línea] Disponible en: https://hal.archives-ouvertes.fr/hal-01245232/document [Accedido el 03 de marzo de 2018]

[2] Wikipedia, El Problema del Viajante. [En línea]

Disponible en:

https://es.wikipedia.org/wiki/Problema_del_viajante

[Accedido el 03 de marzo de 2018]

[3] Ingeniería Industrial Online, El Problema del Agente Viajero-TSP. [En línea] Disponible en:

https://www.ingenieriaindustrialonline.com/herramientas-para-el-ingeniero-industrial/investigaci%C3%B3n-de-operaciones/problema-del-agente-viajero-tsp/

[Accedido el 03 de marzo de 2018]

[4] Ingeniería Industrial Online, El Método Simplex. [En línea] Disponible en:

https://www.ingenieriaindustrialonline.com/herramientas-para-el-ingeniero-industrial/investigaci%C3%B3n-de-operaciones/m%C3%A9todo-simplex/

[Accedido el 03 de marzo de 2018]