ALGORÍTMO PARA RUTEO DE K VEHÍCULOS ELECTRICOS A BASE DE GRAFOS, RECORRIENDO TODOS LOS VÉRTICES DE ESTE

Rafael Villegas Michel
Universidad EAFIT
Colombia
rvillegasm@eafit.edu.co

Felipe Cortés Jaramillo
Universidad EAFIT
Colombia
fcortesj@eafit.edu.co

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

RESUMEN

En el presente documento se analiza el problema de enrutameinto de vehiculos eletricos, y se propone una soluciona base de un algoritmo para dicho problema, tratando de llegar a un resultado lo mas cercano posible a lo óptimo, debido a la dificultad NP (tiempo no polinomial) para hacer un algoritmo completamente óptimo en este caso. Para ello se usa la teoria de grafos de la computación.

Palabras clave: Algoritmo, Grafos, TSP , Computacion, Routing.

1. INTRODUCCIÓN

En el mundo moderno, los vehículos eléctricos están ganando cada vez una mayor importancia gracias a las ventajas que traen hacia el medio ambiente con respecto a los vehículos a base de combustibles fósiles, lo cual ha impulsado su uso en todo tipo de tareas de la vida diaria de las personas.

Un gran conjunto de esas actividades es todas las relacionadas con el transporte de carga y pasajeros de un lado a otro. El problema con tales actividades es que el rango de conducción es limitado, y el tiempo de carga de la batería es relativamente alto. Por esta razón sería de gran ayuda poder saber la ruta más corta por la que debería pasar el vehículo para gasta la menor cantidad de energía y alcanzar a recargar su batería con la carga necesaria para terminar su trayecto, en caso de que esta se agote.

El problema de enrutamiento de vehículos eléctricos, en su implementación, consiste en un conjunto de lugares con ubicaciones determinadas, unidos entre si por vias por las que los camiones transitarán. Esto puede ser pensado como un grafo G = (V,E), en el que cada lugar objetivo es un vértice V del grafo y cada vía entre ellos es una arista E del mismo. Entre los lugares objetivo hay estaciones de carga de batería, en caso de que un vehículo necesite hacer una recarga, y hay una estación central, representada como el vértice más importante, de donde salen todos los vehículos a repartir sus cargas. Todos los camiones salen de la central, recorren el grafo entregando paquetes y recargando su batería cuando sea necesario, y vuelven a la central cuando hayan terminado con todas sus entregas pendientes.

2. PROBLEMA

El problema a resolver consiste en el diseño de un algoritmo para encontrar las rutas óptimas para que un conjunto de camiones eléctricos visite un conjunto de clientes, suponiendo que no hay límites en el tamaño de la flota, la carga de la batería es lineal y puede ser parcial, los vehículos avanzan a velocidad constante y se desplazan en línea recta de un punto a otro.

Esto se hace con el fin de generar una ruta óptima para que los camiones gasten la menor cantidad de energía posible en el menor tiempo posible, mientras entregan todo lo que se necesita entregar.

3. TRABAJOS RELACIONADOS

3.1 Problema del Agente Viajero (TSP)

Una persona quiere visitar muchas ciudades para vender mercancía. Quiere empezar en la ciudad donde vive, visitar cada ciudad exactamente una vez y volver a la ciudad donde empezó. Para ello, ¿en qué orden debería visitar las ciudades de forma que el tiempo que le tome hacer todo el viaje sea el mínimo posible? [1]

Su importancia radica en que muchos otros problemas pueden ser modelados de la misma forma que el agente viajero, y si se encuentra una solución eficiente para uno de ellos, se encuentra para todos, resolviendo muchos problemas de gran relevancia para la ciencia de la computación de una sola vez. [1]

Desafortunadamente, no hay una solución óptima para este problema (o aún no se ha encontrado ninguna), ya que el número de pasos que necesita ejecutar el algoritmo crece de acuerdo al tamaño del problema, en este caso de acuerdo a la cantidad de ciudades, haciendo que incluso un computador que sea capaz de hacer un millón de instrucciones por segundo, para analizar la ruta óptima para 30 ciudades se demore más de 200.00.00.00.00.00.00.000 años, un tiempo inaceptable. Además, hacer el computador unas 10 o 20 veces más rápido, aún no sería suficiente como para que el tiempo que toma resolver el problema sea aceptable. [1]

Entre las soluciones aproximadas que se encuentran para este algoritmola mas relevante es la del vecino mas cercano, que desde el vertice inicial analiza el paso hacia todos sus vecinos, y elige irse por el menor peso, repitiendo este proceso hasta que no hayan vertices por analizar, momento en el cual vuelve al vertice inicial.

3.2 Problema de enrutamiento de vehículos (VRP)

El problema del enrutamiento de vehículos ha sido uno de los problemas mas buscados tanto como por los programadores en si, como por la economía, ya que el problema de mandar varios vehículos por unas rutas optimas con el fin de guardar tiempo y gasolina, ha estado en constante búsqueda por más de un negocio.[2]

Este problema ya ha sido resuelto por la implementación de varios algoritmos la cual implementan una combinación de posibles rutas y resulta la ruta mas optima y que incluya todos los puntos por los cuales tiene que pasar el vehículo para su trayecto diario. Sin embargo, a pesar de que existe una solución no es la mas optima o buena, ya que la creación de un algoritmo mas eficaz o forma de combinar las rutas mas corta puede tumbar a la anterior fácilmente, por lo que la solución definitiva no existe en estos días.[2]

Entre los algoritmos que se implementan en esta solución existen otras variaciones del problema la cual implican que se piensen más factores o variables que pueden afectar las rutas de los vehículos, lo cual complica mas el problema de enrutar varias rutas para los vehículos, entre estos están La entrega de productos, tiempo, telecomunicaciones y muchas más variaciones.[2]

3.3 Sistema de Colonia de Hormigas (ACS)

El problema del agente viajero ha sido uno de los problemas computacionales mas complicados para resolver por las muchas variaciones que pueden surgir en las rutas y que una de todas ellas sea la mas efectiva, sin embargo, unos estudios que se hicieron revelan que esta solución ya esta implementada, pero no para los humanos, sino para las hormigas. Estos diminutos seres por medio de las feromonas y hondas entre sus antenas permiten fijarse rutas para poder recorrer toda la colonia y hacer paradas en los lugares que se necesitan. Estos caminos son creados entre ellas con el propósito de minimizas tiempo y aumentar la productividad.[3]

3.4 Algoritmo Genético (GA)

Una de las soluciones a los problemas de optimización mas frecuentes se desarrolla gracias a un algoritmo el cual sea capaz de abordar todas las posibles soluciones y además de brindar consigo una solución optima al problema. Pero, imagínese que el problema se acerca mas a la realidad y esto conlleva a que un millón de variables se adicionen al problema lo cual hace que el algoritmo creado anteriormente se vuelva obsoleto. Ahora, imagínese un algoritmo que al principio desarrolla el problema como se busca de una forma buena pero no la más efectiva y cuando este problema evolucione el algoritmo también lo hará, porque tendrá en cuenta los estados anteriores y los mejorara de tal forma que el algoritmo nuevo es funcional y eficaz que el anterior, Esto es un algoritmo genético, ya que su principal función o característica es evolucionar o mejorarse ya que esta inspirado en la ley natural, de la selección natural , por lo que el mejor algoritmo sobrevive y se mejora para poder solucionar problemas de una forma mas efectiva y es principalmente utilizado en los problemas que se refieren a combinaciones u optimización.[4]

Aplicado al TSP o VRP, se vería desde el punto en que se llega a una solucion aproximada con un algoritmo, pero se el algoritmo se sigue mejorando a si mismo, hasta llegar a una solucion mas óptima.

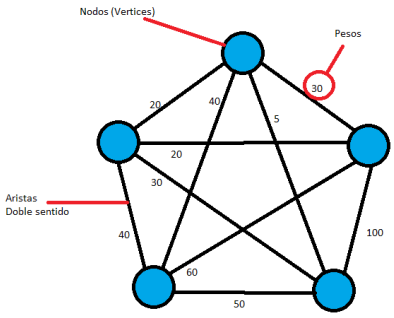
4. Solución planteada: Algoritmo CM (Cortés-Michel)

A continuación se explica la estructura de datos usada y el algoritmo planteado.

4.1 Estructura de datos

La estructura usada para resolver este problema es un grafo, el cual consta de una serie de vertices V distribuidos en el espacio, conectados entre si por aristas E. Estas tienen un peso que simboliza la distancia entre sus dos nodos.

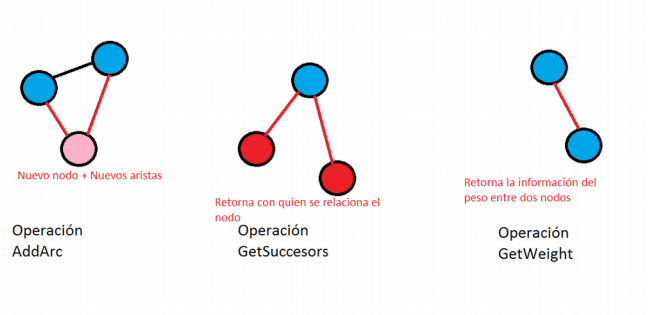
Cada vertice representa una locación, y cada arista representa la forma de llegar de un lado a otro, y el tiempo que demora en hacerlo.



Gráfica 1: Representación de un grafo.

4.2 Operaciones de un grafo

Las operaciones basicas que se necesita que tenga el grafo para que el algoritmo CM funcione son: 1. Añadir aristas entre vertices con un peso determinado, 2. Poder saber cuáles son los vertices con los que se conecta cualquier otro vertice y 3. Obtener el peso de una arista entre dos vertices.



Gráfica 2: Operaciones que se aplican al grafo.

4.3 Criterio de diseño de la estructura de datos

Se decidió usar un grafo para representar el conjunto de locaciones ya que es una forma eficiente para representar un mapa de dos dimensiones, pudiendo almacenar cada locación en un vertice diferente que se debe visitar, y su distancia con respecto a otros mediante aristas.

4.4 Analisis de complejidad del grafo

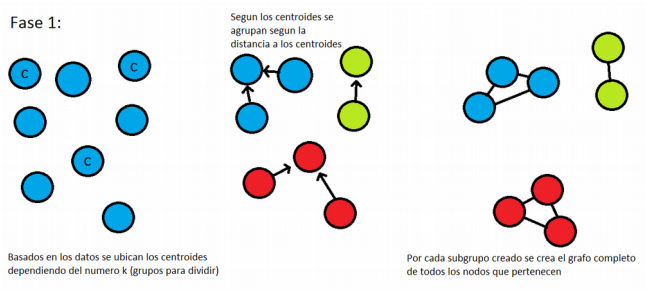
Acción	**Complejidad**
Obtener los vecinos de V	**O**(n)

Obtener el peso entre V y V’	**O(**n**)**
Añadir arista entre V y V’	**O(**1**)**

Tabla 1: Tabla para reportar complejidad de las operaciones de un grafo (con listas).

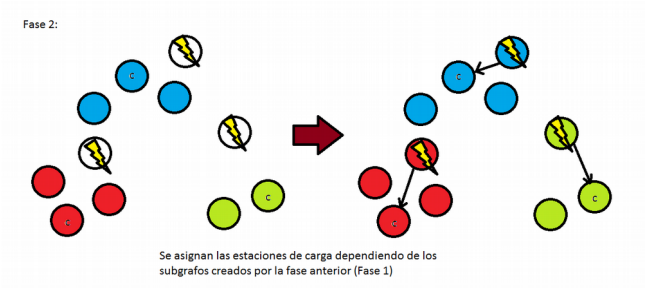
4.5 Funcionamiento del aloritmo CM

Primero se hace una partición del total de vertices para generar k subgrafos diferentes, especificado por teclado, mediante el algoritmo k-means[5], el cual establece k centroides en todo el grafo, e identifica que vertices estan cerca de cada centroide, agrupandolos y repitiendo este proceso hasta que no hayan mas cambios que hacer.



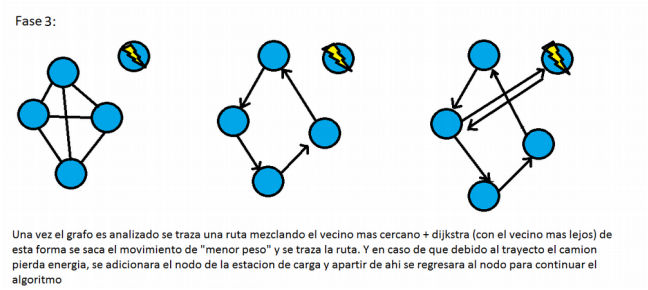
Gráfica 3: Algoritmo K-means.

Luego, se crean los k subgrafos, y se cogen todas las estaciones de carga de la bateria y se analiza el subgrafo mas cercano que no tenga ninguna dentro de si, añadiendola a este. Si todos los subgrafos ya tienen al menos una estación de carga, entonces se añade al subgrafo mas cercano, sin importar que ya tenga otras estaciones dentro de si. Si está el caso en el que hay menor numero de estaciones que de subgrafos, se asigna una estacion a mas de un subgrafo, hasta que todos queden con al menos una estación.



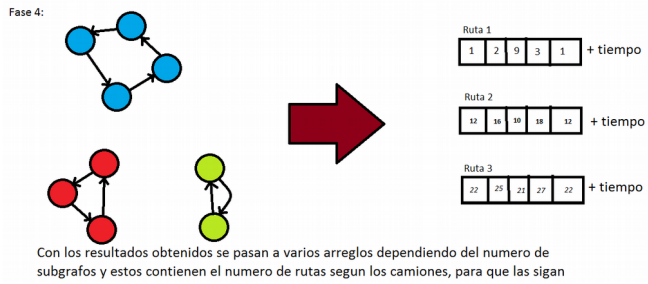
Gráfica 4: Repartición de las estaciones en los subgrafos.

Seguido de lo anterior, para cada subgrafo se aplica el siguiente algoritmo: se marca el vertice inicial como visitado y se analizan los vecinos de este, y se reconoce el mas cercano. Con esa información se analiza si la bateria es capaz de andar esa distancia mas la distancia desde el vertice hasta la estación de carga mas cercana. Si no es capaz, el carro va y carga la bateria hasta que alcance el 100% de carga. Si es capaz, entonces se sigue el camino encontrado, y se marcan los vertices visitados durante el trayecto. Esto se repite hasta que desde un vertice V, todos sus vecinos ya hayan sido visitados, momento en el cual se hace dijkstra hacia el origen y se sigue ese camino. Cada vez que se mueva de un vertice a otro se va sumando a la distancia total recorrida y ademas se le resta a la bateria lo que se consumió en cada camino.



Gráfica 5: Vecino mas cercano mezclado con Dijkstra.

Al final, el algoritmo CM retorna una lista de parejas de un arreglo en donde esta el camino seguido por el vehiculo y el tiempo total que se demoró en hacer el recorrido, esto para cada vehiculo.



Gráfica 6: finalización del algortimo y valor de retorno.

4.6 Complejidad del algortimo

El algoritmo tiene una complejidad de **O(**kn^3**)**

4.7 Criterios de diseño del algoritmo

La particion inicial se hace para distribuir las rutas que debe tomar cada vehiculo, para que ninguno repita la ruta ni los vertices de otro.

La distribucion de las baterias se hizo asi para garantizar que cada subgrafo tenga al menos una estacion.

La trayectoria por los vertices se pensó en estilo voraz-heurístico, usando una variacion de la solucion mas popular para el TSP, debido a su parecido con el EVRP: el vecino mas cercano. No es la optima, pero se pensó que al analizar dijkstra hacia el vecino mas lejano, comparado con el vecino mas carcano se podia llegar a visitar varios vertices de una sola vez en el caso de que la distancia de dijkstra fuera menor, ahorrando tiempo para analisis de vecino mas cercano en un futuro.

La decision de ir a cargar la bateria si no se es capaz de completar un viaje hacia el siguiente vertice mas la distancia hacia la estacion de carga fue tomada para evitar el desgaste extremo de la bateria hasta el punto de que el

vehiculo no pueda andar mas. Es una cantidad heuristica que consideramos suficiente para evitar tal situación.

4.8 Tiempo de Ejecución

Con 345 nodos, y 20 camiones se demora 44569312 ns en leer datos y 654764470 ns en procesarlos.

Con 345 nodos, y 30 camiones se demora 39717287 ns en leer datos y 900885450 ns en procesarlos.

Con 345 nodos, y 5 camiones se demora 42513337 ns en leer datos y 226689213 ns en procesarlos.

REFERENCIAS

1. Shahriari, S. Traveling salesman problem. Salem Press Encyclopedia of Science, 2014.
2. Toth, P & Vigo, D. The Vehicle Routing Problem. Discrete Mathematics of Neural Networks: Selected Topics, 2002.
3. Xiaoxia Zheng, Yang Fu, "Ant Colony Optimization Algorithm Based on Immune Strategy", 2011.
4. Chun-Hao Chen, Tzung-Pei Hong, Vincent S. Tseng, "A SPEA2-based genetic-fuzzy mining algorithm", 2010.
5. Bottou, L. & Bengio, Y. Convergence Properties of the K-Means Algorithm. 1995.