ENRUTAMIENTO DE VEHICULOS ELÉCTRICOS

Eduard Damiam Londoño
Universidad
País
edlondonog@eafit.edu.co

Gonzalo Garcia Hernandez
Universidad
País
ggarciah@eafit.edu.co

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

RESUMEN

En este documento le daremos solución a un problema presentando actualmente que involucra los autos eléctricos y su eficiencia en cuanto a recorrer rutas optimizando el tiempo empleado y los recursos disponibles para este

Palabras clave

Grafo- Agente Viajero – Energía – Rutas – Optimización Combinatoria

Palabras clave de la clasificación de la ACM
Mathematics of computingDiscrete mathematics Combinatorics Combinatorial Optimization

1. INTRODUCCIÓN

Las reservas de petróleo se están agotando, por lo que debemos buscar una nueva alternativa de combustible, es ahí donde surge la idea de implementar la electricidad como combustible para nuestros vehículos, esto conlleva a nuevas problemáticas y limitaciones para las cuales se está buscando una solución.

Entre estas problemáticas se encuentra la duración de la batería para recorrer grandes trayectos y la rapidez con la que abastecemos nuevamente la batería de nuestro vehículo.

En este documento hablaremos de una posible solución a esta gran problemática, para ello emplearemos algunas pruebas demostrando la eficacia de nuestro algoritmo y demás datos en pro de la investigación de esta posible solución.

2. PROBLEMA

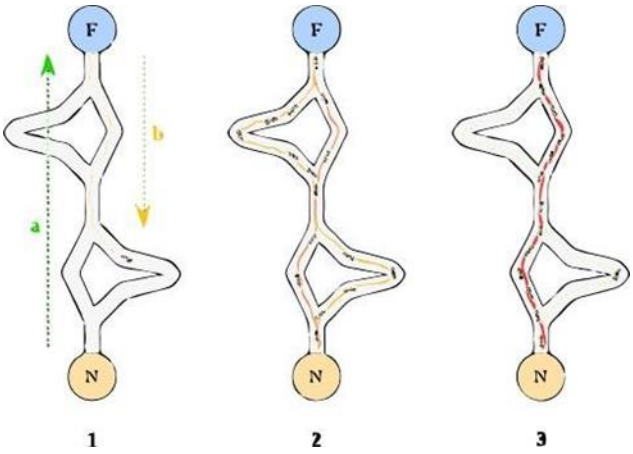
El problema consiste en optimizar las rutas de entregas que hacen los camiones eléctricos actualmente para visitar un listado de clientes. El propósito de resolver este problema radica en disminuir el tiempo en que tarda un camión eléctrico en salir de la empresa, visitar los clientes asignados, recargar batería (en caso de que sea necesario) y volver.

3. TRABAJOS RELACIONADOS

El problema de enrutamiento de vehículos es una generalización del problema del viajero que a su vez es un problema de optimización combinatoria, hay muchos problemas de este estilo, por lo que se han creado varios algoritmos heurísticos para solucionar estos problemas aquí 4 de ellos:

3.1 Algoritmos de la colonia de hormigas

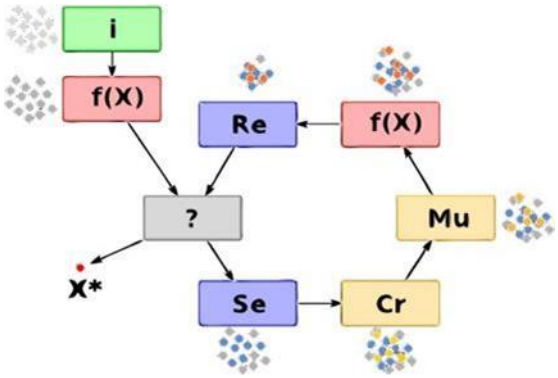
Los algoritmos de colonia de hormigas son un tipo de algoritmos basado en el comportamiento de las hormigas, se empieza con una hormiga que busca un camino al azar, la hormiga va dejando un rastro de feromonas hasta que regresar el inicio [6], entonces algunas de las otras hormigas se ven atraídas por el rastro de feromonas, mientras más tiempo tarde una hormiga en ir y regresar más rápido desaparecerán las feromonas, y las otras hormigas optaran menos por ir a ese camino, lo cual nos da una lista de caminos que son más cortos que los demás.



(grafico que muestra una representación de un algoritmo de colonia de hormigas, se puede ver como en el camino más corto es en el que quedan las feromonas)

3.2 Algoritmo genético

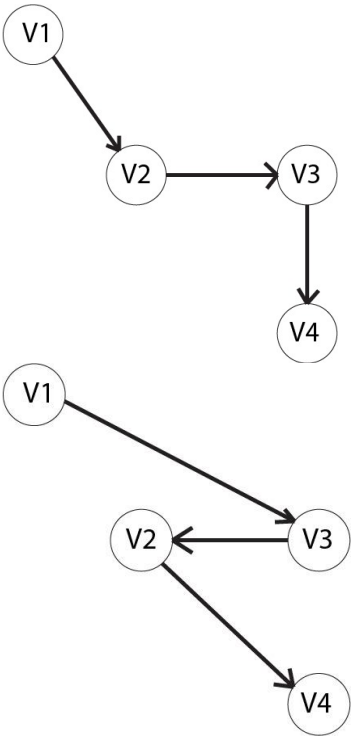
Los algoritmos genéticos están basados en la teoría de evolución y en la genética, su funcionamiento consiste en escoger una población inicial de datos los cuales son llamados cromosomas, estos cromosomas son sometidos a una serie de pruebas para escoger los cromosomas más prometedores [5] y se cruzan para generar nuevos datos más prometedores, a su vez se a los cromosomas prometedores se les pueden asignar mutaciones, lo cual es hacerle alguna modificación al azar para ver si esto lo mejora o lo empeora, lo cual permite alcanzar zonas del espacio de búsqueda no cubiertas por la población original, por ejemplo en caso del TSP la selección se haría en base a cual de todos los genes tiene la menor cantidad de peso total, y luego se haría la combinación entre esos genes, una mutación se haría cambiando intercambiando al azar alguno de los vértices.



(grafico que muestra una representación de un algoritmo genético, i es la inicialización, f(x) es la función, Se es la selección. Cr el cruzamiento, Mu la mutación, la condición de parada y x* las soluciones) [2]

3.3 Búsqueda Tabú

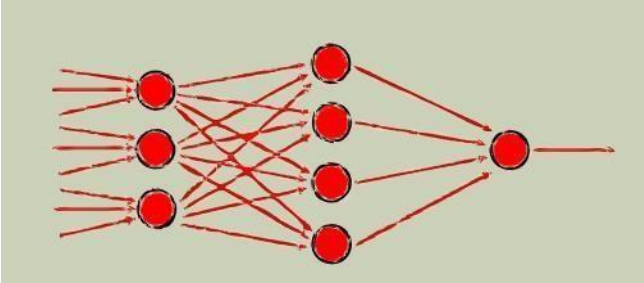
La búsqueda tabú es un método de optimización que consiste en usar estructuras de datos para que una vez se encuentre una posible solución se le marque como “tabú” y no sea visitada otra vez, en ejemplo de esto con el problema del viajero es que una vez se tenga la primera solución, se intercambia en orden de dos ciudades para ver cuál es más prometedora que la otra y así agregarla o no la lista tabú



(grafico que muestra una búsqueda tabú, en esta se intercambia el orden de V2 y V3 para generar una nueva solución)

3.4 Enjambre de Partículas

Se trabaja con una población de soluciones candidatas llamadas partículas, las cuales de desplazan conforme a a reglas matemáticas, el movimiento de las partículas depende de su mejor posición obtenida, y de la mejor posición global y cuando se descubren nuevas y mejores posiciones estas también orientan los movimientos de las partículas, el proceso se repite hasta hallar (no siempre) la mejor solución.

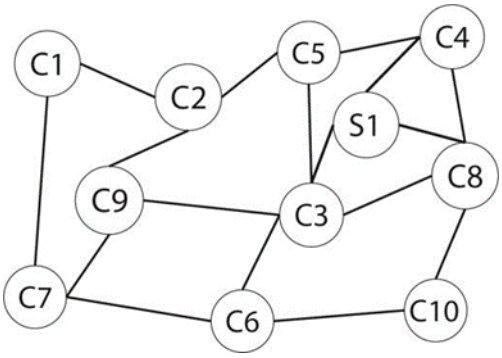


(Grafico que representa un algoritmo de enjambre de partículas)

4. VECINO MAS CERCANO

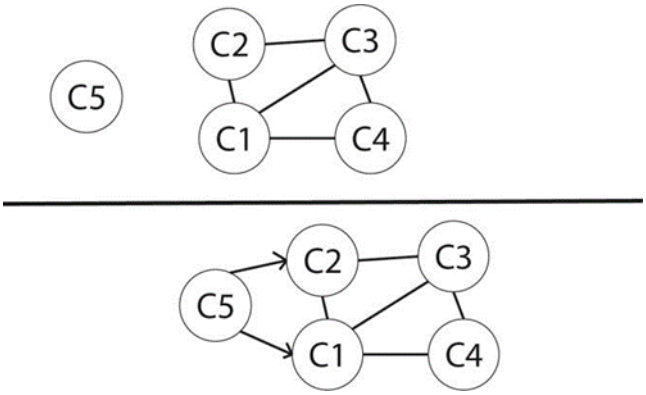
A continuación, explicamos la estructura de datos y el algoritmo.

4.1 Estructura de datos

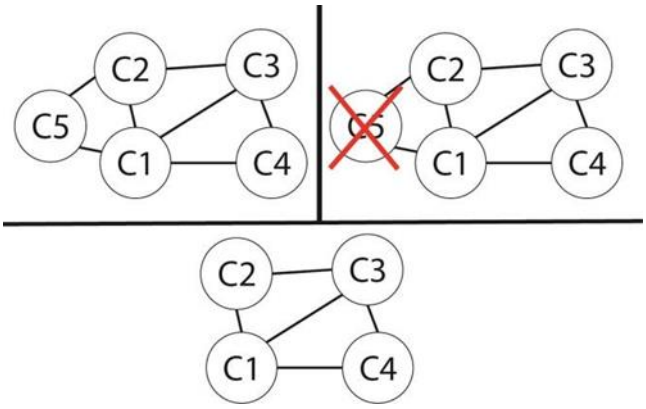


(Grafica 1) Nuestra estructura de datos consiste en un grafo conformado por calles y estaciones de recarga en el cual podremos desplazarnos entre cada uno de los nodos a través de las aristas.

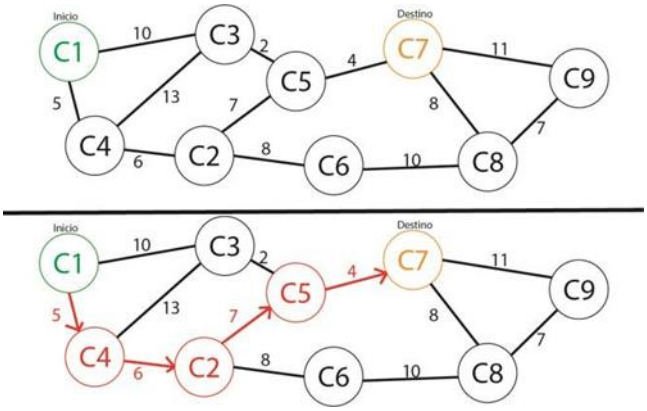
4.2 Operaciones de la estructura de datos



(Grafica 2) Operación para insertar un nuevo vértice en el grafo: Crea un nuevo nodo y lo conecta por medio de aristas a uno o más nodos.



(Grafica 3) Operación para eliminar un vértice del grafo: Se elimina el nodo y sus conexiones con los demás nodos.



(Grafica 4) Operación para encontrar la ruta de un vértice a otro: Se establece un nodo inicial y un nodo destino y luego empezamos a trazar una ruta partiendo desde nuestro nodo inicio considerando cual es la arista de menor peso hasta llegar al nodo destino.

4.3 Criterios de diseño de la estructura de datos

Consideramos que la memoria invertida en un grafo es menor a la memoria invertida en otro método como lo hubiese sido una matriz, aparte de que nos quedaba de una manera más organizada con sus respectivos costos, nombre, vecinos, etc.

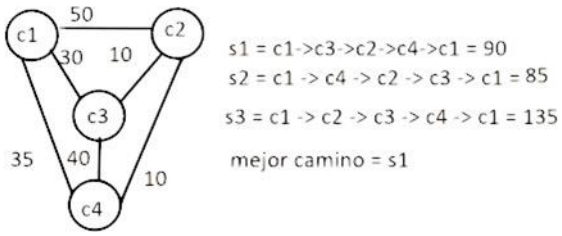
4.4 Análisis de Complejidad

Método	Complejidad
Insertar	O(n)
Eliminar	O(n)
Trazar Ruta	$O(n^n)$

Tabla 1: Tabla para reportar la complejidad

4.5 Algoritmo

Nuestro algoritmo se basa en una implementación del vecino más cercano y de la búsqueda tabú, se empieza aplicando el algoritmo del vecino más cercano que consiste en pararse en un vértice y luego ver cuál de todos los arcos de dicho vértice tiene menos peso para luego irse por ese arco repitiendo el proceso hasta haber recorrido todo el grafo hasta tener una solución.



Cada auto tiene un tiempo límite para hacer el recorrido, para no sobrepasar ese tiempo límite antes de ir de un vértice actual a un vértice destino, se mira el tiempo que se tardaría ir del vértice destino al depósito, si la suma de lo que tarda, más el tiempo que ha pasado (tomando también el tiempo que tardaría en ir del vértice actual al vértice destino) es mayor al tiempo límite del auto, el auto es llevado al depósito y se manda otro auto desde el depósito al vértice destino.

(Gráfica 5) en este ejemplo se toman 3 caminos distintos ya que el algoritmo no puede tomar más (solo hay tres vecinos de c1), y se escoge el camino s1 ya que es el más corto.

Como los autos son eléctricos, se necesita cargarlos para esto hay varias estaciones de cargas, se visita una estación de carga cada vez que la batería baje del 40%, entonces se miran todas las estaciones de carga, y se mira en cual se tarda menos tiempo total, siendo el tiempo total la cantidad de tiempo que tarde en ir desde el vértice actual hasta la estación de carga más lo que se tarde cargando en la estación, luego se mira si la suma de ese tiempo mas el tiempo del recorrido hasta el momento es mayor a la cantidad del tiempo límite, si es mayor se envía el auto al depósito, si no es mayor se envía el auto a la estación de carga y de la estación de carga al vértice más cercano desde donde estaba antes de ir a la estación de carga.

También se genera una segunda solución de una manera muy parecida, con la única diferencia de que cuando un auto llega a su tiempo límite y va a la estación, el nuevo auto va al cliente más cercano no visitado desde la estación.

4.6 Cálculo de la complejidad del algoritmo

Sub problema	Complejidad
Crear el grafo con las estaciones de carga y los clientes	$O(n^2 + l + (n * x))$
Buscar los caminos	$O(n^2 + (n * x))$
Escoger el mejor camino	$O(m)$
Complejidad Total	$O(n^2 + l + (n * x))$

Tabla 2: n es igual a la cantidad de clientes, l es la cantidad de líneas del archivo txt, x es la cantidad de estaciones de carga y m es la cantidad de caminos generados.

4.7 Criterios de diseño del algoritmo

Se escogió este algoritmo por ser un algoritmo voraz con una buena velocidad de ejecución y un gasto moderado de memoria, además de que gracias a que da un conjunto de varias soluciones se puede optimizar con búsqueda local.

4.8 Tiempos de Ejecución

	Conjunto de Datos 1	***Conjunto de Datos 2***	***...Conjunto de Datos n***
Mejor caso	0.234 sg	0.216 sg	0.231 sg
Caso promedio	0.337 sg	0.360 sg	0.96 sg
Peor caso	0.547 sg	0.527 sg	0.531 sg

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

6. CONCLUSIONES

La necesidad de buscar una ruta varias veces para encontrar un camino más eficiente nos permite optimizar en gran medida el resultado que entregaremos por medio de nuestro programa.

Nos llevamos muy buenas sorpresas cuando comenzamos a realizar varias búsquedas para optimizar nuestro resultado, el algoritmo nos estaba arrojando un buen resultado, pero una vez llevamos a cabo la optimización por medio de las búsquedas, notamos que el resultado mejoró significativamente.

Creemos en el potencial de este algoritmo y por ello planeamos seguir en nuestra búsqueda por una mayor optimización incorporándole algoritmos meta-heurísticos.

6.1 Trabajos futuros

El problema de enrutamiento de vehículos es un problema cuya solución ayudara enormemente a una gran cantidad de empresas como al medio ambiente, nos gustaría implementar algoritmos metaheurísticos como algoritmos genéticos o enjambres para mejorar las soluciones dadas por nuestro algoritmo.

AGRADECIMIENTOS

Agradecemos a la institución ICETEX, la cuál nos abrió esta gran puerta para desarrollar proyectos como este y en el futuro mucho más colaborativos con la sociedad.

We thank for assistance with [particular technique, methodology] to [Name Surname, position, institution name] for comments that greatly improved the manuscript.

REFERENCIAS

1. Algoritmo de la colonia de hormigas – 2006 - https://es.wikipedia.org/wiki/Algoritmo_de_la_colonia _de_ hormigas#/media/File:Aco_branches.svg- Accessed 2018- 03-04
2. Algoritmo genetico – 2007 - https://es.wikipedia.org/wiki/Algoritmo_genético#/media/Fi le:Evolutionary_algorithm.svg – Accessed 2018-03-04
3. Solving the capacitated vehicle routing problem using a twophase metaheuristic procedure – 2009- http://www.scielo.org.co/scielo.php?script=sci_arttext &pid=S1794-12372009000200003 – Accessed 2018-03-04 Swarm intelligence – 2013- https://es.slideshare.net/vickmascara/inteligencia- de-enjambre - Accessed 2018-03-04
4. J. H. Holland University of Michigan Press, Ann Arbor. 1975. Adaptation in Natural and Artificial Systems.
5. A. Colorni, M. Dorigo et V. Maniezzo, Distributed Optimization by Ant Colonies, actes de la première conférence européenne sur la vie artificielle, Pari,s 134-142, 1991.
6. optimización por enjambre de particulas- 2010- https://es.wikipedia.org/wiki/Optimización_por_enjam bre_ de_partículas- Acessed 2018-03-04