OPTIMIZACIÓN DE ENTREGAS CON VEHÍCULOS ELÉCTRICOS

Jacobo Rave Londoño
Universidad Eafit
Colombia
jravel@eafit.edu.co

Diego Alejandro Vanegas González
Universidad Eafit
Colombia
davanegasg@eafit.edu.co

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

RESUMEN

El uso de vehículos eléctricos para entregar paquetes a los clientes en las rutas óptimas es muy importante ya que entregas rápidas y eficientes brindan grandes beneficios a sus propietarios y distribuidores. Algunos ejemplos son: un mejor y más rápido servicio a sus clientes, disminución de consumo de energía y aumento de la productividad.

1. INTRODUCCIÓN

Los vehículos eléctricos son el futuro de la movilidad y el envío de mensajería por su eficiencia en cuanto a consumo respecto a viajes y, lo más importante, por su baja huella ecológica, porque no dañan la naturaleza. Sus desventajas son pocas, como un rango de conducción limitado y el tiempo de carga del batería relativamente largo. Por esta razón, es importante encontrar la mejor ruta posible para su desplazamiento. En este caso, se diseña un algoritmo para encontrar las rutas optimas para la movilidad eléctrica con el fin de mejorar el envío de mercadería y servicio de paquetería.

2. PROBLEMA

La mala optimización de las rutas para las entregas de productos con vehículos eléctricos genera mucho consumo energético y el tiempo de recarga en el envío es ineficiente. Estos factores son necesarios tenerlos en cuenta para entregar todos los paquetes del día con menos retrasos y mejor desempeño.

3. RELATED WORK

3.1 Optimización Combinatoria para el mantenimiento de vehiculos electricos (EnSPP)

El EnSPP consiste en encontrar rutas de origen-destino óptimas para EV maximizando la carga de la batería de los vehículos en el nodo de destino.

3.2 Vehicle Routing Problem (VRP)

Es un problema de optimización combinatoria y de programación entera que pregunta "¿Cuál es el conjunto óptimo de rutas que debe atravesar una flota de vehículos para entregar a un conjunto de clientes determinado?"
por lo que el tamaño de los problemas que se pueden resolver de manera óptima mediante programación matemática u optimización combinatoria puede ser limitado. Por lo tanto, los solucionadores comerciales tienden a usar heurísticas debido al tamaño y la frecuencia de los VRP del mundo real que necesitan resolver.

3.3 ElectricVehicle Routing Problem (e-VRP)

Problema de enrutamiento de vehículos eléctricos (EVRP) que encuentra la estrategia de enrutamiento óptima con un costo de tiempo de viaje y un costo de energía mínimos, así como el número de vehículos eléctricos despachados. Eso se puede resolver utilizando la formulación de programación lineal de enteros mixtos MILP (que se puede adaptar para modelar diferentes supuestos de carga) y un conjunto de instancias pequeñas con información real sobre

3.4 Traveling Salesman Problem (TSP)

El problema del vendedor ambulante (TSP) es el desafío de encontrar la ruta más corta pero más eficiente para que una persona la tome dada una lista de destinos específicos. Es un problema algorítmico bien conocido en los campos de la informática y la investigación de operaciones. Algunas de sus soluciones son el método de fuerza bruta, el método de bifurcación y delimitación, el método del vecino más cercano.

4. Optimización de ruteo de Vehículos Eléctricos v1.0

4.1 Estructura de datos

Para resolver este problema, utilizamos varias estructuras de datos. Tales como, Listas de listas, Matrices de adyacencia, Listas de Arreglos.

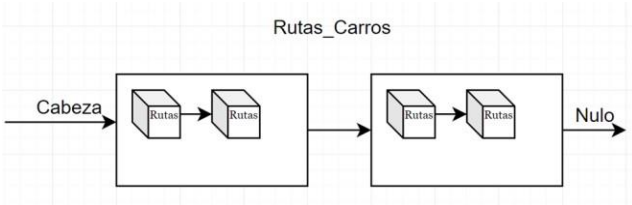


Gráfico 1. Listas de Listas, para almacenar las rutas de los diferentes carros

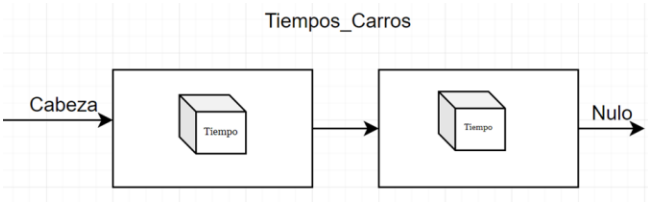


Gráfico 2. Listas de Listas, para almacenar los tiempos que se demora cada carro en su respectiva ruta.

	Deposito	c0	c1
Deposito	null	{D, T}	{D, T}
c0	{D, T}	null	{D, T}
c1	{D, T}	{D, T}	null

Gráfico 3. Matriz de Adyacencia, en las que almacenamos Pares de datos Distancia - Tiempo.

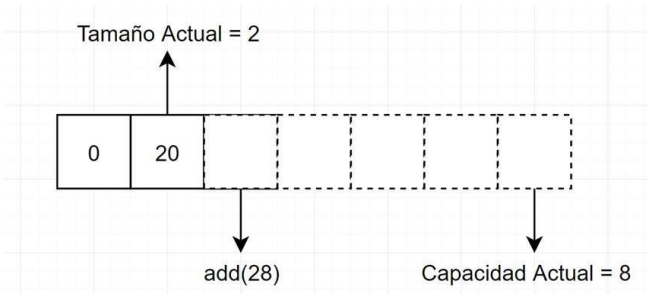


Gráfico 4. Listas de Arreglos, en los que almacenamos la ruta general.

4.2 Operaciones de la estructura de datos

Diseñen las operaciones de la estructura de datos para solucionar el problema eficientemente. Incluyan una imagen explicando cada operación

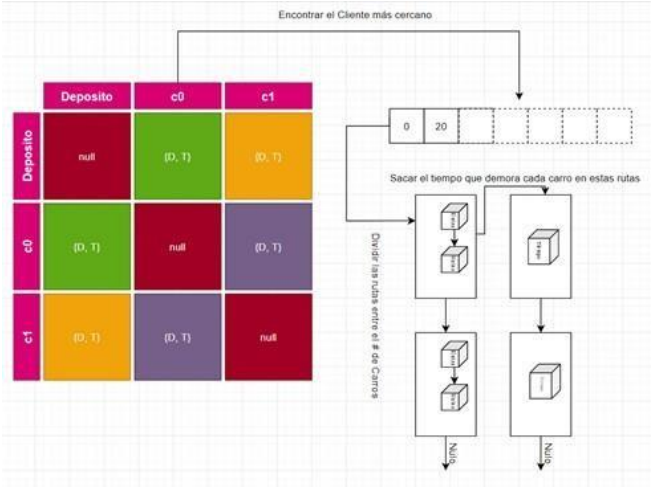
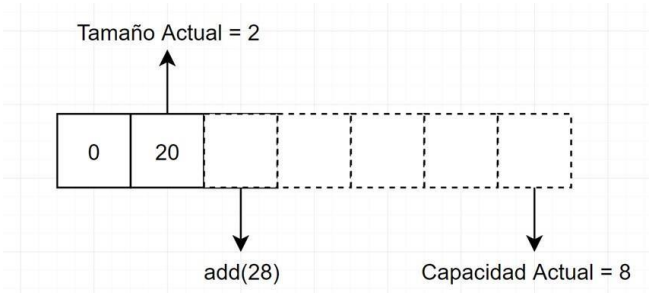
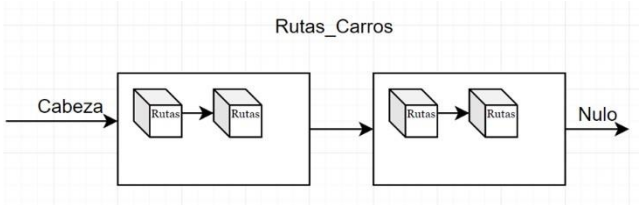


Gráfico 5. Imagen resumen sobre las operaciones del Algoritmo.

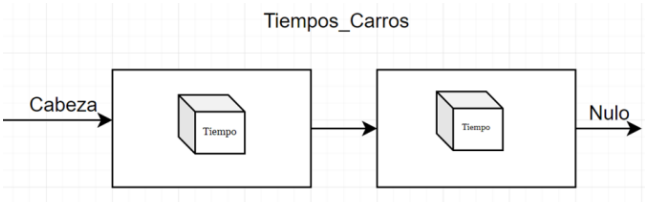
Creación de ruta general, creamos una ruta grande con todos los nodos, teniendo en cuenta los mejores tiempos para así hallar el cliente más cercano y dirigirse a su posición. Después, organizamos todos los clientes por orden de tiempo y de esta forma creamos una ruta gigante con todos los clientes con menor tiempo de viaje entre cada uno y añadimos esto a un ArrayList el cual almacena la ruta gigante.



Después, dividimos esta ruta en varias partes dependiendo del número de vehículos que vayamos a usar y les asignamos una partición de estas rutas para visitar siempre y cuando se cumpla con la restricción del tiempo.



Luego, separamos los tiempos de cada ruta individualmente por sus Vehículos. Para así saber cuánto se va a demorar en completar su ruta



4.3 Criterios de diseño de la estructura de datos

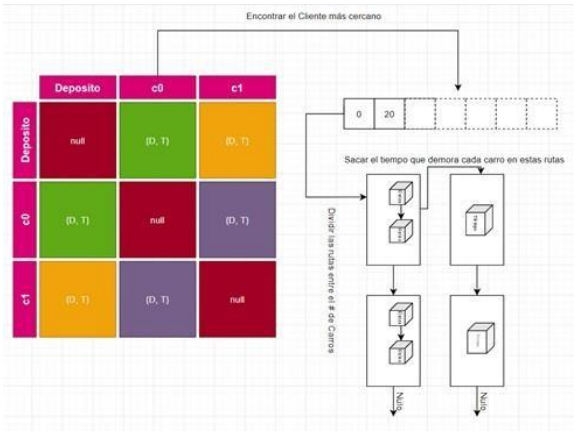
Seleccionamos estas estructura de datos por su fácil acceso y además de su increíble eficiencia en tiempo y memoria.

4.4 Análisis de Complejidad

Método	Complejidad
Leer	O(N)
generarGrafoDT	O(N*M)
calcularRuta	O(N)
calcularTiempo	O(N^2)
generarTiempoTotal	O(N)
generarCarros	O(1)
bestClient	O(N)
asignarRutas	O(N)

Tabla 1: Tabla para reportar la complejidad

4.5 Algoritmo



Gráfica 3: Paso a paso cómo se generan las rutas para cada carro

4.6 Cálculo de la complejidad del algoritmo

Sub problema	Complejidad
Crear el grafo con las distancias y lo tiempos	O(N*M)
Encontrar el mejor cliente	O(N)
Dividir la ruta entre carros	O(C)
Generar los vehículos para las rutas	O(1)
Calcular el tiempo que se demora cada vehículo en su ruta	O(A^2)
Complejidad Total	O(N^2)

Tabla 2: complejidad de cada uno de los sub problemas que componen el algoritmo. Sea N y M nodos de coordenadas, C es el número de Carros y A es el número de clientes a visitar.

4.7 Criterios de diseño del algoritmo

Este algoritmo tiene una gran eficiencia en memoria y tiempo. Ya que nos permite hacer la creación y asignación de rutas en O(N) , además de calcular los tiempos para las rutas y filtrarlos en O(A^2). Lo cual nos brinda una muy buena optimización en memoria y tiempo para resolver este problema

4.8 Tiempos de Ejecución

Calculen, (I) el tiempo de ejecución y (II) la memoria usada del algoritmo, para el Conjunto de Datos que está en el ZIP:

Tomen 100 veces el tiempo de ejecución y memoria de ejecución, para cada conjunto de datos

	Conjunto de Datos 1	***Conjunto de Datos 2***	***...Conjunto de Datos n***
Mejor caso	9.9 sg	10 sg	5 sg
Caso promedio	9.8 sg	9.7sg	6 sg
Peor caso	11 sg	11 sg	8 sg

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

4.9 Memoria

	Conjunto de Datos 1	*Conjunto de Datos 2*	*...Conjunto de Datos n*
Consumo de memoria	257MB	257 MB	257 MB

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

4.10 Análisis de los resultados

Valores durante la ejecución del código

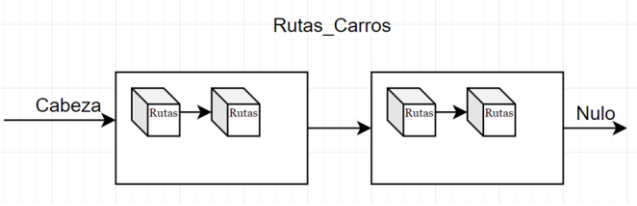
Tabla 5: Análisis de los resultados obtenidos con la implementación del algoritmo

	Matriz	LinkedList	ArrayList
Espacio en el Heap	0,9MB	0,16MB	0,04MB
Tiempo creación ruta	0	0	30MS
Tiempo creación número Carros	0	29MS	0
Tiempo creacion rutas-Carros	0	33MS	0
Impresión Matriz	7535MS	0	0

5. RUTEO CONTROLADO CON TSP

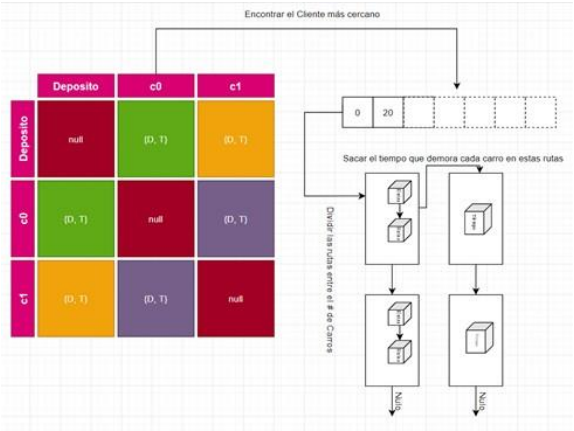
A continuación, explicamos la estructura de datos y el algoritmo.

5.1 Estructura de datos



Gráfica 4: Lista de listas con las rutas de cada carro

5.2 Operaciones de la estructura de datos



Gráfica 5: Imagen de una operación de generación de rutas de carros

5.3 Criterios de diseño de la estructura de datos

Teniendo en cuenta que el problema requiere una solución óptima y eficiente se optó por utilizar estructuras de datos que consumieran poca memoria, también al hacer operaciones y accesos la complejidad en tiempo no fuera demasiada. Como el grafo era completo se decidió hacerle subdivisiones para tener subgrafos más sencillos, esto mejora la manipulación y la optimización. Se utilizo listas de adyacencia ya que son muy eficientes en grafos pequeños, es de fácil acceso. Este acceso seria O(n) n siendo el número de nodos pero como se toman siempre 4 nodos el acceso seria en O(1), muy optimo y rápido.

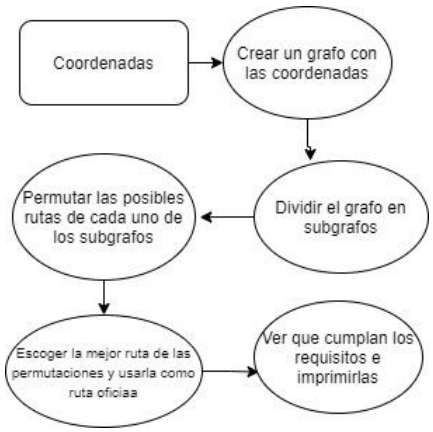
5.4 Análisis de Complejidad

Método	Complejidad
Leer	O(N)
generarGrafoDT	O(N*M)
calcularRuta	O(N)
calcularTiempo	O(N^2)
generarTiempoTotal	O(N)
generarCarros	O(1)
bestClient	O(N)
asignarRutas	O(N)

Método	Complejidad
asignarCargaBateria	O(n)
permutaciones	O(m)
dfs	O(n!)
permute	O(n!)
bestPermu	O(n)

Tabla 6: Tabla para reportar la complejidad

5.5 Algoritmo



Gráfica 6: Paso a paso cómo se genera el ruteo individual de cada carro

5.6 Cálculo de la complejidad del algoritmo

Calculen la complejidad del algoritmo para el peor de los casos, el mejor de los casos y el caso promedio

Sub problema	Complejidad
Crear el grafo con distancias y los tiempos	O(N*M)
Encontrar el mejor cliente	O(N)
Dividir las rutas entre carros	O(C)
Generar los vehículos para las rutas	O(1)
Calcular el tiempo que se demora cada vehículo en su ruta	O(A^2)
Permutaciones	O(P)
Complejidad Total	$O(N^2)$

Tabla 7: complejidad de cada uno de los sub problemas que componen el algoritmo. Sea N y M nodos de coordenadas, C es el número de Carros, A es el número de clientes a visitar y P la cantidad de rutas que se necesitan permutar

5.7 Criterios de diseño del algoritmo

Los criterios que tuvimos para tener en cuenta el diseño del algoritmo fue dividir el problema en pequeños problemas para así hallar una solución pronta, entonces dividimos el grafo en subgrafos y estos los permutamos hasta encontrar la mejor ruta posible de cada subgrafo. Hacerle la permutación a cada subruta seria O(n!). n siendo la cantidad de clientes. En este caso, solo serían 4 ya que cada ruta tiene 4 clientes. Así que podría ser tomado como o(m).

.

5.8 Tiempos de Ejecución

	Conjunto de Datos 1	*Conjunto de Datos 2*	*...Conjunto de Datos n*
Mejor caso	10 ms	11 sg	10sg
Caso promedio	12 sg	12 sg	12 sg
Peor caso	16 sg	16 sg	15 sg

Tabla 8: Tiempos de ejecución del algoritmo con diferentes conjuntos de datos

5.9 Memoria

Mencionar la memoria que consume el programa para varios ejemplos

	Conjunto de Datos 1	*Conjunto de Datos 2*	*...Conjunto de Datos n*
Consumo de memoria	90 MB	98 MB	95 MB

Tabla 9: Consumo de memoria del algoritmo con diferentes conjuntos de datos

5.10 Análisis de los resultados

Como podemos observar en la primera tabla (Es de la primera entrega) el tiempo de creación para rutas-Carros es de 33 MS y ahora que asignamos baterías y hacemos una permutación para encontrar las mejores rutas, es de tan solo 10 MS el tiempo de creación de rutas-Carros

	Matriz	LinkedList	ArrayList
Espacio en el Heap	0,9MB	0,16MB	0,04MB
Tiempo creación ruta	0	0	30MS
Tiempo creación número Carros	0	29MS	0
Tiempo creacion rutas-Carros	0	33MS	0
Impresión Matriz	7535MS	0	0

Permutaciones	0	0	5 MS
AsignarBateria	0	10 MS	0

Tabla 10: Análisis de los resultados obtenidos con la implementación del algoritmo

6. CONCLUSIONES

El TSP tiene muchas complicaciones y es un problema que toca mirar muy detenidamente para lograr encontrar una buena solución, aunque pensemos que nuestra solución es la major, pueden haber muchísimas más que compiten con nosotros en cuanto eficiencia y productividad, esto lo hemos ido observando a medida que vimos los cambios en los resultados con la entrega 1 y la 3

6.1 Trabajos futuros

Respondan ¿Qué les gustaría mejorar en el futuro? ¿Qué les gustaría mejor al algoritmo, estructura de datos, implementación?

AGRADECIMIENTOS

Agradecimientos a

Juan Sebastian Guerra y Juan David Echeverri por ayudarnos a conceptualizar y a aclarar muchas dudas

Profesor Mauricio Toro, por la aclaración de conceptos, guía en el desarrollo y brindarnos un gran espacio de enseñanza y aprendizaje

REFERENCIAS

1. Nora Touati Moungla and Vincent Jost. 2011.(January 2011). Retrieved February 21, 2021 from http://www.icrepq.com/icrepq'11/504-touati.pdf

2 Suzanne Ma. 2020. Understanding the Travelling Salesman Problem (TSP). (January 2020). from https://blog.routific.com/travelling-salesman-problem#:~:text=To%20solve%20the%20TSP%20using,this%20is%20the%20optimal%20solution.&text=This%20method%20breaks%20a%20problem,solved%20into%20several%20sub-problems.

3 Alejandro Montoya. 2017. Electric Vehicle Routing Problems : models and solution approaches. (January2017).from https://tel.archives-ouvertes.fr/tel-0144 1718/document

4 Anon. 2021. Vehicle routing problem. (January 2021). Retrieved February 22, 2021 from https://en.wikipedia.org/wiki/Vehicle_routing_problem