TÍTULO (DESCRIPCIÓN CORTA DEL PROYECTO. ENTRE 8 Y 12 PALABRAS)

Juan Felipe Ortiz Salgado
Universidad EAFIT
Colombia
jfortizs@eafit.edu.co

Samuel Meneses Diaz
Universidad EAFIT
Colombia
smenesesd@eafit.edu.co

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

RESUMEN

Considerando nuevas estrategias para encontrar la ruta más corta para una flota de vehículos que necesita entregar un inventario a un destino específico, traería beneficios en la movilización de vehículos. Sin embargo, es difícil lograr ese objetivo porque hay diferentes variantes de este problema. Cada uno de ellos tiene una restricción específica. Como capacidad de carga limitada, horas restringidas, número insuficiente de coches y algunas otras restricciones, que hacen que muchos algoritmos se hagan para satisfacer esas necesidades. En este caso, el retroceso puede ser una buena solución al problema de buscar la mejor ruta y eliminar lo menos factible.

Palabras clave
Modelado, Algoritmos, Clark-Wright, Método de Ahorros

1. INTRODUCCIÓN

Los vehículos eléctricos han sido una nueva alternativa para reducir la contaminación atmosférica de los coches de gasolina o diésel. La electricidad desempeña un papel importante y representa cero emisiones que mejoran al medio ambiente. Sin embargo, el uso de vehículos eléctricos para la carga y para el transporte de pasajeros tiene una limitación: el alcance de conducción es limitado y el tiempo de carga de la batería es relativamente largo.

2. PROBLEMA

El problema por resolver consiste en diseñar un algoritmo para encontrar rutas óptimas para un grupo de vehículos eléctricos para entregar mercancía a un conjunto de clientes. De esta manera, el tiempo que se tarda en visitar a cada cliente, recargar la batería y terminar toda la ruta se minimizará.

3. TRABAJOS RELACIONADOS

3.1 The Travelling Salesman (TSP)

El vendedor viajero consiste en encontrar la ruta más corta para que una persona complete una tarea dado un grupo específico de destinos. La dificultad de este problema se produce al trabajar con muchos lugares para visitar porque el algoritmo debe estar a cargo para verificar la menor distancia o costo para cada ruta. El famoso problema tiene diferentes soluciones. Uno de ellos es usando la Fuerza Bruta. Este método encuentra la mejor ruta comparando todas las permutaciones posibles de rutas para elegir la solución única más corta. En otras palabras, calcula el tiempo que se tarda en visitar cada distancia y, finalmente, elige el tiempo más corto.

3.2 Random walk algorithm

El problema en este algoritmo es encontrar, después de un tiempo fijo, la función de distribución de probabilidad de la distancia del punto desde el origen. Así que la solución es el algoritmo de caminata aleatoria, que es un proceso para determinar la ubicación probable de un punto sujeto a movimientos aleatorios, dadas las probabilidades de mover alguna distancia en alguna dirección.

3.3 Profitable Vehicle Routing Problem with Multiple Trips

El problema de ruteo de vehículos (VRP) relaciona dos variantes que son el VRP rentable y el VRP con viajes múltiples. En relación con la rentabilidad, consiste en las limitaciones para atender a un grupo de clientes debido a la escasez presupuestaria o por la insuficiencia de la oferta. Por otro lado, en relación con los múltiples viajes, significa que una flota de vehículos limitada tiene que realizar varias rutas con un horario estricto. El problema se solucionó mediante el uso de dos algoritmos basados en tres matrices: algoritmo de escalada en colina y algoritmo de descenso de vecindad variable.

3.4 Problema de enrutamiento de vehículos capacitados (CVRP)

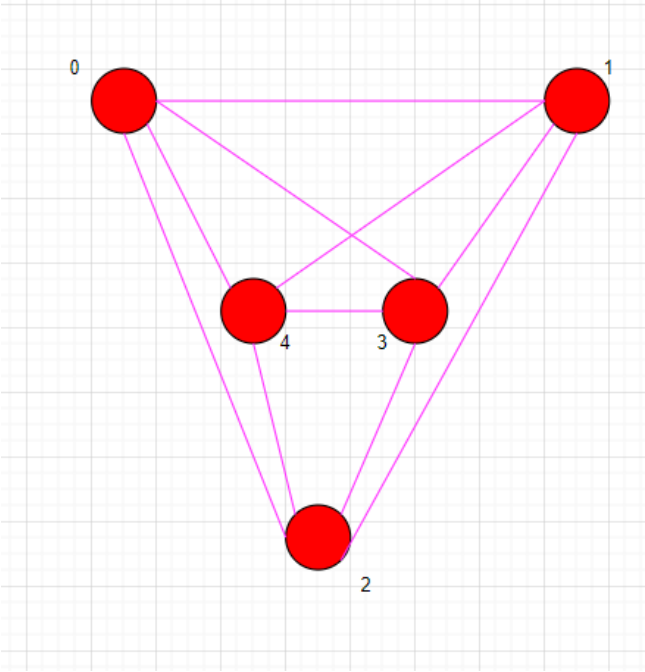
Consiste en encontrar una ruta que no requiera un consumo excesivo de tiempo y recursos. La dificultad se da porque hay una capacidad de carga limitada y debe volver al almacén principal. Este problema tiene diferentes soluciones, pero una de las más famosas son las técnicas de metaheurística, que se basa en el uso de los parámetros proporcionados por el usuario con el fin de encontrar un resultado eficiente. En este caso, se debe realizar una búsqueda en profundidad, encontrando las rutas que no requieren demasiado tiempo, es decir, el retroceso debe ser implementado.

4. TÍTULO DE LA PRIMERA SOLUCIÖN DISEÑADA

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

4.1 Estructura de datos

Esta estructura es una matriz de adyacencia en todo este grafico es un mapa de X ciudad y sus nodos distribuidos en ella, y las líneas representan la distancia entre cada nodo. Es una estructura sencilla pero se le pueden implementar estaciones de carga, parqueadero de los carros y más cosas.



Gráfica 1: lo que se puede ver la distancia entre dos nodos específicos

Grafica1	0	1	2	3	4
0	0	45,7	12,42	41,53	1,23
1	22,42	0	17.1	72	92
2	13,2	11,32	0	13,4	45,2
3	89,3	73,24	22,6	0	23,42
4	63,4	90,39	45,3	93,2	0

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

- **addArc:** Es el método para agregar un nuevo arc, es decir una nueva conexión entre dos vértices. addArc recibe el valor del nodo de origen, el valor del nodo de destino y el peso de estos dos.

Grafica1	0	1	2
0	0	45,7	12,42
1	22,42	10,5	17.1
2	13,2	11,32	0

addArc(1, 1, 10.5)

1: nodoOrigen

1: nodoDestino

10.5: Peso

- **getWeight:** Este es el método para obtener el peso o la distancia entre el nodoOrigen y el nodoDestino

Grafica1	0	1	2
0	0	45,7	12,42
1	22,42	10,5	17.1
2	13,2	11,32	0

getWeight(1,1) = 10.5

- **getSuccessors:** Este método entrega la lista de los sucesores de un vertice. Y el método recibe el numero de un nodo.

Grafica1	0	1	2
0	0	45,7	12,42
1	22,42	10,5	17.1
2	13,2	11,32	0

getSuccessors(1) = 0,1,2

Gráfica 2: Imagen de una operación de borrado de una lista encadenada

4.3 Criterios de diseño de la estructura de datos

Para este proyecto se escogió la matriz de adyacencia, ya que pensamos en lo eficaz que puede ser ya que el tiempo de ejecución de O(1).

También esta matriz ayudo a representar un grafo finito no dirigido, el cual fue mostrado en el punto 4.1 para mostrar los vértices y los puntos de la ciudad. Los elementos de la matriz nos dan a entender la ciudad, los puntos si están separados o no, a que distancia están. Es decir que nos basamos en lo eficaz que puede ser la ejecución de este.

4.4 Análisis de Complejidad

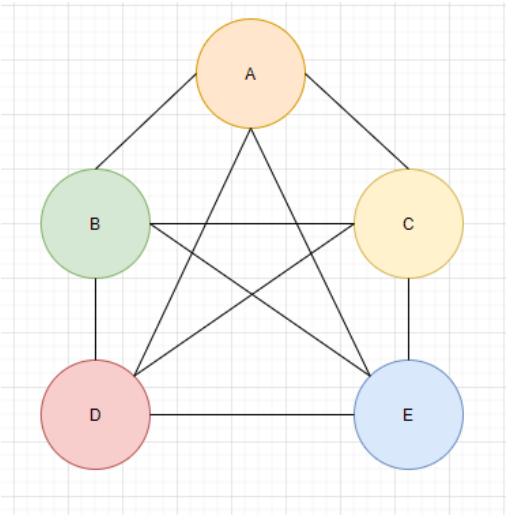
Calculen la complejidad de las operaciones de la estructura de datos para el peor de los casos. Vean un ejemplo para reportarla:

	Complejidad tiempo	Complejidad memoria
Creación tabla de ahorros	$O(N^2)$	$O(N^2)$
Metodo creador de ruta	$O(N^3)$	$O(N^3)$

Nota: La letra n filas por m columnas

Tabla 1: Tabla para reportar la complejidad

4.5 Algoritmo



Ahorro	0-1-0	0-2-0
	91.4	24.84

4.6 Cálculo de la complejidad del algoritmo

Sub problema	Complejidad
Crear la matriz de ahorros	$O(N^2)$
Añadir un camino a la lista de registro	O(1)
Método para validar si se puede realizar el camino	O(N)
Método para eliminar los ahorros de los clientes visitados	$O(N^2)$
Método calculador de ruta	$O(N^3)$
Complejidad Total	$O(N^3)$

Tabla 2: complejidad de cada uno de los subproblemas que componen el algoritmo. N número de clientes a visitar y M numero de estaciones de carga.

4.7 Criterios de diseño del algoritmo

El algoritmo que se implementó se basó en el algoritmo Clarke-Wright. Se conoce el algoritmo de ahorro porque funciona calculando el guardar entre dos rutas para construir caminos diferentes, con el cual se analiza la posibilidad de combinar dos trazados en una sola ruta, y este tiene altas posibilidades de ser la óptima. En este caso, se preocupa más por encontrar una ruta eficiente bajo el tiempo y las restricciones de la batería por lo que garantiza la mejor solución, es eficiente para resolver el problema sin exceder el tiempo máximo de respuesta. Además, durante las diferentes operaciones se utilizaron matrices que permitían buscar en O (1).

Teniendo en cuenta que el objetivo principal del algoritmo consistía en priorizar el tiempo en que este lograba generar el ruteo de vehículos, se decidió priorizar el tiempo.

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 3***
Mejor caso	1 sg	0.8 sg	1.3 sg
Caso promedio	3.5 sg	3.4 sg	3 sg
Peor caso	6 sg	5.5 sg	6.3 sg

Tabla 3: Tiempos de ejecución del algoritmo con diferentes conjuntos de datos, Datos 1 con 320 clientes, datos 2 con 345 clientes, datos 3 con 359 clientes.

4.9 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	0.8 MB	0.8 MB	0.8 MB

Tabla 4: Consumo de memoria del algoritmo con diferentes conjuntos de datos, Datos 1 con 320 clientes, datos 2 con 345 clientes, datos 3 con 359 clientes.

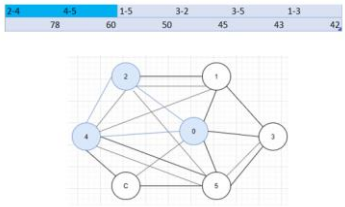
4.10 Análisis de los resultados

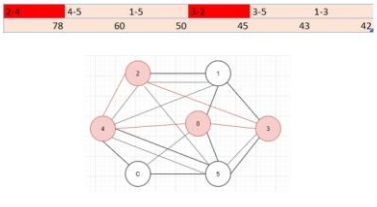
Expliquen los resultados obtenidos. Hagan una gráfica con los datos obtenidos, como, por ejemplo:

	Memoria(KB)	**Tiempo de ejecución**	**Número de clientes**	**Número de vehículos**	**Tiempo total**
(1)	755	14 s	320	33	159.52
(2)	826	9 s	320	27	95.97
(3)	825	11 s	320	29	119.29
(4)	826	12 s	320	33	159.52
(5)	825	16 s	320	27	95.97
(6)	825	14 s	320	29	119.29
(7)	825	13 s	320	33	159.52
(8)	825	10 s	320	27	95.97
(9)	825	11 s	320	29	119.29
(10)	826	14 s	320	33	159.52
(11)	825	16 s	320	27	95.97
(12)	826	14 s	320	29	119.29

Tabla 5: Análisis de los resultados obtenidos de la ejecución del algoritmo.

5. ALGORITMO DE CLARK AND WRIGHT





5.6 Cálculo de la complejidad del algoritmo

Sub problema	**Complejidad**
Crear la matriz de ahorros	$O(N^2)$
Método para validar si se puede realizar el camino	$O(N)$
Método para eliminar los ahorros de los clientes visitados	$O(N^2)$
Método calculador de ruta	$O(N^3)$
Complejidad Total	O(N^3)

5.7 Criterios de diseño del algoritmo

El algoritmo usado para generar el ruteo de vehículos fue el de Clark and Wright que esta modificado para considerar la aleatoriedad, genera las rutas para vehículos eléctricos, priorizando su capacidad de carga y maximizando el tiempo disponible.

En busca de alternativas para mejorar las rutas que genera dicho algoritmo voraz, se plantea la aleatoriedad dentro del mismo, con el fin de encontrar posibles soluciones con mejores resultados y de esta manera escoger la mejor. Dado que el objetivo final era encontrar soluciones en el menor tiempo posible, este algoritmo era el más optimo y aprovechando la brecha de tiempo con que se contaba, se plantea la aleatoriedad dentro del mismo, para así brindar la mejor solución posible.

5.8 Tiempos de Ejecución

	Conjunto de Datos 1	*Conjunto de Datos 2*	*Conjunto de Datos 3*
Mejor caso	25 ms	22 ms	29 ms
Caso promedio	114 ms	55 ms	476 ms
Peor caso	204 ms	88 ms	924 ms

Tabla 8: Tiempos de ejecución del algoritmo con diferentes conjuntos de datos (en milisegundos)

5.9 Memoria

	Conjunto de Datos 1	*Conjunto de Datos 2*	*Conjunto de Datos 3*
Consumo de memoria	0.9 MB	0.9 MB	1 MB

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

5.10 Análisis de los resultados

Memoria (KB)	**Tiempo de ejecución (ms)**	**No. Clientes**	**No. Camiones**	**Tiempo total de las rutas**
965	**301**	**320**	**51**	**360**
957	**168**	**320**	**48**	**290**
956	**125**	**320**	**52**	**342**
957	**120**	**320**	**51**	**368**
958	**194**	**320**	**48**	**299**
956	**105**	**320**	**52**	**346**
1035	**86**	**320**	**51**	**346**
1036	**90**	**320**	**47**	**292**
1034	**95**	**320**	**48**	**305**
1035	**101**	**320**	**50**	**339**
1035	**370**	**320**	**47**	**295**
1034	**148**	**320**	**49**	**315**

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

6. CONCLUSIONES

En relación con la eficiencia, el algoritmo demostró ser eficaz. El tiempo de ejecución fue moderadamente eficiente, pero se puede mejorar utilizando otra forma de establecer a qué estación de carga debe ir el camión después de visitar a un cliente. Por otro lado, el consumo de memoria de la solución fue mayor porque se implementaron tres tipos de estructuras de datos. Entonces, el consumo fue alto. El algoritmo es una buena forma de encontrar una solución de optimización. Sin embargo, se deben agregar algunas mejoras para evitar cualquier tipo de error en cada proceso que se realizó.

6.1 Trabajos futuros

Evolucionar el algoritmo por medio de métodos de Búsqueda Local es el principal objetivo a futuro, además del comparar con soluciones varias de back-tracking, esto último dado que el evaluar más rutas aumenta la probabilidad de encontrar una mejor solución. Otro aspecto clave a mejorar es la complejidad del algoritmo mismo, puesto que, aunque con data set pequeños, como lo trabajados, el tiempo algorítmico es bajo, para data sets más grandes el tiempo de ejecución del problema aumentaría a gran escala.

AGRADECIMIENTOS

REFERENCIAS

[1] *Travelling Salesman Problem | Set 1 (Naive and Dynamic Programming) - GeeksforGeeks*. *GeeksforGeeks.com*, 2018. Disponible en: https://www.geeksforgeeks.org/travelling-salesman-problem-set-1/

[2] Mediorreal, A., 2014. Modelo de ruteo de vehículos para la distribución de las empresas Laboratorios Veterland, Laboratorios Callbest y

Cosméticos Marlioü París. UNIVERSIDAD JAVERIANA.

[3] Chbichib1, A., Mellouli, R. y Habib, C. (2012, julio). Problema de generación de rutas rentables para vehículos con viajes múltiples: modelado y algoritmo de descenso de vecindario variable (N.o 2). Revista estadounidense de investigación operativa. https://doi.org/10.5923/j.ajor.20120206.04

[4] Molina, J. y Salmeron, J. (2020, septiembre). El problema de las rutas de vehículos heterogéneas con ventanas de tiempo y un número limitado de recursos (N.o 94). Elsevier. https://doi.org/10.1016/j.engappai.2020.103745

[5] Alinaghian, M., Kaviani, Z., Khaledan, S. 2015.A Novel Heuristic Algorithm Based on Clark and Wright Algorithm for Green Vehicle Routing Problem. International Journal of Supply and Operations Management, 2(2).784–797.