Ruteo de vehículos eléctricos en una empresa repartidora de mercancía

Juan Pablo Rincón Usma
Universidad Eafit
Colombia
jprinconu@eafit.edu.co

Julián Gómez Benítez
Universidad Eafit
Colombia
jgomezb11@eafit.edu.co

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

RESUMEN
En este trabajo se quiere apoyar el desarrollo de los vehiculos electricos siendo esta la tecnologia mas prometedora, actualmente, para reemplazar el uso del petroleo y así apoyar el medio ambiente. Se piensa desarrolar un algoritmo que encuntre la ruta más optima, teniendo en cuenta las desviaciones, para un vehiculo electrico en su labor de entrega de mercancias.

Para la implementacion de este problema decidimos usar un algoritmo tipo greedy que decide la ruta de toda la flota de carros de la manera mas rapida posible teniendo en cuenta las restricciones que tienen actualmente los carros electricos

Palabras clave
Vehiculos electricos – Grafos – Optimizacion – Rutas – Algoritmos.
Palabras clave de la clasificación de la ACM
Theory of computation → Design and analysis → Approximation algorithms analysis → Routing and network design problems.

Theory of computation → Design and analysis → Graph algorithms analysis → Shortest paths.

Applied computing → Operations research → Trasnportation.

1. INTRODUCCIÓN
Los vehículos eléctricos son una de las tecnologías más prometedoras para reducir la dependencia del petróleo y las emisiones de gas invernadero. Aunque sus beneficios con respecto al cuidado del medio ambiente son muchos, los vehículos eléctricos tienen un problema y es la duración de su batería y el tiempo de carga de las mismas. Por eso la situación que se está modelando en este informe trata de diseñar un algoritmo que encuentre las rutas más optimas que pueden tomar los vehículos eléctricos, teniendo en cuenta las desviaciones para cargar la batería, de una compañía de reparto de mercancía,

2. PROBLEMA
Tratamos de modelar un algoritmo que permita a los vehículos eléctricos encontrar el camino óptimo para entregar mercancías a un grupo de clientes definido, esto con el fin de apoyar y contribuir a un futuro en el que los vehículos eléctricos puedan reemplazar a los impulsados por petróleo para así contrarrestar el impacto ambiental.

3. TRABAJOS RELACIONADOS

3.1 An improved Particle Swarm Optimization Algorithm for the VRP with Simultaneous Pickup and Delivery and Time Windows.
En logística, el principal objetivo de los planificadores es reducir al máximo los costes operativos. Teniendo esto en cuenta, surgen otros aspectos, como utilizar menos tiempo, que son importantes para los clientes. En consecuencia, los planificadores suelen tener que encontrar un equilibrio entre todos estos aspectos y los costes operativos.

Para resolver este complejo problema de optimización combinatoria utilizamos la optimización por enjambre de partículas (PSO), un algoritmo de inspiración social. El objetivo de PSO es encontrar un conjunto de caminos que minimice la distancia total de los mismos y que atienda, simultáneamente, las demandas de entrega y recogida de los clientes. Además, en este trabajo también se tienen en cuenta las restricciones de las ventanas de tiempo, que hacen que el problema sea más difícil de resolver.

3.2 Solución del Problema de Conformación de Lotes en Almacenes utilizando Algoritmos Genéticos.

Las operaciones de almacenamiento tienen una gran influencia en los costos logísticos, tanto en costos de inversión como en costos operacionales directos. Los almacenes y centros de distribución deben realizar esfuerzos para ofrecer una preparación de pedidos eficiente recuperando órdenes de clientes de forma económica, y minimizando los costos relacionados con distancias recorridas. Los algoritmos genéticos son metaheurísticos de búsqueda global, que se basan en un algoritmo poblacional que en cada iteración evalúa varias soluciones de forma simultánea, lo cual genera mayor eficiencia a la hora de encontrar una solución cercana al óptimo. Para un problema como el de conformación de lotes, los algoritmos genéticos han demostrado ser un método de solución apropiado, debido a que este problema es altamente combinatorio y complejo por su naturaleza, que se basa en encontrar la mejor combinación de órdenes en lotes para satisfacer una función objetivo establecida.

3.3 Algoritmo para el cálculo de la velocidad media óptima en una ruta (ASGA).

El eco-drivinges una técnica de conducción que permite ahorrar combustible, optimizando el comportamiento del conductor en la conducción. Está técnica permite ahorrar combustible con independencia de la tecnología y consiste en la aplicación de un conjunto de reglas.
El algoritmo ASGA propuesto para obtener la velocidad media óptima se basa en los algoritmos genéticos. Este tipo de algoritmos consta de un conjunto de etapas bien definidas: representación, función de optimización, inicialización, selección, cruce, mutación y reemplazamiento. A continuación, describiremos cada una de las etapas. Representación. El problema de optimización se define como un problema de optimización combinatorio donde los individuos se representan como vectores. Cada posición del vector representa un tramo del viaje y contiene la velocidad media en dicho tramo y el tiempo que tarda en completarlo (duración) asumiendo que el vehículo circula a dicha velocidad de forma constante.

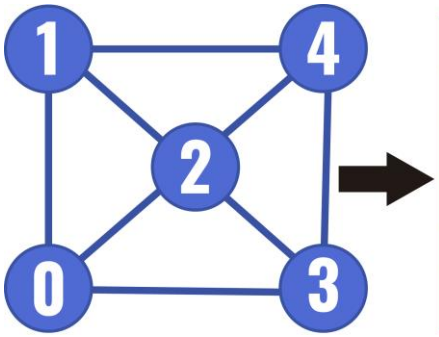
3.4 Desarrollo de un sistema capaz de optimizar rutas de entrega utilizando algoritmos genéticos.

Actualmente y gracias a la globalización los productos que se comercializan no tienen el mismo lugar de origen y de consumo, sino que viajan grandes distancias desde donde se producen o maquilan hasta llegar al consumidor final; esto genera grandes costos en la cadena productiva para el desplazamiento, por lo tanto, se han hecho esfuerzos para que estos desplazamientos sean los óptimos, con esto nos referimos a que el desplazamiento sea el mínimo para la entrega de productos.
La ruta optima se seleccionan las colonias donde se realizará la entrega sobre el mapa de la ciudad de León, Gto, sobre este mapa se coloca el mouse sobre la colonia a la que se va a llevar la mercancía, y se agrega dándole un nombre. Al seleccionar varias colonias está información se introducirá al algoritmo. El resultado que se obtiene del algoritmo genético es un grupo de cromosomas que representan al mejor individuo y gracias a que la función objetivo para este tipo de problemas se asigna para el cálculo de la ruta más corta, entonces el mejor individuo que resulta es la ruta óptima, lo que permite un ahorro considerable en la logística de la entrega de mercancías.

4. e-VRP v1

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

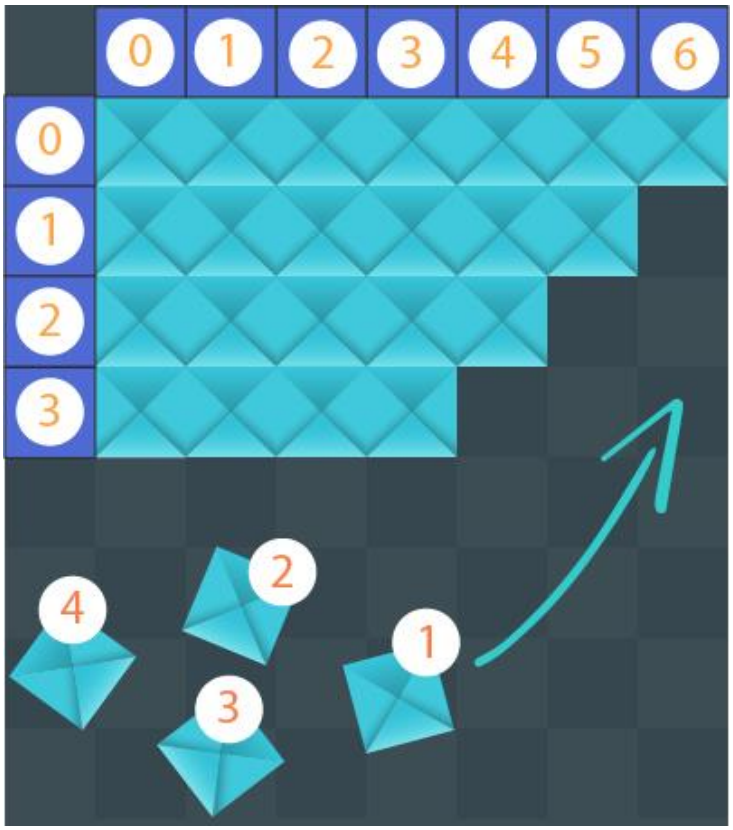
4.1 Estructura de datos



	0	1	2	3	4
0	0	1	1	1	1
1	1	0	1	1	1
2	1	1	0	1	1
3	1	1	1	0	1
4	1	1	1	1	0

4.2 Operaciones de la estructura de datos

Agregar:



Búsqueda:

4.3 Criterios de diseño de la estructura de datos

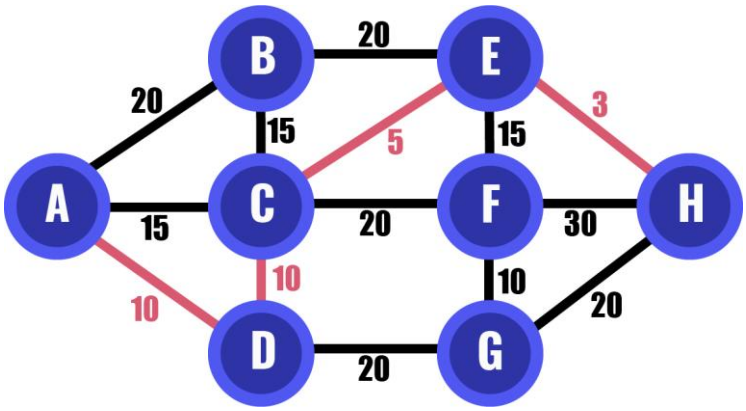
Elegimos representar el grafo con una matriz de adyacencia debido a que su complejidad en el acceso a los datos es de O(1). Con esta complejidad el algoritmo se hace mucho más eficiente a la hora de realizar las operaciones que requieren acceder a los pesos de las aristas de nuestro grafo. Además, teniendo en cuenta que los datasets van desde 320 hasta 360 nodos, la cantidad de memoria que se utiliza en esta matriz no es exorbitante comparada con la eficiencia en tiempo que ganamos.

4.4 Análisis de Complejidad

Método	Complejidad
Acceder a una posición de la matriz	O(1)
Imprimir la matriz	$O(n^2)$
Agregar en la matriz	O(n)
Modificar una posición de la matriz	O(n)

Tabla 1: Tabla para reportar la complejidad

4.5 Algoritmo



Gráfica 3: Grafica que explica cómo funciona la técnica del vecino más cercano.

4.6 Cálculo de la complejidad del algoritmo

Sub problema	Complejidad
Crear matriz de adyacencia	O(N/2)
Solucion Greedy	$O(M^2)$
Optimizacion Tabu Search	$O(I*N*V^2)$
Complejidad Total	$O (N/2 + M^2 + I*N*V^2)$

Tabla 2: complejidad de cada uno de los subproblemas que componen el algoritmo. Sea N la cantidad total de nodos, M la cantidad de clientes, V la cantidad de vehículos de la solucion e I la cantidad de iteraciones Tabú que se especifican manualmente en el programa.

4.7 Criterios de diseño del algoritmo

En nuestro programa decidimos usar dos algoritmos que se complementan. El primero es un algoritmo tipo Greedy (también conocido como el algoritmo del vecino más cercano) que se guía por una heurística que consiste en elegir la opción local optima con la esperanza de llegar a una solución general lo más optima posible, la razón por la que decidimos usar un Greedy para la solución del problema es más que todo por la rapidez con la que devuelve una solución, aunque la mayoría de las veces esas soluciones no son las más eficientes esto no es un problema porque estas soluciones nos sirven de base para el segundo algoritmo que implementamos en nuestro programa.

La optimización matemática Tabu Search, es un algoritmo metaheurístico que toma como punto de inicio una solución, en este caso toma la solución tipo Greedy que se realizó anteriormente, y genera nuevas soluciones a partir de esta.

Para generar las nuevas soluciones, el orden en el que dos nodos son visitados es intercambiado y se usa la distancia total de la ruta para juzgar cual de todas es la más optima. Esta búsqueda se repite un numero definido de veces. Decidimos usar este algoritmo debido a que se complementa muy bien con la solución Greedy que planteamos, ya que mientras el Greedy saca una posible solución lo más rápido posible el Tabu Search la optimiza para asegurase de que sea mucho más eficaz, generando así un algoritmo que es eficaz y a la vez preciso.

4.8 Tiempos de Ejecución

	Conjunto de Datos 1 (5 nodos)	*Conjunto de Datos 2(345 nodos)*	*Conjunto de Datos 3(359 nodos)*
Mejor caso	222 ms	1193 ms	1196 ms
Caso promedio	227.8 ms	1333 ms	1313 ms
Peor caso	239 ms	1369 ms	1482 ms

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

4.9 Memoria

	Conjunto de Datos 1(5 nodos)	*Conjunto de Datos 2(345 nodos)*	*...Conjunto de Datos 3 (359 nodos)*
Consumo de memoria	20 MB	28 MB	29 MB

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

4.10 Análisis de los resultados

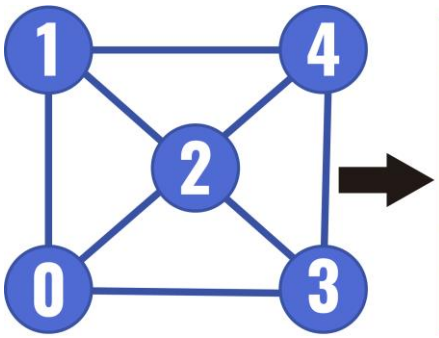
Memoria	Tiempo de ejecucion	Numero de Nodos	Numero de camiones
20MB	229 ms	2	1
28MB	1275ms	345	30
29MB	1320 ms	359	26

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

5. e-VRP v2

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

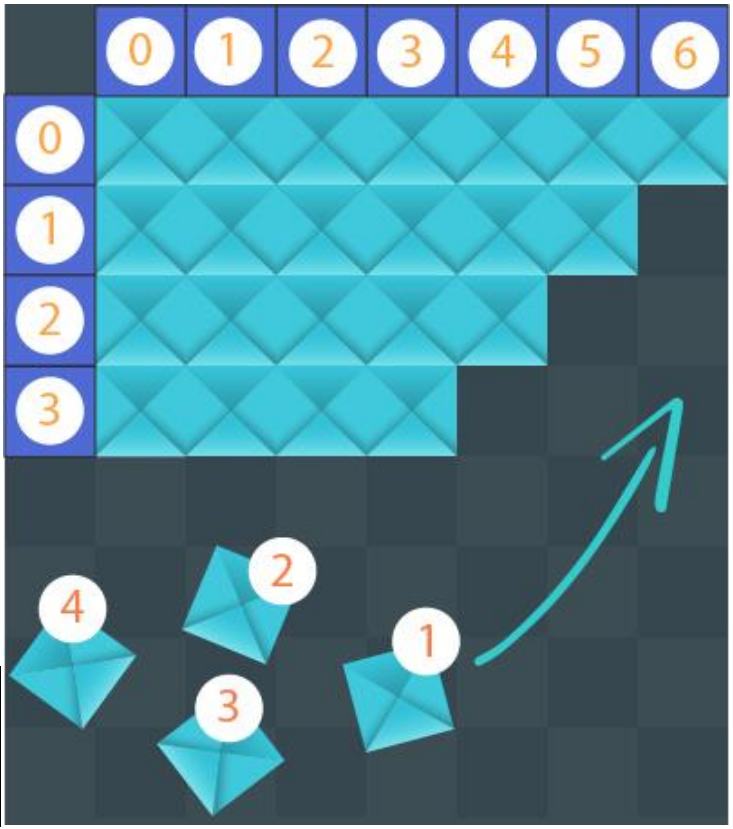
5.1 Estructura de datos



	0	1	2	3	4
0	0	1	1	1	1
1	1	0	1	1	1
2	1	1	0	1	1
3	1	1	1	0	1
4	1	1	1	1	0

5.2 Operaciones de la estructura de datos

Agregar:



Búsqueda:

5.3 Criterios de diseño de la estructura de datos

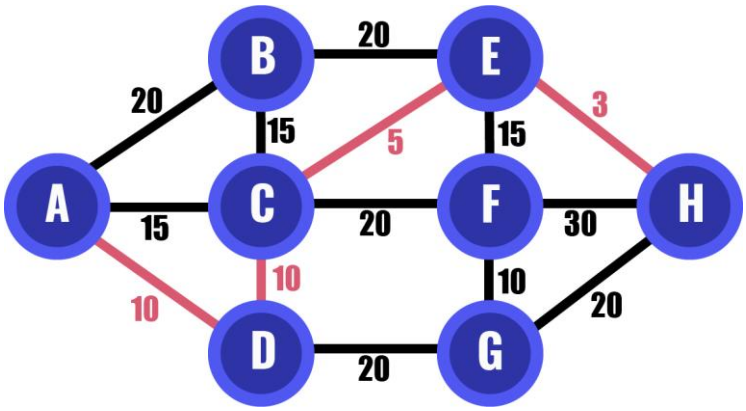
Elegimos representar el grafo con una matriz de adyacencia debido a que su complejidad en el acceso a los datos es de O(1). Con esta complejidad el algoritmo se hace mucho más eficiente a la hora de realizar las operaciones que requieren acceder a los pesos de las aristas de nuestro grafo. Además, teniendo en cuenta que los datasets van desde 320 hasta 360 nodos, la cantidad de memoria que se utiliza en esta matriz no es exorbitante comparada con la eficiencia en tiempo que ganamos.

5.4 Análisis de Complejidad

Tabla 6: Tabla para reportar la complejidad

Método	Complejidad
Acceder a una posición de la matriz	O(1)
Imprimir la matriz	$O(n^2)$
Agregar en la matriz	$O(n^2)$
Modificar una posición de la matriz	O(n)

5.5 Algoritmo



Gráfica 3: Grafica que explica cómo funciona la técnica del vecino más cercano.

5.6 Cálculo de la complejidad del algoritmo

Sub problema	Complejidad
Crear matriz de adyacencia	O(N/2)
Solucion Greedy	$O(M^2)$
Optimizacion Tabu Search	$O(I*N*V^2)$
Complejidad Total	$O (N/2 + M^2 + I*N*V^2)$

Tabla 2: complejidad de cada uno de los subproblemas que componen el algoritmo. Sea N la cantidad total de nodos, M la cantidad de clientes, V la cantidad de vehículos de la solucion e I la cantidad de iteraciones Tabú que se especifican manualmente en el programa.

5.7 Criterios de diseño del algoritmo

En nuestro programa decidimos usar dos algoritmos que se complementan. El primero es un algoritmo tipo Greedy (también conocido como el algoritmo del vecino más cercano) que se guía por una heurística que consiste en elegir la opción local optima con la esperanza de llegar a una solución general lo más optima posible, la razón por la que decidimos usar un Greedy para la solución del problema es más que todo por la rapidez con la que devuelve una solución, aunque la mayoría de las veces esas soluciones no son las más eficientes esto no es un problema porque estas soluciones nos sirven de base para el segundo algoritmo que implementamos en nuestro programa.

La optimización matemática Tabu Search, es un algoritmo metaheurístico que toma como punto de inicio una solución, en este caso toma la solución tipo Greedy que se realizó anteriormente, y genera nuevas soluciones a partir de esta. Para generar las nuevas soluciones, el orden en el que dos nodos son visitados es intercambiado y se usa la distancia

total de la ruta para juzgar cual de todas es la más optima. Esta búsqueda se repite un numero definido de veces. Decidimos usar este algoritmo debido a que se complementa muy bien con la solución Greedy que planteamos, ya que mientras el Greedy saca una posible solución lo más rápido posible el Tabu Search la optimiza para asegurase de que sea mucho más eficaz, generando así un algoritmo que es eficaz y a la vez preciso.

5.8 Tiempos de Ejecución

	Conjuntos de Datos (Todos)
Mejor caso	21329 ms
Caso promedio	24060 ms
Peor caso	26932 ms

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

5.9 Memoria

	Conjunto de Datos 1(5 nodos)	***Conjunto de Datos 2(345 nodos)***	***...Conjunto de Datos 3 (359 nodos)***
Consumo de memoria	20 MB	28 MB	29 MB

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

5.10 Análisis de los resultados

Memoria	**Tiempo de ejecucion**	**Numero de Nodos**	**Numero de camiones**
20MB	175ms	2	1
28MB	1772ms	345	34
29MB	1156 ms	359	35

6. CONCLUSIONES

Al realizar nuestro trabajo de ruteo de vehículos decidimos que lo más importante era el tiempo de ejecución del algoritmo, por eso optamos por una solución tipo greedy que nos ayuda a encontrar un posible resultado en el menor tiempo posible que satisfaga las condiciones propuestas.

Esto resulto siendo una buena aproximación a la solución óptima, pero en vista de que nos quedaba tiempo sin aprovechar decidimos implementar un algoritmo de optimización llamado Tabu Search, lo cual nos funcionó hasta el punto en el cual las rutas optimizadas incumplen las restricciones del problema mientras que las rutas generadas por el algoritmo greedy son una solución viable.

Entre la primera solución y la final la diferencia es la optimización en tiempo del algoritmo greedy ya que gracias a la invención de nuevos métodos y casos de parada el tiempo de ejecución se acorta de manera considerable. No se cambiaron las estructuras de datos usadas en la primera solución por lo cual el consumo de memoria sigue siendo el mismo. La parte de la implementación del tabu a la solución final no avanzo demasiado debido a que a la hora de considerar las restricciones del problema se nos complicó el hecho de validar que las rutas “optimizadas” cumplieran estrictamente las condiciones de las propuestas.

6.1 Trabajos futuros

Nos gustaría implementar de una manera más completa el algoritmo de optimización que se adapte de la mejor manera a nuestra solución tipo greedy para asi tener una solucion tanto rapida como optima y que a su vez cumpla con todos los requerimientos solicitados.

REFERENCIAS

Referenciar las fuentes usando el formato para referencias de la ACM. Léase en http://bit.ly/2pZnE5g Vean un ejemplo:

1.Adobe Acrobat Reader 7, Be sure that the references sections text is Ragged Right, Not Justified. http://www.adobe.com/products/acrobat/.

2. Fischer, G. and Nakakoji, K. Amplifying designers’ creativity with domainoriented design environments. in Dartnall, T. ed. Artificial Intelligence and Creativity: An Interdisciplinary Approach, Kluwer Academic Publishers, Dordrecht, 1994, 343-364.

3. C. Lagos, G. Guerrero, E. Cabrera, A. Moltedo, F. Johnson and F. Paredes, "An improved Particle Swarm Optimization Algorithm for the VRP with Simultaneous Pickup and Delivery and Time Windows," in IEEE Latin America Transactions, vol. 16, no. 6, pp. 1732-1740, June 2018. https://ieeexplore.ieee.org/abstract/document/8444393?casa_token=pX67a3S1bdkAAAAA:cL0cgUWNWwuJM5qt9txqk8XydOdYqXXm8wifA3CR3ZG20jbS6BZbz-etiU_J6KjKRrR3jkTUKBAg.

4. Jose A. Cano, Alexander A. Correa-Espinal and Rodrigo A. Gómez-Montoya. Solving the Order Batching Problem in Warehouses using Genetic Algorithms. vol.29 no.6 dic. 2018. https://scielo.conicyt.cl/scielo.php?pid=S0718-07642018000600235&script=sci_arttext.

5. V. Corcoba Magaña and M. Muñoz Organero, Algoritmo para el cálculo de la velocidad media óptima en una ruta(ASGA), Revista Iberoamericana de Automática e Informática industrial 11 (2014) 435–443.

https://polipapers.upv.es/index.php/RIAI/article/view/9436/9408

6. Rosario Baltazar, Judith Esquivel Vázquez, Andrea Rada and Claudia Díaz. Desarrollo de un sistema capaz de optimizar rutas de entrega utilizando algoritmos genéticos, Inteligencia artificial y TICs, 2010.