ALGORITMO PARA OPTIMIZACIÓN DE LAS RUTAS EN LA CIUDAD

Santiago Albisser
Universidad Eafit
Colombia
Salbisserc@eafit.edu.com

Juan Pablo Leal
Universidad Eafit
Colombia
Jplealj@eafit.edu.co

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

RESUMEN

Hoy en día presentamos una gran congestión en las vías de nuestra ciudad, debido a la gran cantidad de carros que salen hoy en día sabiendo que muchos de estos se podrían aprovechar para ir varias personas dentro del mismo vehículo y de esta manera presentarse menos congestión en las vías. Este problema es importante no solo por el tiempo que podemos perder atascados en un trancón en el día a día, sino también por el hecho de que la polución generada por los carros en un trancón es nociva para nuestra salud. La solución es crear un algoritmo que nos permita ubicar de distintos grupos, la manera en la que se pueda compartir un vehículo y así, generar menos contaminación y menos trancones.

1. INTRODUCCIÓN

Hoy en día presentamos altos índices de contaminación del aire en la ciudad debido al incremento masivo de carros. En este documento hablaremos sobre el problema del alto tráfico debido a la gran cantidad de personas que utilizan el carro ellos solos y encontrar la manera para evitar que el uso de carros sea reducido de tal manera que mejore las condiciones para todos en la ciudad. La historia de este problema comienza con el primer carro que trajeron a Medellín que fue en el año 1899. A partir de ese año, con el paso del tiempo fueron llegando más carros a la ciudad, hasta llegar a lo que nos hemos convertido hoy, una ciudad donde uno puede pasar metido en un "taco" para llegar del tesoro hasta la universidad eafit fácilmente 1-2 horas dependiendo de la hora a la que intente salir. Por lo que por medio de esto buscaremos la manera de mejorar esta situación.

2. PROBLEMA

Hoy en día en la ciudad de Medellín presentamos una congestión debido al alto tráfico que se presenta cada día y va aumentando por la gran cantidad de carros que llegan a la ciudad día a día, estos carros se puede evidenciar que dentro de la gran mayoría solo se moviliza 1 persona, donde se podrían movilizar fácilmente 3-5 personas. Este problema es importante resolverlo debido a varios factores como lo viene siendo los siguientes: 1) Temas Ambientales. 2) Temas de trabajo, ejemplo esto puede afectar la hora de llegada ya sea a la casa o al trabajo debido a la alta congestión. 3) Seguridad: Estando metidos en un taco, somos más susceptibles a que nos atraquen o nos pase algo. Resolviendo este problema buscamos mejorar la calidad de vida no solo de nosotros sino también de las personas que nos rodean.

3. TRABAJOS RELACIONADOS

3.1 Problema "De enrutamiento de vehículos VRP "

El problema de enrutamiento, conocido como VRP. Es un problema complejo de optimización que tiene como objetivo minimizar los costos de transportes asociados a rutas de reparto, la complejidad de resolver un problema de este depende de distintas restricciones que puede presentar ejemplo: que sea de flota heterogénea, de rutas abiertas, ventanas de tiempo o unas de recogida y entrega. La solución de este algoritmo comienza con una solución inicial factible de rutas, la cual es comparada con otra solución factible. Entre las 2 se escoge la que mejores resultados, es decir, la que optimice mejor los objetivos definidos también conocido como (Local Search). Este proceso se repite miles de veces hasta encontrar la solución más cercana al óptimo posible respetando todas aquellas condiciones que se hayan establecido tanto para los vehículos como para los puntos a ser visitados (las variantes).

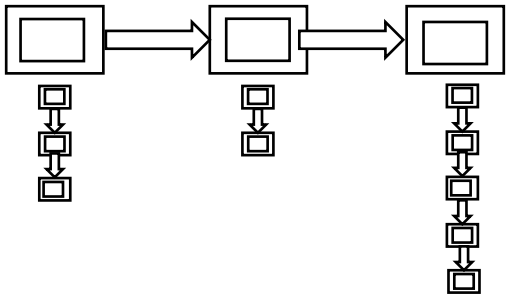
3.2 Muddy City El problema de muddy city necesita que las calles de una ciudad estén pavimentadas de modo tal que se pueda llegar desde una casa cualquiera a otra casa cualquiera, usando otras casas como medio si es necesario para que la pavimentación se haga al mínimo costo. La solución que se le encontró al problema fue usando árboles recubridores mínimos es una estructura de datos que parte de un grafo conexo y no dirigido. Es un subgrafo de ese grafo inicial que debe de ser un árbol y que contiene todos los vértices del grafo inicial.

3.3 Problema del viajante Dada una lista de ciudades y la distancia entre cada ciudad, el problema consiste en hallar la ruta más corta posible en la que se visite cada una de las ciudades exactamente una vez y volver al punto de partida. Su solución más común es utilizar el algoritmo de Boruvka, que encuentra el mínimo árbol de expansión en un grafo

3.4 Problema de conexiones de cable Dada una lista de lugares se necesita unirlas con la longitud de cable más corta debido a su precio. El problema necesita que todos los lugares estén conectados aunque no es necesario que sea directamente, es decir, que debe existir un camino por el que cualquier lugar se conecte a cualquier otro.

4. ALGORITMO PARA LA OPTIMIZACIÓN DE RUTAS EN LA CIUDAD

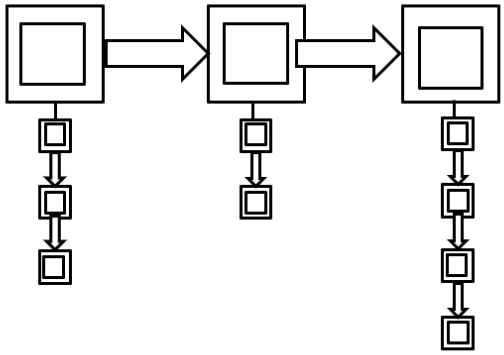
4.1 Estructura de datos



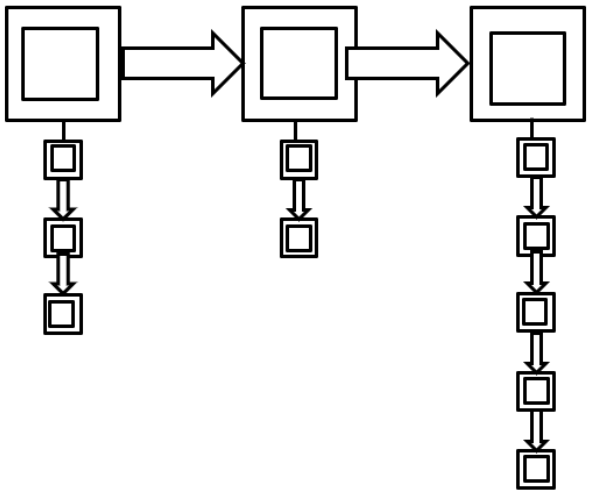
Gráfica 1: Lista doblemente enlazada que almacena la asignación de vehículos. En la lista de más afuera se almacenan los carros y en la de más adentro se almacena cada uno de los nodos/personas que se agregan al carro.

4.2 Operaciones de la estructura de datos

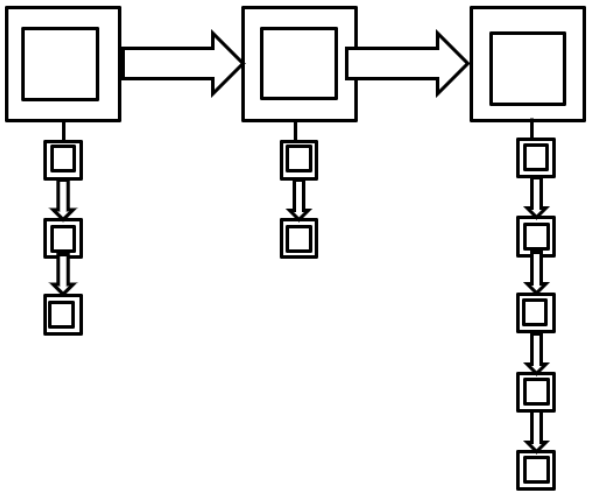
La operación de agregar sería así:



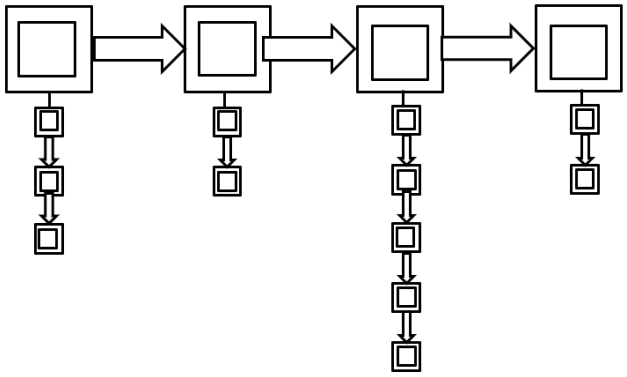
Si se le agrega un vértice más al carro de la posición 2, quedaría así:



La operación de agregar un carro sería así:



Al agregarse un carro, la lista quedaría así:



4.3 Criterios de diseño de la estructura de datos
Seleccionamos este algoritmo con esta estructura de datos debido a que su complejidad era más baja por lo tanto hay menos tiempo de ejecución.

Ya que con las linkedlist enlazadas, al usar el método de agregar nos iba a dar una complejidad constante y no se iba a seguir aumentando el tiempo de ejecución del programa.

Tuvimos en cuenta que la suma del camino nuevo no fuera mayor a la distancia que había desde el nodo inicial hasta la universidad multiplicada por la constante.

4.4 Análisis de Complejidad

SUBPROBLEMA	COMPLEJIDAD
Leer el archivo	$O(n)$
Hallar la distancia desde un nodo cualquiera hasta la universidad	c
Hallar la distancia mayor que se encuentra en una lista	$O(n^2)$
Asignar nodos a los vehículos	$O(n^2)$
Guardar el archivo	$O(m^2)$

Donde n es el número de nodos del grafo y m es la longitud de la lista de carros luego de que se asignan los vehículos.

Tabla 1: Tabla para reportar la complejidad

4.5 Algoritmo

El algoritmo primero calcula todas las distancias que hay entre cada nodo y la universidad y se almacena en una lista. Luego se halla la mayor distancia de esa lista y empieza visitando el vértice más lejano. Cuando está en este vértice, se calcula el sucesor más cercano y se agrega al carro junto con el del vértice, luego, a ese sucesor más cercano se le halla el más cercano y se agrega al carro. Este proceso se repite hasta que se llene el carro con 5 vértices. Apenas el carro está lleno, se agrega a la lista que almacena todos los carros y se vacía para que se repita el mismo proceso con el nodo más lejano a la universidad pero que ya no sea el que se utilizó para la primera iteración.

4.6 Criterios de diseño del algoritmo

Seleccionamos este algoritmo con esta estructura de datos debido a que su complejidad era más baja por lo tanto hay menos tiempo de ejecución.

Ya que con las linkedlist enlazadas, al usar el método de agregar nos iba a dar una complejidad constante y no se iba a seguir aumentando el tiempo de ejecución del programa.

Tuvimos en cuenta que la suma del camino nuevo no fuera mayor a la distancia que había desde el nodo inicial hasta la universidad multiplicada por la constante.

4.7 Tiempos de Ejecución

	Mejor tiempo	Peor tiempo	Tiempo promedio
P=1.1	1 ms	1 ms	1 ms
P=1.2	0 ms	1 ms	0,167 ms
p=1.3	0 ms	1 ms	0,4 ms

Tabla 3: Tiempo de ejecución del algoritmo para otimización de rutas en la ciudad para un grafo de 11 nodos con diferentes p

	Mejor tiempo	Peor tiempo	Tiempo promedio
P=1.1	11 ms	24 ms	17 ms
P=1.2	14 ms	21 ms	18,2 ms
p=1.3	15 ms	22 ms	17,66 ms

Tabla 4: Tiempo de ejecución del algoritmo para otimización de rutas en la ciudad para un grafo de 205 nodos con diferentes p

4.8 Análisis de los resultados

Numero de vértices	p	Numero de carros
205	1.1	79
205	1.2	71
205	1.3	60

Tabla 5: Resultados del algoritmo para un grafo de 205 vértices.

En la tabla 5 se pueden ver los resultados del algoritmo con el grafo de 205 vértices. Se puede ver que optimiza la solución ya que baja el número de carros. Si cada persona usara su propio carro serían 205 carros. Y el número de carros óptimo se debería acercar a 41. Esto significa que la solución que nos da es óptima, pero se podría mejorar más.

5. CONCLUSIONES

La idea del proyecto era organizar unas rutas de modo que las personas pudieran recoger a otras sin que se aumentara el tiempo que se demoraba el conductor inicialmente.

Concluimos que a pesar de bajar considerablemente el número de carros, debemos buscar la manera de que sea más óptimo.

Quisiéramos reducir la complejidad del algoritmo ya que consideramos que puede ser más óptimo.

5.1 Trabajos futuros

Nos gustaría mejorar la complejidad en el código que implementamos, para que este sea mejor. También quisiéramos tener en cuenta otras cosas como posibles accidentes en la vía o que un carro tenga pico y placa y no pueda ser usado en algún día específico.

AGRADECIMIENTOS

Agradecemos al monitor Rafael Villegas, por ayudarnos con nuestras inquietudes a lo largo del trayecto recorrido en semestre.

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.