Algoritmo para disminuir el trafico urbano asignando rutas de vehículos compartido a los empleados de una empresa

Juan Camilo Guerrero Alarcón
Universidad Eafit
Colombia
jcguerrera@eafit.edu.co

Juan José Escudero Valencia
Universidad Eafit
Colombia
jjescuderv@eafit.edu.co

Santiago Pulgarín Vásquez
Universidad Eafit
Colombia
spulgarinv@eafit.edu.co

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

RESUMEN

El problema se basa en la disminución de uso de vehículos particulares en la ciudad de Medellín. Actualmente la ciudad está enfrentando un gran problema ambiental en cuanto a la calidad del aire, y las entidades de movilidad han implementado técnicas como el pico y placa ambiental, la cual contribuye con disminuir la emisión de gases, pero no da una solución permanente. Para encontrar posibles soluciones, analizamos problemas que son parecidos al nuestro en varios aspectos, teniendo como interés la búsqueda de caminos más cortos pasando por varios vértices con un destino común. La solución propuesta por el equipo de trabajo fue empezar a escoger los nodos más lejanos al destino propuesto, para eso organizamos las distancias desde la más cercana hasta la más lejana, revisando el más lejano y buscando que nodos cercanos estaban a él, y verificando si se puede recoger a un nodo cercano del más lejano. En cuanto a los resultados obtenemos una reducción en el tráfico de la ciudad que nos permiten solucionar el problema de cierta manera, dichos resultados son que reducimos a más de la mitad de los nodos previos. Concluimos que el tráfico de la ciudad se reduce de manera considerable y que es posible hacerlo de manera eficiente, también dicha implementación nos ayuda a reducir la polución generada en la ciudad.

Palabras clave

Diseño - Algoritmo – Camino más Corto – Optimización

Palabras clave de la clasificación de la ACM
Theory of computation → Design and analysis of algorithms → Graph algorithms analysis → Shortest paths

1. INTRODUCCIÓN

El aumento de tráfico en la ciudad de Medellín se ha convertido en un problema serio que deriva en más de estos, uno de ellos es la polución del aire, mayor tiempo en las transiciones diarias, se reduce la productividad, etc. Una de las posibles soluciones es que los dueños de los vehículos compartan con las demás personas dichos vehículos, con esto no se tendrá vehículo por persona, esto claro se puede aplicar siempre y cuando las personas se dirijan a una misma institución o sitio de empleo, en estos casos sería muy eficiente, reduciendo también el tiempo de espera en un parqueadero.

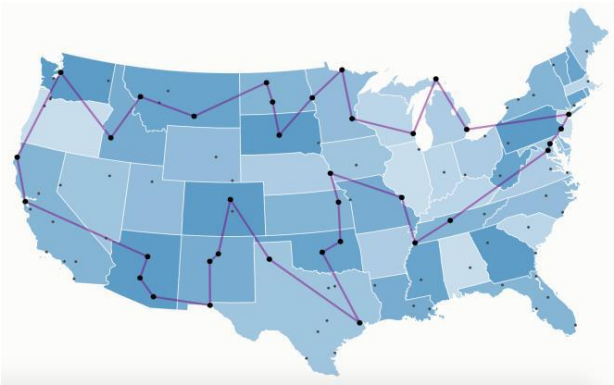
2. PROBLEMA

El problema radica en la disminución de tráfico en la ciudad, una de las estrategias es que cada uno de los dueños de vehículos de determinada empresa recoja a otras personas que trabajan en esta misma, el conductor puede llevar un máximo de 4 pasajeros y todos se dirigen a al mismo lugar de trabajo. El problema radica en que el tiempo en el que se demora recogiendo a los demás ocupantes no puede aumentar más de lo debido.

3. TRABAJOS RELACIONADOS

3.1 El problema del viajero

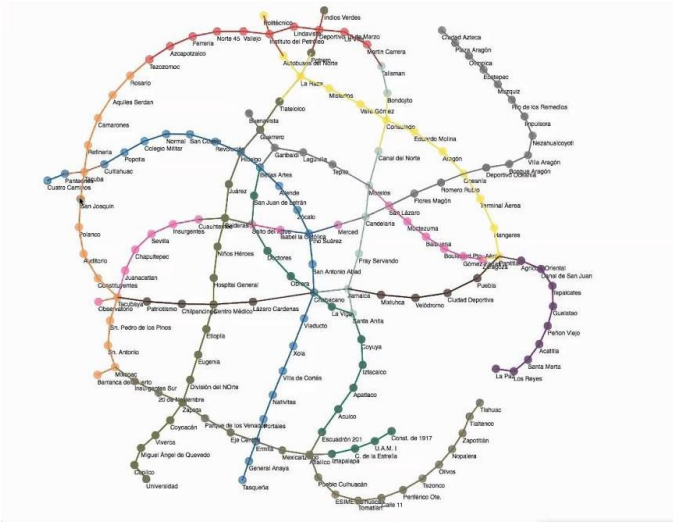
Los orígenes del problema del vendedor ambulante no están claros. El problema del vendedor ambulante fue formulado matemáticamente en la década de 1800 por el matemático irlandés WR Hamilton y por el matemático británico Thomas Kirkman. El juego Icosian de Hamilton era un rompecabezas recreativo basado en encontrar un ciclo hamiltoniano. El problema del vendedor viajero (TSP) hace la siguiente pregunta: "Dada una lista de ciudades y las distancias entre cada par de ciudades, ¿cuál es la ruta más corta posible que visita cada ciudad y regresa a la ciudad de origen?". El algoritmo define que el uso de fuerza bruta, aunque esto no denota un rendimiento óptimo, este problema actualmente es de los más estudiados en cuestión de optimización.



3.2 El problema del camino más corto sobre la red del Metrobús y del metro de la Ciudad de México

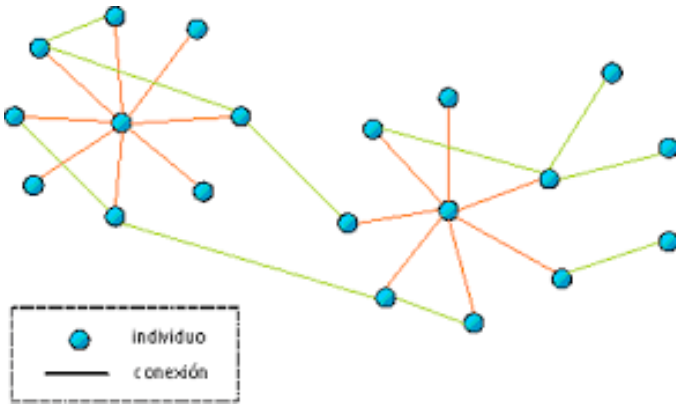
Los orígenes del problema se dan cuando en la fase de diseño para construir la red de Metrobús y metro en ciudad de

México, surge la necesidad de encontrar el camino más corto entre la estación “Cuatro Caminos” y “Tláhuac”, sobre la red de metro y Metrobús. Y en donde el diseñador se pregunta ¿Cuál es la ruta de metro y Metrobús más corta posible de recorrer el grafo y volver al punto de origen? Todo esto relacionado con la necesidad de optimizar el tiempo en este tipo de problemas.



3.3 El problema de rutas de vehículos (mensajería)

Este es uno de los problemas más estudiados en investigación operativa. En donde se plantea la necesidad de diseñar el conjunto óptimo de rutas para una flota de vehículos que deben servir a un determinado número de clientes. En donde aparece una empresa repartidora de mensajería que desea encontrar la ruta optima de reparto o recogida desde uno o varios centros de acopio a un número determinado de ciudades o clientes de tal manera que el coste total originado de ese reparto o recogida sea mínimo.



3.4 The School Bus Routing Problem

Muchas escuelas tienen contratos con empresas privadas de transporte para tener cierto número de buses a disposición de sus estudiantes. Estos contratos se basan en costos por número de buses y la distancia que estos recorran. El problema que enfrentan las escuelas es encontrar rutas para sus buses de tal manera que se utilicen el menor número de estos, se cubran todas las estaciones sin recorrer mucha distancia y que los recorridos no duren más de 45 minutos.

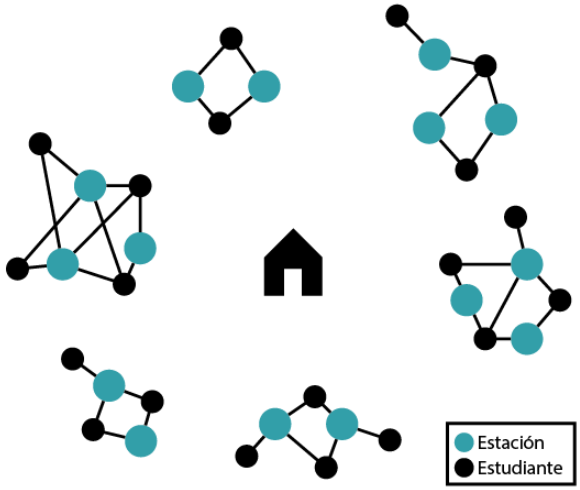


Figura 1. Representación de un posible caso.

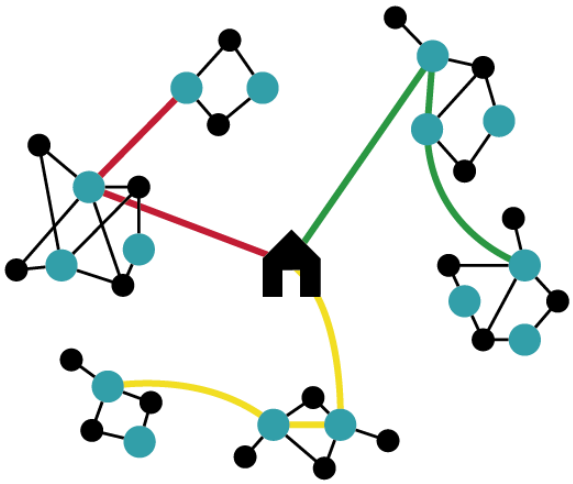
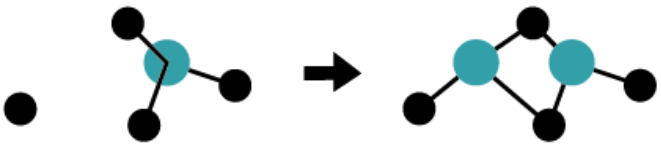


Figura 2. Representación de una solución con 3 buses.

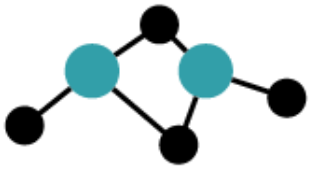
Solución: Primero generamos un grafo que contenga la cantidad mínima de estaciones de buses posible, tal como se ve en la figura 1. Para realizar este análisis, buscamos las estaciones que conecten con la mayor cantidad de casas de los estudiantes, con la restricción de que ningún estudiante se quede por fuera del análisis, y que, de ser necesario, se

agregue otra estación para un estudiante.



Luego, se buscan las estaciones que conecten con todos los estudiantes de un subconjunto, y si esa estación no existe, entonces se empieza una búsqueda del camino más corto entre ellas, para luego añadirlas a una ruta final que no exceda más de 45 minutos de recorrido. Creando entonces varias paradas y varias rutas para poder recoger a todos los estudiantes.

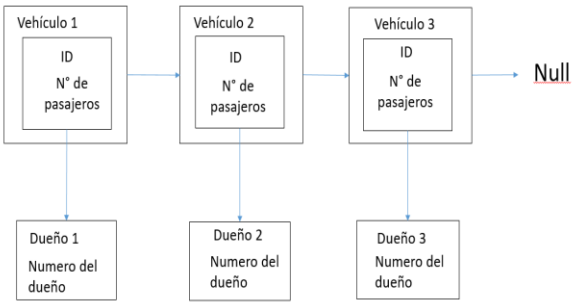
Caso 1. Una única estación para todo el subconjunto



Caso 2. Se agrega otra estación para poder recoger a todos los estudiantes

4. Doble Lista Enlazada con PermutacionesA continuación, explicamos la estructura de datos y el algoritmo.

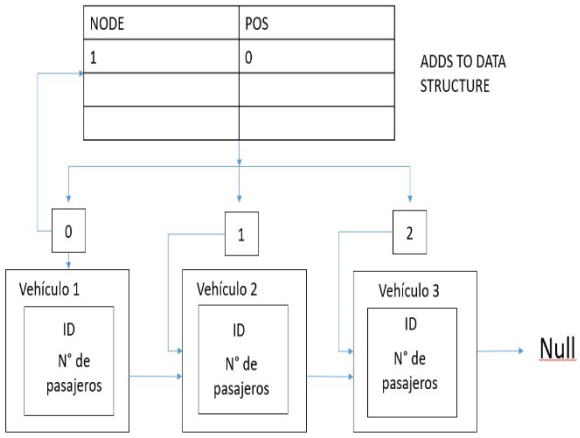
4.1 Estructura de datos



Gráfica 1: LinkedList doble con permutaciones, en las cuales incluyes cada lista un cierto número de pasajeros

4.2 Operaciones de la estructura de datos

Adición de datos a LinkedList.



Gráfica 2: Ilustración de la adición en la estructura de datos escogida previamente

4.3 Criterios de diseño de la estructura de datos

La selección de dicha estructura de datos fue implementada ya que esta nos permitía dar una solución previa a la definitiva, ya que con dichas listas podíamos simular tanto los conjuntos como los subconjuntos para realizar las correspondientes permutaciones y así empezar a comparar todos los casos, eso además de que no proporciona un gasto inmenso o considerable de memoria, aunque cabe aclarar que puede tener cierto margen de error.

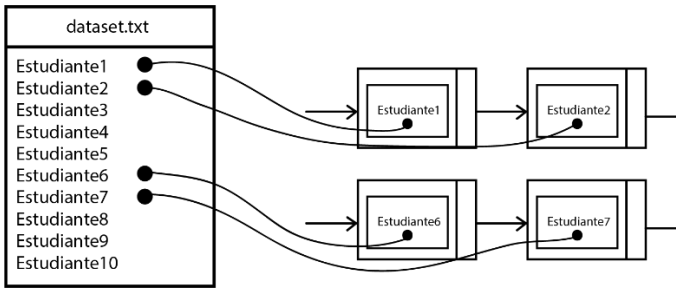
Métodos LinkedList	Complejidad
Búsqueda	O(n)
Adición	O(1)
Eliminación	O(1)

4.4 Análisis de Complejidad

Tabla 1: Tabla para reportar la complejidad

4.5 Algoritmo

Crear una LinkedList de LinkedLists, en la que las segundas son cada carro que se va a utilizar.



Gráfica 3: Ilustración de la funcionalidad del algoritmo.

Como se ve en la *Gráfica 3* el algoritmo toma el dataset y crea una LinkedList de carro1 desde el primer estudiante, luego lee la siguiente línea del dataset y lo añade a la LinkedList, esto hasta que alcance un total de 5 que sería el límite de personas que caben en un carro. Cuando se acabe la LinkedList (En este caso con Estudiante5), el estudiante de la siguiente línea se añade a una Lista nueva, y así hasta que acabe el dataset.

4.6 Cálculo de la complejidad del algoritmo

Sub problema	Complejidad
Leer el archivo y guardar los datos en la estructura de datos	O(n^2)
Asignar los vehículos compartidos	O(n)
Guarda el archivo en un archivo .txt	O(n)
Complejidad Total	O(n^2)

Tabla 2: Complejidad de cada uno de los subproblemas que componen el algoritmo. Sea n el número de dueños de vehículos y la empresa.

4.7 Criterios de diseño del algoritmo

El algoritmo se realizó de esta manera ya que nos permite asemejar el problema como un conjunto de datos con el cual podemos realizar las permutaciones para así comparar con los demás vecinos y cada subconjunto de dichos datos, también porque la estructura de los datos no presenta un mayor inconveniente para su implementación y al momento de realizar las restricciones correspondientes se disminuye el margen de error, ya que podemos agrupar estos datos y asignar a el conjunto del que hablamos anteriormente. Otro factor importante fue el uso de memoria, ya que su ejecución no requería cantidades considerables.

4.8 Tiempos de Ejecución

	Conjunto de Datos 1 (205, 1.1)	***Conjunto de Datos 2 (205, 1.2)***	***Conjunto de Datos 3 (205, 1.3)***
Mejor caso	0 ms	0 ms	0 ms
Caso promedio	0 ms	4 ms	5 ms
Peor caso	1 ms	16 ms	16 ms

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

4.9 Memoria

	Conjunto de Datos 1 (205, 1.1)	***Conjunto de Datos 2 (205, 1.2)***	***Conjunto de Datos 3 (205, 1.3)***
Consumo de memoria	371 MB	400 MB	437 MB

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

4.10 Análisis de los resultados

Resultados Algoritmo:

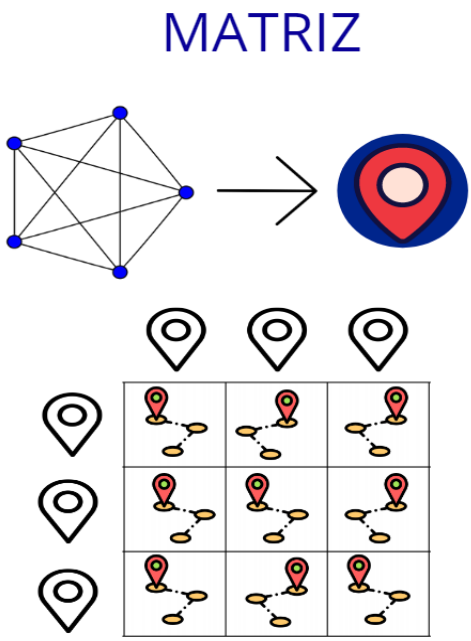
Sub problema	Complejidad
Leer el archivo y guardar los datos en la estructura de datos	O(n^2)
Asignar los vehículos compartidos	O(n)
Guarda el archivo en un archivo .txt	O(n)
Complejidad Total	O(n^2)

Resultados Estructura de Datos:

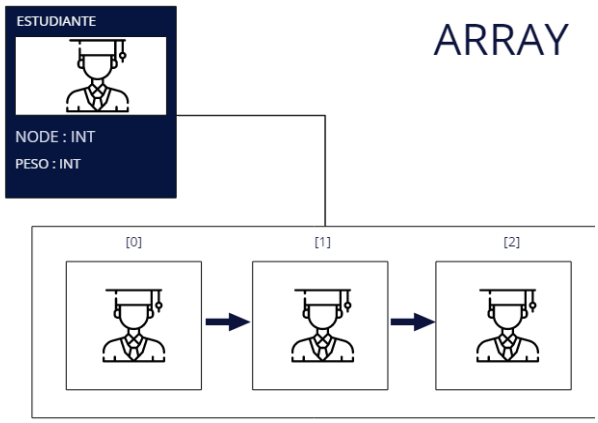
Métodos LinkedList	Complejidad
Búsqueda	O(n)
Adición	O(1)
Eliminación	O(1)

5. UNBELIEVABLE CARPOOLING

5.1 Estructura de datos

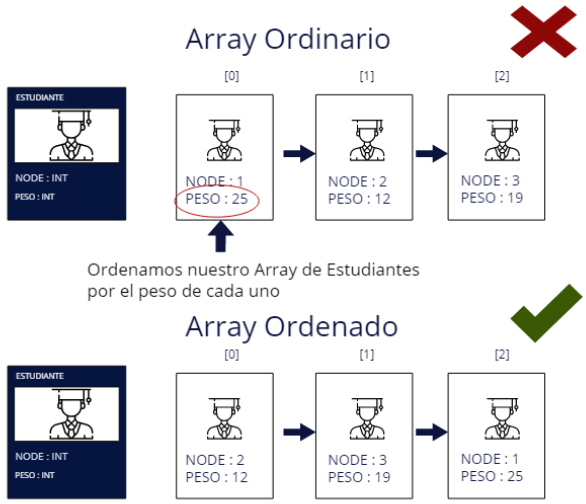


***Grafica 4**: Implementación de la matriz para guardar el grafo*



***Grafica 4.1**: Implementación del array como estructura de datos*

5.2 Operaciones de la estructura de datos



***Grafica 5**: Ordenamiento del array, para manipular los datos*

5.3 Criterios de diseño de la estructura de datos

El criterio por el cual decidimos emplear una matriz de adyacencia para guardar el grafo es porque al momento de acceder en ella es O(1) independiente de la posición en la que se requiera, el uso de memoria es mínimo y trabaja de una manera eficiente. Al usar dicha estructura de datos manipular dicho grafo es mucho más fácil de usar. Como estructura de datos también utilizamos un arreglo el cual al igual que la matriz acceder a el es eficiente, y no consume grandes cantidades de memoria, además que dado el caso de ordenar dicho array nos brinda una complejidad eficaz.

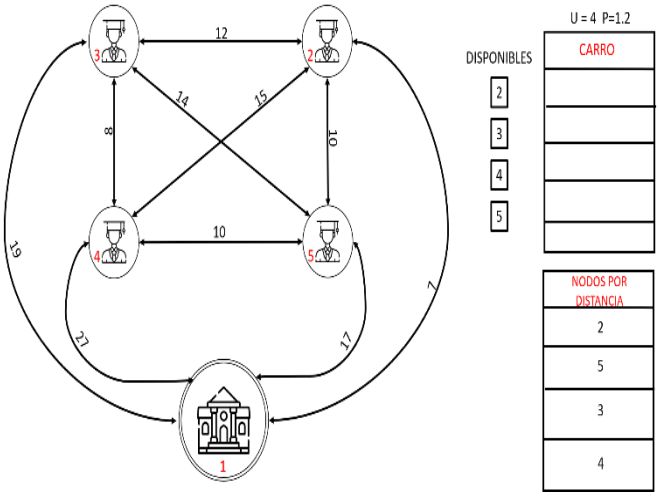
5.4 Análisis de Complejidad

Método	Complejidad
Adición en matriz	O(1)
Adición en arreglo	O(1)
Obtención de matriz	O(1)
Obtención de arreglo	O(1)
Ordenamiento de arreglo	O(nlogn)

Siendo n el número de elementos en el arreglo, Java brinda un método que permite ordenar arreglos con O(nlogn) sin importar el arreglo.

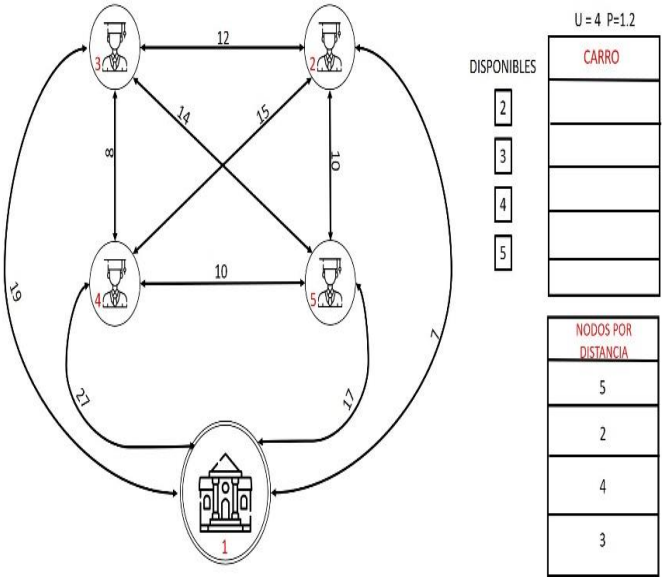
Mètodo	Complejidad
Bùsqueda Fonètica	O(1)
Imprimir búsqueda fonètica	O(m)
Insertar palabra busqueda fonètica	O(1)
Búsqueda autocompletado	O(s + t)
Insertar palabra en TrieHash	O(s)
Añadir bùsqueda	O(s)

Tabla 6: Tabla para reportar la complejidad

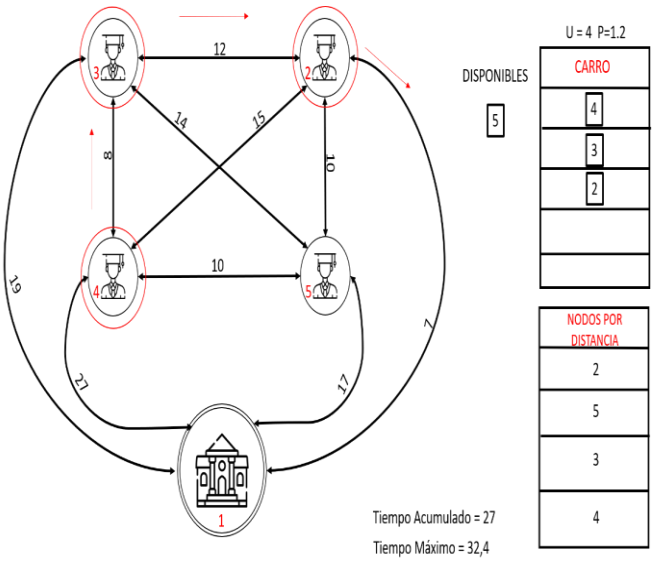


Gráfica 7: *Ordenamiento descendente de los nodos en el array con respecto a su distancia a la Universidad*

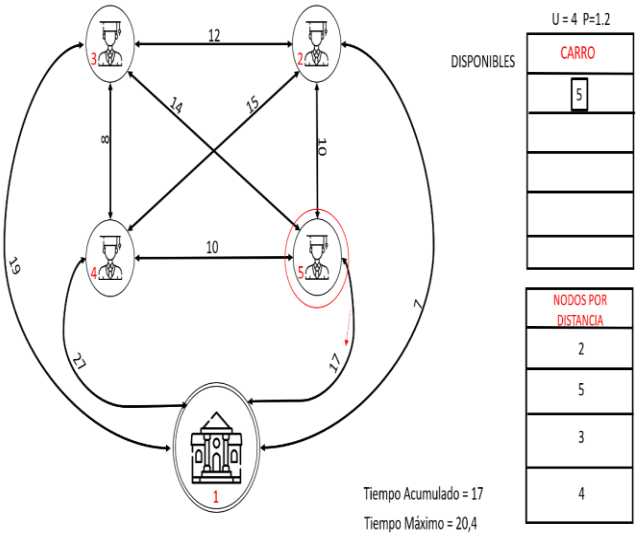
5.5 Algoritmo



Gráfica 6: *Inserción de los nodos en un array llamado Nodos por Distancia*



Gráfica 8: *Escogencia de la ruta más corta y asignación a un carro*



Gráfica 9: *Escogencia de la ruta más corta con el nodo disponible y asignación a un carro*

5.6 Cálculo de la complejidad del algoritmo

Sub problema	Complejidad
Leer el dataset y crear la estructura de datos	O(n)
Ordenar el arreglo de sucesores	O(nlogn)
Llenar los carros	$O(n^2)$
Complejidad Total	$O(n^2 \log n)$

Tabla 7: complejidad de cada uno de los sub problemas que componen el algoritmo. Siendo n el número de personas que deseamos distribuir en carros.

5.7 Criterios de diseño del algoritmo

El algoritmo se diseñó de esta manera debido a que almacenar el grafo mediante una matriz de adyacencia le brinda alta eficiencia en tiempo, además el hecho de que el algoritmo escoja a los nodos más distantes de la universidad como posibles candidatos a ser conductores permite que el número limitante que se determina con la constante sea mayor, por lo tanto, permita recorrer más nodos y distribuir mejor los carros disponibles, donde posteriormente centrado en el nodo conductor buscara cual es el más cercano a él, revisando en todos los casos que no exceda el tiempo máximo permitido para ese nodo y que cada vez se esté acercando más a la universidad nos asegura una alta precisión en el número optimo total de vehículos y en el tiempo de ejecución del algoritmo.

5.8 Tiempos de Ejecución

Teniendo que U (Usuarios) es el número de personas a las que deseamos aplicar nuestro algoritmo de carpooling. Y teniendo la constante p como el multiplicador para determinar cuanto sería el tiempo máximo que puede demorar el conductor recogiendo a sus compañeros y llegando a su destino.

Con 5 usuarios:

	Constante (p) = 1,2	**Constante (p) = 1,7**
Mejor caso	0,02 ms	0,02 ms
Caso promedio	0,06 ms	0,06 ms
Peor caso	2,52 ms	2,53 ms

Con 11 usuarios:

	Constante (p) = 1,1	**Constante (p) = 1,2**	**Constante (p) = 1,3**
Mejor caso	0,02 ms	0,02 ms	0,3 ms
Caso promedio	0,12 ms	0,11 ms	0,12 ms
Peor caso	2,85 ms	2,95 ms	3,01 ms

Con 205 usuarios:

	Constante (p) = 1,1	**Constante (p) = 1,2**	**Constante (p) = 1,3**
Mejor caso	0,76 ms	0,77 ms	1,23 ms
Caso promedio	2,43 ms	2,74 ms	2,74 ms
Peor caso	23,2 ms	32,1 ms	38,7 ms

5.9 Memoria

Mencionar la memoria que consume el programa para varios ejemplos

Con 5 usuarios:

	Constante (p) = 1,2	**Constante (p) = 1,7**
Consumo de memoria	1,57 MB	1,14 MB

Con 11 usuarios:

	Constante (p) = 1,1	**Constante (p) = 1,2**	**Constante (p) = 1,3**
Consumo de memoria	1,13 MB	1,82 MB	1,82 MB

Con 205 usuarios:

	Constante (p) = 1,1	**Constante (p) = 1,2**	**Constante (p) = 1,3**
Consumo de memoria	16,7 MB	16,6 MB	16,1 ms

5.10 Análisis de los resultados

Dataset	**Resultados**
U = 5 y P = 1,2	3 carros
U = 5 y P = 1,7	2 carros
U = 11 y P = 1,1	6 carros
U = 11 y P = 1,2	6 carros
U = 11 y P = 1,3	5 carros
U = 205 y P = 1,1	56 carros
U = 205 y P = 1,2	52 carros
U = 520 y P = 1,3	51 carros

Notamos que nuestro algoritmo funciona de manera más eficiente cuando tenemos más usuarios por analizar, es decir, mientras mayor sea la cantidad de nodos, tendremos un resultado más óptimo.

6. CONCLUSIONES

- De los criterios más importantes que tuvimos en cuenta fue almacenar los datos en una estructura eficiente en memoria y tiempos de ejecución, también teniendo como prioridad la complejidad del algoritmo.
- Obtuvimos como uno de los resultados más importantes la reducción de más de la mitad de los vehículos en la ciudad en un buen tiempo de ejecución.
- La primera solución planteada no satisfacía en lo absoluto ya que no cumplía con las expectativas y ningún criterio visto, la manera de solucionarlo no cumplía tampoco con el planeamiento.
- Uno de los problemas que se presentó en el desarrollo del proyecto fue la poca experiencia en temas de carpooling y utilización de grafos, además de buscar una solución óptima al inicio del mencionado proyecto.
- En trabajos futuros se desearía hacer del algoritmo mucho más preciso a la hora de asignar estudiantes a vehículos, además de que sea mas dinámico y funcional y se pueda usar en más puntos de destino.

6.1 Trabajos futuros

En trabajos futuros se desearía hacer del algoritmo mucho mas preciso a la hora de asignar estudiantes a vehículos, además de reducir considerablemente la complejidad que tiene asociada el algoritmo esto con el fin de poder implementar el algoritmo en la vida real.

AGRADECIMIENTOS

Queremos dedicar un especial agradecimiento a todas las personas que directamente o indirectamente favorecieron el desarrollo de este proyecto.

REFERENCIAS

[1] D. L.; Bixby, R. M.; Chvátal, V.; Cook, W. J (2007). Travelling_salesman_problem. https://en.wikipedia.org/wiki/Travellingsalesman_problem#History.

[2] (Hois, s.f.). El problema del camino más corto sobre la red del Metrobús y del metro de la Ciudad de México ´ (Solución por Dijkstra y Simplex Dual) http://eenube.com/images/pdf/reporte.pdf

[3] Calviño, A. (2011). Cooperación en los problemas del viajante (TSP) y de rutas de vehículos (VRP): una panorámica.

[4] R. Lewis, K. Smith-Miles, K. Phillips. (s.f). The School Bus Routing Problem: An Analysis and Algorithm. https://core.ac.uk/download/pdf/158355938.pdf