JK Shared Vehicle Strategic System (JK-SVSS algorithm)

Kevin Alexander Herrera
Universidad Eafit
Colombia
kaherrerag@eafit.edu.co

Jose Joab Romero Humba
Universidad Eafit
Colombia
jjromeroh@eafit.edu.co

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

RESUMEN

El alto tráfico trae grandes problemas a la calidad de vida de las personas debido a que afecta diversos factores como la calidad de aire y la disminución de la productividad; la estrategia "vehículo compartido" es significativamente funcional y efectiva sobre entornos como empresas, esta consiste en que las personas compartan su vehículo reduciendo de esta manera el número de vehículos particulares que circulan diariamente, es necesario implementar un algoritmo que determine la ruta más corta en la que una persona puede recoger a otras de modo que se evite que las mismas vayan en su propio carro individualmente, con la condición que el tiempo no puede aumentarse más de cierto porcentaje en comparación al original.

PALABRAS CLAVES

Grafo, Pathfinding, Tabla Hash, Carpooling, Complejidad, Memoria, Tráfico, Estructura de datos, Algoritmo, Automóvil, Camino mínimo.

PALABRAS CLAVES ACM

Theory of computation → Design and analysis of algorithms and problem complexity→ Data structures and algorithms → Graph algorithms analysis → Shortest paths.

1. INTRODUCCIÓN

Los elevados niveles del tráfico en las ciudades son una de las principales causas de problemas en diversos aspectos con relación a la calidad de vida de las personas, esto se debe a que afecta drásticamente el ambiente, la productividad y la movilidad dentro de las ciudades.

El siguiente informe enseña cómo a partir de una estrategia denominada "vehículo compartido" se puede reducir considerablemente la cantidad de vehículos que circulan por la ciudad a la vez, consiste en que las personas usen su vehículo de manera compartida con las personas que se dirijan a su mismo destino, evitando asì que usen sus vehículos particulares individualmente, sin embargo, se debe cumplir la siguiente condición: el tiempo de viaje no debe exceder cierto porcentaje respecto al viaje original para ello es necesario implementar un algoritmo que indique qué personas pueden recoger a otras de la manera más eficiente respecto a movilidad-tiempo.

2. PROBLEMA

La problemática a la cual se enfrenta la estrategia "vehículo compartido" es el diseño y la implementación de un algoritmo que logre encontrar de manera eficiente y rápida el grupo de personas que pueden ir juntas al mismo destino sin aumentar de manera considerable el tiempo de viaje; se hace con el objetivo de establecer métodos que se pueden aplicar en empresas y otras entidades para disminuir el uso de vehículos particulares y por consiguiente reducir el tráfico.

3. TRABAJOS RELACIONADOS

3.1 Problema del agente viajero:

Dada una lista de ciudades y las distancias entre cada par de ellas, ¿cuál es la ruta más corta posible que visita cada ciudad exactamente una vez y al finalizar regresa a la ciudad origen?

Solución: el algoritmo de programación lineal y entera; ramificación y acotación.

La primera forma de atacar el problema del viajante es utilizar un algoritmo de programación lineal y entera. Este es un algoritmo exacto pero con un coste de tiempo muy elevado. En muchos casos, la utilización de este programa es inviable debido a que el coste de tiempo es muy alto a pesar de que llegue siempre a obtener la solución del problema.

Solo se debe expresar el TSP como un problema de programación lineal entera y programarlo.

la fórmula es la siguiente:

$$\text{minimizar} \quad \sum_{i,j=0}^{n} c_{ij}x_{ij}$$

$$\begin{aligned}
\text{sujeto a} \quad (1) \quad & \sum_{i=0}^{n} x_{ij} = 1 && j = 0, ..., n \\
(2) \quad & \sum_{j=0}^{n} x_{ij} = 1 && i = 0, ..., n \\
(3) \quad & u_i - u_j + nx_{ij} \leq n - 1 && 1 \leq i \neq j \leq n \\
& 0 \leq x_{ij} \leq 1, && x_{ij} \in Z, \forall i, j
\end{aligned}$$

Figura 1: Ecuación agente viajero.

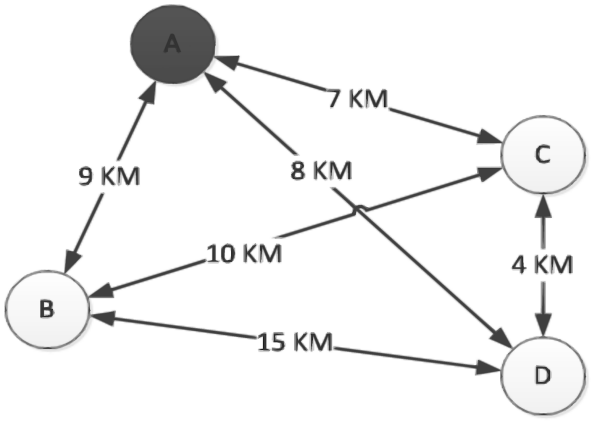


Figura 2: Ciudades y sus distancias como un grafo.

3.2 Persecución en los videojuegos:

En videojuegos como Pac Man, se presenta una situación donde el personaje principal, el jugador, es perseguido por unos enemigos que intentan atraparlo. el problema está en hacer que estos enemigos puedan localizar y perseguir al jugador.

Para esto se utiliza el algoritmo Pathfinding A*, Este algoritmo es una mejora del algoritmo de Dijkstra y trabaja considerando el escenario como una rejilla con muchas celdas: tenemos el punto de inicio dentro de una y nuestro punto objetivo en otra. El algoritmo consiste en calcular un "costo general" para cada celda que rodee la casilla que se esté analizando, se elige la celda de menor "costo general" y luego esta misma se convertirá en la celda de análisis, teniendo como casilla padre aquélla que acabamos de abandonar.

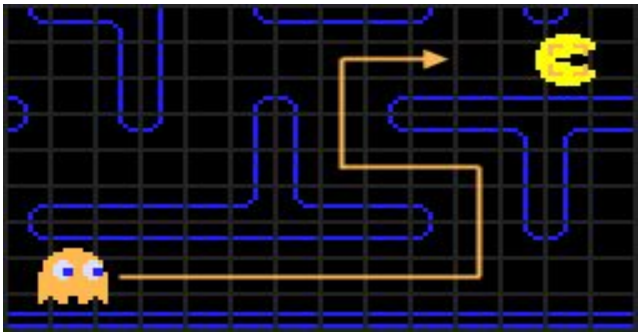


Figura 3: Pathfinding en el videojuego Pac Man.

3.3 Sistema de aparcado guiado inteligente:

En un parqueadero se desea establecer un sistema que permita otorgar el lugar más cercano a sus clientes para que estos estacionen su vehículo, de manera que el servicio sea más eficiente y rápido; para ello se necesita de una máquina que entregue un ticket al cliente indicando la ubicación correspondiente. Por lo tanto se ha empleado el algoritmo de Dijkstra de manera que tenga en cuenta los lugares disponibles y ocupados y calcule cual es el lugar más cercano para estacionar.

Figura 4: Sistema de aparcado inteligente.

3.4 Planificación del camino y asistencia de control de seguimiento de trayectoria global a vehículo autónomo:

Los vehículos autónomos tienen incorporado un sistema GPS que les permite ubicarse, un dilema que se presenta desde los orígenes de esta idea es indicar al vehículo el camino más corto para que llegue a su destino, para ello se han implementado diversos algoritmos de búsqueda de caminos cortos y combinaciones de los mismos con el objetivo de darle solución a este problema; actualmente se encuentra varias versiones de algoritmos "pathfinding" pero aún no son lo suficientemente óptimas.

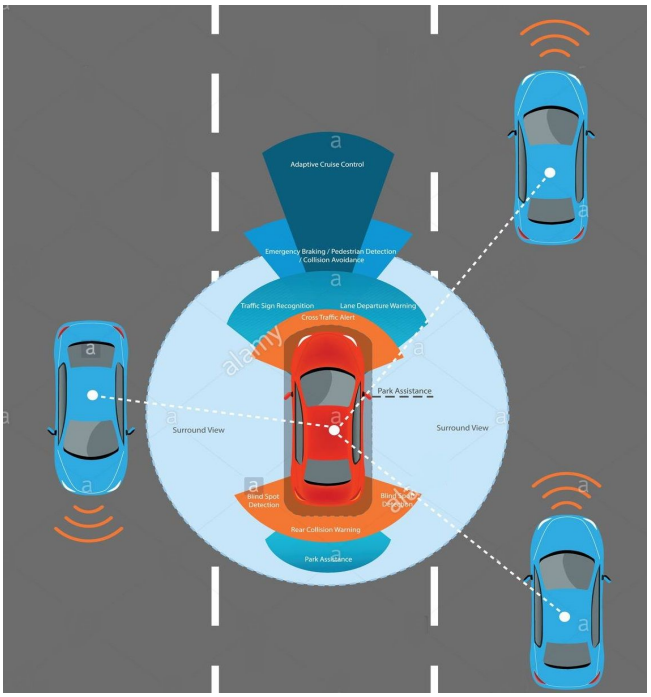
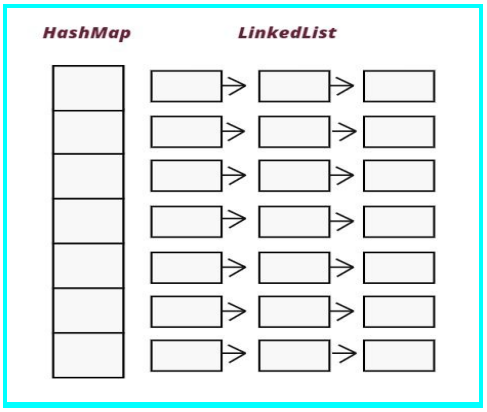


Figura 5: Sistema GPS en Automóvil Autónomo

4. TABLA HASH - LISTA ENLAZADA

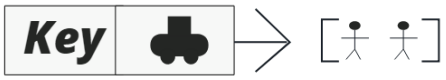
4.1 Estructura de datos:

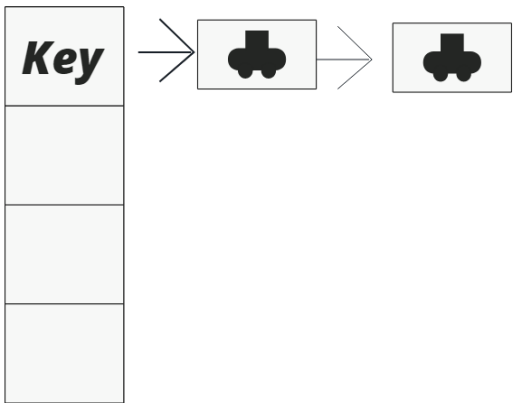


Gráfica 1: Gráfica de la estructura de datos.

4.2 Operaciones de la estructura de datos:

Gráfica 2: Operación para añadir datos.





Gráfica 3: Estructura para almacenar carros que contienen personas.

4.3 Criterio de diseño:

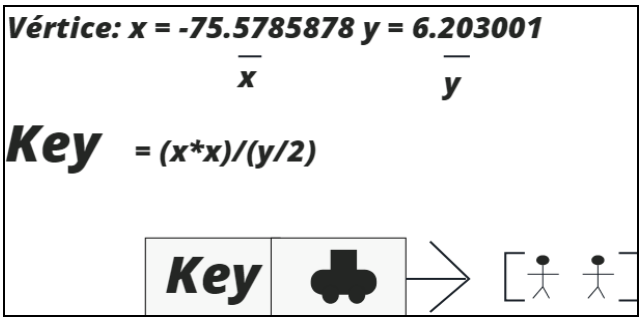
Se implementa una tabla Hash unida a una lista enlazada debido a la eficiencia que presentan las operaciones de las mismas, otorgan una gran facilidad y rendimiento (uso de memoria y tiempo) que permite la versatilidad dentro del algoritmo a la hora de agregar o consultar elementos, de esta manera se reduce la complejidad y los tiempos de ejecución.

4.4 Análisis De complejidad:

Método	Complejidad (n = Vértices)
LeerArchivo	O(n^2)
agrupadorDeCarros	O(n)
EscribirArchivo	O(n)

Tabla 1: Reporte de complejidad de los métodos.

4.5 Algoritmo:



Gráfica 4: Método para agrupar personas en carros.

4.6 Cálculo de complejidad del algoritmo:

Sub problema	Complejidad
Leer Archivo y extraer los vértices	O(n^2)
separar los vértices en el hashmap y asignarlo a un carro	O(n)
Escribir el archivo	O(n)
Complejidad Total	O(n^2)

Tabla 2: complejidad de cada uno de los sub problemas que componen el algoritmo. sea n el número de vértices.

4.7 Criterios de diseño del algoritmo:

Antes de la implementación del algoritmo se analizó que existe una relación directa entre las coordenadas X y Y de un vértice y su distancia-tiempo a otros vértices, lo cual permite establecer una serie de condiciones que agilizan el procesado de información, las cuales son la base para el algoritmo. De esta manera, empleando estructuras de datos que se acoplan a la arquitectura es posible implementar un diseño que permite al algoritmo ser más óptimo, rápido y eficiente (complejidad-memoria).

4.8 Tiempos de ejecución:

	Tiempo de ejecución (ms)
Mejor caso	192
Caso promedio	197
Peor caso	256

Tabla 3: Tiempos de ejecución del algoritmo.

4.9 Memoria:

	Conjunto de Datos 1	*Conjunto de Datos 2*
Consumo de memoria	0.152 MB	0.982 MB

Tabla 3: Consumo de memoria del algoritmo con diferentes conjuntos de datos.

4.10 Análisis de resultados:

	HashMap	LinkedList
Memoria	7.5 MB	3.47 MB
busqueda (ms)	145 ms	113 ms
agrupación (ms)	190 ms	198 ms

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

REFERENCIAS:

1. Es.wikipedia.org. (2019). *Problema del viajante*. [online] Available at:
2. https://es.wikipedia.org/wiki/Problema_del_viajante [Accessed 3 Mar. 2019].
3. Eio.usc.es. (2019). [online] Available at: http://eio.usc.es/pub/mte/descargas/ProyectosFinMaster/Proyecto_774.pdf [Accessed 3 Mar. 2019].
4. Rodríguez Puente, R. and Lazo Cortés, M. (2019). *Shortest path search: current applications*. [online] Availableat:https://www.researchgate.net/profile/Rafael_Rodriguez_Puente/publication/295861940_Busqueda_de_caminos_minimos_aplicaciones_actuales_Shortest_path_search_current_applications/links/56cf1f5a08ae4d8d649f9b09.pdf [Accessed 4 Mar. 2019