INSTITUTO TECNOLÓGICO AUTÓNOMO DE MÉXICO

SISTEMA DE INTELIGENCIA ARTIFICIAL CON MÚLTIPLES ESTRATEGIAS DE RAZONAMIENTO PARA PLANEAR RUTAS

INTELIGENCIA ARTIFICIAL

Presentan:

DIEGO VILLALVAZO SULZER 155844
JOSÉ DE JESÚS SÁNCHEZ AGUILAR 156190
SEBASTIÁN ARANDA BASSEGODA 157465
SANTIAGO CORTIZO BARREIRO 164596

PROFESOR:
ANDRÉS GÓMEZ DE SILVA GARZA

Mini-Router

Resumen

El presente documento comprende la elaboración de un programa escrito en SWI-Prolog que implementa un sistema de inteligencia artificial mini-router, el cual combina aspectos de razonamiento basado en casos con razonamiento basado en modelos e implementa el algoritmo de búsqueda A*. Con esto en mente, se desarrolló un sistema capaz de encontrar la ruta óptima entre dos estaciones dadas.

Este programa representa una solución de inteligencia artificial al ser genérica ya que permite ser adaptada para resolver problemas parecidos. A partir de una investigación previa se llegó a una teoría formal, sistemática y específica acerca del dominio de aplicación del sistema.

I. INTRODUCCIÓN Y MARCO TEÓRICO

La Ciudad de México cuenta con uno de los sistemas de transporte colectivo más grandes del mundo. En el presente proyecto se utilizaron dos diferentes sistemas, el metro que cuenta con 12 líneas y el metrobús que cuenta con 7.

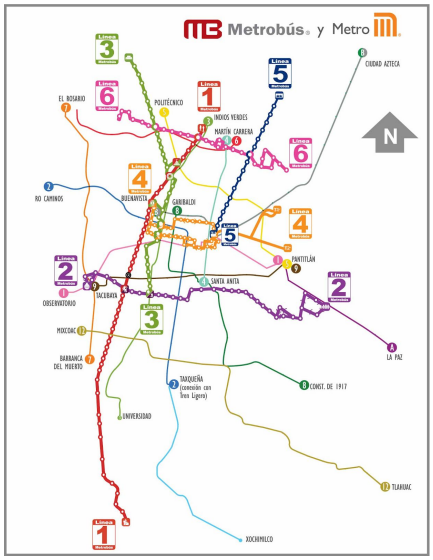

Fig. 1 Metro y Metrobús de la Ciudad de México

En total existen 6 líneas de metrobús y 12 de metro, con 208 y 195 estaciones respectivamente. Varias de estas estaciones cuentan con transbordos y permiten cambiar entre una línea de metro y otra. También existen otras estaciones que, aunque no cuentan con transbordos directos, permiten cambiar de sistema de transporte entre el metro y el metrobús, ya que las estaciones de los diferentes métodos de transporte están al costado una de otra y tienen el mismo nombre.

Una vez definidos los aspectos a tomar en cuenta respecto al sistema de movilidad colectivo de la CDMX se definió el diseño de la solución, la cual consiste en tres partes esenciales:

Primero, la implementación de todos los aspectos técnicos del del sistema de movilidad, es decir, cómo se podían representar las estaciones, cómo se iban a encontrar las conexiones, cómo se diferenciarán las líneas, las estaciones con conexiones y cualquier otra información que pensáramos pudiera ser relevante que estuviera al acceso del programa.

La segunda parte de la solución consiste en la implementación de una función heurística. Una función heurística consiste en maximizar o minimizar los aspectos de un problema. La función heurística asigna un valor a cada estado, es decir, dónde se encuentra el usuario en relación a la ubicación final, además de que es una herramienta útil para la búsqueda A*.

La tercera parte de la solución consiste en hacer que el sistema encuentre el movimiento óptimo dependiendo del estado actual. Para esto se utiliza el método de búsqueda A*.

La cuarta parte de la solución consiste en implementar un modelo de mini-router. El modelo consiste en combinar aspectos de razonamiento basado en casos y razonamiento de modelos/mapas organizados jerárquicamente. De esta manera, solo resta realizar la búsqueda A* para buscar dentro de una sola zona del mapa jerárquico.

II. DESARROLLO DE LA SOLUCIÓN

1.Implementación del Sistema y la función Heurística.

Se obtuvieron las estaciones de ambos sistemas del portal de datos de la Ciudad de México en formato JSON. Posteriormente se limpiaron los datos para representar cada estación en el lenguaje de PROLOG,

```
station(system, name, lat, lon, line, index).
```

Donde para cada estación se tiene:

- ***system***: sistema que pertenece (metro o metrobús).
- ***name***: nombre de la estación.
- ***lat*** y ***lon*** coordenadas.
- ***line***: línea del sistema a la que pertenece.
- ***index***: índice que ordena un conjunto de estaciones de la misma línea.

Representar las estaciones en este formato permite encontrar fácilmente las estaciones que tienen conexión con otras líneas o en otro sistema. De igual forma el ***index*** permite conocer el qué estaciones están conectadas entre sí y encontrar las terminales de cada línea, se ordenan de Norte-Sur y de Oeste-Este.

*2. Búsqueda A**

Para el algoritmo A*, el cual consiste de una función que estima qué camino es conveniente seguir, se calculó el valor mediante la norma euclidiana del nodo a evaluar a uno de los nodos hijo (estaciones compatibles) y del nodo padre al nodo destino para obtener:

$$f(n) = g(n) + h(n).$$

Donde:

$g(n)$: indica la distancia acumulada del nodo origen "S" al "N".
$h(n)$: expresa la distancia estimada desde el nodo "N" hasta el nodo destino "D".

Se programaron una serie de métodos/reglas auxiliares en PROLOG.

Cola de prioridades: una lista que recibe una estación con los valores g(n) y h(n), y los inserta en la lista calculando el valor f(n).

Estaciones adyacentes: para una estación en particular se obtienen las estaciones cercanas a la misma y con ayuda del nodo destino y de la norma euclidiana se decide a qué estación avanzar.

Estaciones de conexión: busca las estaciones en la base de conocimiento que tengan el mismo nombre, esto nos regresa una lista con las estaciones de conexión tanto en un sistema como en otro.

Proceso de búsqueda:

1. Tomar el nodo inicial o el primero en la cola de prioridades.
2. Expandir el nodo tomado, obtener sus nodos hijos (siguiente estación y conexiones), obtener sus valores heurísticos y añadirlos a la cola de prioridades.
3. Ir al paso 1 hasta que el nodo inicial sea el nodo destino.
4. Encontrado el destino, realizar "brack-tracking" con el camino encontrado.

El método *a_star()* regresa una lista con todas las estaciones para llegar hasta su destino, esta lista se guarda en la memoria de casos para ser utilizada en el futuro.

3. Mini-Router

Para llevar a cabo el Mini-Router, usamos la estrategia de Razonamiento Basado en Casos. En primer lugar definimos un caso, que es una ruta previamente calculada por el algoritmo A*, en forma de una lista con el nombre de las estaciones. Después, cada que se haga otra consulta, es decir, cada que el usuario quisiera calcular una ruta nueva, se revisa en la memoria de casos si hay un caso similar. En caso afirmativo se toma ese caso y se adapta a la consulta que haga el usuario y se regresa la ruta correcta.

A medida que fuera más utilizado el programa, nos encontraríamos con menos casos en los que fuera necesario evaluar la ruta en el A* ya que habría sido previamente evaluada y serían guardados en la memoria de casos.

Fig. 2 Memoria de casos

Para volver el mini-router más útil para el usuario, hemos programado un método que al dar sus coordenadas actuales encuentra la estación más cercana y que pueda comenzar su trayecto. El usuario puede elegir darle al sistema unas coordenadas de salida y llegada o si prefiere los nombres de las estaciones. El predicado *closestStation* recorre el nombre de todas las estaciones, calcula la norma entre las coordenadas que recibe y la estación. Va guardando la norma más pequeña y al acabar regresa el nombre de la estación más cercana. Con el nombre de las estaciones ya puede revisar si hay un caso utilizable o aplicar el A*.

4. Usabilidad

Al correr la terminal el programa pregunta el origen y la llegada del trayecto del usuario. Después de ello el sistema arroja una ruta en orden de primero a final con flechas que indican la el sentido del trayecto. Si el usuario desea calcular otro viaje responde “sí.” y el programa vuelve a preguntarle las estaciones. Podemos observar el funcionamiento en la figura 4. Si el usuario llega a dar nombres incorrectos de estaciones el programa lo detecta y vuelve a preguntar la estación.

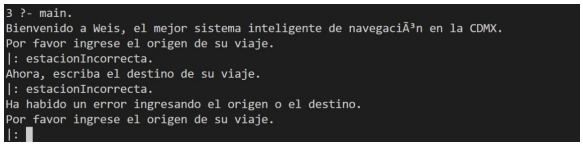

Fig. 3 Estación incorrecta

III. RESULTADOS

En las pruebas nos encontramos con un sistema de fácil uso, capaz de encontrar rápida y correctamente el camino óptimo entre dos estaciones de metro o metrobús.

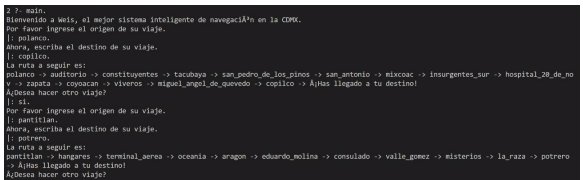

Fig.4 Ejecución del programa

Como se puede observar en la figura anterior se imprimen claramente las estaciones presentes en el recorrido.

IV. TRABAJO FUTURO

Una vez comprendido el funcionamiento y desarrollada la solución para un dominio “pequeño”, gracias al diseño de la misma, es posible ampliar aún más el dominio de conocimiento, integrando otros sistemas de transporte en la ciudad como:

- Ecobici,
- RTP,
- Transportes eléctricos,
- Cablebús,
- Tren suburbano, entre otros.

También es posible eficientar la recuperación y adaptación de casos para encontrar una ruta aún más óptima. Además se pueden mejorar aspectos de la heurística para minimizar factores como:

- Tiempo de traslado total,
- Número de estaciones recorridas,
- Número de transbordos,
- Número de cambios de sistema,
- Costo del recorrido,
- Horas ‘pico’, entre otros.

V. CONCLUSIONES

Al final del proyecto los resultados fueron satisfactorios, además de que se logró afianzar los conocimientos obtenidos en clase sobre Prolog, mini-router y búsqueda A*. También se pudo observar y comprender la eficiencia que presenta la la implementación de un mini-router en conjunto con la búsqueda A*.

Usando por primera vez la memoria de casos fue posible darse cuenta que era lo más eficiente al trazar una ruta, ya que toda la parte pesada de esta, es decir, la búsqueda A*, había sido hecha previamente y solamente era necesario identificar el caso previo útil. Era posible asumir que en la memoria de casos se encontraría la ruta más eficiente gracias a que existía en la memoria tras haber sido obtenida del A*, que siempre te regresaría la ruta más eficiente, evitando errores.

Al comenzar el proyecto fue posible observar que funcionaba a la perfección con la búsqueda A*. El usuario siempre obtendría la ruta más eficiente, pero de la misma manera era posible observar que no necesariamente era lo más eficiente para la

computadora. Bajo este razonamiento es que se decide utilizar una memoria de casos, no hay necesidad de hacer que el programa evalúe nuevamente un trayecto si ya lo hizo previamente. Es claro que la combinación de los diferentes métodos puede ser contraproducente, como puede llegar a ser el caso en el que revisar la memoria de casos tome más tiempo que la búsqueda A*, es por esto que se necesita hacer un balance entre ambos para asegurarse que nunca crezca excesivamente la memoria.

V. REFERENCIAS

1.- Russel, S and Norvig, P, Artificial Intelligence: A Modern Approach, Prentice Hall, 2010.
2.- Agencia Digital de Inovación Pública CDMX - https://datos.cdmx.gob.mx/pages/home/
3.- Movilidad Integrada de la Ciudad de México - https://semovi.cdmx.gob.mx