Seguimiento 2

Seguimiento tarea 2

Daniel Tomas Gallego

Objetivo:

El objetivo de la tarea 2 es definir formalmente el problema de buscar una ruta desde un punto de partida que debe visitar un conjunto de puntos de interés (nodos) en cualquier orden, basado en un grafo creado a partir de un archivo GraphML.
También se asume que ya se ha implementado una clase Estado, que incluye el espacio de estados, el estado inicial, la función sucesora y la función objetivo, que se utilizarán como la estructura básica para los algoritmos de búsqueda en tareas posteriores.

Diseño del código ejecutado

Clase Estado (estado.py)
La clase Estado es un modelo de un estado de un problema, que se define por el nodo actual y la lista de nodos a visitar. Sus principales características son:
• Inicialización: Se le da current_node (id del nodo donde se encuentra actualmente el agente) y places_to_visit (lista de nodos pendientes en el orden de corto a largo).
• Representación: la función make_str genera una cadena compacta sin espacios en la forma (current_node,[places_to_visit]), que satisface las condiciones de uniformidad.
• Identificador Único: La función id_state utiliza la función de hash MD5 para devolver un valor hash único de la representación en cadena, para realizar búsquedas rápidas en estructuras indexadas como diccionarios.
• Función Sucesora: La función successors construye una lista de tuplas (acción, nuevo estado, costo) para cada vecino del estado actual utilizando la lista de adyacencia. Si un vecino está en places_to_visit, entonces se elimina de la lista en el nuevo estado. Los sucesores se ordenan por el id del nodo de destino.
• Función Objetivo: la función is_goal determina si no hay nodos restantes por visitar (len(places_to_visit) == 0).

Integración del Grafo (main.py)
El código en general utiliza la clase Estado y, a su vez, la función parse_graphml de creargrafo.py para:
• Cargar el grafo desde CR_Capital.graphml.
• Comenzar creando un estado inicial que consiste en un nodo inicial ('70') y una lista de nodos restantes (['68', '40', '50', '300']).
• Generar un archivo test2.txt que muestre:
– La representación del estado inicial.
– Su identificador (hash MD5).
– Si el estado es un objetivo (si no hay más nodos por visitar).
– La lista de sucesores junto con sus acciones, estados que son el resultado y costos.

Una Clase GraphMLHandler (creargrafo.py)
La clase GraphMLHandler (discutida en la Tarea 1) sigue siendo crucial, ya que nos proporciona la lista de adyacencia que consultamos con la función successors en la clase Estado. En el sistema SAX, el grafo es accesible en memoria para ayudar en la generación de estados sucesores.

Aspectos destacados

• Espacio de Estado: La clase Estado agrupa de manera sucinta y eficiente el nodo actual y los nodos restantes por visitar en un formato correcto/ordenado.
• Función Sucesora: La generación ordenada de sucesores garantiza regularidad y trazabilidad en futuras técnicas de búsqueda.
• Identificador único: El uso de MD5 asegura que los estados puedan ser comparados o almacenados en estructuras tipo índice sin duplicados.

• Enlace con el Grafo: La manipulación con la lista de adyacencia obtenida en la Tarea 1 permite la posibilidad de simular transiciones reales a partir de las conexiones que tiene el grafo Ciudad Real.
• Flexibilidad: El método es flexible en el sentido de que el estado inicial y los nodos objetivo pueden ser modificados para corresponder a los requisitos del problema.

Conclusión

La Tarea 2 define apropiadamente el problema de la tarea de búsqueda de rutas, mediante la implementación de la clase Estado, que incluye la siguiente información: el espacio de estados, el estado inicial, la función sucesora y la función objetivo.
La conexión con el grafo del Ejercicio 1 y la creación de un archivo de salida con los resultados preparan todo para comenzar a trabajar en algoritmos de búsqueda que resuelvan el problema de encontrar una ruta óptima en el marco de los datos de OpenStreetMap.