Seguimiento 4

Seguimiento tarea 4

Daniel Tomas Gallego

Objetivo:

En esta tarea, el laboratorio se completa implementando la estrategia de búsqueda A* en el algoritmo creado en la Tarea 3, utilizando 2 heurísticas:
- Una siguiendo la distancia euclidiana (Heurística)
- La otra la longitud mínima del arco del grafo (Harco).

Estas heurísticas predicen el costo de alcanzar un estado objetivo y guían la búsqueda a través de un grafo que corresponde a los nodos visitados por el entretenedor en el grafo de OpenStreetMap.

Descripción del Código Implementacion:

Clase Problema (problema.py).
La clase Problema es responsable de configurar el problema:
• Leer un grafo de un archivo GraphML con Graph y xml.sax.
• Estado inicial: (S, s0, G1, …, Gk ≡ {g1, …, gk})
• s0 es el nodo desde el cual comenzar.
• Calcula D1, que es la distancia más corta entre dos nodos objetivo en términos de distancia euclidiana (para usar en la heurística euclidiana).
• Define el método es_objetivo, utilizado para probar si un estado es un objetivo (nodos restantes por visitar = 0).

Clase Estado (estado.py).
La clase Estado es un modelo del estado del problema:
• Lleva un seguimiento de nodo_actual y la lista de nodos_por_visitar que está ordenada.
• Crea una cadena única con MD5 para prevenir entradas duplicadas.
• Tiene un método obtener_sucesores que produce estados sucesores dados los vecinos del grafo, de modo que, si son objetivos, se eliminan de la lista.

Clase NodoBusqueda (arbolbusqueda.py).
SearchNode es una clase que modela un nodo del árbol de búsqueda:
• Consiste en: id, padre, estado, valor, profundidad, costo, heurística, acción.
• Implementa los métodos obtener_camino (devuelve el camino desde la raíz) y __lt__ (para ordenar nodos en la frontera) según el valor y el id.

Clase Grafo y Clase Arista (creargrafo.py, arista.py).
La clase Grafo (heredada de xml.sax.ContentHandler y Arista), así como Arista del archivo GraphML:
• Grafo genera listas de nodos, aristas, adyacencias, mientras almacena la mínima distancia del grafo.
• Arista guarda algunas de las etiquetas de OpenStreetMap como fuente, destino, distancia y otras.

Funciones Heurísticas (utilidades.py).
Las heurísticas se implementan en el módulo py:
• heuristica_euclidea: Devuelve min(D1, D2) * k, donde D1 es la distancia mínima entre los nodos objetivo, D2 es la distancia mínima a un nodo por visitar desde el nodo actual, y k es el número de nodos que quedan por visitar.
• heuristica_arco_minimo : A1 * k, con A1 la longitud mínima de un arco en el grafo.
• calcular_valor: Establece el valor de los nodos según la estrategia (BFS, DFS, costo uniforme, A*).

Algoritmo de búsqueda (algoritmoBusqueda.py).
La función realizar_busqueda utiliza A* y algo de lógica:
• Mantiene la frontera con una cola de prioridad (heapq) ordenada por el valor del nodo.
• Lleva un seguimiento de los estados recorridos utilizando un diccionario basado en un hash MD5.
• Expande nodos con límite de profundidad y calcula el valor h basado en el tipo de heurística de entrada.
• Devuelve la trayectoria de solución si se alcanza un estado objetivo.

Integración (main.py).
El script main.py orquesta la ejecución:
• Verifica la estrategia (bfs, dfs, costo uniforme, A) y la heurística (euclidiana, minimum_arc).
• Solo especifica qué heurística se utiliza.
• Si se pueden encontrar soluciones, informa el camino tomado para encontrarla; de lo contrario, se rinde.
• Construye un objeto Problema, luego invoca el método de búsqueda, imprime el camino o informa que no se encontró solución.

Aspectos destacados:
• Heurísticas permisibles: Heurística estima el costo (su valor A*) con la distancia en línea recta, y Harco utiliza la distancia mínima según el grafo, garantizando la optimalidad en A*.
• Complejidad: Para hacer la búsqueda de manera eficiente, hemos utilizado una cola de prioridad en la frontera y un diccionario para los estados explotados.
• Flexibilidad: El código está implementado para múltiples heurísticas, estrategias y es fácil de aplicar a otros problemas.
• Reutilización: Las estructuras utilizadas en las clases Problema, Estado y Grafo son las mismas que las de las tareas anteriores.

Conclusión.
El laboratorio 4 logra esto al introducir los componentes heurísticos Heurística y Harco a la combinación A*, que optimiza la búsqueda de rutas en el grafo de OpenStreetMap.