Tarea 3

Universidad de Castilla La Mancha

Sistemas inteligentes

Daniel Tomás Gallego
Oussama Bolbaroud

Introducción:

En este trabajo implementamos un algoritmo de búsqueda general para explorar espacios de estados utilizando estrategias como búsqueda de amplitud, profundidad y costo uniforme. Para ello, se definen elementos clave como los nodos de árbol de búsqueda, limites ordenados y conjuntos de estados de acceso para garantizar un procesamiento eficiente y estructurado. Además, se incorpora un sistema para cargar datos iniciales de archivos XML en directorios, demostrando la funcionalidad del algoritmo a diferentes escenarios. Este documento detalla el desarrollo del algoritmo, sus componentes y los resultados obtenidos.

Código:

1.Clase Nodo

Representa un estado del grafo y almacena información relevante como:

- Estado actual
- Nodo padre
- Acción que llevo al estado actual
- Costo acumulado
- Heurística
- Profundidad en el grafo
- Id y valor según la estrategia de búsqueda

Métodos clave:

- Camino(): Reconstruye el camino desde el nodo inicial hasta el nodo actual.
- __repr__: Representa el nodo de forma legible, con un identificador único basado en el estado.

2. Clase Frontera

Maneja los nodos en la frontera usando una cola de prioridad (heapq), lo que permite extraer nodos con el menor valor según la estrategia seleccionada.

Métodos:

- insertar(nodo): Agrega un nodo a la frontera
- extraer(): Extrae el nodo con el menor valor de prioridad.
- vacia(): Verifica si la frontera está vacía.

3. Clase EstadosVisitados

Almacene un conjunto de estados visitados para evitar bucles.

Métodos:

- agregar(estado): Marca un estado como visitado.
- contiene(estado): Verifica si un estado ya fue visitado.

4. Clase Problema

Cargue datos desde un archivo GraphML y defina el problema en función de:

- Nodos y aristas del grafo.
- Estado inicial y objetivo.
- Generación de sucesores para un estado dado.

Métodos clave:

- es_objetivo(estado): Verifica si un estado es el objetivo.
- cargar_datos(): Lee el archivo GraphML y carga nodos y aristas.
- sucesores(estado): Genera sucesores del estado actual.

5. Función algoritmo_busqueda

Implemente algoritmos generales para el recorrido de gráficos utilizando diferentes estrategias:

- Anchura: Se priorizan nodos con menor profundidad.
- Profundidad: Se priorizan nodos con mayor profundidad.
- Costo Uniforme: Se prioriza el nodo con el menor costo acumulado.

Ejemplo de Ejecución

```
Expandiendo nodo: [6995][23.00,8e805e,6884,803->806,23,0.00,23.00]
Generando sucesor: Acción=806->808, Estado=808, Costo=1.0
Generando sucesor: Acción=806->812, Estado=812, Costo=1.0
Expandiendo nodo: [6996][23.00,150cf7,6887,790->789,23,0.00,23.00]
Se ignora nodo [6996][23.00,150cf7,6887,790->789,23,0.00,23.00] porque ya fue visitado.
Expandiendo nodo: [6997][23.00,d6fc33,6888,784->777,23,0.00,23.00]
Generando sucesor: Acción=777->776, Estado=776, Costo=1.0
Generando sucesor: Acción=777->784, Estado=784, Costo=1.0
Generando sucesor: Acción=777->780, Estado=780, Costo=1.0
Expandiendo nodo: [6998][23.00,1e9eb0,6888,784->782,23,0.00,23.00]
Generando sucesor: Acción=782->781, Estado=781, Costo=1.0
Generando sucesor: Acción=782->783, Estado=783, Costo=1.0
Generando sucesor: Acción=782->780, Estado=780, Costo=1.0
Generando sucesor: Acción=782->784, Estado=784, Costo=1.0
Expandiendo nodo: [6999][23.00,4fe1cf,6888,784->791,23,0.00,23.00]
Generando sucesor: Acción=791->792, Estado=792, Costo=1.0
Expandiendo nodo: [7000][23.00,161c65,6888,784->785,23,0.00,23.00]
Se ignora nodo [7000][23.00,161c65,6888,784->785,23,0.00,23.00] porque ya fue visitado.
```

Conclusión:

El código carga un archivo GraphML que contiene información sobre un grafo, define un estado inicial y un estado objetivo, y ejecuta el algoritmo con diferentes estrategias de búsqueda, mostrando los pasos y el camino solución.