ESTRATEGIAS DE BÚSQUEDA

Asignatura:
Sistemas Inteligentes
Nombre y apellidos:
Yago García Araújo
Brais Varela Sieiro
Fecha:
Marzo 2022

Contenido

1. **Ejercicio 1**
   a) Se crea una clase “Nodo” donde definimos el constructor, que tiene como atributos la acción a tomar, su estado y el nodo padre. Implementamos un método “getEstado” y “getPadre”, que devuelve el estado y el padre respectivamente.

      En la clase “Estrategia4” modificamos la lista que venía por defecto para hacer una lista de nodos, y aplicamos el método “reconstruye_sol”, que devuelve una lista de nodos con los estados explorados.

   b) Implementamos una clase denominada “EstrategiaBusquedaGrafo”, que a diferencia de “Estrategia4”, se crea una lista para guardar los estados explorados y una pila para almacenar los estados de los nodos no explorados.

2. **Ejercicio 2**
   a) Se crea una clase “ProblemaCuadradoMagico”, la cual hereda de “ProblemaBusqueda”. En la clase hija, se detalla un método “esMeta” en el que se recorre la matriz de diversas formas para comprobar si el resultado de la suma de sus casillas según los criterios requeridos en el enunciado es correcto.

      En las clases “BusquedaAnchura” y “BusquedaProfundidad” se implementan la búsqueda en anchura y en profundidad, respectivamente. En la primera estrategia de búsqueda se emplea una cola para almacenar los nodos frontera y una lista para los nodos explorados, y en la segunda una pila para los frontera y otra lista para los explorados.

      La búsqueda en profundidad es más adecuada, ya que emplea menos pasos que su alternativa para llegar al estado meta. Empleando como matriz inicial {{4,9,2},{3,5,0},{0,1,0}}, cuenta 166 nodos expandidos y 336 creados, en cambio, su homóloga determina 662 expandidos y 1944 creados.

   b) En la clase “HeuristicaCuadradoMagico”, se ha decidido implementar como heurística para resolver el problema del cuadrado mágico un método que consiste en comprobar si la suma de las n filas, columnas y diagonales exceden el resultado de la operación $n * \frac{n*n+1}{2}$, o si para completar las casillas vacías habría que emplear un número superior a $n * n$. Si estas condiciones resultan ciertas, la ejecución del programa se detendría. Es una implementación robusta dado que comprueba los posibles casos erróneos para evitar erratas.

      En la clase “EstrategiaAasterisco” se emplean métodos detallados en la interfaz “EstrategiaBusquedaInformada”. Emplea un funcionamiento similar a los algoritmos de búsqueda en anchura y profundidad, sin embargo, incluye el coste del camino empleado (se ha modificado la clase Nodo, en la cual se ha incluido como nuevo atributo) y el valor de la función f.

c) Para resolver el tercer ejemplo detallado en el enunciado, el programa requiere de cierto tiempo, dado que es una matriz de mayor tamaño con un estado inicial muy alejado del final.