PRÁCTICA 1: PACMAN

Algoritmos de búsqueda

19 DE NOVIEMBRE DE 2018

UNIVERSITAT DE LLEIDA

Marc Alba Cerveró i Francesc Contreras Pérez

1. Implementación de los algoritmos

Para implementar los algoritmos hemos hecho uso de la clase *PriorityQueue* ya implementada en el fichero *útil.py* y la creación de una nueva clase *Node* creada por nosotros.

La clase *Node* constara de un constructor que tendrá los siguientes atributos: *parent, state, action ,cost* , esta clase también constara de dos métodos *__eq__* y *getRootPath*. El método *__eq__* comprobara si dos objetos de la clase *Node* tienen el mismo estado y retornara un valor booleano, en el método *getRootPhat* le pasaremos como parámetro un objeto de la clase *Node* con la finalidad de retornar una lista con las acciones que han sido necesarias hasta encontrar el objetivo.

En caso de que un nodo expandido ya pertenezca a la cola con prioridad, *fringe*. Es decir, sus estados son iguales, debe actualizarse la prioridad de este nodo en caso de tener un coste menor. Usaremos el método *Update*, ya implementado.

El método *Update* busca el nodo en su cola, si lo encuentra compara prioridades, si es menor lo actualiza, en caso contrario no modifica la cola.

2. Evaluación experimental

2.1 UCS

	TinyMaze	**SmallMaze**	**MediumMaze**	**BigMaze**
Cost	8	19	68	210
Expanded	15	92	269	620

2.2 Best-H first

	TinyMaze	**SmallMaze**	**MediumMaze**	**BigMaze**
Cost Manhattan	8	29	74	210
Cost Euclidean	8	29	152	210
Expanded Manhattan	8	39	78	466
Expanded Euclidean	8	39	159	471

2.3 A*

	TinyMaze	SmallMaze	MediumMaze	BigMaze
Cost Manhattan	8	19	68	210
Cost Euclidean	8	19	68	210
Expanded Manhattan	14	53	221	549
Expanded Euclidean	13	56	236	557

2.4 Análisis

Para evaluar los algoritmos utilizaremos las tablas mostradas, estas tablas muestran los nodos que se expanden un algoritmo según el mapa que se utiliza y la heurística utilizada.

En los mapas utilizados el algoritmo con menor número de nodos expandidos es BFS-H, esto es debido ya que el algoritmo trata de acercarse en cada paso tanto como puede al objetivo.

El algoritmo de búsqueda BFS-H visita el siguiente estado basado en la función heurística f (n) = h(n) con el valor heurístico más bajo. No considera el costo del camino a ese estado particular. Todo lo que importa es lo que el siguiente estado del estado actual tiene las heurísticas más bajas.

El algoritmo de búsqueda A * visita el siguiente estado según las estadísticas f (n) = h(n) + g(n) donde *h* componente es la misma heurística aplicada que en BFS, pero *g* componente es la ruta desde el estado inicial al estado particular. Por lo tanto, no elige el siguiente estado solo con el valor de heurística más bajo, sino uno que otorgue el valor más bajo al considerar su heurística y el costo de llegar a ese estado.

En conclusión, según las tablas mostrada anteriormente el algoritmo con mayor eficiencia es el BFS-H ya que no expande tantos nodos en los mapas analizados. No obstante, en el mapa medio este algoritmo tiene un coste más elevado que los demás. También es importante la utilización de las heurísticas en los algoritmos BFS-H y A* ya que en todos los casos la utilización de la heurística euclídea se obtiene unos resultados menos favorables que utilizando la heurística de Manhattan.

3. Klotski

- **Estado inicial:**
  Un tablero de N X M dimensiones el cual estará completado por K figuras rectangulares, con la obligación de dejar dos espacios libres disponibles y una pieza objetivo.

- **Acciones:**
  Desplazamiento horizontal o vertical de las figuras.

- **Modelo de transición:**
  Cada acción tiene su efecto esperado siempre y cuando haya un espacio vacío para hacer el desplazamiento.

- **Test de objetivo:**
  Se determina cuando la pieza objetivo P llega a su posición objetivo determinada en el tablero.

- **Coste de un camino:**
  Definimos el coste de cada desplazamiento igual a 1, donde el coste total vendrá dado por el sumatorio de los costes del total de movimientos dados para resolver el problema.

- **Heurística:**
  La heurística utilizada consistirá en devolver el coste que hay desde la posición actual hasta el objetivo en el tablero (distancia de Manhattan). Y será admisible ya que nunca habrá una sobrestimación en obtener este valor.

- **Algoritmo de búsqueda:**
  Para resolver el puzle Klotski el algoritmo de búsqueda que más se ajusta para encontrar la solución de forma más optima es haciendo uso del algoritmo BFS-H, ya que este algoritmo nos ayudara a encontrar la solución más corta suponiendo que un bloque en un espacio cuenta como un movimiento.