Daniel Berdie Pardo y Alex Satour Perucho
Inteligencia Artificial

Práctica 1

Búsqueda

En esta práctica se pedia la implementacion de los algoritmos de busqueda explicados en clase aplicados al mítico videojuego *Pacman,* así como la codificación de una modificación del planteamiento original y una heurística para este.

Para empezar, modificamos la estructura de la clase nodo para añadir la función *path()* que nos ayudará a obtener el camino correspondiente a la solución.

En todos los algoritmos usamos dos estructuras de datos como mínimo, un diccionario donde guardamos los nodos visitados llamada *generated* y una pila/cola/lista con prioridad llamada *fringe* donde se guardan los que aún no han sido visitados.

En el caso de los algoritmos DFS y BFS se usa una pila y una cola respectivamente para que se recorran los diferentes nodos de manera correcta.

En el caso del UCS y el A*, también deberemos saber si el nodo ha sido expandido anteriormente, así que se añade un dato más en cada entrada del diccionario *gen* en la que con un 0 indicaremos que el nodo correspondiente a esa *key* aún no ha sido expandido y con un 1, lo contrario.

En el algoritmo UCS, asignamos la prioridad con el coste acumulado más el de ese mismo nodo y en el caso del A* también sumaremos el valor de una determinada heurística.

La representación del problema *Corners Problem* se ha hecho considerando que los estados contienen el nodo actual y, además, las esquinas que aún tenemos que visitar. De esta manera llegaremos al estado objetivo cuando la lista que contiene las esquinas vacías esté vacía.

Para obtener los sucesores de un nodo, simplemente se generan aquellos con los que el resultado de aplicar algún movimiento de los posibles (*North, West, East, South*) no se corresponda con la posición de una pared. Si no se corresponde con alguna pared, se mirará si con este movimiento se estará en alguna de las esquinas, y si lo estará, se eliminará el *corner* correspondiente de la lista. Si no corresponde a ninguno, se le asignara la misma lista que su nodo antecesor.

La heurística implementada se basa en considerar la distancia al *corner* más lejano, ya que de esta manera nunca sobreestimaremos esta información, porque en el mejor de los casos, pasaremos por los demás puntos antes de llegar a la esquina más lejana y esta distancia seguirá siendo la mínima distancia a recorrer posible para que el problema se resuelva.