Inteligencia Artificial: PL1.B

Búsqueda A*

Para crear la función A* Search simplemente hemos modificado la implementación del Uniform Cost Search sumándole a la prioridad asignada a la expansión de un estado el valor calculado mediante un heurístico.

Buscando todas las esquinas

La única dificultad de implementar un problema en el que pacman ha de visitar las cuatro esquinas es el cambio de la representación de un estado. Hasta ahora, bastaba con que un estado estuviera codificado mediante la posición de pacman, pero ahora, también ha de indicar por qué esquinas ha pasado. Es importante notar que añadir un entero que exprese por cuantas esquinas ha pasado no es una solución, ya que pacman podría limitarse a pasar cuatro veces por la misma esquina. Por ello, hemos representado un estado mediante la posición de pacman y cuatro booleanos que indican si este ha pasado por la esquina correspondiente.

Buscando todas las esquinas: Heurístico

Tras probar con varios heurísticos, el que mejores resultados nos ha dado ha sido la distancia Manhattan máxima a una esquina sin visitar. Es admisible ya que es la distancia más corta posible entre pacman y la esquina más lejana, la cual va a tener que visitar.

Comiendo todos los puntos

Tras una no pequeña cantidad de tiempo invertido en probar varios heurísticos, el que mejores resultados nos ha dado ha sido el siguiente:

La distancia Manhattan entre pacman y el punto más cercano más la distancia entre este punto y su punto más lejano. No es posible añadir más puntos intermedios ya que no se puede asegurar que pacman no pasaría a través de ellos en el camino al punto más cercano. Es importante recordar que el punto que se considera más cercano es aquel que minimiza la distancia Manhattan pero que sin embargo, no tiene porque corresponder con el punto al que pacman puede llegar con el menor número de pasos teniendo en cuenta las paredes del laberinto.

Iñigo Mendizabal y Martín Jiménez

Búsqueda subóptima

Para este caso, definimos un problema para el que definimos su estado final, que es cuando hay comida en el punto en el que está, este problema nos ayuda a encontrar un camino para cualquier comida. Luego para la búsqueda del camino más corto a una comida utilizamos BFS(búsqueda en anchura) para encontrar la comida más cercana.