Práctica 2: Representación y búsqueda en la librería AIMA
Inteligencia Artificial

Autores:
José Javier Cortés Tejada
Pedro David González Vázquez

Cuestión 2: 8-puzzle con ejemplo extremo

	Coste del camino	Nodos expandidos	Tam. de cola	Tam. de cola (máx)
Anchura	30	181058	365	24048
Voraz (MT [1])	116	803	525	526
Voraz (MAN [2])	66	211	142	143
A* (MT)	30	95920	23489	23530
A* (MAN)	30	10439	5101	5102

En el algoritmo voraz, la diferencia de los datos obtenidos viene dada por la heurística que se está aplicando. En el caso de la heurística MT se cuentan las casillas mal posicionadas y se procede a intercambiarlas entre sí, lo que implica que el número de nodos expandido sea mayor, pues cada nodo se puede intercambiar por $n-1$ nodos (siendo n el número de fichas mal colocadas), aunque esto no es tan descabellado pues el método voraz solo busca la primera solución válida.

Por otro lado, el algoritmo voraz con heurística MAN trata de expandir el nodo con menor coste, es decir, solo expande los necesarios para llegar a la solución con menor coste, lo cual evita que como en el caso anterior expandamos tantos nodos, por lo tanto el coste del camino es menor que con MT.

Con A* se nos presenta la misma situación que con el algoritmo voraz; aunque solo que tomamos como referencia los nodos en expansión, no el coste del camino, pues al ser la mejor opción, será igual en todos los casos (siempre y cuando las heurísticas sean optimistas). Así pues , nos fijaremos en MT tiene muchos más nodos (cerca de 9 veces más) expandidos, pues dado que cada casilla puede intercambiarse con $n-1$ casillas (n = número casillas desconocidas totales) por lo tanto, la exploración se ejecuta sobre todas las casillas en el caso extremo y todas sus permutaciones, sin embargo, con MAN al haber evaluado la distancia y no los nodos expandir, muchos caminos no se expandirán al haber encontrado

[1] *Misplaced Tille Heuristic*
[2] *Manhattan Heuristic*

una solución mas óptima para la misma configuración.

En general, vemos que si lo que necesitamos son optimizaciones de memoria, pero buscando el camino mínimo, la mejor situación (en el caso peor) es la búsqueda en anchura, pues su tamaño de cola es bastante menor que en el resto de casos ya que el número de nodos en cola son los del nivel en curso. En cambio, buscando tiempos reducidos, aunque la opción no sea la mas óptima, el mejor algoritmo sería el voraz, pues sus tiempos de resolución son mucho menores, sacrificando el tiempo de la tarea realizada (tarda más en encontrar una solución a cambio de que sea más rápido).

Usaremos A* cuando necesitemos la solución la más óptima en tiempo de resolución y más rápida en ejecución, aunque también deberíamos darnos cuenta de que en este caso no compensa usar A* pues la distancia de una ejecución a otra no es muy pronunciada respecto de un algoritmo voraz, ya que hay una diferencia de aproximadamente 2*coste del camino de A* (usando el mejor coste de ambas heurísticas).

La diferencia entre ambas heurísticas (sin tener el cuenta el algoritmo aplicado) es bastante obvia; en los casos peores nos encontramos con que MAN ofrece mejores soluciones tanto a nivel memoria como en tiempo de ejecución de la tarea, ya que el número de nodos al expandir será bastante menor dado que no esta en coste factorial (uno de las peores situaciones).

Cuestión 3: búsqueda de caminos (Arad - Budapest)

	Coste del camino	Nodos expandidos	Tam. de cola	Tam. de cola (máx)
Profundidad	733	10	1	3
Anchura	450	5	3	5
A* (DR [3])	418	5	4	6

En estos casos la memoria no debería preocuparnos pues la diferencia de nodos en muy pequeña (en casos más complejos, si se diesen grandes diferencias entre estas nos iríamos por profundidad o anchura), pero aún así el mejor coste es el de A*, que en este caso, es el único que asegura un camino relativamente óptimo pues el hecho de que anchura tenga un coste similar es meramente selección del destino y el origen, de tal manera que en la simulación siguiente apreciamos una gran diferencia entre el coste de A* y anchura:

Coste del camino desde Arad hasta Neamd:

- 1939 profundidad.
- 856 anchura.
- 824 A*.

En este caso tenemos un mapa expandido (más grande, con más nodos), de manera que compensa tener un A*, pues en anchura (muy probablemente) tendríamos los mismos problemas que en profundidad.

[3] *Rect Line Heuristic*

Cuestión 4: búsqueda de caminos (Blaustein - Libie)

Los ejemplos de ejecución hacen referencia a las ciudades Blaustein y Libie.

	Coste del camino	Nodos expandidos	Tam. de cola	Tam. de cola (máx)
Profundidad	558	25984	3933	3938
A* (DR)	17	23204	508	574
Coste uniforme	16	62739	228	336

Como podemos observar, la búsqueda en profundidad demuestra ser muy ineficiente a la hora de encontrar un camino corto para ir de A a B con mapas muy complejos (siendo este el caso más común), pues hay una diferencia de más de 500 km (no ahorraríamos en gasolina). Incluso en este caso no ahorraríamos tampoco en nodos a expandir ni en el tamaño de la cola, pues al final, aunque vayas por el primer camino que encuentres terminarás entrando a estados erróneos y, consecuentemente, expandirás más nodos que son innecesarios.

En cambio, en A* distancia recta (evaluando la distancia como heurística) nos fijamos en que el coste, pese a no ser el mejor, empieza a ser mucho más aceptable que en el caso de la profundidad. Sin embargo, pese a ser aceptable no es la mejor respecto del tamaño de la cola y el máximo de ésta; aun así, si es la más rápida en encontrar la solución óptima. Por ello, este algoritmo podría ser usado en el cálculo de rutas complejas con bajo tiempo de reacción.

El coste uniforme nos permite encontrar una ruta que minimice la distancia recorrida, aunque para ello, en este caso, necesitaríamos un tiempo casi tres veces mayor para el cálculo de la ruta. Dicha observación deriba en el siguiente razonamiento: el coste uniforme permitirá calcular la mejor ruta al comienzo del viaje, procesando dicha información mientras el conductor enciende el coche o se pone en marcha, pero no será válida para recalcular la ruta en medio de un trayecto, dicho recálculo podría ser debido a distintos problemas; carreteras cortadas, accidentes, atascos u obras.

Cuestión 5: definición de estados y operadores en 8-puzzle

La definición de estado está en el fichero *State.java* y queda representado por un vector de 9 enteros donde cada posición del mismo corresponde con el valor de una pieza:

$$\text{state} = \text{new int}[]\ \ 5, 4, 0, 6, 1, 8, 7, 3, 2\ ;$$

Los operadores están también representados en el fichero *State.java*. En el caso del 8-puzzle son definidos 4 operadores: *UP, DOWN, LEFT y RIGHT.*

La definición básica de las heurísticas está en el fichero *HeuristicFunction.java* dentro del paquete *aima.core.search.framework*, y la que implementa el número de fichas mal colocadas se encuentra en el fichero *MisplacedTilleHeuristicFunction.java*, dentro del paquete *aima.core.environment.eightpuzzle*.

Cuestión 6: definición de estados y operadores en la búsqueda de caminos

El estado queda definido en el fichero *AgentAppFrame.java*, juntando el indice de la ciudad y su nombre para defir un mapa, y un índice del actual (el método donde se define es *selectionChanged*).

La definición de los operadores se encuentran en la misma clase, en el método *RouteFindingAgentController*. Para definir el operador genera distintos posibles caminos y si no encuentra uno genera un aleatorio por el que avanzar.