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

Autores:
José Javier Cortés Tejada
Pedro David González Vázquez

1. 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 H)	116	803	525	526
Voraz (MAN H)	66	211	142	143
A* (MT H)	30	95920	23489	23530
A* (MAN H)	30	10439	5101	5102

Algoritmo voraz: le diferencia de los datos obtenidos en este algoritmo 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, luego el coste del camino es menor que con TMH.

DISCUSION A*: es la misma que voraz, solo que tomamos como ref los nodos en expansion, no el pathCost, pues al ser la mejor opcion, sera igual en todos los casos (siempre y cuando las heristicas sean optimistas) Asi pues , nos fijaremos en que la MTH tiene muchos mas nodos (unas 9 veces mas) expandidos, pues dado que cada casilla puede intercambiarse con $n-1$ casillas (n === numero casillas desc totales), luiego la exploracion se ejecuta sobre todas las casillas en el caso extremo y todas sus permutaciones En MANH, al haber evaluado la diatncia y no los nodos expandir, por lo que hay muchos caminos que no se expandiran al haber encontrado una solucion mas optima para la misma configuracion

DISCUSION GEN: al compara todo, vemos que si lo que necesitamos son optimizaciones de memoria, pero buscando el camino minimo, la mejor simulacion (en el caso peor) es la busqueda en anchura, pues su tamaño de cola es bastante menor que en el resto de casos ya que el numero de nodos en cola son los del nivel en curso. En cambio, buscando tiempos

reducidos aunque la opción no sea la mas optima, el mejor algoritmo sería el voraz, pues sus tiempos de resolución son mucho menores, sacrificando el tiempo de la tarea realizada (lo que das puede tardar mas, pero ser luego mas lenta).

Usando el A*, si necesitamos la mas óptima en tiempo de resolución y que sea la mas rápida en ejecución, también deberíamos darnos cuenta de que en este caso no compensa usar a estrella pues la distancia de una ejecución a otra no es muy pronunciada entre un estrella y un voraz ya que hay una diferencia de aproximadamente 2*pathCost 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 MANH ofrece mejores soluciones tanto a nivel memoria como en tiempo de ejecución de la tarea, ya que el numero de nodos al expandir sera bastante menor en MANH dado que no esta en coste factorial (uno de las peores situaciones)

2. Cuestión 3: búsqueda de caminos

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

Discusión general: en estos casos la memoria no debería preocuparnos pues la diferencia de nodos en muy pequeña (en casos más complejos, si se diesen grande 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, pues viendo la simulación siguiente apreciamos una gran diferencia entre el coste de A* y anchura: Arad – Neamd
1939 prof
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. Cuestión 4: 8-puzzle con ejemplo extremo

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 H)	17	23204	508	574
Coste uniforme	16	62739	228	336

Discusión general: 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 algorítmo podría ser usado en el cálculo de rutas complejas bajo tiempo de reacción.
El coste uniforme nos permite encontrar una ruta que minimice la distancia recorrida, aunque para ello (en este caso concreto) necesitará un tiempo tres veces mayor de procesamiento, lo que no la hace apta para rutas complejas (planificar de la ruta de un coche), pues no es capaz de reaccionar ante un imprevisto a tiempo, por ejemplo: uno de los nuevos Tesla, aún habiendo planificado una ruta puede haber habido un accidente, teniendo que reaccionar rápido ante este estímulo.

4. Cuestión 5: 8-puzzle con ejemplo extremo

La definicion de estado esta en el fichero State.java, y cada clase que la implemente definira su propio estado

Ruta de los operadores: package aima.core.agent;
Action execute(Percept percept);

El estado esta representado por
state = new int[] 5, 4, 0, 6, 1, 8, 7, 3, 2 ;
que indica el valor de cada pieza

Cada heuristica tiene su propia construccion y se van modificando entre ellas

package aima.core.search.framework;

public interface HeuristicFunction

package aima.core.environment.eightpuzzle;

MisplacedTilleHeuristicFunction

5. Cuestión 6: 8-puzzle con ejemplo extremo

Cuadro 1: My caption