INTELIGENCIA ARTIFICIAL

Práctica 2

2018/2019

Jesús Aguas Acín -- 736935@unizar.es

Introducción

El objetivo de esta práctica es realizar experimentos y recopilar información relevante acerca de la eficiencia de distintas búsqueda, informadas y no informadas, aplicadas al juego 8-puzzle. Para ello se mostrará por pantalla el número de nodos generados y el factor de ramificación efectivo b* de cada tipo de búsqueda para distintas profundidades.

Para asegurar la consistencia de los datos se realizarán 100 pruebas distintas, con cada algoritmo de búsqueda y probándolo con soluciones hasta en una profundidad d=24.
Además se generará un estado inicial y final aleatorio en cada una de estas pruebas.

Los algoritmos de búsqueda no informada serán BFS (Breadth First Search) y IDS (Iterative Deepening Search)

Los algoritmos de búsqueda informada serán A* con las heurísticas de fichas descolocadas y Manhattan.

Metodología

El funcionamiento del EightPuzzle ya estaba implementado en *aima.core.environment.eightpuzzle*, por lo que no fue necesario modificar nada de este, excepto el EightPuzzleGoalTest, que fue necesario crear un EightPuzzleGoalTest2 con funciones que permitiesen modificar el estado final.

Las búsquedas informadas y no informadas se encuentran en *aima.core.search.informed* y *aima.core.search.uninformed*, respectivamente. Sin embargo, fue necesario modificar las heurísticas de las búsquedas informadas para adaptarlas según el estado final, además de añadir en el algoritmo de Iterative Deepening Search (IDS) la métrica nodos generados, para poder mostrarlos posteriormente.

Para mostrar los nodos generados fue necesario añadir la métrica de nodos generados en la clase NodeExpander en aima.core.search.framework, y para mostrar el factor de ramificación efectivo fue necesario añadir la clase Biseccion en aima.core.util.math.

Para generar los estados iniciales y finales aleatorios fue necesario añadir la clase GenerateInitialEightPuzzleBoard en aima.gui.demo.search.

Finalmente queda hacer la clase EighPuzzlePract2 en aima.gui.demo.search, que contiene el método main() que mostrará por pantalla los nodos generados y el factor de ramificación efectivo para cada búsqueda en distintas profundidades. Para realizar cada una de las búsquedas desde el main() se llamará a un método *public static int eightPuzzleSearc(Search search, int Depth)*.
El cual realizará la búsqueda especificada en *search* un total de 100 veces con un estado final que se encuentra a *Depth* pasos del estado inicial, ambos generados de forma aleatoria para cada uno de los experimentos. El resultado de dicho método es la media de nodos generados en los 100 experimentos realizados.

A partir de los nodos generados podremos obtener el factor de ramificación llamando al método *public static double getRamificacion(int depth, int GeneratedNodes)* que devuelve un dato de tipo double que equivale el factor de ramificación efectivo b*.

Finalmente se muestra todo por pantalla obteniendo así la siguiente salida:

	Nodos Generados				b*			
d	BFS	IDS	A*h(1)	A*h(2)	BFS	IDS	A*h(1)	A*h(2)
2	7	10	5	5	2,19	2,70	1,79	1,79
3	19	33	9	8	2,26	2,81	1,66	1,58
4	39	93	12	12	2,17	2,79	1,49	1,49
5	69	275	16	15	2,05	2,82	1,42	1,39
6	120	766	25	20	1,98	2,81	1,43	1,36
7	218	2297	33	24	1,95	2,84	1,40	1,31
8	363	5794	47	30	1,91	2,79	1,39	1,29
9	623	17554	72	42	1,88	2,82	1,41	1,30
10	992	47026	100	50	1,84	2,81	1,40	1,28
11	1636	---	179	67	1,82	---	1,44	1,29
12	2736	---	259	89	1,81	---	1,44	1,29
13	4285	---	438	156	1,79	---	1,46	1,33
14	6937	---	615	189	1,77	---	1,46	1,32
15	11063	---	945	291	1,76	---	1,46	1,33
16	17522	---	1513	291	1,75	---	1,47	1,30
17	27630	---	2410	551	1,74	---	1,48	1,34
18	39857	---	3465	676	1,72	---	1,48	1,33
19	63882	---	5384	993	1,71	---	1,48	1,34
20	92408	---	9307	1405	1,69	---	1,49	1,34
21	130696	---	12747	2108	1,68	---	1,49	1,35
22	175442	---	21502	3009	1,66	---	1,50	1,35
23	229512	---	30902	3928	1,64	---	1,49	1,35
24	284931	---	46885	4557	1,62	---	1,49	1,34

Conclusiones

A partir de los resultados obtenidos se puede observar que algoritmo de búsqueda más eficiente para todas las profundidades es el algoritmo de búsqueda A*, concretamente con la heurística Manhattan, ya que esta genera muchos menos nodos que el resto de búsquedas y es la que tiene el menor factor de ramificación. Su valor se aproxima a 1, por lo que podemos afirmar que se trata de una heurística de gran calidad.

También se puede observar que a partir de una profundidad mayor a 10, se deja de realizar búsquedas IDS (Iterative Deepening Search), esto es debido a que el coste en tiempo crece considerablemente con cada nivel de profundidad, hasta el punto de no poderse resolver en un tiempo razonable.