Universidad Zaragoza

INTELIGENCIA ARTIFICIAL (30223)

Grado en Ingeniería Informática

Área de Lenguajes y Sistemas Informáticos

Dpto. Informática e Ingeniería de Sistemas

# Práctica 2 :

# Resolución de problemas y búsqueda.

# Búsqueda informada

## 1. Objetivo de la práctica

Familiarizarse con el código en java para resolver problemas de búsqueda informada y diferenciar sus características. Volvemos a utilizar el código en http://code.google.com/p/aima-java/.

Tu trabajo consistirá básicamente en realizar experimentos y recopilar información relevante de la búsqueda relacionada con aspectos de eficiencia. Realizaremos con el 8-puzzle búsquedas ciegas (BFS e IDS) y búsquedas informadas A* con las heurísticas de fichas descolocadas y Manhatan. En esta práctica vamos a utilizar la búsqueda A*, y a comparar la eficiencia de los algoritmos mostrando el número de nodos generados y el factor de ramificación efectivo b* para distintas profundidades.

## 2. Tareas

Se deben entregar en un zip solos las clases escritas o modificadas, y una memoria con la tabla(s) obtenidas y los comentarios que estimes oportunos.

Notas de las tareas a realizar:

1. Factor de ramificación efectivo con heurísticas del problema del 8 puzle. Se mostrará una tabla con los resultados tal como se muestra en el ejemplo:
    - Debes incluir en las métricas el número de nodos generados. En el código se define la métrica de nodos expandidos, pero no los nodos generados. Mira las notas al final de la práctica para ver que clases tienes que modificar en función de la versión del código que tengas.
    - Necesitarás implementar una clase `Biseccion` en `aima.core.util.math` que permite obtener los ceros de la función $N = b^* (b^{*d} - 1) / (b^* - 1)$ por aproximaciones sucesivas, donde b* es el factor de ramificación efectivo, N el número de nodos generados y d la profundidad de la solución.

- Debes generar 100 experimentos aleatorios de la profundidad deseada y calcular la media de los nodos generados. Puedes utilizar la clase `GenerateInitialEightPuzzleBoard` suministrada para que generar aleatoriamente estados iniciales, y estados finales de la profundidad deseada. El método `random` ejecutad acciones aleatorias desde el estado inicial, donde d es la profundidad deseada sin generar estados repetidos. Aún así, el que se hayan dado d pasos al estado final, no garantiza que no haya caminos más cortos al estado final. Por lo que en los experimentos, o al generar los estados inicial y final, deberás comprobar que la solución óptima es del coste deseado.
- Tendrás que reescribir las clases `ManhattanHeuristicFunction` y `MisplacedTilleHeuristicFunction` para que sean útiles para cualquier estado final.

| | Nodos Generados | | | | b* | | | |
|---|---|---|---|---|---|---|---|---|
| d | BFS | IDS | A*h(1) | A*h(2) | BFS | IDS | A*h(1) | A*h(2) |
| 2 | 8 | 11 | 5 | 5 | 2,37 | 2,85 | 1,79 | 1,79 |
| 3 | 19 | 33 | 10 | 9 | 2,26 | 2,81 | 1,74 | 1,66 |
| 4 | 38 | 96 | 13 | 11 | 2,15 | 2,82 | 1,53 | 1,45 |
| 5 | 68 | 265 | 17 | 14 | 2,04 | 2,80 | 1,44 | 1,37 |
| 6 | 127 | 820 | 25 | 18 | 2,00 | 2,85 | 1,43 | 1,33 |
| 7 | 221 | 2242 | 35 | 23 | 1,95 | 2,83 | 1,41 | 1,30 |
| 8 | 375 | 6344 | 50 | 28 | 1,91 | 2,83 | 1,41 | 1,28 |
| 9 | 634 | 18168 | 72 | 36 | 1,88 | 2,83 | 1,41 | 1,27 |
| 10 | 1024 | 49619 | 108 | 46 | 1,85 | 2,82 | 1,42 | 1,27 |
| 11 | 1635 | --- | 169 | 63 | 1,82 | --- | 1,43 | 1,28 |
| 12 | 2711 | --- | 266 | 84 | 1,81 | --- | 1,45 | 1,28 |
| 13 | 4202 | --- | 425 | 127 | 1,78 | --- | 1,46 | 1,30 |
| 14 | 7095 | --- | 611 | 144 | 1,78 | --- | 1,46 | 1,28 |
| 15 | 11178 | --- | 978 | 193 | 1,76 | --- | 1,47 | 1,29 |
| 16 | 16992 | --- | 1473 | 313 | 1,74 | --- | 1,47 | 1,31 |
| 17 | 27552 | --- | 2385 | 373 | 1,73 | --- | 1,48 | 1,30 |
| 18 | 42609 | --- | 3708 | 524 | 1,72 | --- | 1,48 | 1,31 |
| 19 | 61937 | --- | 5432 | 699 | 1,71 | --- | 1,48 | 1,31 |
| 20 | 91339 | --- | 9033 | 957 | 1,69 | --- | 1,49 | 1,31 |
| 21 | 127078 | --- | 12912 | 1140 | 1,68 | --- | 1,49 | 1,30 |
| 22 | 175503 | --- | 20481 | 1736 | 1,66 | --- | 1,49 | 1,32 |
| 23 | 229049 | --- | 31352 | 2091 | 1,64 | --- | 1,49 | 1,31 |
| 24 | 287542 | --- | 43422 | 3152 | 1,62 | --- | 1,49 | 1,32 |

NOTAS:

En la versión 1.8 hay que añadir la métrica de nodos generados en las clases NodeExpander en aima.core.search.framework e IterativeDeepingingSearch. Toma como modelo la métrica nodos expandidos.

En la versión 1.9 hay que añadir un contador en la clase Node de la misma manera que se contabilizan los nodos expandidos en la clase NodeExpander. Tendrás que añadir el manejo de esta métrica en QueueSearch (aima.core.search.qsearch), y en las clases DepthLimitedSearch e IterativeDeepeningSearch (aima.core.search.uninformed).

OJO. Las heurísticas ManhattanHeuristicFunction y MisplacedTilleHeuristicFunction están pensadas para un único estado objetivo. En este problema tendrás que definir otra clase EightPuzzleGoalTest2 para poder modificar el estado objetivo y las heurísticas ManhattanHeuristicFunction2 y MisplacedTilleHeuristicFunction2 que tengan en cuenta el estado objetivo definido.