INTELIGENCIA ARTIFICIAL

Memoria de la práctica 1

Grupo Jueves B

Hecho por
GUILLERMO BAJO LABORDA, 842748@unizar.es

Universidad Zaragoza
1542

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

Introducción y resumen

En esta primera práctica de la asignatura, se han abordado distintos problemas. Primeramente, se han analizado diversos datos obtenidos según los algoritmos de búsqueda utilizados en el problema del Eight Puzzle, así como la profundidad, los nodos expandidos, el tiempo, etc…

Por otra parte, se ha resuelto el problema de los caníbales y los misioneros, explicado en las sesiones teóricas. Para esto, se ha utilizado de "plantilla" los ficheros utilizados para la resolución de Eight Puzzle, ya que muchas cosas sin bastante similares, pero hay que modificar otras adaptándolas a cada problema concreto.

Finalmente, se ha resuelto el problema del lobo, la cabra y la col,una variante del problema de los misioneros y los caníbales.

Problema del Eight Puzzle

Para realizar el trabajo solicitado en el enunciado respecto a este problema, se ha creado el fichero EightPuzzlePr1, que contiene un main con las llamadas a los diferentes métodos de búsqueda, así como la implementación de la función EightPuzzleSearch, que representa la información solicitada por el problema.

Obviamente, los algoritmos producen distintos resultados, por tanto, a continuación se mostrará la salida obtenida y se analizarán los resultados obtenidos:

Problema	Profundidad	Expand	Q.Size	MaxQS	tiempo
BFS-G-3	3	5	4	5	9
BFS-T-3	3	6	9	10	1
DFS-G-3	59123	120491	39830	42913	987
DFS-T-3	---	---	---	---	(1)
DLS-9-3	9	10	0	0	2
DLS-3-3	3	4	0	0	1
IDS-3	3	9	0	0	0
UCS-G-3	3	16	9	10	3
UCS-T-3	3	32	57	58	4
BFS-G-9	9	288	198	199	9
BFS-T-9	9	5821	11055	11056	54
DFS-G-9	44665	141452	32012	42967	833
DFS-T-9	---	---	---	---	(1)
DLS-9-9	9	5474	0	0	11
DLS-3-9	0	12	0	0	0
IDS-9	9	9063	0	0	14
UCS-G-9	9	385	235	239	2
UCS-T-9	9	18070	31593	31594	75
BFS-G-30	30	181058	365	24048	839
BFS-T-30	---	---	---	---	(2)
DFS-G-30	62856	80569	41533	41534	569
DFS-T-30	---	---	---	---	(1)
DLS-9-30	0	4681	0	0	5
DLS-3-30	0	9	0	0	0
IDS-30	---	---	---	---	(1)
UCS-G-30	30	181390	49	24209	935
UCS-T-30	---	---	---	---	(2)

BFS

- Graph Search: El algoritmo de búsqueda en grafo siempre encuentra solución, y como se puede observar, la profundidad del problema coincide con el número de pasos a resolver en el mismo (3, 9 y 30).
- Tree Search: En este caso, pese a tener un comportamiento bastante similar al del grafo, no encuentra solución para el problema en 30 pasos.

DFS

- Graph Search: El algoritmo de búsqueda en grafo es claramente muy ineficiente, en el problema resoluble en 3 pasos tiene una profundidad de 59.123 y llega a expandir más de 120.000 nodos. Estas estadísticas son similares en para 9 y 30 pasos.
- Tree Search: Este algoritmo resulta de los menos eficientes para este problema, ya que no es capaz de encontrar solución para ninguno de los problemas en un tiempo razonable.

DLS

Es destacable que en este algoritmo, tanto el tamaño de la frontera como su tamaño máximo es siempre 0. A su vez, este algoritmo encuentra solución siempre que sea en un nivel de profundidad menor o igual al límite (3 o 9, respectivamente). Esto se debe a que el límite es siempre menor a la profundidad en la que se encuentra la solución de menor coste.

IDS

Este algoritmo encuentra una solución bastante rápida para los problemas resolubles en 3 y 9 pasos. Sin embargo, dado que la complejidad temporal de este algoritmo es exponencial, para el problema resoluble en 30 pasos tarda mucho tiempo. Al igual que sucede en el algoritmo Depth Limited Search (DLS), el tamaño de la frontera es 0 para este algoritmo.

UCS

- Graph Search: El algoritmo de búsqueda de coste uniforme encuentra siempre una solución en la profundidad equivalente al número de pasos en los que se resuelve el problema. Sin embargo, es destacable como el número de nodos expandidos aumenta exponencialmente conforme aumentan el número de pasos en los que se resuelve el problema.
- Tree Search: Al contrario del algoritmo en grafo, este no siempre encuentra solución, ya que lo hace para los problemas en 3 y 9 pasos pero no en 30, ya que en este ocupa demasiada memoria. Al igual que en la búsqueda en grafo, cuando encuentra solución lo hace en la profundidad equivalente al número de pasos en los que se resuelve el problema. El número de nodos expandidos también aumenta exponencialmente,

Problema de los misioneros y los caníbales

El problema de los misioneros y los caníbales consiste en que 3 misioneros y 3 caníbales crucen de un lado del río al otro con una barca con capacidad máxima para 2 personas. En ningún momento puede haber en ninguna de las dos orillas un número de misioneros inferior al de caníbales (esto se aplica si hay al menos un misionero en ese lado de la orilla).

Para resolver este problema se ha creado el directorio *aima.core.enviroment.canibales* el cual contiene los ficheros *CanibalesBoard.java, CanibalesFunctionFactory.java* y *CanibalesGoalTest.java*, con estructura bastante similar a los utilizados para resolver el problema del Eight Puzzle.

Se ha utilizado la misma nomenclatura a la utilizada en las trazas del enunciado proporcionado en el recurso de la plataforma Moodle.

A continuación se mostrará la salida resultante de la ejecución del problema:

```
Misioneros y canibales BFS -->
pathCost : 11
nodesExpanded : 13
queueSize : 1
maxQueueSize : 3
Tiempo : 9
SOLUCIÓN:
GOAL STATE
RIBERA-IZQ --RIO-- BOTE M M M C C C RIBERA-DCH
CAMINO ENCONTRADO
INITIAL STATE
RIBERA-IZQ M M M C C C BOTE --RIO-- RIBERA-DCH
Action[name==M2C]
RIBERA-IZQ M M M C --RIO-- BOTE C C RIBERA-DCH
- - -
Action[name==M1C]
RIBERA-IZQ M M M C C BOTE --RIO-- C RIBERA-DCH
- - -
Action[name==M2C]
RIBERA-IZQ M M M --RIO-- BOTE C C C RIBERA-DCH
- - -
Action[name==M1C]
RIBERA-IZQ M M M C BOTE --RIO-- C C RIBERA-DCH
- - -
Action[name==M2M]
RIBERA-IZQ M C --RIO-- BOTE M M C C RIBERA-DCH
- - -
Action[name==M1M1C]
RIBERA-IZQ M M C C BOTE --RIO-- M C RIBERA-DCH
```

```
- - -
Action[name==M2M]
RIBERA-IZQ C C --RIO-- BOTE M M M C RIBERA-DCH
- - -
Action[name==M1C]
RIBERA-IZQ C C C BOTE --RIO-- M M M RIBERA-DCH
- - -
Action[name==M2C]
RIBERA-IZQ C --RIO-- BOTE M M M C C RIBERA-DCH
- - -
Action[name==M1C]
RIBERA-IZQ C C BOTE --RIO-- M M M C RIBERA-DCH
- - -
Action[name==M2C]
RIBERA-IZQ --RIO-- BOTE M M M C C C RIBERA-DCH
- - -


Misioneros y canibales DLS(11) -->
pathCost : 11
nodesExpanded : 2199
Tiempo : 38
SOLUCIÓN:
GOAL STATE
RIBERA-IZQ --RIO-- BOTE M M M C C C RIBERA-DCH
CAMINO ENCONTRADO
INITIAL STATE
RIBERA-IZQ M M M C C C BOTE --RIO-- RIBERA-DCH
Action[name==M2C]
RIBERA-IZQ M M M C --RIO-- BOTE C C RIBERA-DCH
- - -
Action[name==M1C]
RIBERA-IZQ M M M C C BOTE --RIO-- C RIBERA-DCH
- - -
Action[name==M2C]
RIBERA-IZQ M M M --RIO-- BOTE C C C RIBERA-DCH
- - -
Action[name==M1C]
RIBERA-IZQ M M M C BOTE --RIO-- C C RIBERA-DCH
- - -
Action[name==M2M]
RIBERA-IZQ M C --RIO-- BOTE M M C C RIBERA-DCH
- - -
Action[name==M1M1C]
RIBERA-IZQ M M C C BOTE --RIO-- M C RIBERA-DCH
- - -
Action[name==M2M]
RIBERA-IZQ C C --RIO-- BOTE M M M C RIBERA-DCH
- - -
Action[name==M1C]
RIBERA-IZQ C C C BOTE --RIO-- M M M RIBERA-DCH
- - -
Action[name==M2C]
```

```
RIBERA-IZQ C --RIO-- BOTE M M M C C RIBERA-DCH
- - -
Action[name==M1C]
RIBERA-IZQ C C BOTE --RIO-- M M M C RIBERA-DCH
- - -
Action[name==M2C]
RIBERA-IZQ --RIO-- BOTE M M M C C C RIBERA-DCH
- - -


Misioneros y canibales IDLS -->
pathCost : 11
nodesExpanded : 8504
Tiempo : 54
SOLUCIÓN:
GOAL STATE
RIBERA-IZQ --RIO-- BOTE M M M C C C RIBERA-DCH
CAMINO ENCONTRADO
INITIAL STATE
RIBERA-IZQ M M M C C C BOTE --RIO-- RIBERA-DCH
Action[name==M2C]
RIBERA-IZQ M M M C --RIO-- BOTE C C RIBERA-DCH
- - -
Action[name==M1C]
RIBERA-IZQ M M M C C BOTE --RIO-- C RIBERA-DCH
- - -
Action[name==M2C]
RIBERA-IZQ M M M --RIO-- BOTE C C C RIBERA-DCH
- - -
Action[name==M1C]
RIBERA-IZQ M M M C BOTE --RIO-- C C RIBERA-DCH
- - -
Action[name==M2M]
RIBERA-IZQ M C --RIO-- BOTE M M C C RIBERA-DCH
- - -
Action[name==M1M1C]
RIBERA-IZQ M M C C BOTE --RIO-- M C RIBERA-DCH
- - -
Action[name==M2M]
RIBERA-IZQ C C --RIO-- BOTE M M M C RIBERA-DCH
- - -
Action[name==M1C]
RIBERA-IZQ C C C BOTE --RIO-- M M M RIBERA-DCH
- - -
Action[name==M2C]
RIBERA-IZQ C --RIO-- BOTE M M M C C RIBERA-DCH
- - -
Action[name==M1C]
RIBERA-IZQ C C BOTE --RIO-- M M M C RIBERA-DCH
- - -
Action[name==M2C]
RIBERA-IZQ --RIO-- BOTE M M M C C C RIBERA-DCH
- - -
```

Como se puede apreciar en las trazas, los tres algoritmos de búsqueda llegan a la solución y a todos ellos les cuesta el mismo número de pasos, 11.

Respecto a los nodos expandidos, el algoritmo Breadth-First Search (BFS), que expande los nodos en orden de profundidad, explorando primero todos los nodos a profundidad 1 antes de pasar a profundidad 2, expande un total de 13 nodos.

En cambio, el algoritmo DLS con límite de profundidad 11 (Depth-Limited Search) expande un total de 2199 nodos. Es normal puesto que esta búsqueda tiende a expandir muchos nodos antes de encontrar una solución al problema.

Finalmente, el algoritmo Iterative Deepening Depth-First Search (IDLS), expande todavía más nodos que DLS(11), 8504 nodos.

A modo de resumen, la cantidad de nodos expandidos varía según el algoritmo de búsqueda utilizado. BFS tiende a expandir menos nodos en problemas con una solución cercana a la raíz del árbol de búsqueda, mientras que la búsqueda en profundidad (como DLS e IDLS) tiende a expandir más nodos antes de encontrar una solución.