Sistemas de Inteligencia Artificial
Métodos De Búsqueda No Informados
Gridlock Problem

José Ignacio Galindo (50211), Federico Homovc (50418), Nicolás Loreti (49479)
Instituto Tecnológico de Buenos Aires (ITBA)
25 de Marzo 2013

Palabras clave - BFS, DFS, Iterative DFS, Greedy, A star, Gridlock Problem, heurísticas, optimización.

1 - Introducción

Este trabajo analiza la resolución del juego *Gridlock* o *Rush Hour* mediante el uso de un sistema de producción y distintas estrategias de búsqueda. Las estrategias utilizadas incluyen algoritmos BFS, DFS, IDFS, Greedy y A*.

2 - Descripción del problema

El juego consiste en un conjunto de piezas dispuestas de forma horizontal y vertical en un tablero, las cuales se deben mover a lo largo o ancho del mismo para poder sacar una pieza distinguida de color azul por una ranura en un lado del tablero. Las piezas verticales sólo se pueden mover hacia arriba y hacia abajo, y las piezas horizontales sólo hacia los lados. Ninguna pieza se puede superponer con otra en ningún momento ni puede salirse del tablero (sólo la pieza azul al finalizar el juego).

3 - Resolución

3.1 - Modelado

Todas las reglas consisten del movimiento de una pieza a lo largo de los espacios contiguos disponibles, por lo que la aplicación de cada una de ellas tiene siempre costo 1. En este caso tomamos que solo las piezas del tipo *espacio* se pudiesen mover ya que esto hacía a la solución más fácil de implementar.

3.2 - Heurística 1

La primera heurística planteada para este problema es contar la cantidad de piezas verticales que se interponen entre la pieza distinguida y la salida. Esta heurística favorece aquellos tableros donde la pieza distinguida tiene el camino más libre hacia la salida.

Como para sacar la pieza distinguida primero hay que mover de a una todas las piezas que están frente a ella, el valor devuelto por esta heurística es seguro menor que el costo de la solución óptima, por lo que es admisible.

3.3 - Heurística 2

La segunda heurística consiste en contar, además de la cantidad de piezas interpuestas entre la distinguida y la salida, si estas piezas tienen espacio hacia arriba o hacia abajo para moverse lo suficiente para dejar libre el camino de la pieza distinguida. En el primer caso se cuenta 1 por cada pieza interpuesta y de lo contrario 2, ya que al menos hay que mover una pieza más para liberarla.

Funciona de forma muy similar a la heurística 1 pero da mayor prioridad a aquellos tableros donde las piezas que bloquean la salida son más fácilmente removibles. Sin embargo, esta heurística no es admisible ya que una pieza puede estar bloqueando a otras dos, con lo que basta un movimiento para desbloquearlas y otros dos para moverlas; en este caso la heurística tiene un valor de 5 y el costo óptimo es de 4.

3.4 - Diferencias entre las heurísticas

- La primera de admisible, mientras que la segunda no.
- La segunda es más sofisticada ya que evalúa si las piezas se encuentran o no en proximidad de blancos para asignar el valor de la heurística.
- La segunda al ser no admisible puede dar lugar a soluciones no óptimas.

3.5 - Función de Costo

El costo de aplicar cualquier regla como se dijo anteriormente es uno. Esto da lugar a que fácilmente se tengan estados repetidos con el mismo costo. Para hacer un poco más eficiente la búsqueda de una solución se decidió cambiar en el motor el método *checkOpenAndClosed()* de la cátedra para que tomara como repetidos aquellos estados que correspondían al mismo tablero con mismo costo. Una vez encontrada una solución, el costo final viene representado por la cantidad de movimientos necesarios para encontrar esa solución particular. Es decir, que dado nuestra función tenemos una relación directa 1 a 1 entre el costo y los movimientos realizados.

4 - Análisis de resultados

Los resultados obtenidos se pueden apreciar en el anexo (1).

Las pruebas fueron realizadas sobre cuatro tableros, cada uno con mayor grado de dificultad que el anterior. El primer tablero utilizado es de muy baja complejidad (sólo se requieren 2 movimientos para encontrar la solución) y se tuvo en cuenta para poder obtener una comparación del comportamiento de los distintos algoritmos en un entorno de muy baja

complejidad. Los demás tableros son más complejos, presentando una complejidad ascendente.

Se corrió cada uno de los algoritmos una vez con cada tablero respetando el orden en que se aplicaban la reglas con el objetivo de poder hacer una mejor comparación de los mismos. En el caso del algoritmo de profundización iterativa fue menester cambiar el orden de aplicación de las reglas cada vez que se alcanza la profundidad establecida, ya que de lo contrario, el algoritmo carecería de sentido. Luego, se decidió correr dicho algoritmo 10 veces quedándonos con una performance mínima y máxima. Como se puede observar en las tablas, cuando se aplicó el algoritmo en los tableros con mayor dificultad hubo casos en los que al pasar más de un minuto no se encontraba la solución. Esto se debe a la naturaleza aleatoria de la aplicación de las reglas y del factor de ramificación del problema. Esto hace que ante un ordenamiento no favorable de las reglas se avance mucho sobre el árbol por un camino que no lleve a la solución, lo que hacer que cada vez haya más nodos por explorar y el algoritmo se vaya haciendo cada vez más lento.

Analizando los resultados del algoritmo A estrella, se puede ver como con la heurística h1 , admisible, siempre se encuentra la solución óptima, es decir, la que se alcanza con menor cantidad de movimientos. Se puede ver también que en el caso del tablero de nivel alto 2 (board 3) el algoritmo en cuestión con la heurística h2 no encuentra la solución óptima, lo que se desprende directamente de su condición de heurística no admisible. También haciendo una comparación de los resultados del algoritmo en cuestión para las dos heurísticas se termina viendo que la performance en relación al tiempo de procesamiento y a la cantidad de nodos expandidos es similar, por lo que no se puede afirmar contundentemente que alguna de la heurísticas funcione mejor para el problema en cuestión, al menos en lo que a A estrella se refiere.

También se pueden apreciar resultados similares cuando se observan los resultados obtenidos con el algoritmo greedy. Si bien la heurística h2 funcionó mejor para los dos últimos tableros en el sentido que se encontró una mejor solución (menor cantidad de movimientos) y con menor tiempo de procesamiento; se puede ver como en el segundo tablero ocurre lo contrario. Por lo tanto no se puede afirmar que ninguna de las heurísticas utilizadas sea mejor que la otra, a pesar de ser diferentes en su funcionamiento.

Pasando a describir las diferencias entre el algoritmo A estrella y greedy se puede apreciar claramente lo que se esperaría si se tiene en cuenta la teoría. El algoritmo greedy encuentra una solución no óptima, pero sí la encuentra considerablemente más rápido, sobre todo si se tienen en cuenta los dos tableros con mayor dificultad. También se puede ver como, exceptuando el primer tablero, el algoritmo A estrella expande considerablemente más nodos que el algoritmo greedy, lo que es lógico teniendo en cuenta que greedy siempre va eligiendo el nodo con menor valuación de la heurística para cada nivel.

Haciendo una comparativa entre DFS y BFS, se puede ver claramente que BFS encuentra siempre una solución con una profundidad menor a la de DFS, lo que tiene mucho sentido teniendo en cuenta como estos dos algoritmos funcionan.

También se puede ver como el tiempo de procesamiento de DFS es menor al de BFS lo que está relacionado con el alto factor de ramificación del problema ((# de *espacios* * 4)^profundidad).

Finalmente analizando los resultados obtenidos en general se puede ver como los algoritmos informados son como regla general más rápidos y encuentran una solución mejor, es decir con menor cantidad de movimientos, lo que tiene total sentido teniendo en cuenta la teoría.

5 - Conclusión

En todos los casos como era de esperar las búsquedas informadas resultaron ser mucho más efectivas que con los algoritmos no informados.

En el caso de Greedy generalmente está lejos de encontrar la solución óptima pero al ser informada con una buena heurística demuestra que obtienen resultados muy superiores a un BFS o DFS en tiempos reales. En un caso al usar una heurística no admisible como h2, se obtuvieron peores resultados, con mayor profundidad y estados generados que usando un DFS.

Por otro lado, utilizando Greedy siempre se garantiza obtener al menos una respuesta en tiempo mínimo, como se dijo anteriormente, difícilmente sea la óptima pero se puede contar con una posible respuesta en tiempos muy cortos. En la mayoría de los casos por debajo del segundo.

BFS como era de esperar siempre encuentra la solución óptima en cuanto a cantidad de movimientos pero al comparar este algoritmo contra A* son evidentes las mejoras y beneficios de utilizar este último.

Al utilizar una heurística no admisible con A* se puede observar también (board 3) como existe el riesgo de llegar a soluciones no óptimas.

Por último, IDFS depende mucho de que reglas se empiece aplicando primero. Si se aplican las reglas en un orden que favorezca a encontrar la solución se puede obtener resultados sorprendentemente buenos, al nivel de los algoritmos informados. Pero el problema se encuentra en que la obtención de estas soluciones son dependientes del azar en una medida determinante, lo que no pasa en los algoritmos informados.

Referencias

- Apuntes de la cátedra
- Inteligencia Artificial: un enfoque moderno, Russell y Norvig, Segunda edición, Prentice Hall.

Anexo

Resultados

Tablero nivel bajo (board 1)								
Estrategia	DFS	BFS	IDFS (mín)	IDFS (máx)	Greedy (h1)	Greedy (h2)	A* (h1)	A* (h2)
Profundidad	55	2	2	117	16	16	2	2
# de nodos expandidos	55	90	2	168	16	16	8	8
# de nodos frontera	310	241	28	508	129	129	81	81
# de estados generados	31056	24191	283	508118	12917	12917	819	819
tiempo de procesamiento [s]	0.096326	0.11532	0.007777	0.183623	0.050233	0.045126	0.026384	0.030412
Tablero nivel alto I (board 2)								
Estrategia	DFS	BFS	IDFS (mín)	IDFS (máx)	Greedy (h1)	Greedy (h2)	A* (h1)	A* (h2)
Profundidad	141	8	208	N/A	112	230	8	8
# de nodos expandidos	141	3034	440		112	360	678	529
# de nodos frontera	785	1410	893		589	1001	804	755
# de estados generados	785142	14103035	893238		589113	1001361	804679	755530
tiempo de procesamiento [s]	0.183321	3.805601	0.332527		0.156115	0.343572	0.521176	0.442526
Tablero nivel alto II (board 3)								
Estrategia	DFS	BFS	IDFS (mín)	IDFS (máx)	Greedy (h1)	Greedy (h2)	A* (h1)	A* (h2)
Profundidad	710	16	540	N/A	623	360	16	17
# de nodos expandidos	716	8286	1800		685	367	4004	4275
# de nodos frontera	3165	1092	2448		1809	1500	1446	1992
# de estados generados	3165717	10928287	2448542		1809686	1500368	14464005	19924276
tiempo de procesamiento [s]	1.25594	21.129889	1.566352		0.807168	0.42991	4.544817	5.212651
Tablero nivel alto III (board 4)								
Estrategia	DFS	BFS	IDFS (mín)	IDFS (máx)	Greedy (h1)	Greedy (h2)	A* (h1)	A* (h2)
profundidad	673	26	318	N/A	373	193	26	26
# de nodos expandidos	1715	3515	781		871	194	3117	2860
# de nodos frontera	2478	510	1228		1331	726	902	1031
# de estados generados	24781716	5103516	1228322		1331872	726195	9023118	10312861
tiempo de procesamiento [s]	1.934207	3.209046	0.49758		0.614303	0.208537	2.749087	2.513319

Tableros

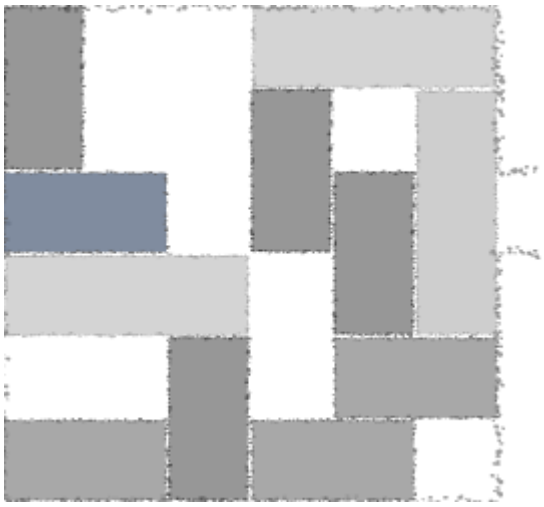

Figura 1: Tablero nivel alto I (board 2).

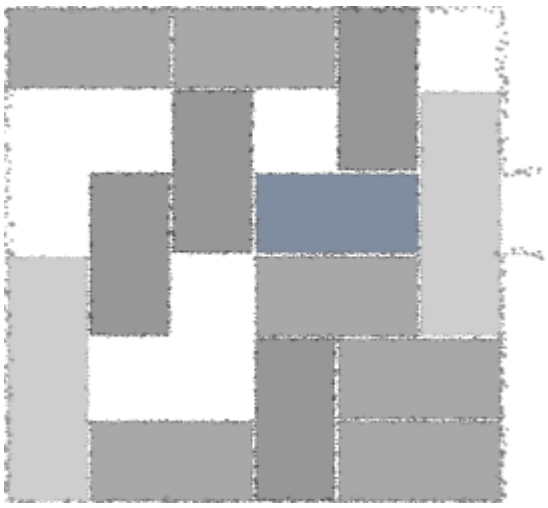

Figura 2: Tablero nivel alto II (board 3).

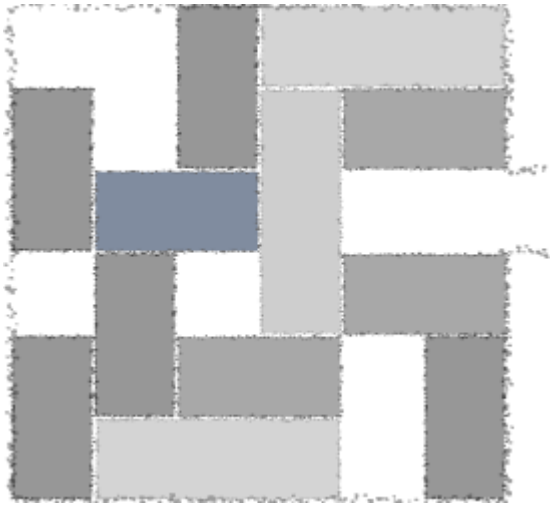

Figura 3: Tablero nivel alto III (board 4).