Sistemas de Inteligencia Artificial - Métodos de Búsqueda No Informados - Informe

Daniel Goldberg(48421), *I.T.B.A,* Tomás Mehdi(51014), *I.T.B.A,* Cristián Prieto(47573), *I.T.B.A,*

Abstract—En el presente artículo se describe el desarrollo de un sistema de producción para la resolución del juego *GridLock* (http://www.addictinggames.com/puzzle-games/gridlock.jsp), para el cual se utiliza un motor de inferencia realizado en Java.

Index Terms—Solución óptima, nodos expandidos, costo de búsqueda, nodos frontera, eficiencia, reglas.

1 Introducción

Mediante un sistema de producción y con distintas estrategias de búsqueda se pueden resolver problemas para los cuales no hay un algoritmo establecido. En este caso se analizará el desarrollo de un sistema de producción para la resolución del juego GridLock y se utilizarán como estrategias de búsqueda depth first search (DFS), breadth first searh (BFS), iterative deeping (ID), Greedy y Astar
En la segunda sección se realiza una breve descripción del juego. En la tercera sección se analizan detalles de implementación, las reglas del problema, la función de costo y las distintas heurísticas a implementar. En la cuarta sección se muestran las pruebas realizadas y los resultados obtenidos. En la quinta y última sección se analizan los resultados de la sección anterior y se sacan conclusiones. En las secciones anexas se pueden ver las tablas de resultados y los tableros a analizar.

2 Descripción del problema

El juego GridLock se trata de un tablero con piezas dispuestas en forma horizontal y vertical. Las mismas tienen distintos tamaños y están ubicadas cada una en un posición distinta dentro del tablero sin que se superpongan. Una de estas piezas es la pieza objetivo, la pieza azul (*ver imágenes anexas*), que debe ser llevada a la salida. Los movimientos permitidos son, para las piezas verticales, arriba y abajo, cualquier cantidad de posiciones, siempre y cuando no se superpongan con otras piezas ni se salgan del tablero. De manera equivalente las piezas horizontales pueden moverse a izquierda y derecha. Cualquiera de estas alternativas se toma como un sólo movimiento ya sea la pieza desplazada una o varias posiciones.

3 Implementación

Se representó el tablero como una lista de piezas y a su vez contiene una matriz que facilita la visualización del mismo. Se definieron cuatro metareglas: mover izquierda, mover derecha, mover arriba y mover abajo. Las primeras dos aplican sólo a piezas horizontales y las últimas dos a piezas verticales. Las reglas correspondientes a esas metareglas consisten en aplicar cada una de esas metareglas el número de veces necesario para moverse la cantidad de posiciones deseada, con un máximo de cinco posiciones debido a las dimensiones del tablero (*ver imágenes anexas*).

3.1 Función de costo

El costo de aplicar cualquiera de las reglas es uno. Debido a esto se realizó una modificación en el motor de inferencia para hacer más eficiente la búsqueda de soluciones. La misma consiste en descartar los estados nuevos y repetidos con igual costo. Originalmente se descartaban los estados nuevos y repetidos con costo mayor. Se agregó que se descarten los estados nuevos y repetidos con costo igual ya que aplicar cualquiera de nuestras reglas tiene costo uno y los estados con el mismo costo van a estar a la misma profundidad en el árbol.

3.2 Heurística 1

La $h1(n)$ consiste en contar la cantidad de casilleros de distancia que separan a la pieza azul de la salida sin tener en cuenta las piezas que puedan haber en el medio. En el tablero 1 y 2 ésta heurística daría 4 mientras que en el tablero 3 daría 5.
La intención de esta heurística es darle mayor prioridad a los tableros en donde la pieza azul se encuentra más cerca de la salida.
Esta heurística no es admisible ya que si la pieza azul tuviera el camino a la salida libre la solución óptima tendría costo 1 pero la heurística seguiría devolviendo la cantidad de casilleros de distancia entre la pieza azul y la salida.

3.3 Heurística 2

La $h2(n)$ consiste en contar la cantidad de piezas que hay entre la pieza azul y la salida. En el tablero 1 daría 2, en el tablero 2 daría 1 y en el 3 3.
La intención de la heurística es cuantificar los movimientos mínimos necesarios para ganar.
Al devolver la cantidad mínima de movimientos necesarios para que la pieza azul llegue a la salida, ésta heurística nunca va a sobreestimar al costo real. Por lo tanto es admisible.

4 Pruebas realizadas y resultados obtenidos

Se realizaron cien corridas con los métodos de búsqueda DFS, BFS, ID, Greedy y Astar para cada uno de los tres tableros con niveles de dificultad baja, media y alta. Se tomó como medida de dificultad del tablero el nivel del juego, cuanto mayor es el nivel del juego, mayor es la dificultad del tablero (es importante aclarar que esta dificultad no tiene relación con la cantidad de movimientos para sacar la ficha azul, sino en como hacer los movimientos para sacarla). Los resultados mostrados son un promedio de esas pruebas. La disposición de las piezas en los tableros puede verse en las imágenes anexas.

Todas las pruebas fueron realizadas con un orden de aplicación de reglas aleatorio. La generación de números aleatorios en todos los casos fue inicializada con la misma semilla.

La primer batería de pruebas fue realizada sobre el tablero de dificultad baja sin ninguna poda más que la eliminación de ciclos
Con estas primeras pruebas se busca analizar si los resultados obtenidos con los distintos métodos de búsqueda se corresponden con la teoría.

Los resultados de las pruebas pueden verse en la sección anexa.

La segunda batería de pruebas fue realizada con los métodos de poda propios del motor y además se agregó la eliminación de nodos repetidos con el mismo costo. Esto se puede realizar ya que en este problema todas las reglas tienen el mismo costo asociado. Por lo tanto dos nodos repetidos con el mismo costo van a estar siempre en el mismo nivel del árbol y no va a ocurrir que uno pueda ser solución óptima y el otro no.
Con estas segundas pruebas se busca analizar cómo se comportan los distintos métodos de búsqueda para este problema en particular, con las correspondientes mejoras en cuanto a eficiencia, que pueden ser aplicadas debido a las características del mismo.

Los resultados de las pruebas pueden verse en la sección anexa.

Aclaraciones: La fila "nodos visitados sin reiniciar" es un valor relevante sólo para el método ID y cuenta la cantidad de nodos visitados, sin reiniciar el contador en cada iteración. Con algunos métodos en determinadas circunstancias (dificultad del tablero, tipo de heurística, etc.) no se llegaron a completar las cien pruebas debido a tiempo de ejecución o falta de memoria. En estos casos el promedio se calculó sobre el número de pruebas alcanzadas.

5 Conclusiones

De la primer batería de pruebas se puede ver que los datos obtenidos se corresponden con la teoría.
Comparando BFS con ID se ve que ambos llegan a la misma profundidad. Ambos métodos son completos cuando el máximo número de sucesores de un nodo es finito y óptimos cuando el costo no decrece con la profundidad. Estas dos cosas suceden en el presente problema con lo cual ambos métodos son completos y óptimos. Sin embargo se puede ver que ID en el nivel final expande una cantidad de nodos menor que BFS. Además BFS tiene una cantidad mucho mayor de nodos frontera que ID. Esto se debe a que ID va visitando nodos y si no son solución los expande, siempre y cuando no estén en el último nivel de la iteración. Por otro lado BFS expande siempre que visita un nodo que no es solución independientemente del nivel y es por esto que va acumulando más nodos frontera. Esto hace que gaste mayor cantidad de memoria.
Comparando DFS con Greedy se ve que Greedy llega a la solución en una profundidad menor debido a que es un método de búsqueda informado.
Se puede ver también que Astar es el método que llego a la solución óptima expandiendo la menor cantidad de nodos. Comparando las heurísticas en este método se observa que la segunda al ser admisible expande menor cantidad de nodos que la primera ya que encuentra antes la solución.

De la segunda batería de pruebas se puede ver que la poda beneficia a todos los métodos de búsqueda. El que más se beneficia es BFS en el que se ve que por ejemplo la cantidad de nodos frontera disminuyó considerablemente. Esto se debe a que expande una cantidad de nodos mucho menor porque no visita nodos repetidos.
A medida que se aumenta la dificultad del tablero se ve que algunos métodos de búsqueda comienzan a fallar en algunas situaciones. Por ejemplo en el tablero de dificultad alta ni DFS ni Greedy pueden completar todas las iteraciones. En este tablero también se puede ver que a este nivel de dificultad Astar comienza a ser el método más óptimo en cuanto a cantidad de nodos expandidos y nodos frontera (también en cuanto al tiempo de ejecución a pesar de que no sea un parámetro de comparación tan preciso como el resto) y se separa

claramente de los demás. Además se puede observar que la heurística admisible da resultados mucho mejores que la no admisible.

REFERENCES

[1] Stuart Rusell and Peter Norvig, *Artificial Intelligence. A Modern Aproach. Third edition. Uninformed search.*
[2] Maria Cristina Parpaglione, *Clase 1-2-3.*
[3] http://www.addictinggames.com/puzzle-games/gridlock.jsp, *niveles del juego*

Pruebas sin poda

Pruebas tablero dificultad baja

Estrategia	DFS	BFS	ID	Greedy (h1)	Greedy (h2)(*1)	Astar (h1)	Astar (h2)
Profundidad	128	4	4	121	69	4	4
Nodos expandidos	130	4091	3382	121	69	3289	2030
Nodos expandidos sin reiniciar	130	4091	4355	121	69	3289	2030
Nodos frontera	815	31985	22	776	529	25764	18017
Total estados	947	36077	3405	898	599	29055	20048
Tiempo (ms)	14	369	47	14	7	2415	1434

*(*1) Promedio de 4 corridas*

Pruebas con poda

Pruebas tablero dificultad baja

Estrategia	DFS	BFS	ID	Greedy (h1)	Greedy (h2)	Astar (h1)	Astar (h2)
Profundidad	66	4	4	66	49	4	4
Nodos expandidos	125	256	313	97	346	221	192
Nodos expandidos sin reiniciar	125	256	558	97	346	221	192
Nodos frontera	264	155	12	266	234	172	266
Total estados	391	412	327	365	582	394	460
Tiempo (ms)	27	38	27	18	144	33	33

Pruebas tablero dificultad media

Estrategia	DFS	BFS	ID	Greedy (h1)	Greedy (h2)	Astar (h1)	Astar (h2)
Profundidad	211	14	14	202	174	14	14
Nodos expandidos	976	1215	5371	727	588	1188	1283
Nodos expandidos sin reiniciar	976	1215	22528	727	588	1188	1283
Nodos frontera	702	60	23	671	586	73	179
Total estados	1679	1276	5395	1399	1175	1262	1463
Tiempo (ms)	698	297	7161	405	441	297	345

Pruebas tablero dificultad alta

Estrategia	DFS(*1)	BFS	ID	Greedy (h1)(*2)	Greedy (h2)(*3)	Astar (h1)	Astar (h2)
Profundidad	293	10	10	105	57	10	10
Nodos expandidos	312	1605	3360	111	57	1430	784
Nodos expandidos sin reiniciar	312	1605	9596	111	57	1430	784
Nodos frontera	1249	716	22	389	214	737	539
Total estados	1562	2323	3384	501	272	2169	1325
Tiempo (ms)	90	491	1847	20	77	426	200

*(*1) Promedio de 11 corridas*

*(*2) Promedio de 7 corridas*

*(*3) Una corrida*

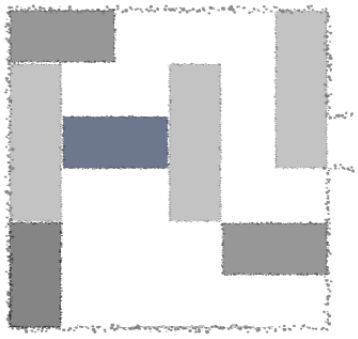

Fig. 1. Tablero dificultad baja

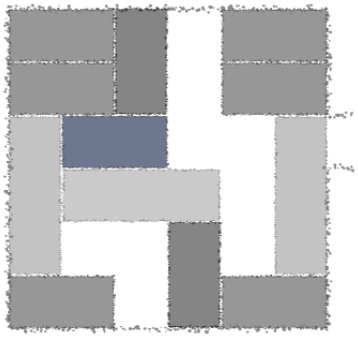

Fig. 2. Tablero dificultad media

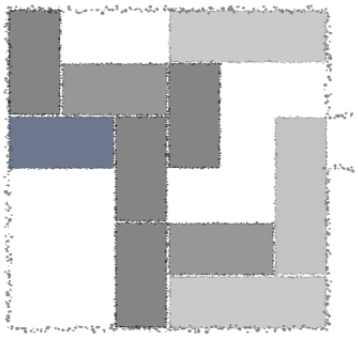

Fig. 3. Tablero dificultad alta