Trabajo Práctico 2:
Algoritmos de Búsqueda

Axel Fratoni
Fernán Oviedo
Gastón Rodríguez
Julián Antonielli

Sistemas de Inteligencia Artificial
1er cuatrimestre 2018

Objetivo

Crear un sistema de producción para resolver el juego Sokoban usando diferentes algoritmos de búsqueda.

Sokoban

El Sokoban es un juego de rompecabezas que consiste de un tablero rectangular en forma de grilla en donde cada posición puede ser ocupada por una pared, una caja, una meta o estar vacío. El jugador empieza en una posición arbitraria del tablero y puede moverse por los casilleros vacíos y empujar cajas que contengan un casillero vacío adyacente en la posición que están siendo empujados. El objetivo del juego es lograr que todas las cajas estén en el mismo casillero que una meta.

Metodología

Para resolver el problema fue necesario crear un motor de inferencia que dado un estado inicial genere un árbol de búsqueda expandiendo los nodos según el algoritmo pedido. Nuestro motor es capaz de utilizar los algoritmos de búsqueda desinformados *depth first*, *breadth first* y *profundización iterativa*; y los algoritmos informados *greedy* y *A estrella*.

Reglas

Se consideraron dos posibilidades como conjuntos de reglas aplicables:

- Que cada movimiento del jugador sea una regla.
- Que cada movimiento de una caja sea una regla.

Finalmente se optó por la segunda ya que generaría muchas menos estados que la primera y aceleraría el proceso de búsqueda. La regla consiste en fijarse en todos los lugares a los que el jugador se podría mover, y si encuentra una caja y la puede empujar, que lo haga.

Consideremos la figura 1 como ejemplo. En este caso las reglas aplicables serían:

- Mover caja en (1,3) hacia arriba.
- Mover caja en (1,3) hacia abajo.

Las reglas que moverían la caja hacia la izquierda o la derecha no son aplicables ya que no se puede mover la caja en esas direcciones, así que no son consideradas. Todas las reglas tienen costo 1. Esto se debe a que consideramos la optimalidad de la solución en función de la cantidad de movimientos de cajas.

Heurísticas

Utilizamos dos heurísticas admisibles para la resolución con los algoritmos informados:

- **Cajas posicionadas:** heurística basada en cuantas cajas no están posicionadas en metas. Por ejemplo en el estado de la figura 4 el valor de la heurística sería 3 ya que una de las cuatro cajas está posicionada en el lugar correcto. Es fácil ver que esta heurística es consistente: cada movimiento de una caja puede disminuir el valor de la heurística en 1 (poniendo una caja que no estaba en una meta sobre una meta), subirla en 1 (sacando una caja de una meta) o dejarla igual (en otro caso). Como el costo de una regla siempre es 1, siempre la heurística en un estado es menor a la heurística en el estado siguiente más el costo de la regla para pasar de estado.
- **Distancia:** heurística basada en la suma de las distancias de Manhattan entre las cajas y su meta más cercana. Por ejemplo en la figura 3 el valor de la heurística sería 5 ya que la caja de arriba se encuentra a tres casilleros de su meta más cercana y la caja de abajo se encuentra a dos casilleros de la suya. Esta heurística también es consistente. Al igual que en el caso anterior, mover una caja sólo puede disminuir el valor de la heurística en 1. Con el mismo razonamiento que antes, la heurística es consistente.

Problema

El problema se encarga de generar las reglas para un estado determinado, fijarse si un estado está resuelto y devolver un estado inicial. Un estado tiene un tablero armado y sabe cuántas cajas hay, cuántas cajas posicionadas hay y dónde se encuentra el jugador.

Para saber si el problema está resuelto lo único que hay que saber es saber si todas las cajas están posicionadas.

Resultados

Una vez implementado el motor de inferencia genérico y resueltas todas las cuestiones correspondientes al Sokoban se plantearon 4 distintos “niveles” del juego para ser resuelto con todos los algoritmos disponibles.

En base a esos niveles se obtuvieron métricas al finalizar la ejecución correspondientes a la profundidad del árbol, el tiempo en encontrar la solución y la cantidad de nodos expandidos, de frontera y generados. El primer nivel, al ser tan trivial, fue resuelto de igual manera con todos los algoritmos, pero en los otros tres se pudieron observar diferencias significativas.

En los siguientes cuadros A*P significa *A estrella con heurística de cajas posicionadas* y A*D significa *A estrella con heurística de distancia*. GRP significa *Greedy con heurística de cajas posicionadas,* GRD significa *Greedy con heurística de distancia* e IDA significa *Iterative Deepening Search*.

- **Nivel 2:** éste fue el único nivel en el cual algún método no encontró la respuesta (agotó la memoria después de nueve minutos de ejecución), y fue el método BFS.
  En la figura cinco se puede observar que A*D fue el segundo peor, mientras que A*P fue el más rápido.

- **Nivel 3:** en este nivel se puede observar en la figura seis cómo el método greedy tuvo una profundidad de más del doble que los demás, y la cantidad de nodos generados hasta 66 veces la cantidad de nodos generados por el algoritmo más adecuado (A*P).

- **Nivel 4:** este nivel es el primero en el cual a A*D le va mejor que a A*P, como se puede observar en la figura 7, mostrándonos que no necesariamente siempre la segunda heurística es mejor que la primera. También a dfs es al que mejor le va, demostrando que no necesariamente los algoritmos informados son mejores.

Conclusiones

Con estos tres niveles se puede observar cómo ambos algoritmos A* son superiores a cualquiera de los otros cuatro mientras que se elija bien la heurística, con *Iterative Deepening Search* siguiéndoles de cerca y Greedy, BFS y DFS demostrando los peores resultados.

También se puede observar cómo una mala heurística puede dañar los resultados de la búsqueda, en el segundo nivel mirando A*D, ya que este demostró los peores resultados mientras que A*P mostró los mejores.

Gráficos

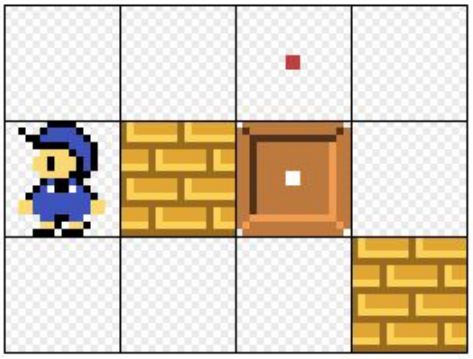

Figura 1: Nivel 1

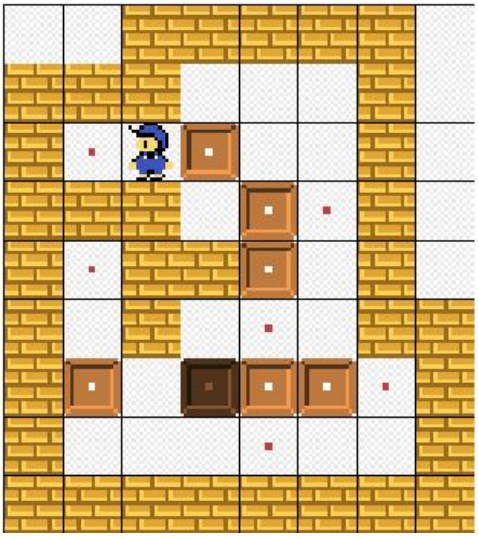

Figura 2: Nivel 2

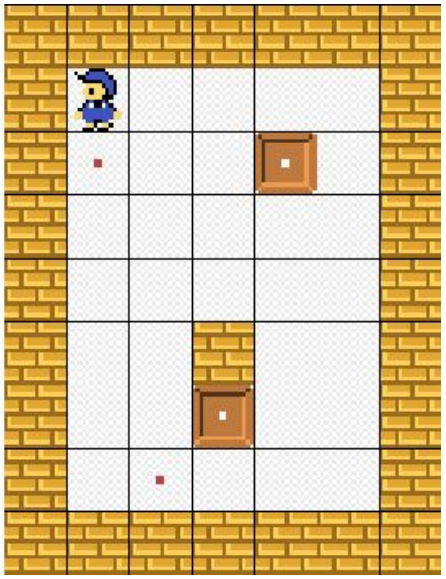

Figura 3: Nivel 3

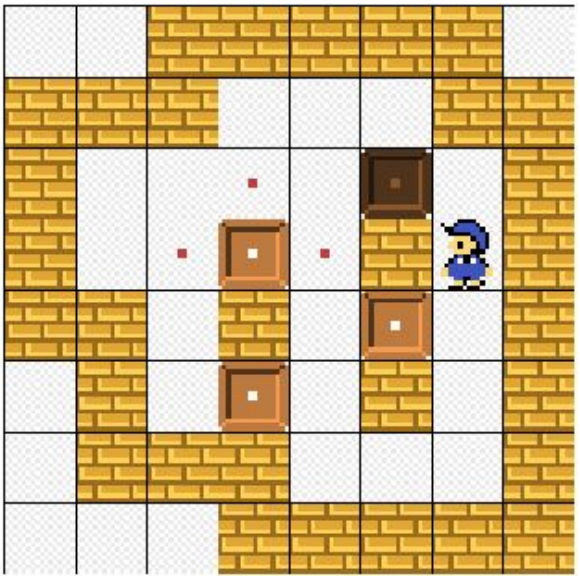

Figura 4: Nivel 4

	A*D	A*P	BFS	DFS	GRD	GRP	IDDFS
Depth	12	12	-	20	20	12	12
Time (seconds)	12,96	0,10	-	7,15	52,68	0,56	52,75
Expanded Nodes	551057	6746	-	745852	3000334	29513	627274
Frontier Nodes	1074723	23500	-	52	865989	26166	35
Generated nodes	2249971	34382	-	3470238	12557538	107476	2840192

Figura 5: Métricas tomadas del nivel 2.

	A*D	A*P	BFS	DFS	GRD	GRP	IDDFS
Depth	5	5	5	7	15	7	5
Time (seconds)	0,019	0,013	0,022	0,042	0,114	0,012	0,03
Expanded Nodes	186	60	506	1631	5832	37	574
Frontier Nodes	391	262	1557	31	1513	90	22
Generated nodes	769	347	2737	6731	23081	145	2855

Figura 6: Métricas tomadas del nivel 3.

	A*D	A*P	BFS	DFS	GRD	GRP	IDDFS
Depth	12	12	12	58	24	34	12
Time (seconds)	0,453	4,163	8,22	0,227	0,629	0,594	125,04
Expanded Nodes	58370	416753	993732	16600	40384	46921	27118392
Frontier Nodes	95213	357004	858127	160	13842	16733	29
Generated nodes	221484	1215284	2749620	58197	153807	175899	98050126

Figura 7: Métricas tomadas del nivel 4.