Sistemas de Inteligencia Artificial

Trabajo Práctico Especial 1

Métodos de Búsqueda No Informados e Informados

Abril 2016

Integrantes

- Cavo, María Victoria - 53202
- Di Nucci, Nicolás Santiago - 54091
- Kljenak, Iván - 54019

Índice

- Introducción ……... 2
- Trabajo Realizado ……. 3
    - Estrategias de búsqueda ……... 3
    - Reglas …….. 3
    - Heurísticas ……... 4
- Análisis de resultados y conclusiones ……... 5
- Anexo ……...…….. 7

Introducción

El presente documento describe la implementación de métodos de búsqueda informados y no informados para hallar soluciones óptimas para el juego “Edificios”.
El juego consiste en un tablero cuadrado, inicialmente vacío, dividido en *NxN* casillas, cada una de las cuales corresponde a un edificio cuya altura, que es un número entre 1 y *N*, se debe determinar.
Además, se provee una serie de datos que representan la cantidad de edificios visibles desde cada uno de los cuatro puntos cardinales.
El objetivo del juego es determinar las alturas de los edificios, logrando que desde cada perspectiva sólo sea visible la cantidad de edificios indicada. Una condición adicional conocida es que por cada número entre 1 y *N* hay exactamente *N* edificios de esa altura y que no es posible ubicar dos edificios de una misma altura en una misma fila o en una misma columna.
Ejemplificamos el juego, a partir de la siguiente serie de tableros de *2x2* (esto implica que se deben ubicar cuatro edificios, dos de altura 2 y dos de altura 1):

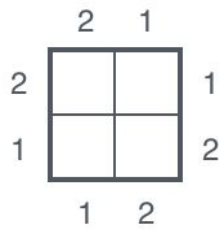

Figura 1. Ejemplo de tablero de 2x2.

Este tablero indica que alguien observando desde el norte puede ver dos edificios en la primera columna y uno en la segunda; que si alguien observa desde el oeste, verá dos edificios en la primera fila y uno en la segunda; y el razonamiento es análogo para quien mira desde el sur o del este. La única solución posible para este caso, es la siguiente:

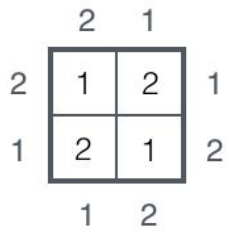

Figura 2. Ejemplo de tablero de 2x2 resuelto.

Se puede ver que se respeta la restricción de que un número no se repita en una fila o en una columna, y además, se cumple la condición de la cantidad de edificios visibles. Para verificar esto, por ejemplo, se puede observar que alguien al sur del mapa puede efectivamente ver dos edificios en la segunda columna: el de altura 1 que está al frente y el de altura 2, que sobresale por detrás. Con el mismo análisis se ve que todas las demás filas y columnas cumplen la condición.

Trabajo realizado

Extendiendo el solucionador general de problemas aportado por la cátedra, se implementó código para hallar soluciones al juego “Edificios”, usando diferentes métodos de búsqueda. Como punto de partida, se decidió aprovechar el concepto de “reparación heurística”, iniciando la exploración de soluciones a partir de un tablero previamente completo garantizando que en cada fila no haya números repetidos (en principio, si podría haberlos en las columnas), que progresivamente se acerca a la solución realizando, en cada paso, intercambios de los números de dos casillas. La razón para tomar esta decisión reside en la complejidad de implementar una solución que progresivamente fuera completando el tablero, ya que en este juego, la posición de un número depende del lugar dónde estén ubicados los demás. Con un tablero incompleto, se cuenta con menor información sobre qué tan apropiado es ubicar un edificio en una determinada casilla.

Reglas

Con la inicialización previa del tablero, se conoce que inicialmente el tablero ya cuenta con *N* edificios de cada una de las alturas comprendidas entre 1 y *N*. Además, en cada fila hay exactamente un edificio de cada altura. A continuación, realizando sucesivamente intercambios, es posible alcanzar una solución. Cada intercambio define un nuevo estado del juego. La regla aplicada para pasar de un estado a otro consiste en seleccionar dos casillas cualesquiera en una misma fila y modificar las alturas de sus edificios respectivamente. Es decir, el conjunto de reglas se reduce a, para cada fila, realizar todos los intercambios posibles entre dos casillas.
Al no haber diferencias que dependan de cuáles son las casillas elegidas, aplicar cualquiera de las reglas tiene el mismo costo, y éste es constante.
En un momento del desarrollo del trabajo, se evaluó agregar reglas que intercambiaran filas y columnas completas. Sin embargo, se observó que en estas condiciones, no era posible representar todos los estados posibles que podía alcanzar el tablero, impidiendo el hallazgo de una solución.

Estrategias de búsqueda

Se utilizaron las estrategias de búsqueda no informadas *DFS*, *BFS* e *IDDFS*, que usando un tablero inicial como raíz, construyen un árbol, donde en cada nivel se agregan los estados alcanzables aplicando una de las reglas a cada nodo del nivel anterior. Estas tres estrategias difieren en su manera de recorrer ese árbol. *DFS* evalúa cada una de las ramas del árbol, desde la raíz, hasta la hoja antes de proceder con la siguiente rama. *BFS*, en cambio, recorre cada uno de los niveles del árbol antes de pasar al siguiente. *IDDFS*, actúa como *DFS*, pero con una restricción adicional, que es un límite a la profundidad evaluada: si, por ejemplo, al método se le agrega como parámetro una profundidad límite *n,* el algoritmo consiste en utilizar la estrategia *DFS* recorriendo el árbol hasta llegar a esa profundidad, y seguir buscando sólo en caso de no haber hallado una solución al alcanzar esa profundidad.
Asimismo, se usaron estrategias de búsqueda informada, agregando conocimiento del juego para restringir la búsqueda en base a una táctica aplicable al mismo que permita

orientar la búsqueda por aquellos caminos por los que resulta más probable hallar la solución. Este es el caso de las estrategias greedy y el algoritmo *A**.

Heurísticas

Considerando que el juego en cuestión tiene dos restricciones principales (que no haya alturas repetidas en una misma fila o columna y la cantidad de edificios visibles) se intentó definir una heurística para cada uno de los casos y combinar ambas para obtener una nueva que contemplase con más precisión el universo del juego.
Para la primera condición, la heurística considera la cantidad de números diferentes en cada columna. De cumplirse esta condición, tal cantidad debería ser igual a *N*. La estrategia implementada consiste en evaluar en cada columna la diferencia entre *N* y la cantidad de números diferentes, realizar una suma entre todos esos valores y finalmente retornar ese resultado dividido por dos, pensando en que cada intercambio puede modificar los valores de hasta dos columnas.

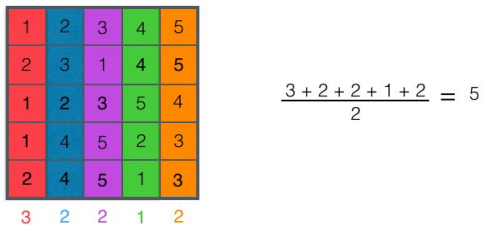

Figura 3. Cálculo de heurística para verificar condición de números repetidos.

En cuanto a la segunda condición, se evalúa, en cada columna y en cada fila, la diferencia entre la cantidad de edificios que debería ser visible y la cantidad de edificios efectivamente visibles. Aquellas filas o columnas en que alguna de esas diferencias no sea igual a cero, desde cualquiera de los dos puntos cardinales que la afectan, incrementan una cuenta que intenta calcular cuántas filas o columnas aún es necesario corregir. Así, se obtiene una cantidad de columnas a corregir y otra de filas a corregir. Como cada regla puede potencialmente afectar dos columnas, pero sólo una fila, la primera cuenta es dividida por dos. Luego, se toma el máximo entre ambas sumas como valor de retorno, considerando que es necesario como mínimo aplicar esa cantidad de reglas para llegar a un estado final.

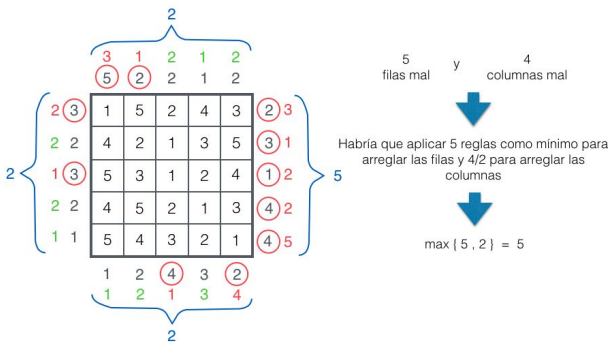

Figura 4. Cálculo de heurística para verificar condición de visibilidad de edificios.

Finalmente, una vez calculadas ambas estrategias, el valor de la heurística combinada es igual al máximo entre los dos valores obtenidos.

Para una segunda heurística, se propuso una modificación en la estrategia para evaluar la condición de la cantidad de edificios visibles. En este caso, aquellas filas o columnas en las cuales la diferencia entre la cantidad de edificios efectivamente visibles y la cantidad que deberían serlo no es igual a cero para ambos puntos cardinales afectando a la fila o columna, tienen un mayor peso en el cálculo de cuál sería el movimiento más adecuado en el siguiente paso. Para el ejemplo de la figura 4, por ejemplo, con este cambio el valor obtenido sería 7. Las filas 1 y 3 aportan dos unidades a una de las dos cuentas consideradas (son las mismas que fueron mencionadas en el caso anterior) para resolver el valor que devolverá esta estrategia y las filas 2, 4 y 5 aportan una unidad cada una. En cuanto a las columnas, las 1, 2, 3 y 5. Adicionalmente, en este caso no se realiza la división por dos en la cuenta de las "columnas a corregir". El cálculo de la heurística se reduce entonces a obtener el máximo entre 7 y 4. Esta heurística no resulta admisible pues eliminamos contemplaciones hechas en el caso anterior que servían para evitar sobreestimaciones, como la división por dos de la cuenta que se usa para evaluar las columnas. No obstante eso, como se explicará en la sección de Análisis de resultados, en las pruebas realizadas, la heurística obtiene un mejor rendimiento al ser usada con el algoritmo greedy que cuando se utiliza la primera estrategia descripta.

Análisis de resultados y conclusiones

Se realizaron pruebas para evaluar y comparar distintos aspectos del funcionamiento de cada uno de los métodos de búsqueda implementados. Los variables analizadas fueron: cantidad de nodos expandidos, cantidad de nodos borde (que eventualmente podrían ser expandidos), cantidad de estados analizados, costo de la solución (equivalente a cantidad de pasos necesarios para llegar a la solución), profundidad alcanzada en el árbol de estados y tiempo de ejecución.

La primera serie de pruebas consistió en definir un conjunto de condiciones de juego iniciales (tablero y cantidad de edificios visibles desde cada perspectiva) y aplicar los 5 métodos de búsqueda a cada una de esas condiciones iniciales, para luego calcular un promedio que permitiera evaluar el rendimiento de cada método. El tamaño de los tableros iniciales varía desde 3 x 3 hasta 4 x 4.
A partir de los datos obtenidos, se puede concluir que el método IDDFS es el que encuentra la solución en el menor tiempo. Sin embargo, en una comparación entre métodos no informados e informados, se observa que los segundos resultan más eficientes en cuanto a la cantidad de nodos analizados y no consumen un tiempo que pueda ser considerado irracional. En efecto, un comparación entre los métodos DFS y BFS con los métodos A* y greedy, muestran que los segundos aportan una solución óptima o cercana a la óptima en un tiempo menor a los primeros. Otro dato destacado luego de haber realizado las pruebas, es que el método DFS tiene un rendimiento significativamente menor a cualquiera de los demás métodos. La explicación de esta diferencia reside en que cuando son pocos los movimientos necesarios, no es necesario recorrer una gran profundidad del árbol de estados, pero el método DFS recorre cada una de las ramas hasta llegar a una hoja, lo que implica que muchos nodos sean evaluados si la solución no se encuentra en alguna de las primeras ramas. *(Ver tabla 1)*

En una segunda ronda de pruebas, se procuró estudiar las diferencias entre los dos métodos de búsqueda informada, A* y greedy. Para ello, nuevamente se definió un conjunto de tableros (en este caso de dimensión 5 x 5) y luego se calculó un promedio de los resultados obtenidos. Se utilizó en ambos casos la primera de las heurísticas descriptas en la sección correspondiente.
Los datos obtenidos indican que la estrategia greedy encuentra una solución más rápidamente que A*. Asimismo, la cantidad de nodos expandidos es menor. No obstante eso, la solución aportada por A* es más óptima, dado que este algoritmo evalúa tanto el costo de haber llegado hasta el estado actual como el costo estimado de continuar el camino hasta la solución. La decisión de usar uno u otro depende de un balance entre la exactitud que se necesite de la solución y la cantidad de recursos disponibles. *(Ver tabla 2)*

Finalmente, se realizó una tercera sesión de pruebas. Esta vez el objetivo fue medir el grado de optimización aportado por cada una de las heurísticas en el uso de la estrategia greedy. Una vez más se definió un conjunto de tableros iniciales y los resultados surgieron de promediar los datos conseguidos en cada una de las sucesivas ejecuciones de nuestra implementación. La conclusión es que aplicando la heurística que no es admisible, la estrategia greedy obtiene un rendimiento superior que al usar la otra heurística de acuerdo a todas las variables contempladas. Incluso se puede apreciar que la solución hallada requiere aplicar menos reglas. *(Ver tabla 3)*

Anexo

Propiedades	Estrategias de Búsqueda				
	No informadas			Informadas	
	BFS	DFS	IDDFS	GREEDY	A*
Nodos expandidos	353	3414	64	22	17
Nodos borde	3771	49275	28	253	134
Estados analizados	145	3414	64	20	7
Costo de la solución	3	3414	3	4	3
Profundidad de la solución	3	3414	3	4	3
Tiempo [milisegundos]	67	43757	9	28	16

Tabla 1

Propiedades	Estrategias de Búsqueda Informadas	
	GREEDY	A*
Nodos expandidos	1424	3123
Nodos borde	67402	143164
Estados analizados	756	1854
Costo de la solución	15	5
Profundidad de la solución	15	5
Tiempo [milisegundos]	4517	67402

Tabla 2

Propiedades	Heurísticas	
	Admisible	No Admisible
Nodos expandidos	1424	38
Nodos borde	67402	1813
Estados analizados	756	36
Costo de la solución	15	6
Profundidad de la solución	15	6
Tiempo [milisegundos]	4517	47

Tabla 3