25 de Diciembre, 2019

Sistemas Inteligentes

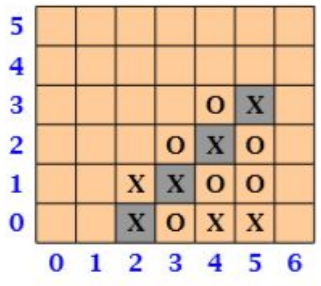
PRÁCTICA 2: CONECTA 4

Víctor Calabuig Rodríguez

Índice

1 Introducción

Esta memoria corresponde a la realización de la segunda práctica de la asignatura Sistemas Inteligentes, del Grado en Ingeniería Informática. Esta práctica consiste en crear un programa inteligente que sea capaz de jugar al juego tradicional Conecta 4 (o 4 en Raya), en el que cada jugador debe intentar colocar 4 de sus fichas de forma consecutiva, en horizontal, vertical, o diagonal, antes que su rival. El tablero es una matriz de 7 columnas por 6 filas, y las fichas siempre caen desde arriba.



El algoritmo inteligente implementado en esta práctica será el que utilice uno de los jugadores, al que llamaremos Min, y este jugará contra otro programa implementado por el profesor (y por tanto oculto), al que llamaremos Max. Para implementar la parte inteligente de nuestro programa se utilizará un algoritmo de búsqueda entre adversarios llamado Minimax. Este algoritmo realizará búsquedas en profundidad, con un límite establecido (que se introducirá como parámetro para realizar varias ejecuciones con distintos límites de profundidad), para averiguar qué estado sucesor es el que más próximo está de una jugada ganadora. Teóricamente, a mayor nivel de profundidad, más inteligente será el programa ya que tomará su decisión en base a una cantidad mayor de estados (disposiciones del tablero) hipotéticos. Esta hipótesis se pondrá a prueba ejecutando el programa con distintos niveles de profundidad y anotando los resultados.

La memoria está formada por 2 partes principales:

- **Resultados**: Aquí se recogen los resultados obtenidos tras realizar varias ejecuciones con distintas combinaciones en los límites de profundidad de cada jugador.
- **Análisis**: Esta parte trata de entender y explicar los resultados obtenidos en la parte anterior.

**Junto con esta memoria se entrega el proyecto Java comprimido en el archivo Conecta4.zip.

2 Resultados

Tras implementar la práctica, he realizado varias ejecuciones con distintas combinaciones de límites de profundidad para cada jugador (Min y Max). Para cada ejecución se ha medido el jugador que gana la partida, los nodos que se han explorado, y el tiempo de ejecución. A continuación se muestran los resultados:

2.1 Jugador ganador

gana min
gana max
empate

					Max					
	Prof.	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
	1									
	2									
	3									
Min	**4**									
	5									
	6									
	7									
	8									
	9									

Imagen 1

2.2 Nodos explorados

					Max	
	Profundidad	1	2	3	4	5
	1	117	121	38	38	120
	2	648	673	582	598	228
	3	3.588	3.164	3.195	3.208	3.390
Min	4	20.338	20.844	18.632	19.097	22.335
	5	118.005	137.937	131.298	130.949	161.970
	6	646.220	692.439	677.754	663.065	896.979
	7	4.165.945	4.165.945	3.919.448	3.878.142	6.201.738
	8	27.919.589	32.042.591	28.202.094	28.115.663	32.863.983
	9	165.572.477	179.457.241	151.513.046	152.337.590	193.544.594

Imagen 2

					Max	
	Profundidad	5	6	7	8	9
	1	120	24	24	24	120
	2	228	171	171	171	171
	3	3.390	1.116	1.116	1.116	3.357
Min	4	22.335	18.883	18.883	19.998	21.923
	5	161.970	123.815	124.203	124.433	134.589
	6	896.979	932.444	857.592	766.811	661.464
	7	6.201.738	4.491.217	4.406.194	4.991.009	5.056.568
	8	32.863.983	34.467.553	34.603.451	28.988.344	28.026.038
	9	193.544.594	174.900.529	175.546.146	194.876.741	210.095.730

Imagen 3

2.3 Tiempo de ejecución (segundos)

					Max	
	Profundidad	1	2	3	4	5
	1	0,347	0,421	0,384	0,419	0,612
	2	0,447	0,480	0,462	0,590	0,538
	3	0,414	0,461	0,514	0,569	0,671
Min	4	0,668	0,613	0,835	0,662	0,949
	5	1,039	0,968	1,235	1,364	1,693
	6	2,918	3,312	3,574	3,410	4,502
	7	16,383	16,555	16,277	15,672	22,813
	8	210,647	141,667	125,810	124,083	231,561
	9	1392,753	1443,080	1445,732	1498,133	1420,088

Imagen 4

					Max	
	Profundidad	5	6	7	8	9
	1	0,612	0,559	0,656	1,173	4,230
	2	0,538	0,620	0,802	1,422	2,572
	3	0,671	0,647	0,856	1,415	4,580
Min	4	0,949	0,972	1,588	2,561	6,481
	5	1,693	1,561	2,241	3,118	6,960
	6	4,502	4,985	4,896	5,978	7,502
	7	22,813	21,472	22,562	26,313	26,411
	8	231,561	237,568	248,523	222,049	227,881
	9	1469,89	1506,004	1570,878	1499,099	1569,054

Imagen 5

3 Análisis de Resultados

En primer lugar, analizando la tabla de victorias de cada jugador, puede observarse que a medida que aumenta el límite de profundidad de un jugador también aumenta el número de veces que gana la partida. Esto es lo que se esperaba, ya que con un mayor límite de profundidad, un jugador explora más estados hipotéticos, y por tanto es capaz de tomar mejores de decisiones de juego al contemplar escenarios más avanzados en el tiempo (en cuanto a número de movimientos). Aún así, se dan casos anómalos en los que gana el jugador que juega con un límite de profundidad inferior, como es el caso en el que Min juega con límite 6 y Max con límite 9, y gana Min, o también el caso en el que Min juega con límite 1, Max con límite 9, y el resultado es empate (en lugar de ganar Max, que sería lo esperado).

La tabla de nodos explorados refleja el número de nodos que explora Min a lo largo de la partida para decidir qué movimientos realizar. Por ello, como muestra la tabla, a medida que aumentamos la profundidad de la búsqueda de Min, este explora más estados para tomar cada decisión, y por lo tanto aumenta el número de nodos explorados totales al final de la partida.

En tercer lugar, observando la tabla del tiempo que tarda en ejecutarse cada partida, vemos que se parece a la tabla de nodos explorados en cuanto a su comportamiento en relación al límite de profundidad: A medida que aumenta el límite de profundidad (tanto de Min como de Max), aumenta el tiempo de ejecución. Esto se corresponde con los nodos explorados, ya que explorar más nodos consume más tiempo. Este comportamiento es el esperado ya que es de esperar que para explorar más estados y tomar una mejor decisión, cada jugador utilice más tiempo para cada jugada.

Destacar aquí que aunque el tiempo aumenta cuando aumenta la profundidad tanto de Min como de Max, el ritmo al que lo hace es sustancialmente distinto. Por ejemplo, para un límite de profundidad de Min y Max de 1, el tiempo es 0,347 segundos. Si aumentamos el límite de profundidad de Min a 9 (manteniendo el límite de Max en 1), el tiempo pasa a unos 23 minutos, que significa un aumento del 397.600%. Sin embargo, si aumentamos el límite de profundidad de Max a 9 (manteniendo el límite de Min a 1), el tiempo pasa a ser 4,23 segundos, aumentando tan solo en un 1219%. Esto significa que el algoritmo empleado por la clase Profesor (que es el que utiliza el jugador Max para realizar sus

jugadas) tiene una complejidad temporal mejor que el algoritmo que he implementado yo en la práctica (que es el que utiliza el jugador Min). Por lo tanto, a medida que el tamaño de problema crece (con límites de profundidad mayores), se aprecia cada vez más la diferencia entre un algoritmo y otro, siendo el jugador Max muchísimo más rápido en calcular sus jugadas. Otra prueba de que el algoritmo de la clase Profesor que utiliza el jugador Max es más rápido es el tiempo que tarda en realizar un jugada cada jugador. Por ejemplo, si ejecutamos el programa con un límite de 7 para Min y un límite de 9 para Max, Max tarda aproximadamente medio segundo en realizar sus primeras jugadas (las que más tiempo consumen), mientras que Min tarda unos 3 segundos para cada uno de sus primeros movimientos.

Por último, una observación adicional en cuanto al comportamiento del programa durante su ejecución. Cuando se inicia la partida, los jugadores tardan más en calcular su jugada que en los momentos finales de esta. Específicamente, la primera jugada es la que más tiempo le consume a ambos jugadores, y a partir de ahí, cada jugada adicional se realiza de forma más rápida que la jugada anterior. Esto se debe a que el tablero vacío es el estado que más posibilidades distintas (jugadas posibles) tiene, por lo que la búsqueda en profundidad explora más estados. Así, a medida que se va llenando el tablero con fichas de ambos jugadores, el espacio de posibles jugadas se va reduciendo cada vez más y más, y por tanto las búsquedas exploran menos estados y son más rápidas.

4 Bibliografía

- Oracle, “Java SE - JDK 13 Documentation”, https://docs.oracle.com/en/java/javase/13/
- Vega, O. Resolución de problemas mediante búsqueda [Diapositivas de PowerPoint]. Recuperado 10 de Noviembre de 2019, de https://poliformat.upv.es/access/content/group/GRA_11683_2019/teor%C3%ADa/tema_2.pdf