UNIVERSIDAD SIMÓN BOLÍVAR
DPTO. COMPUTACIÓN Y TEC. DE INF.
CI-5437 INTELIGENCIA ARTIFICIAL I

Informe Proyecto 2

Estudiantes
LEONEL GUERRERO
EROS CEDEÑO

Profesor
CARLOS INFANTE

22 de junio de 2023

Contenido General

1. Introducción

La inteligencia artificial actualmente es casi sinónimo al machine learning, sin embargo la IA va mas alla. Es posible desarrollar algoritmos que nos permiten simular inteligencia sin necesidad de apoyarse en redes neuronales, y aun así y tomar desiciones de acuerdo a diferentes escenarios y según diferentes criterios.

La teoria de juegos ha causado gran interés dado que presentan retos para las computadoras y estos son modelables mediante espacios de estados. Aun mas cuando la computadora de IBM: Deep Blue fue capaz de vencer en una partida de ajedrez a uno de los mejores jugadores de la historia, marcando un precedente en la historia de la inteligencia artificial. Deep Blue vs Garry Kasparov es la denominación genérica para los famosos matches de ajedrez, jugados a 6 partidas entre la supercomputadora de IBM Deep Blue y el campeón del mundo de ajedrez: Garry Kaspárov. Estos encuentros son referidos también con la expresión de .El hombre contra la máquina", debido a la naturaleza del evento. El primer encuentro se jugó en febrero de 1996 en Filadelfia, (Pensilvania). Kasparov lo ganó 4–2, perdiendo una partida, empatando 2 y ganando 3. Lo que en notación ajedrecística se escribe +3 =2 -1.

Así como el ajedrez existe un conjunto de juegos con características similares denominados Juegos de suma cero que se definirán a continuación en este informe.

En este proyecto se realizaran pruebas sobre el desempeño de 4 algoritmos: Negamax, Negamax con poda alpha beta, Scout y Negascout sobre el popular juego de suma cero Othello.

2. Juegos de Suma Cero

2.1. Definición

Se denominan juegos de suma cero a aquellos en los que las ganancias de un jugador se equilibran con las pérdidas de otro. En otras palabras, son aquellos juegos en los que si hacemos una resta entre las ganancias totales de los participantes y las pérdidas totales, el resultado siempre va a ser cero.

Los juegos de suma cero se encuentran en muchos contextos. El póquer y los juegos de azar son ejemplos populares de juegos de suma cero, ya que la suma de las cantidades ganadas por algunos jugadores es igual a las pérdidas combinadas de otros. Juegos como el ajedrez y el Othello, donde hay un ganador y un perdedor, también son juegos de suma cero.

3. Caso de estudio: Othello 6x6

3.1. Othelo 6x6

El juego Othello estándar o también conocido como Reversi, es un juego entre dos personas, que comparten 64 fichas iguales, de caras distintas, que se van colocando por turnos en un tablero dividido en 64 escaques. Las caras de las fichas se distinguen por su color y cada jugador tiene asignado uno de esos colores, ganando quien tenga más fichas sobre el tablero al finalizar la partida. Tradicionalmente el tablero es 8x8 sin embargo para fines prácticos e ilustrativos de este proyecto se estará trabajando con una representación simplificada del juego, restringiendo el tablero a 6x6. A continuacion se establecen las reglas del juego.

1. Las fichas negras siempre juegan primero.
2. Dos fichas deben estar colocadas en el centro de ambos colores para empezar.
3. Coloca el primer disco en un lugar que rodee a un disco de tu oponente. A esto también se le conoce como flanquear en Othello. Una 'fila' está conformada por uno o más discos formando una línea horizontal, vertical o diagonal.

	B	C	D	E	F	G
2						
3						
4						
5						
6						
7						

Tabla 1: Estructura e identificacion tradicional de las casillas de Othello

4. Se le da la vuelta al disco flanqueado para mostrar su lado contrario. Cuando se flanquea un disco, se debe girar para mostrar el color contrario y pertenecera al jugar del color correspondiente. Sin embargo, es posible volver a darle vuelta a ese mismo disco en caso de que forme parte de una fila flanqueada por el oponente.
5. Se siguen turnando los contrincantes para colocar discos hasta que no sea posible realizar un movimiento legal. Siempre debes colocar los discos en una posición en la que puedan flanquear una fila de discos de tu oponente.
6. Si para un jugador no es posible realizar un movimiento valido este renuncia a su turno.
7. Si no hay movimientos validos para ninguno de los jugadores se considera la partida como finalizada y ganara el jugador con mas fichas de su color.

3.2. Implementacion

Se utilijo el lenguaje de programacion C++ para la implementacion del juego. Para representar el tablero del juego se utilizan tuplas de 64 bits. Dado que en el tablero 6x6 dispone de 36 casillas unicamente, los primeros 36 bits serán los bits significativos considerados en cada tupla. Tradicionalmente se nombran con una letra seguida de un numero que inidican la columna y la fila respectivamente como se muestra en la tabla 1 1

Sin embargo en la implementación se renombran las casillas con un identificador numérico único para cada una. La función de mapeo que se realizo para definir la correspondencia entre casillas se ilustra en la tabla 2.

Cómo se puede observar la numeración de las casillas enumera inicialmente las cuatro casillas centrales del tablero. Luego continua numerando las casillas restantes pero está vez llenando de izquierda a derecha y de arriba a abajo para obtener finalmente como última casilla la casilla número 35.

Para la tupla de bits del tablero se asignan con 1's las casillas que están siendo ocupadas por fichas negras y 0's en las casillas ocupadas por fichas blancas. Adicionalmente de lleva en otra variable llamada 'free' una tupla se bits que indican cuando una casilla está libre o está ocupada por otra ficha. Marca con 1's los espacios ocupados y con 0 los vacíos.

Realizar una jugada consiste en ocupar uno de los espacios válidos para el jugador en turno por la ficha correspondiente y actualizar el tablero de acuerdo a los resultados de la jugada. Por lo tanto la jugada de expresa adecuadamente mediante el número de casilla a ocupar dentro del tablero. Cuando un jugador no tiene jugada posible entonces se dice que

	B	C	D	E	F	G
2	4	5	6	7	8	9
3	10	11	12	13	14	15
4	16	17	0	1	18	19
5	20	21	2	3	22	23
6	24	25	26	27	28	29
7	30	31	32	33	34	35

Tabla 2: Mapeo de los identificadores de las casillas dentro de la implementación

ocupa la casilla 36 (valor indicativo de paso de turno) y no se reversa ninguna ficha ni de ocupa ninguna casilla dentro del tablero.

Adicionalmente se implementa un conjunto de métodos útiles para manipular los estados y obtener información de forma adecuada y simple.

Por cada casilla del tablero se almacenan 4 arreglos de enteros que indican las casillas adyacentes en casa una de las siguientes orientaciones: Horizontal, Vertical, Diagonal ascendente y Diagonal descendente. Estos son particularmente útiles para determinar si un movimiento dado es válido, para obtener los estados adyacentes a un estado inicial y para determinar las fichas flanqueadas o reversibles como resultado de una jugada.

4. Algoritmos de Juego

Con la estructura del juego Othello explicada en el inciso anterior se ejecuta un conjunto de algoritmos de desicion sobre el arbol de juego representado por un variacion principal siguiente:

1.d3 2.c5 3.d6 4.e3 5.f5 6.f4 7.e2 8.d2 9.c2 10.e6 11.e7 12.g5 13.c4 14.c3 15.g4 16.g3 17.f3 18.c7 19.b5 20.d7 21.b7 22.b3 23.c6 24.b6 25.f7 26.f6 27.b4 28.b2 29.g6 30.g7 31.pass 32.f2 33.g2.

Siguiendo la notación de la implementación establecida anteriormente corresponde a la siguiente PV:

1.12 2.21 3.26 4.13 5.22 6.18 7.7 8.6 9.5 10.27 11.33 12.23 13.17 14.11 15.19 16.15 17.14 18.31 19.20 20.32 21.30 22.10 23.25 24.24 25.34 26.28 27.16 28.4 29.29 30.35 31.36 32.8 33.9

Para asegurarnos que la implementación de los algoritmos es correcta, evaluamos los mismos a lo largo de la variación principal. El valor del juego es -4 y por lo tanto, todo nodo sobre la variación principal debe evaluar a dicho valor.

Para cada algoritmo, la evaluación se hace comenzando sobre el tablero terminal en la variación principal, y subimos a lo largo de la misma. Cada vez que subimos, volvemos a ejecutar el algoritmo de solución a partir del nodo actual hasta que termine o hasta que el límite de tiempo expire.

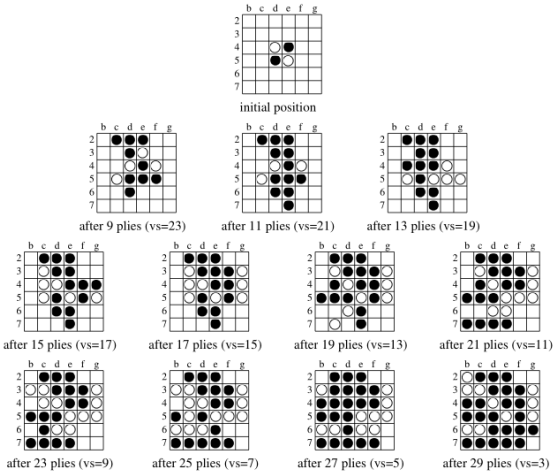

Figura 1: Estados intermedios de la variación principal

4.1. Definición: Negamax

El algoritmo Negamax es una variante del algoritmo minimax, el cual consiste en que cada nodo independiente si fuese MIN o MAX toma el valor maximo de sus hijos (como si todos los nodo fueran MAX), solo que los valores de los nodos MAX se cambian de signo, de ahi podemos ver de donde viene su nombre ya que en vez de tomar los maximos y minimos toma siempre maximos pero con los valores cambiados. Pero de esta manera se logra el mismo efecto ya que tomar el mayor valor pero cambiado de signo es en realidad tomar el menor asi cuando el algoritmo esta en un nivel MIN cuando toma el mayor de sus hijos en realidad esta tomando al menor.

Los inicios de Minimax fueron con el matemático francés Emile Borel que fue el primero en ofrecer en 1921 un tratamiento riguroso a los juegos competitivos y en estudiar las estrategias aplicables a los juegos de suma cero, sin embargo se suele atribuir a John Von Neumann el principal merito de la concepción del principio minimax, ya que el fue el quien, en su articulo de 1928 «Zur Theorie der Gesellschaftsspiele»(Sobre la teoría de los juegos de sociedad), publicado en la revista Mathematische Annalen, puso las bases de la moderna teoria de juegos y probo el teorema fundamental del minimax, por el que se demuestra que para juegos de suma cero con información imperfecta entre dos competidores existe una única solución optima.

4.2. Definición: Negamax con poda alpha-beta

Los inicios de la poda Alfa-Beta fueron con personajes como Arthur Samuel, D.J Edwards y T.P. HArt, Alan Kotok, Alexander Brudno, Donald Knuth y Ronald W. Moore. Esta tecnica de busqueda consiste en reducir el numero de nodos evaluados en un arbol de juego por el algortimo minimax, dando una ayuda al problema del algoritmo minimax que es el numero de estados a explorar es expononencial al numero de movimientos.

La busqueda minimax se utiliza el algoritmo de primero en profundidad, por ello en cualquier momento solo se deben considerar los nodos a lo largo de un camino en el arbol. Para la poda alfa-beta se utilizan dos parametros que describen los limites sobre los valores hacia atras que aparecen a lo largo de cada camino.

α: Que es el valor de la mejor opción hasta el momento a lo largo del camino para MAX, esto implicara por lo tanto la elección del valor mas alto.

β: Que es el valor de la mejor opcion hasta el momento a lo largo del camino para MIN,

esto implicara por lo tanto la elección del valor mas bajo.

En esta búsqueda se va actualizando el valor de los parámetros según se recorre el árbol. El método realizara la poda de las ramas restantes cuando el valor actual que se esta examinando sea peor que el valor actual de α o β para MAX o MIN, respectivamente.

4.3. Definición: Scout

Desarrollado por Judea Pearl en los años 80, este algoritmo tiene como idea básica que mucho esfuerzo computacional se puede evitar si se tiene una forma rápida de comprobar desigualdades. De nuevo la potencia del algoritmo se deberá a su capacidad de poda: no explorará ramas que no lleven a una mejor solución. Por lo tanto con SCOUT necesitaremos, indispensablemente, un método rápido para comprobar desigualdades. Este método, que comprobará si el valor MiniMax de un nodo es menor o mayor que un determinado valor, lo denominaremos función TEST. En consecuencia, para SCOUT necesitaremos dos rutinas: "TEST" y "EVAL". La función TEST se encargará de resolver la desigualdad. Sus parámetros de entrada serán: el nodo a aplicar el test, el valor fijo de la comparación, y la desigualdad a aplicar (>ó >=) . Su salida será un booleano indicándonos si se cumple o no la desigualdad. Por otra parte, la función EVAL calculará el valor MiniMax de un nodo, usando a TEST para comprobar cuándo deberá o no explorar una rama determinada. Su único parámetro de entrada será el nodo a evaluar.

4.4. Definición: Negamax Scout

También llamado "búsqueda de la variante principal" este algoritmo puede ofrecer rendimientos incluso mayores a la poda alfa-beta si los nodos se encuentran correctamente ordenados.

Al igual que alfa-beta, negascout es un algoritmo de búsqueda direccional para calcular el valor minimax de un nodo en un árbol. La mejora del algoritmo negascout sobre el algoritmo alfa-beta es que el primero no examinará un nodo que el segundo podaría.

Este algoritmo es útil siempre y cuando los nodos estén bien ordenados ya que supone que el primer nodo explorado será el mejor, así podrá confirmarlo en otros nodos usando la ventana nula. En caso de fallo, como el primer nodo no era el de máximo nivel, se seguirá buscando el mejor nodo en el árbol del mismo modo que en el algoritmo alfa-beta.

5. Resultados

Los algoritmos explicados anteriormente se utilizaron para abordar el problema del Othello 6x6 sobre cada uno de los estados de la variacion principal y se analizan 5 variables de interes:

- Nivel: Nivel de profundidad del arbol de busqueda, en donde entre mas bajo el numero mas cerca esta del estado inicial.
- Valor: Valor del tablero en el estado final, el cual es la diferencia entre las fichas del jugador y las del oponente.
- Expandidos: Nodos expandidos, es decir, nodos que se generaron, y se evaluaron todos sus hijos.
- Generados: Nodos generados, es decir, nodos que se generaron de algun nodo padre
- Segundos: Segundos de ejecucion del algorimo
- Velocidad Generados/Segs: Generación de nodos por segundo.

La ejecuciones se realizaron en una computadora de escritorio con procesador Intel i9-12900H (20) de 12th con 4.9 GHz de frecuencia del reloj, en donde se ejecutaron los algoritmos durante 15 minutos recorriendo cada estado de la variacion principal, para analizar cual es el que llega mas alto en los estados de la variacion principal, es decir el que llega mas cerca del estado inicial y logra llegar a un estado final. .

Los resultados obtenidos son consistentes con los resultados esperados. Estos se resumen en las tablas siguientes, en donde en la mayoria el valor del juego es -4, lo cual es un resultado esperado ya que el jugador 1 es el que pierde, con 4 fichas menos que el jugador 2, sin embargo en la tabla del algoritmo scout se observara que el valor se va alternando entre 4 y -4, esto debido al funcionamiento del algoritmo, en donde cunado las blancas juegan (niveles pares en las tablas) el valor es 4 representando que el gano con 4 fichas a su favor y cuando las negras juegan (niveles impares en las tablas) el valor es -4 representando que el jugador negro perdio con 4 fichas en contra.

Negamax					
Nivel	Valor	Expandido	Generado	Segundos	Velocidad (Generados/Seg)
34	-4	0	1	1.00001e-06	999992
33	-4	1	2	1.00001e-06	1.99998e+06
32	-4	3	5	1.00001e-06	4.99996e+06
31	-4	4	7	1.00001e-06	6.99995e+06
30	-4	9	14	2.00002e-06	6.99995e+06
29	-4	10	15	2.00002e-06	7.49994e+06
28	-4	64	97	1.5e-05	6.46667e+06
27	-4	125	191	2.7e-05	7.07407e+06
26	-4	744	1163	0.000163	7.13497e+06
25	-4	3168	4965	0.000654	7.59174e+06
24	-4	8597	13480	0.001708	7.89227e+06
23	-4	55127	86964	0.012415	7.00475e+06
22	-4	308479	479379	0.067935	7.05644e+06
21	-4	2525249	3941913	0.574536	6.86104e+06
20	-4	9459570	14835904	2.16035	6.86737e+06
19	-4	65121519	102380150	15.1728	6.74762e+06
18	-4	625084814	984849508	208.603	4.72118e+06

Cuadro 3: Resultados Negamax

Negamax con poda alpha-beta

Nivel	Valor	Expandido	Generado	Segundos	Velocidad (Generados/Seg)
34	-4	0	1	1.00001e-06	999992
33	-4	1	2	9.99949e-07	2.0001e+06
32	-4	3	5	2.00002e-06	2.49998e+06
31	-4	4	7	9.99949e-07	7.00035e+06
30	-4	9	14	3.00002e-06	4.66663e+06
29	-4	10	15	3.00002e-06	4.99996e+06
28	-4	21	30	5.99999e-06	5.00001e+06
27	-4	62	89	1.6e-05	5.5625e+06
26	-4	186	265	5.1e-05	5.19608e+06
25	-4	769	1111	0.000188	5.90957e+06
24	-4	1152	1665	0.000263	6.3308e+06
23	-4	3168	4557	0.000784	5.8125e+06
22	-4	7031	10164	0.001667	6.09718e+06
21	-4	76021	110226	0.018376	5.99837e+06
20	-4	98129	142929	0.024011	5.95265e+06
19	-4	205017	300939	0.04996	6.0236e+06
18	-4	960343	1394469	0.241701	5.7694e+06
17	-4	1549785	2252573	0.391216	5.75788e+06
16	-4	22325108	32446284	5.45289	5.95029e+06
15	-4	32949019	47710205	8.30147	5.7472e+06
14	-4	82016158	117768173	28.9797	4.06382e+06
13	-4	315074162	453291654	120.02	3.77679e+06

Cuadro 4: Resultados Negamax con poda alpha-beta

Scout

Nivel	Valor	Expandido	Generado	Segundos	Velocidad (Generados/Seg)
34	4	0	1	0	inf
33	-4	1	2	9.99949e-07	2.0001e+06
32	4	3	5	4.00003e-06	1.24999e+06
31	-4	4	7	2.00002e-06	3.49997e+06
30	4	14	21	5.99999e-06	3.50001e+06
29	-4	15	22	4.99998e-06	4.40002e+06
28	4	26	35	7e-06	5e+06
27	-4	64	85	1.7e-05	5e+06
26	4	314	404	7.9e-05	5.11392e+06
25	-4	1334	1760	0.000321	5.48287e+06
24	4	2011	2642	0.000481	5.49272e+06
23	-4	3232	4167	0.00079	5.27468e+06
22	4	10214	13368	0.002443	5.47196e+06
21	-4	42358	54696	0.01039	5.26429e+06
20	4	68853	89391	0.017101	5.22724e+06
19	-4	157458	204736	0.039048	5.24319e+06
18	4	497954	649816	0.125919	5.16059e+06
17	-4	911296	1190773	0.2284	5.21354e+06
16	4	6096169	7994314	1.52998	5.2251e+06
15	-4	23572285	30962884	7.85414	3.94224e+06
14	4	57114374	74595522	19.0055	3.92493e+06
13	-4	211427675	276967910	61.0344	4.5379e+06
12	4	545805549	713445579	168.982	4.22203e+06

Cuadro 5: Resultados Scout

Negascout

Nivel	Valor	Expandido	Generado	Segundos	Velocidad (Generados/Seg)
34	-4	0	1	1.00001e-06	999992
33	-4	1	2	1.00001e-06	1.99998e+06
32	-4	3	5	2.00002e-06	2.49998e+06
31	-4	4	7	1.00001e-06	6.99995e+06
30	-4	14	22	3.99997e-06	5.50004e+06
29	-4	15	23	3.00002e-06	7.66661e+06
28	-4	26	38	7e-06	5.42858e+06
27	-4	64	92	1.5e-05	6.13333e+06
26	-4	312	446	7.1e-05	6.28169e+06
25	-4	1275	1818	0.000298	6.10067e+06
24	-4	1894	2710	0.000432	6.27315e+06
23	-4	3051	4290	0.00074	5.7973e+06
22	-4	9329	13452	0.002154	6.24512e+06
21	-4	37988	54043	0.009386	5.75783e+06
20	-4	63570	91230	0.015868	5.74931e+06
19	-4	142595	206739	0.035299	5.8568e+06
18	-4	466161	675195	0.116025	5.81939e+06
17	-4	870050	1266108	0.221928	5.70504e+06
16	-4	5518091	7922363	1.40299	5.64677e+06
15	-4	19705373	28280234	5.6808	4.97821e+06
14	-4	47600678	67767026	17.8732	3.79155e+06
13	-4	185297022	263797632	67.9737	3.88088e+06
12	-4	477003110	680946014	154.465	4.40841e+06

Cuadro 6: Resultados Negascout

6. Análisis de resultados

De manera general se puede observar que los algoritmos que utilizan poda son mas eficientes que los que no la utilizan. Esto se debe a que la poda permite descartar ramas del arbol de busqueda que no son necesarias de explorar. En el caso de Negamax con poda se puede observar que es mas eficiente que Scout, esto se debe a que Scout no utiliza una poda tan agresiva como Negamax con poda. En el caso de Negascout se puede observar que es mas eficiente que Negamax con poda, esto se debe a que Negascout utiliza una poda mas agresiva que Negamax con poda, tambien se puede observar que scout y negascout logran llegar mas arriba en los estados del juego llegando hasta el nnivel 12, el mas alto entre todos, sin embargo se puede apreciar que negascout es ligeramente mas eficiente que scout ya que logra podar mas ramas, por lo que explora menos nodos y logra analizar todo el arbol en menos tiempo.

7. Conclusiones

Se analizo el desempeño de 4 algoritmos escenciales en la inteligencia artificial como son Negamax, Negamax con poda, Scout y Negascout. El problema a resolver era el popular juego de mesa Othello y se obtuvieron resultados diversos que fueron analizados en el inciso anterior.

Junto a este informe se anexa el código fuente con su documentación de uso así como un archivo de texto con los resultados de ejecución en la computadoras de prueba.

Se plantea como tarea futura refactorizar esta actividad incorporando las redes neuronales para generar heuristicas que permitan una poda mas adecuada dentro del espacio de estados generados y expandidos por los algoritmos.