Universidad Simón Bolívar
Dpto. de Computación y Tecnología de la Información
Inteligencia Artificial I

Nathalia Silvera 12-10921
Carolina Rivas 13-11209

Informe Proyecto IV

El juego de Connect 4 es un desafío estratégico clásico que involucra a dos jugadores que compiten para conectar cuatro fichas del mismo color en línea, ya sea horizontal, vertical o diagonalmente, dentro de un tablero vertical de 7 columnas y 6 filas. Este juego ha sido objeto de interés tanto en el campo de la inteligencia artificial como en el de la investigación de juegos, debido a su complejidad y amplias posibilidades de movimientos.

El objetivo de este proyecto es abordar la resolución del juego de Connect 4 mediante el uso de dos conocidos algoritmos de inteligencia artificial: Montecarlo y Minimax. Estos algoritmos son ampliamente utilizados para tomar decisiones en juegos de estrategia, y su aplicación en Connect 4 nos permitirá evaluar su desempeño y comparar los resultados obtenidos.

El algoritmo de Montecarlo se basa en simulaciones aleatorias y se utiliza para explorar movimientos posibles y estimar la probabilidad de victoria en un estado de juego dado. Por otro lado, el algoritmo Minimax es una estrategia de búsqueda exhaustiva que evalúa todas las combinaciones de movimientos posibles hasta cierta profundidad para encontrar el mejor movimiento.

En este proyecto, implementaremos ambos algoritmos y los pondremos a prueba en partidas de Connect 4. Se analizarán factores como el tiempo de ejecución, la eficacia en la toma de decisiones y la capacidad para encontrar estrategias ganadoras. A partir de los resultados obtenidos, se determinará cuál de los dos algoritmos es más óptimo para resolver este tipo de juegos de estrategia.

Aplicación de los Algoritmos

Algoritmo Minimax

El algoritmo Minimax se implementa en el código mediante la función minimax. Esta función realiza una búsqueda en profundidad para explorar todas las posibles combinaciones de movimientos hasta una profundidad máxima de 15, para evitar la explosión combinatoria en juegos más complejos.
En cada nivel de profundidad, la función evalúa el estado actual del tablero y calcula un valor que representa la ventaja o desventaja del jugador en ese estado. Se utiliza una función de evaluación heurística que toma en cuenta el número de fichas conectadas en línea y su importancia estratégica.

El algoritmo Minimax se beneficia del uso de poda alfa-beta, que permite eliminar ramas del árbol de búsqueda que no serán seleccionadas y así reducir la cantidad de nodos explorados, mejorando significativamente la eficiencia del algoritmo.
Una vez que se han explorado todas las opciones posibles hasta la profundidad establecida, el algoritmo Minimax selecciona el movimiento que lleva al mejor resultado para el jugador en turno.

En el código, la función get_best_move utiliza el algoritmo Minimax para encontrar el mejor movimiento posible para el jugador actual. Itera a través de los movimientos posibles y evalúa la utilidad de cada movimiento utilizando el algoritmo Minimax. Luego, selecciona el movimiento que produce el mejor resultado para el jugador actual.

Algoritmo de Montecarlo

El algoritmo de Montecarlo se implementa en el código mediante varias funciones relacionadas: expand, best_child, selection, simulation y backpropagation.

El proceso de búsqueda de Montecarlo comienza con la función selection, que selecciona un nodo en el árbol de búsqueda para ser explorado. Si el nodo no está completamente explorado, se expande el árbol creando nuevos nodos hijos correspondientes a movimientos no explorados. Si el nodo ya está completamente explorado, se utiliza la función best_child para seleccionar el mejor nodo hijo según la fórmula UCT (Upper Confidence Bound applied to Trees).

Después de seleccionar un nodo para explorar, la función simulation realiza una simulación aleatoria del juego desde el estado del nodo seleccionado hasta alcanzar un estado terminal (ganador o empate). Esta simulación se basa en movimientos aleatorios, lo que permite estimar la probabilidad de victoria desde ese estado.

Una vez que se ha completado la simulación, la función backpropagation actualiza los valores de recompensa y visitas en el árbol de búsqueda, retrocediendo desde el nodo terminal hasta la raíz del árbol. Esto permite que los nodos explorados en la fase de simulación influyan en las decisiones futuras del algoritmo.

La función MCTS es el núcleo del algoritmo de Montecarlo. Realiza iteraciones del proceso de búsqueda de Montecarlo para encontrar el mejor movimiento posible desde el estado actual del juego.

Ejecución del Programa

Para llevar a cabo este proyecto, se utilizó una computadora con un procesador Core i7 y 8GB de RAM.

El programa ofrece al jugador la posibilidad de elegir entre varias opciones, numeradas del 1 al 5, para jugar partidas de Connect 4 con diferentes configuraciones:

Opción 1: MCTS VS MCTS En esta opción, dos algoritmos de Montecarlo (MCTS) jugarán entre sí. Ambos algoritmos utilizarán simulaciones aleatorias para tomar decisiones y se enfrentarán en una partida estratégica.
Opción 2: MCTS VS Jugador En esta opción, el jugador podrá enfrentarse a un algoritmo de Montecarlo (MCTS). El jugador tomará decisiones estratégicas de manera interactiva mientras el algoritmo MCTS realizará simulaciones para determinar sus movimientos.
Opción 3: Minimax VS MCTS En esta opción, un algoritmo Minimax se enfrentará a un algoritmo de Montecarlo (MCTS). El algoritmo Minimax realizará una búsqueda exhaustiva en el árbol de juego hasta cierta profundidad, mientras que MCTS utilizará simulaciones aleatorias.
Opción 4: Minimax VS Jugador En esta opción, el jugador podrá enfrentarse a un algoritmo Minimax. El jugador tomará decisiones interactivamente mientras el algoritmo Minimax realiza una búsqueda exhaustiva en el árbol de juego para tomar sus decisiones.

Opción 5: Minimax VS Minimax En esta opción, dos algoritmos Minimax jugarán entre sí. Ambos algoritmos realizarán búsquedas exhaustivas en el árbol de juego y se desafiarán en una partida estratégica.

Para ejecutar el programa, simplemente nos ubicamos en el directorio correspondiente y utilizamos el comando "make" para compilar. Luego, utilizamos el comando "./connect4 <opcion> <intial_state>", donde "<opcion>" debe ser un número del 1 al 5, según la opción que desees jugar, y "<initial_state>" es una secuencia de números que representa la disposición inicial de las fichas en el tablero antes de que los algoritmos de búsqueda tomen decisiones.

La secuencia inicial "<initial_state>" permite configurar el tablero para comenzar el juego desde una posición específica, lo que brinda la oportunidad de simular partidas en diferentes etapas y analizar cómo los algoritmos toman decisiones en situaciones variadas.

A continuación, se presentarán los resultados obtenidos al evaluar los diferentes algoritmos en diversas situaciones y con distintos estados iniciales.

MCTS VS MCTS

Estado Inicial	Ganador	Tiempo
573621356135	MCTS 2 O	2.84637 seconds
624243215565644753161762 23	DRAW	3.56304 seconds
2213455412641157123667	MCTS 2 O	1.25443 seconds
3313455412641157123667	MCTS 2 O	0.34132 seconds
7225753343413131154411	MCTS 2 O	4.73674 seconds
37333774543316557562711	MCTS 1 X	3.58768 seconds

Análisis de MCTS VS MCTS: Observamos que el mejor tiempo fue de 0.34132 segundos y el peor caso fue de 4.73674 segundos, también podemos observar una MCTS 2 tuvo un mejor rendimiento que MCTS 1, ya que fue el que ganó la mayoría de las partidas, de 6 partidas, hubo 1 empate y 4 fueron ganadas por MCTS 2. En general, tuvieron un buen rendimiento en cuanto a tiempo, el promedio fue de 2.72159

MCTS VS Jugador

Estado Inicial	Ganador	Tiempo
573621356135	MCTS X	31.2041 seconds
624243215565644753161762 23	MCTS X	31.346 seconds
2213455412641157123667	MCTS X	5.57759 seconds
3313455412641157123667	MCTS X	11.4205 seconds
7225753343413131154411	MCTS X	54.4157 seconds
3733377454331655756271 1	MCTS X	29.1033 seconds

Análisis MCTS VS Jugador: En este caso se observa que todas las partidas fueron ganadas por el algoritmo MCTS, pero tenemos que recordar que esto depende de los datos proporcionados (estrategias usadas) por el jugador. El tiempo de decisión para el algoritmo en el mejor caso fue de 5.57759 segundos y en el peor caso fue de 54.4157 segundos, el tiempo promedio fue de 27.1778 segundos.

MCTS VS Minimax

Estado Inicial	Ganador	Tiempo
573621356135	KLL	KILL(30min)
624243215565644753161762 23	MINIMAX O	30.4038 seconds
2213455412641157123667	MINIMAX O	0.373343 seconds
3313455412641157123667	MINIMAX O	1.63722 seconds
7225753343413131154411	MINIMAX O	18.648 seconds
3733377454331655756271 1	MINIMAX O	7.66526 seconds

Análisis MCTS VS Minimax: en este caso obtuvimos un peor caso que se tuvo que detener la ejecución a los 30 minutos, esto se puede deber a que la implementación utilizada para minimax tomó mucho tiempo en explorar el árbol hasta cierta profundidad y

así poder tomar una decisión. Evaluando los otros casos se observa un mejor tiempo de 0.373343 segundos y el segundo peor caso 30.4038 segundos, el promedio de tiempo fue de 11.7455 sin incluir el proceso terminado forzosamente. El algoritmo de Minimax ganó las partidas frente al algoritmo de MCTS, lo que nos indica su capacidad de elección de estrategia y búsqueda.

Minimax VS Jugador

Estado Inicial	Ganador	Tiempo
573621356135	MINIMAX X	0.0800171 seconds
624243215565644753161762 23	MINIMAX X	42.7277 seconds
2213455412641157123667	PLAYER O	5.3385 seconds
3313455412641157123667	PLAYER O	2.9441 seconds
7225753343413131154411	MINIMAX X	154.844 seconds
37333774543316557562711	MINIMAX X	79.583 seconds

Análisis Minimax VS Jugador: observamos que de 6 partidas, 4 los ganó el algoritmo de Minimax y 2 por el jugador, pero como se mencionó anteriormente esto también depende de la estrategia que siga el jugador. En general, el algoritmo tomó estrategias superiores al jugador. El mejor tiempo fue de 0.0800171 segundos y el peor tiempo fue de 154.844 segundos y el tiempo promedio fue de 47.58622

Minimax VS Minimax

Estado Inicial	Ganador	Tiempo
573621356135	MINIMAX 1 X	0.0524868 seconds
624243215565644753161762 23	MINIMAX 1 X	52.9532 seconds
2213455412641157123667	MINIMAX 2 O	0.19679 seconds
3313455412641157123667	MINIMAX 2 O	1.48509 seconds
7225753343413131154411	MINIMAX 1 X	513.09 seconds

3733377454331655756271 1	MINIMAX 1 X	11.5338 seconds

Análisis Minimax VS Minimax: en este caso observamos que de 6 partidas Minimax 1 ganó 4 y Minimax 2 ganó 2 partidas. Comparando con MCTS VS MCTS, se tiene una mayor variabilidad en el tiempo con respecto a un estado inicial y al otro. El mejor caso fue de 0.0524868 segundos y el peor caso tomó 513.09 segundos, lo que da un promedio de 96.5519 segundos.

Conclusiones

Basándonos en los resultados de todas las corridas, podemos concluir lo siguiente sobre los algoritmos de Minimax y Montecarlo en el juego Connect 4:

- En el enfrentamiento "MCTS vs MCTS", ambos jugadores MCTS logran empates en ciertos estados iniciales, lo que sugiere estrategias similares. Sin embargo, en otras partidas, uno de los jugadores MCTS obtiene la victoria.
- En el escenario "Minimax vs Montecarlo", el algoritmo de Minimax muestra un rendimiento sólido y gana la mayoría de las partidas. El tiempo de decisión puede variar significativamente para ambos algoritmos.
- En "Montecarlo vs Jugador", el algoritmo de Montecarlo domina y gana todas las partidas contra el jugador humano. Sin embargo, su tiempo de decisión es más largo cuando se enfrenta al jugador humano.
- En "Minimax vs Jugador", el algoritmo de Minimax también muestra un rendimiento sólido y gana la mayoría de las partidas contra el jugador humano. Tiende a tomar decisiones más rápidas que el algoritmo de Montecarlo en este escenario.
- En "Minimax vs Minimax", los resultados son variados, con diferentes variantes del algoritmo Minimax obteniendo victorias en distintas partidas. Los tiempos de decisión varían según la complejidad de las posiciones iniciales y el tamaño del árbol de juego.

En general, tanto Minimax como Montecarlo muestran un rendimiento sólido en Connect 4. La elección entre ellos dependerá de las preferencias específicas y la complejidad del juego. Minimax realiza búsquedas exhaustivas, mientras que Montecarlo utiliza simulaciones aleatorias, lo que afecta su rendimiento y tiempo de decisión en diferentes escenarios. Otra característica relevante es que el algoritmo de Montecarlo cuando se ejecuta contra el mismo da un tiempo muchísimo menor con respecto al Minimax ejecutado

con sí mismo, pero al hacer la ejecución de un algoritmo contra el otro siempre destaca Minimax debido a su estrategia.