Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información
CI-5437 Inteligencia Artificial I
Trimestre Abril - Julio 2023

Informe Proyecto II

Estudiantes:

Br. Gamboa, Roberto

Br. Bandez, Jesús

Carnés:

16-10394

17-10046

Profesor

Carlos Infante

21 de junio de 2023

Índice

Introducción

En el ámbito de la inteligencia artificial y la toma de decisiones en juegos, los árboles de juego y los algoritmos de solución desempeñan un papel fundamental. Los árboles de juego proporcionan una representación estructurada de las posibles decisiones a tomar y sus consecuencias en un juego determinado, permitiendo así el análisis exhaustivo de las diferentes estrategias y movimientos posibles.

El presente informe se centra en el estudio de los árboles de juego mediante el uso de algoritmos básicos de solución utilizados comúnmente en este campo, tales como Negascout, Scout y Negamax. Estos algoritmos son ampliamente conocidos y se han aplicado en la resolución de problemas en juegos de estrategia, como el ajedrez, el go, las damas y muchos otros.

A lo largo de este informe, se presentaran y compararan los resultados de ejecutar los algoritmos antes mencionados sobre una representación 6x6 del juego Othello para concluir cual de ellos es el mas eficiente además expandir nuestro conocimiento sobre árboles de juego y la búsqueda de soluciones en los mismos.

Decisiones de diseño e implementación

Para abordar el presente proyecto, inicialmente se completó la representación 6x6 del juego Othello suministrada, donde los métodos outflank y move, que permiten saber cuándo de puede tomar una pieza del rival y cuales piezas deben ser capturadas al colocar una propia no realizaban las comprobaciones correspondientes cuando se debía capturar en diagonal.

Para arreglar los métodos antes mencionados se siguió como ejemplo la forma en que se examinaban las piezas horizontal y verticalmente, inspeccionando las matrices dia1 y dia2

Al completar la representación del Othello, se realizaron pruebas de que los métodos funcionaran acorde a lo que se deseaba con el archivo main suministrado para posteriormente implementar los algoritmos de búsqueda para arboles de juego.

Los algoritmos de búsqueda implementados son los siguientes:

- Negamax, versión minmax
- Negamax, con poda α β
- Scout
- Negascout

Todos estos algoritmos se implementaron en el archivo main.cc y se realizaron pruebas sobre la variación principal del problema para verificar que trabajaran correctamente. Los resultados de ejecutar los 4 algoritmos se encuentran en la sección siguiente

Resultados

Para obtener los siguientes resultados se realizaron las llamadas a cada algoritmo con una profundidad de 35, además para cada algoritmo se usaron adicionalmente tablas de transposición, que permiten examinar un mayor numero de estados ya que en ellas se guardan los estados previamente visitados para un fácil acceso al valor de los mismos. Se limitó la capacidad de las tablas de transposición a 32 estados.

Con cada algoritmo se partió desde el estado final de la partida de un juego de Othello, se examinan todos los nodos hijos del estado actual para seleccionar aquel con el mejor valor y posteriormente llamar recursivamente al método consigo mismo para evaluar a los hijos de dicho estado, procedimiento que se repite hasta alcanzar la raíz del árbol de juego. Los algoritmos implementados se ejecutaron durante 1 hora cada uno, y los resultados obtenidos con cada uno se resumen en las siguiente tabla:

Algoritmo	Profundidad máxima	Nodos generados	Nodos expandidos	Tiempo de ejecución (segundos)
Negamax (versión minmax)	17	1.305.006.090	614.384.516	1.711,4
Negamax (versión minmax) con TT	17	1.305.006.090	614.384.516	2.082,49
Negamax (poda α β)	12	2.661.793.153	1.746.703.473	1.125,38
Negamax (poda α β) con TT	12	2.437.181.576	1.544.108.632	1.145.81
Scout	11	5.105.257	4.760.651	1.679.81
Scout con TT	11	4.697.691	4.449.146	1.689.17
Negascout	11	3.061.101.779	1.959.116.476	1.368.93
Negascout con TT	12	125.558.102	80.864.583	158,832

De los resultados presentados en la tabla se pueden extraer las siguientes conclusiones:

- La aplicación de tablas de transposición para agilizar los cálculos en cada algoritmo es notable, permite explorar el árbol de juego generando y expandiendo menos nodos por cada profundidad, en los resultados presentados se nota claramente en el algoritmo Negascout con TT, en los otros algoritmos no se presenta una gran variación debido a la limitación impuesta sobre la cantidad de estados que la tabla puede almacenar

- Al aplicar la estrategia de poda α β se pueden explorar un mayor numero de nodos y llegar a una profundidad mayor. Esto es evidente en el algoritmo Negascout, que aplica dicha estrategia y en el algoritmo Negamax a menor medida, donde se alcanza una mayor profundidad que su contra parte sin poda α β

- El algoritmo Scout es una muy buena alternativa entre todos los algoritmos, ya que alcanza profundidades altas en tiempos comparables a los del algoritmo Negamax.

Conclusión

Al implementar los algoritmos de búsqueda, hemos logrado ampliar nuestro conocimiento sobre el modelo de árboles de juego y como usarlos para obtener estrategias óptimas para la resolución de problemas. Mediante el planteamiento de un problema bajo este modelo podemos analizar estrategias, evaluar movimientos y anticipar consecuencias, lo que nos lleva a tomar decisiones más eficientes que nos lleven a la solución del mismo.

Mediante el uso de algoritmos para la búsqueda de soluciones en árboles de juego, se puede mejorar significativamente el rendimiento y la eficiencia en la toma de decisiones. Además, hemos explorado el uso de tablas de transposición junto con estos algoritmos como una técnica para mejorar aún más su rendimiento. Estas tablas nos permiten almacenar y reutilizar información sobre las posiciones ya evaluadas, evitando así cálculos repetidos y acelerando el proceso de búsqueda. Mediante el uso de tablas de transposición, hemos observado una reducción en el tiempo de ejecución y por ende una obtención más rápida de la solución.