Proyecto 2 IA

Ajedrez de Alicia

Integrantes:

Carlos Stiven Ruiz Rojas - 2259629
Jhony Fernando Duque Villada - 2259398
Jairo Andres Gomez Cardona - 2259332
Juan Camilo Diaz Valencia - 2259583
Juan David Rojas Narvaez - 2259673

Docente

Joshua David Triana Madrid

Universidad del Valle - Seccional Tuluá

2024

Universidad del Valle

Representación del problema

Como situación problema se presenta el modo de juego de ajedrez “Alicia”
El modo de juego consiste en lo siguiente:

Dos tableros son puestos a disposición de los jugadores, quienes ubican las fichas en la posición inicial del tablero A como en un juego de ajedrez clásico.

Los movimientos de las fichas serán los del ajedrez clásico con la diferencia de que las fichas al mover serán ubicadas en el tablero opuesto.

Para capturar las fichas del adversario la única forma permitida es que ambas piezas estén en el mismo tablero, ya sea el tablero A o el tablero B.

Las condiciones de empate y victoria continúan siendo las condiciones clásicas del ajedrez.

El objetivo del proyecto es diseñar un programa en Python que a través del algoritmo minimax y la implementación de poda alfa-beta y una heurística se implementará un jugador que realiza movimientos en contra del jugador B, quien es humano, evaluando las opciones disponibles y escogiendo el movimiento más óptimo.

Implementación de la solución:

Para la implementación de la solución, se diseña un algoritmo con las siguientes características:

1. Heurística: Para la definición de la heurística tenemos la **diferencia material de cada jugador.** Esta representa la calidad del estado del juego para cada jugador basado en su color.

   El valor será almacenado como atributo de la clase EvaluarEstado, para posteriormente evaluar en base a ella el estado del juego por parte de otros métodos.

   La heurística también se ve afectada por diversos estados de juego, como en el caso de jaque mate para un jugador, cuando un jugador recibe un jaque mate, retorna un puntaje muy alto positivo, indicando victoria, de lo contrario retorna un valor muy bajo negativo.

   Si el juego tiene un estado de empate o **Tablas**, se devuelve 0.

   Si el juego no ha terminado con las dos condiciones anteriores, entonces se calcula el valor material de ese estado de juego.

Universidad del Valle

2. **Cálculo de la heurística**:

Este método calcula la ventaja material del jugador, está dado por la suma del N valor de cada pieza por la cantidad que se encuentre en el tablero.

h(n) = Ventaja material jugador A - Ventaja material jugador B

h(n) = [(n * Valor_Peon) + (n * Valor_Alfil) + (n* Valor_Torre) + (n*Valor_Caballo) + n*Reina)] - [(n * Valor_PeonR) + (n * Valor_AlfilR) + (n* Valor_TorreR) + (n*Valor_CaballoR) + n*ReinaR)]

Con esta heurística y el tamaño N que se le asigna a la profundidad del árbol, se permite que la Inteligencia artificial pueda explorar hasta N variantes por cada movimiento a ejecutar buscando siempre la ruta que mayor ganancia genere a favor del jugador.

Descripción de la lógica:

El puntaje inicial es 0. A partir de aquí se recorren todas las piezas obtenidas mediante el método state.yield_all_pieces(). Y para cada pieza, obtiene el valor mediante el método get(piece.type.name, 0). Que es un diccionario que fue definido con los valores de cada pieza.

Si la pieza es del jugador que está en evaluación, el valor se suma. Si la pieza es del oponente, se resta del puntaje.

Finalmente, se retorna el puntaje total, que refleja la diferencia a favor o en contra del jugador que se está evaluando.

Algoritmo Minimax:

El árbol explora las jugadas posibles hasta una cierta profundidad, evalúa las jugadas por medio de la heurística para poder puntuar los estados.

La poda alfa beta optimiza la búsqueda al descartar ramas que no puedan mejorar el resultado.

También se hace uso de MoveSelector, que selecciona la mejor jugada mediante concurrencia con ThreadPoolExecutor, esto genera de forma concurrente todas las jugadas legales.

Evalúa cada jugada por medio de Max y Min. Finalmente, selecciona la jugada con la mejor puntuación dependiendo de si se maximiza o minimiza

Universidad del Valle

Poda Alfa-Beta

La clase **“MinimaxStrategy”** implementa el algoritmo Minimax con Poda Alpha-Beta, una técnica utilizada en juegos de decisión (como el ajedrez) para encontrar la mejor jugada posible. La poda Alpha-Beta optimiza la búsqueda reduciendo el número de nodos evaluados.

Funcionamiento:

1. **Evaluación del estado del juego:** Se utiliza un evaluador externo llamado “Evaluator” para asignar una puntuación a cada estado del juego.
2. **Condiciones de finalización:** Si se alcanza la profundidad máxima o el juego termina, se devuelve el puntaje del estado actual.
3. **Recursión:** El algoritmo explora los movimientos posibles para ambos jugadores alternando entre:
    a. **Maximización:** Busca el puntaje máximo para el jugador maximizador.
    b. **Minimización:** Busca el puntaje mínimo para el jugador minimizador.
4. **Poda Alfa - Beta:** Evita evaluar ramas innecesarias cuando se determina que no afecta el resultado final.

Flujo de Ejecución

1. El método **“Alpha_beta”** es el punto de entrada, recibe el estado inicial del juego y los valores iniciales de alpha y beta.
2. Dependiendo del valor de maximizing:
    - Si es Verdadero, delega el cálculo a “maximize”.
    - Sí es Falso, delega el cálculo a “minimize”.
3. Los métodos **“maximize”** y **“minimize”**:
    - Iteran sobre los movimientos posibles, evaluando recursivamente los estados futuros.
    - Actualizan los valores de alpha y beta.
    - Terminan anticipadamente mediante poda Alpha-Beta si encuentran estados irrelevantes para la decisión óptima.

Ejemplo de Comportamiento

Supongamos un estado donde:

- El maximizador tiene 2 posibles movimientos:
    - move1: conduce a un puntaje de 10.
    - move2: conduce a un puntaje de 15.

- El minimizador en un nivel futuro podría reducir este puntaje dependiendo de sus movimientos.

La lógica del algoritmo:

1. Evaluará move1, asignándole un puntaje temporal de 10 y actualizando alpha.
2. Evaluará move2, asignándole un puntaje de 15 y descartando ramas inferiores mediante poda Alpha-Beta.
3. Retornará 15 como el mejor resultado para el maximizador.

INTERFAZ DE USUARIO (LOS DOS TABLEROS)

Para la interfaz gráfica del proyecto se implementó la API "**Alice Chess**" que brinda la posibilidad de tener los dos tableros y los movimiento de cada ficha que son necesarias para la ejecución de una partida en el modo de juego de ajedrez de Alicia.

Su guía de instalación y documentación se encuentra en el siguiente repositorios de GitHub:

Enlace al repositorio

En base a la documentación brindada por el creador de la librería se logró adaptar la lógica de la IA, con el uso de Árboles, Minimax y poda Alfa-beta.

Conclusión

El proyecto implementa una solución eficiente y robusta para el modo de ajedrez "Alicia" utilizando el algoritmo **Minimax** con **Poda Alpha-Beta**, optimizado con un enfoque heurístico basado en la ventaja material. Esta solución permite a la Inteligencia Artificial (IA) tomar decisiones óptimas en cada turno, evaluando jugadas a través de la diferencia de valores de piezas entre los jugadores.

1. **Representación del Problema:**
   La principal diferencia del ajedrez "Alicia" respecto al ajedrez clásico radica en el uso de **dos tableros** y en la regla especial de captura, que requiere que las piezas estén en el mismo tablero. Esto añade una nueva dimensión estratégica que el algoritmo debe manejar.

2. **Evaluación del Estado:**

   - Se implementa una heurística que evalúa la **ventaja material** de un estado de juego.
   - La heurística incorpora condiciones de **jaque mate** (retornando valores extremos) y **empate** (retornando 0).

- La fórmula utilizada calcula la suma ponderada del valor de las piezas presentes en el tablero, diferenciando entre el jugador evaluado y su oponente.

3. **Funcionamiento del Minimax con Poda Alpha-Beta:**

   - El algoritmo recorre las jugadas posibles hasta una profundidad definida.
   - En cada nivel, se alterna entre **maximización** (movimiento del jugador evaluado) y **minimización** (movimiento del oponente).
   - La **poda Alpha-Beta** optimiza el proceso descartando ramas que no influyen en el resultado final, reduciendo el número de nodos evaluados.

4. **Resultados Esperados:**

   La IA será capaz de:

   - Evaluar el estado del juego considerando las reglas específicas del ajedrez "Alicia".
   - Seleccionar jugadas óptimas basadas en la heurística de ventaja material.
   - Optimizar el tiempo de búsqueda gracias a la poda Alpha-Beta y la concurrencia.

En conclusión, la implementación logra integrar técnicas avanzadas de búsqueda en árboles y optimización heurística, resolviendo el problema con una IA eficiente y capaz de competir contra un jugador humano en el modo de ajedrez "Alicia".