Proyecto de Inteligencia Artificial: Jugador Autónomo para HEX

Kendry Javier del Pino Barbosa

Abril 2025

Estrategia

La estrategia implementada en este proyecto combina dos técnicas principales: **Minimax con poda alfa-beta** y una **heurística basada en A***. La elección de estos métodos depende de la fase de la partida, priorizando la eficiencia y la profundidad según la complejidad del tablero. Además, se incorporan optimizaciones clave, como:

- **Uso de caches:** Se utilizan caches para almacenar evaluaciones de estados ya procesados, evitando el recálculo redundante en distintas ramas del árbol de búsqueda.
- **Ordenación de jugadas:** Antes de explorar los nodos mediante Minimax, las jugadas posibles se ordenan según criterios estratégicos (proximidad al centro, conexión con piezas propias, bloqueo de jugadas enemigas y posiciones en los bordes críticos). Esto permite mejorar la eficiencia en la poda alfa-beta.

Esta combinación permite que el agente actúe de manera rápida y efectiva, ajustando la profundidad de búsqueda de forma dinámica en función del estado del juego.

1. Búsqueda Minimax con Poda Alfa-Beta

El agente explora el árbol de jugadas usando el algoritmo *Minimax* complementado con poda alfa-beta. Este enfoque:

- Permite descartar ramas del árbol que no resultarán en una mejora del resultado, reduciendo el tiempo de cómputo.
- Se beneficia del ajuste dinámico de la profundidad, lo cual es crucial para gestionar los altos factores de ramificación en las primeras etapas del juego.

2. Evaluación del Estado y Uso de Caches

La función `evaluate_game_state` asigna un puntaje a cada configuración del tablero combinando:

- **Costos de Conexión:** Se estima el número mínimo de movimientos necesarios para conectar los lados correspondientes, utilizando una búsqueda A*.

- **Conexiones Directas:** Se cuentan las conexiones existentes entre las piezas.
- **Amenazas:** Se penaliza la situación si el oponente se acerca a un estado ganador.

Para evitar realizar cálculos repetitivos, se utilizan caches:

- `eval_cache`: Guarda la evaluación de estados previamente analizados. Esto acelera la búsqueda, ya que permite reutilizar resultados en estados idénticos que se encuentran en distintas ramas.

3. Ordenación de Jugadas

Antes de la exploración en el árbol de Minimax, se ordenan las posibles jugadas utilizando la función `_sort_moves`. Esta ordenación se basa en:

- **Proximidad al centro:** Se prefiere jugar en el centro del tablero.
- **Conexión con Piezas Aliadas y Enemigas:** Se da mayor puntuación a movimientos adyacentes a piezas propias o que bloqueen la expansión del oponente.
- **Posición en Bordes Relevantes:** Se otorga un bonus si el movimiento se ubica en los bordes que son críticos para la conexión (para el Jugador 1 en el eje horizontal y para el Jugador 2 en el eje vertical).

4. Función Heurística y Búsqueda A*

La evaluación de rutas se realiza mediante búsqueda A*. La función heurística, esencial en este proceso, estima la distancia restante al objetivo utilizando una variante unidimensional de la **distancia de Manhattan**. Específicamente:

$$h(x,y) = \begin{cases} |\text{board\_size} - 1 - y|\,, & \text{para el Jugador 1 (objetivo: conexión horizontal)} \\ |\text{board\_size} - 1 - x|\,, & \text{para el Jugador 2 (objetivo: conexión vertical)} \end{cases}$$