Reporte del Proyecto de Inteligencia Artificial: Hex Game

Luis Alejandro Arteaga Morales

11 de abril de 2025

1. Estrategia General (MordecaiBot)

El agente implementado primero intentará descubrir si alguno de los movimientos disponibles en el tablero lo conlleva a una victoria inmediata, de ser así juega este movimiento, en otro caso dispone de un límite de tiempo (`time_limit`) para realizar su movimiento. Este parámetro asegura que no consuma tiempo excesivo explorando el espacio de búsqueda, permitiendo un balance entre rendimiento y jugadas coherentes. Durante este intervalo, el agente emplea **Memory-enhanced Test Driver** a.k.a **MTD(f)** con una **transposition table** para determinar la mejor jugada posible. En caso de que el tiempo límite expire sin encontrar una solución óptima y existan movimientos disponibles en el tablero, el agente seleccionará un movimiento prometedor, priorizando aquellos con mayor potencial estratégico.

1.1. Movimientos Prometedores

Los movimientos prometedores son identificados como aquellas posiciones en el tablero que presentan una mayor posibilidad de generar un resultado favorable, basándose en su proximidad a posiciones ya ocupadas. Este enfoque permite calcular movimientos de manera eficiente, especialmente en las fases iniciales de la partida, donde el tablero se encuentra más abierto y el espacio de búsqueda es considerablemente amplio.

En este contexto, los movimientos prometedores corresponden a las posiciones adyacentes a las ya ocupadas, ordenadas según su distancia al centro del tablero. Este criterio de ordenación busca maximizar la conectividad y el control estratégico del tablero.

1.2. Selección de Movimientos en Caso de Ausencia de Prometedores

Si no se identifican movimientos prometedores, el agente recurre al conjunto de posiciones libres del tablero, obtenidas mediante el método `get_possible_moves`. En este caso, se selecciona la primera posición disponible, garantizando que el agente siempre realice un movimiento válido.

Este enfoque en el diseño del agente **MordecaiBot**, busca un equilibrio entre la exploración exhaustiva del espacio de búsqueda y la capacidad de tomar decisiones rápidas y sensatas en escenarios donde el tiempo es un recurso limitado.

2. Evaluación de los Estados del Tablero para MTD(f) (Minimax)

El algoritmo asigna un valor a cada estado del tablero utilizando una función de evaluación (`evaluate`). Esta función considera los siguientes factores:

1. **Distancia al Camino Más Corto** (`shortest_path_distance`):
   - Evalúa la distancia mínima necesaria para que el jugador actual y el oponente completen su conexión en el tablero.
   - **Ponderación**: $(d_{\text{op}} - d_{\text{jug}}) \times 50$
   - Un menor valor para el jugador actual y un mayor valor para el oponente resultan en un puntaje más alto.
2. **Distancia al Segundo Camino Más Corto** (`two_distance`):
   - Considera caminos secundarios que podrían ser útiles para reforzar la estrategia del jugador o bloquear al oponente.
   - **Ponderación**: $(s_{\text{op}} - s_{\text{jug}}) \times 20$
   - Promueve la creación de múltiples opciones estratégicas mientras penaliza las del oponente.
3. **Bonificación por Conectividad**:
   - Calcula una bonificación basada en la proximidad de las piezas del jugador al centro del tablero, incentivando una mayor conectividad y control estratégico.
4. **Bonificación por Bloqueo** (`blocking_bonus`):
   - Evalúa la capacidad del jugador para bloquear las jugadas del oponente, dificultando su progreso hacia una conexión completa.
   - **Cálculo**: Se basa en la distancia mínima del oponente al camino más corto (`opp_distance`) y asigna una bonificación proporcional.
   - **Ponderación**: $w_b \times (d_{\text{op}}/100)$
   - Este factor incentiva movimientos que obstaculicen estratégicamente al oponente, aumentando las probabilidades de éxito del jugador.

3. Iterative Deepening

El agente utiliza un enfoque de búsqueda con profundización iterativa para explorar el espacio de búsqueda de manera incremental.

4. Transposition Table

La **Transposition Table** es una estructura de datos utilizada para almacenar información sobre estados previamente evaluados del tablero, con el objetivo de evitar cálculos redundantes y acelerar el proceso de búsqueda. En el contexto del agente **MordecaiBot**, esta tabla guarda:

- Valor de evaluación
- Profundidad de búsqueda alcanzada
- Tipo de límite (`exact`, `lower`, `upper`)
- Mejor movimiento asociado

Cada estado del tablero se representa mediante un hash único generado por el método `get_board_hash`. Durante la búsqueda, si un estado ya está en la tabla y su profundidad es suficiente, se reutiliza la información almacenada para evitar evaluaciones adicionales.

5. Null Window Search

El agente **MordecaiBot** utiliza una técnica conocida como **Null Window Search** dentro del algoritmo **MTD(f)** para optimizar la búsqueda de movimientos. Esta técnica consiste en realizar búsquedas con un rango de valores (α y β) muy estrecho, típicamente de tamaño cero, lo que permite determinar rápidamente si un movimiento es mejor o peor que un valor de referencia.

6. Observaciones

- El agente trata de jugar en **5.0s**
- A medida que crecen las dimensiones del tablero, la calidad de las jugadas se reduce
- Los parámetros se pueden tunear para que juegue más rápido o "piense"mejor
- **Es recomendable pasar una copia del tablero, ya que se usan algunos algoritmos que modifican la copia dada del tablero.**