Agente inteligente para el juego Hex

Eduardo Brito Labrada

Facultad de Matemática y Computación
Universidad de La Habana

11 de abril de 2025

1. ¿Cómo se ejecuta?

La clase donde el jugador está implementado tiene como nombre `PopPlayer` que hereda de la clase `Player`. Para inicializar el jugador lo único necesario es darle el `player_id` que tendrá durante el juego.

Para que el jugador realice una jugada se debe llamar a la función `play` que recibe el tablero actual y opcionalmente el tiempo esperado para realizar una jugada (por defecto, este tiempo está fijado en 10 segundos), está función retornará una tupla que representa la casilla en la que el jugador hará la jugada.

2. ¿Cómo funciona?

Primero, se representa la matriz como un grafo no dirigido y conexo, donde hay una arista entre dos nodos si son casillas adyacentes (se puede llegar de una casilla a la otra). Este grafo no se construye explícitamente pues es suficiente con tener las direcciones a las cuales puede llegar cada casilla, pero por comodidad a la hora de representar la matriz se numera cada casilla con un número de $[0, N^2 - 1]$, donde N es la cantidad de filas y columnas del tablero.

El jugador utiliza el algoritmo mini-max with alpha-beta pruning. Por cuestión de complejidad del algoritmo se visitan a lo más dos niveles de profundidad con respecto al tablero actual. Durante el algoritmo, además de tener en cuenta estados terminales (debido a la profundidad o un estado terminal del juego) también se tiene en cuenta el tiempo que se ha demorado el jugador en realizar la jugada actual. En el algoritmo el jugador 1 quiere maximizar, mientras el jugador 2 quiere minimizar; una vez que se llega a un estado terminal, se evalúa que tan bueno es el tablero resultante para el jugador 1 (en otras palabras, que tan "probable" es que gane el jugador 1 terminando con este tablero).

2.1. Heurística

Para calcular, que tan bueno es el tablero resultante para el jugador 1 se tienen en cuenta varios factores, que se mencionan a continuación.

1. En el tablero, que tan largo es el camino mínimo en el grafo (mínima cantidad de jugadas) que tiene que recorrer el jugador 2 con respecto al camino mínimo que tiene que recorrer el jugador 1: intuitivamente esto nos da una idea de la cantidad de si ambos jugadores decidieran

moverse por su camino de longitud mínima quien ganaría. Además, mientras mayor sea el camino mínimo del jugador 2 con respecto al camino mínimo del jugador 1 significa que es más probable que el jugador 1 tiene más posibilidades de ganar si juega de forma óptima.

2. La proporción entre la cantidad de caminos de longitud mínima en el grafo que tiene el jugador 1 con respecto a la cantidad de caminos de longitud mínima en el grafo que tiene el jugador 2: esto nos da una idea de cuánto importa (o no) que el jugador 2 se dedique a bloquear los posibles caminos mínimos del jugador 1 (y viceversa), mientras mayor cantidad de caminos mínimos tenga un jugador significa que tiene mayor cantidad de formas de ganar siguiendo una secuencia óptima de movimientos.

Esta es básicamente la heurística que se tiene en cuenta para evaluar la posición de un tablero. Además, puede pasar que varias jugadas conduzcan al mismo tablero de alguna forma, y por tanto su valor de heurísitica va a ser el mismo, por lo que es necesario definir un criterio para escoger entre todas esas jugadas. Para ello, se decidió incluir más valores a la heurística que en la práctica tuvieron mejores resultados.

- De todas las posibles jugadas $(x_1, y_1), (x_2, y_2), \ldots, (x_k, y_k)$, escoger a (x_i, y_i) que de alguna forma minimice el valor del camino de longitud mínima que favorece al jugador y que maximiza el valor del camino de longitud mínima de su oponente.

Además, se probó y tuvo buenos resultados en la práctica randomizar las posibles jugadas, para que en caso de que no se puedan probar todas las posibles jugadas (por el tiempo límite y la profundidad), hubiera un poco más de probabilidad de que jugadas que pueden ser óptimas para algún estado del tablero se consideraran como una posible solución en algún momento.

La fórmula general de la heurística para una jugada (x, y) queda de la siguiente forma:

$$alpha\_beta(board, depth = 2) + (total\_verts - player\_sp) \cdot 7 + opponent\_sp \cdot 5$$

donde *board* es el tablero después de hacer la jugada (x, y); *alpha_beta* es el algoritmo mini-max with alpha-beta pruning, en este caso se llamada con profundidad 2; *total_verts* es la cantidad de vértices del grafo mencionado al inicio; *player_sp* es el camino de longitud mínima del jugador después de la jugada (x, y), mientras que *opponent_sp* es el camino de longitud mínima del oponente, notar que estos valores se multiplican por una constante que le dan significancia a un valor u a otro.

En el algoritmo *alpha_beta* para evaluar el estado del tablero se utiliza la siguiente fórmula:

$$1000 * ((min\_path2 + 1)/(min\_path1 + 1)) + (num\_path1 + 1)/(num\_path2 + 1) * 18$$

donde *min_path* y *num_path* representan la longitud del camino de longitud mínima y la cantidad de caminos de longitud mínima que hay en el tablero para cada jugador, respectivamente. Estos valores se utilizan sobre todo en proporción para medir cuán mejor es un valor sobre otro en el tablero y se multiplican por una constante para darle signficancia.

Para calcular cada uno de los valores que dependen del grafo generado, se utilizó 0-1 BFS (multisource) que funciona más rápido que el algoritmo de Dijkstra, para contar la cantitdad de caminos de longitud mínima se utilizó Programación Dinámica.