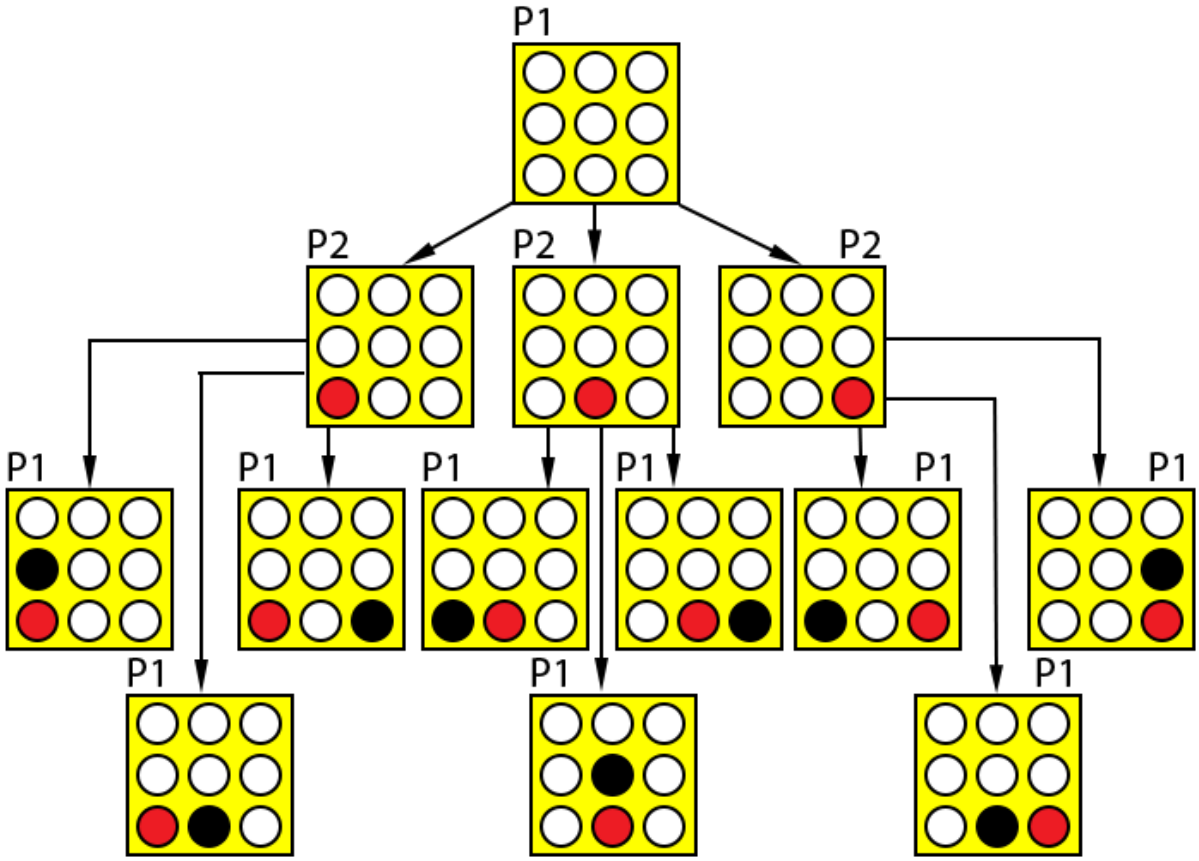
Desconecta 4 BOOM

Sergio Cervilla Ortega



1. **Análisis del problema.**
2. **Descripción de la solución planteada.**

1. Análisis del problema.

Esta práctica está basada en el clásico juego ***Connect Four*** pero con algunos cambios significativos, como pueden ser:

- En el juego original la meta del juego era alinear **4 fichas del mismo color**, es decir, del mismo jugador. En este momento habríamos ganado la partida. En nuestro caso el objetivo es justamente el contrario, debemos evitar alinear 4 fichas del mismo color, por tanto, queremos que gane, en la versión clásica del juego, el contrincante.
- Se ha añadido un tipo especial de ficha, llamada ***ficha bomba***, la cual se proporcionará a los jugadores cada 5 turnos siempre y cuando no tengan ya una ficha en el tablero. La utilidad de esta ficha es eliminar todas las fichas de nuestro color de la fila, haciendo que caigan las fichas que tenemos encima nuestra, tanto si son nuestras como del enemigo. Cabe destacar que detonar la bomba consumirá un turno y por tanto no podremos poner una ficha en ese turno.

Para abordar este problema debemos de utilizar técnicas de búsqueda con adversario, en concreto, el objetivo de esta práctica es la implementación de la ***Poda Alfa-Beta con profundidad limitada****,* dotando de esta manera a nuestro agente con un comportamiento deliberativo y proporcionándole alguna posibilidad de ganar a un humano.

A nivel de implementación, el *tablero de juego* va a estar representado como una matriz llamada ***maze_*** y por tanto la esquina superior izquierda será la coordenada (0,0) y la esquina inferior derecha, la (6,6). Por tanto lo que aparenta estar debajo en el simulador realmente está en las coordenadas más altas de nuestra matriz.

2. Descripción de la solución planteada.

Para solucionar este problema debemos de comentar los siguientes métodos:

Métodos <u>check</u>.

Para comenzar, reciben como parámetros:

- **const Environment &estado:** El tablero de juego actual.
- **int fila, int columna:** Ambos forman la coordenada de la ficha a analizar.
- **int jugador:** Identificador del jugador.
- **int secuencia_meta:** Longitud de la secuencia a buscar.

La funcionalidad es simple: Comprueban si existe una secuencia de longitud *secuencia_meta* sobre el tablero *estado* y en concreto sobre las coordenadas *(fila,columna).* Como los propios nombres lo indican, los métodos comprueban la vertical, horizontal o diagonal incluida en su nombre, como por ejemplo, *checkVerticalAscendente* busca en las verticales ascendentes.

Para comprobarlo, nos situamos en las coordenadas pasadas como parámetro y vamos aumentando un contador llamado ***secuencia_actual*** mientras que las fichas sean nuestras, en el caso de que sea del contrario, paramos. Después de esto comprobamos si la secuencia que hemos obtenido es igual o mayor que la que hemos pasado como parámetro, si es así, devolvemos 1 indicando que hemos encontrado una secuencia que como mínimo tiene longitud *secuencia_meta.* En el caso de las diagonales, volvemos a repetir este proceso e incrementamos la variable llamada *total* para devolverla y que como máximo podrá valer 2, indicando que existe sobre esa ficha una secuencia diagonal ascendente y a la vez descendente.

Dentro de las funciones check tenemos una más especial, llamada ***checkAll,*** la cual es la responsable de devolver el número de secuencias de longitud ***longitud_secuencia*** sobre todas las fichas del jugador ***jugador*** y sobre el tablero ***estado***. Esta función devuelve el número total de secuencias de longitud ***longitud_secuencia*** para todas y cada una de nuestras fichas del tablero.

Método <u>Valoración.</u>

Este método es el que vamos a utilizar en la poda para comprobar la puntuación total del tablero y esta se va a calcular de la siguiente manera:

- **Primero:** Almacenamos en variables el número de secuencias de 4, 3 y 2 fichas sobre nuestro jugador.
- **Segundo**: Almacenamos en variables el número de secuencias de 4, 3 y 2 fichas del contrincante.
- **Tercero:** Calculamos el valor heurístico multiplicando las secuencias largas del rival y las más cortas nuestras por números altos, las medianas de ambos jugadores por un número algo más pequeño y por último las más largas nuestras y las más cortas suyas por un valor muy pequeño. De esta manera, siempre vamos a buscar hacer nosotros las secuencias más cortas y que las del rival sean más largas.

Poda <u>Alfa-Beta.</u>

Nuestra poda se realiza mediante recursividad e irá avanzando secuencialmente en los niveles de profundidad del árbol de estados hasta llegar a la máxima profundidad permitida en esta práctica, que es **profundidad 8** produciéndose de esta manera el efecto horizonte.

Nuestro algoritmo consiste en:

- Si hemos llegado a la profundidad tope, evaluaremos el estado del tablero actual y le asignaremos una puntuación, calculada mediante la función ***Valoración*.**

- Si no hemos alcanzado la profundidad máxima, creamos un nuevo tablero, que será una copia del actual, y le aplicamos uno de los posibles movimientos. Ahora es cuando empezaremos a distinguir entre los tipos de nodo, los que se tienen que maximizar y los que se tienen que minimizar. Para distinguirlos utilizaremos los pares como nodos a maximizar y los impares como nodos a minimizar.

- Si estamos situados en un nodo a maximizar: Llamamos de nuevo a la función de la poda y almacenamos este valor en una variable temporal. Al estar situados en un nodo a maximizar comparamos el valor de la variable ***temporal*** con el valor de **alfa.** Si el valor de temporal es mayor que alfa,

actualizamos alfa y la acción a devolver y si no, podamos. Finalmente cuando beta sea menor o igual que alfa, devolvemos alfa.

- Si estamos situados en un nodo a minimizar: Llamamos de nuevo a la función de la poda y almacenamos este valor en una variable temporal. Al estar situados en un nodo a minimizar comparamos el valor de la variable ***temporal*** con el valor de **beta.** Si el valor de temporal es menor que beta, actualizamos beta y la acción a devolver y si no, podamos. Finalmente cuando beta sea menor o igual que alfa, devolvemos beta.