MEMORIA PRÁCTICA 3
DESCONECTA 4 BOOM

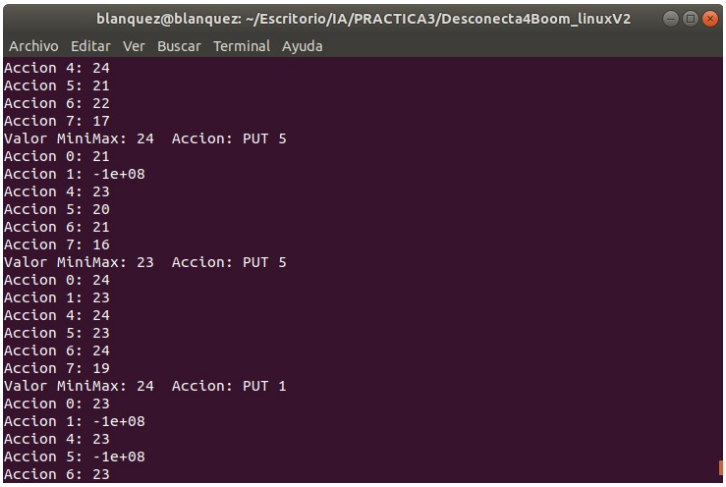
Antonio José Blánquez Pérez
2ºD2 – IA

1. ANÁLISIS DEL PROBLEMA

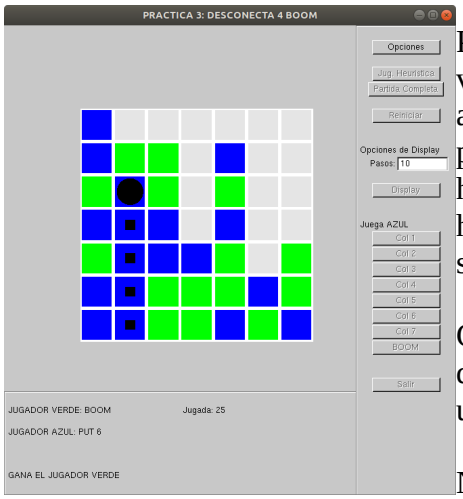
Necesitamos desarrollar un agente deliberativo que juegue al Desconecta 4 BOOM. Para ello hay que desarrollar un algoritmo para seleccionar la mejor jugada y, para que éste funcione, una función heurística para denotar lo bueno o malo que es un tablero. Para ello, la primera intención es implementar un algoritmo de poda alfa-beta y una función heurística aún por definir.

2. DESARROLLO DEL ALGORITMO DE SELECCIÓN

Se va a comenzar el desarrollo implementando el algoritmo minimax para posteriormente transformarlo en poda alfa-beta, usando en principio la heurística ValoracionTest. En el método think se ha comentado casi todo el código, excepto la declaración de algunas variables, la llamada a la función Poda_AlfaBeta(se ha dejado ese nombre para no ir cambiándolo) y el output del valor minimax y la acción elegida.



Tras implementar minimax con una estrategia retroactiva(utilizando la función GenerateNextMove(act) de Environment), mostrando las acciones disponibles y su valor correspondiente solo a profundidad 0, obtenemos resultados positivos. El algoritmo parece seleccionar correctamente el máximo entre las jugadas posibles, aquí tenemos una muestra:



Por lo tanto pasaremos a implementar la poda alfa-beta. Tras varias pruebas con un resultado erróneo devolviendo valor 0 y acción ??? se descubrió el fallo: alfa y beta se estaban pasando por referencia. Después de obtener un resultado satisfactorio se hicieron varias pruebas de que funcionara correctamente, de hecho gana al ninja de nivel 1 con la heurística por defecto(como se puede ver en la imagen de la izquierda).

Con el algoritmo de selección pasamos a buscar un heurística que de valores al tablero para que el algoritmo pueda seleccionar con un criterio suficientemente bueno para ganar partidas.

No obstante, durante las pruebas me he dado cuenta de un pequeño detalle que podría considerarse parte de la poda(o no, diría que está en un término medio). Existe un problema cuando el valor minimax dé más de una opción con valor $+\infty$ o $-\infty$. Ilustrémoslo con el siguiente ejemplo de ejecución, que es el enfrentamiento contra el ninja 1 como jugador verde, usando la poda y la heurística por defecto.

Como podemos ver, en la jugada 21 explotar la bomba es una jugada ganadora, sin embargo no es hasta la jugada 25 cuando esto ocurre. Este hecho se da ya que al haber varias jugadas que garantizan la victoria se cogen en orden PUT1,PUT2,…,BOOM y BOOM es la última, y el problema es que la opción BOOM se traduce en una victoria instantánea mientras que la que PUT2 tiene que realizar 4 jugadas más para ganar. Aunque no sea un gran problema, por asegurar la victoria(¿podría poner una bomba que nos quite la victoria?) y por elegancia, se va a añadir un pequeño control para que se devuelva la solución que más cerca esté de la victoria, para ello se ha tenido en cuenta la profundidad del nodo terminal. Al mismo tiempo se ha implementado lo contrario, de manera que cuando vea que va a perder alarga la partida todo lo posible por si se consigue ver una salida.

3. PLANTEAMIENTO DE LA FUNCIÓN HEURÍSTICA

Ahora comencemos con la heurística, la primera idea es penalizar las fichas cercanas al centro, ya que la columna central permite formar cuatro en raya de manera más sencilla. No es necesario implementar nada ya que la heurística por defecto cumple esta misma función, así que el primer paso es copiar la función ValoracionTest a nuestra función. Posteriormente se pensaron situaciones de tablero beneficiosas para el jugador:

- Es importante situar bien las bombas, cuantas más fichas pongamos en la misma fila sin perder la partida, más fichas propias eliminaremos del tablero, lo cual nos beneficia.

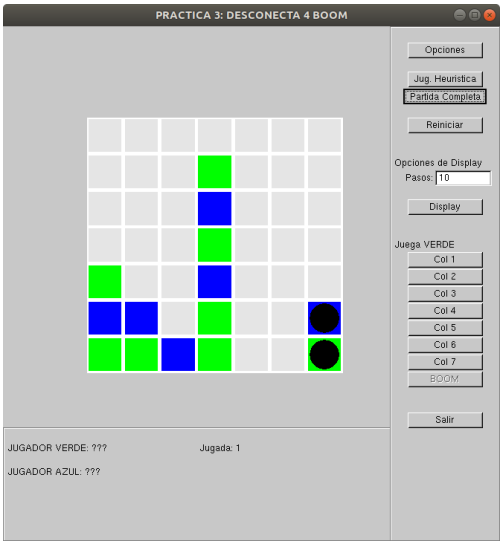
- Hay que evitar que el otro jugador elimine fichas con bombas.

- Menos fichas propias y más fichas del adversario son mejores para nosotros.

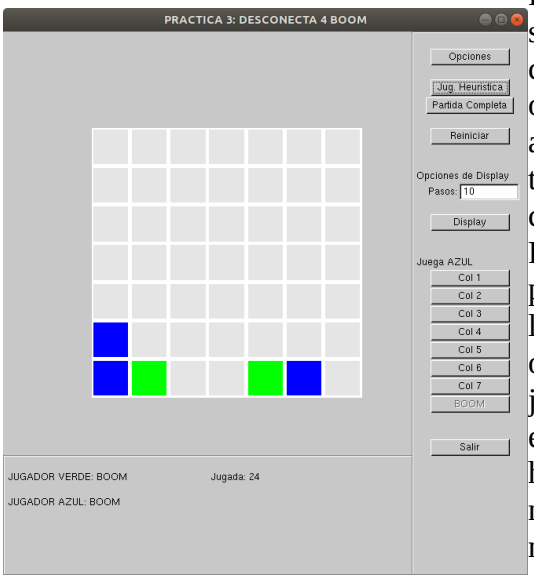
Tras pensar sobre como implementar una heurística para las dos primeras situaciones, he llegado a la conclusión de que ambas están contenidas en la tercera situación, ya que poner bien las bombas e intentar que el oponente no lo haga significa que vamos a quitarnos muchas fichas y el contrario pocas. Así que vamos a basar la función en la diferencia de fichas entre los jugadores (oponente-jugador), ponderadas por su lejanía del centro del tablero.

4. EVOLUCIÓN DE LA FUNCIÓN HEURÍSTICA

Con la anterior heurística comienzo a hacer pruebas, haciendo que la heurística juegue contra sí misma. Ésta primera aproximación parece ser bastante mala, ya que mi idea es que la mejor manera de jugar es no pensar tanto en que el otro jugador haga cuatro en raya si no pensar en colocar bien las fichas para eliminar el máximo número posible. El problema es que ésta primera aproximación llena el tablero de fichas y en cierto momento estás muy condicionado a tus anteriores movimientos, mientras que el verdadero dominio del tablero es tener muy pocas fichas para ahogar al oponente en las suyas. Es prácticamente el mismo tablero que con la heurística por defecto.

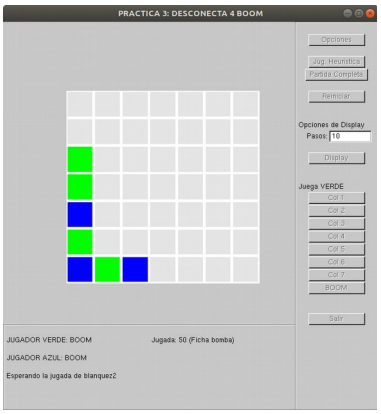


Por tanto cambiamos de estrategia, penalizando solo el centro, dándole un valor de una unidad mientras que el resto de casillas valen dos unidades(éstas unidades se han duplicado para clarificar los resultado, simplemente por comodidad). Con ésta heurística ya entramos en un bucle lo cual es bueno porque es lo que iba buscando, minimizar fichas, y si ambos jugadores lo hacen bien se entra en un bucle infinito. Es interesante ver como se llena la columna central intentando que el oponente tenga problemas con ella, lo cual no sé si es bueno o malo.



Por esto último, voy a suprimir la condición central y simplemente dejar la diferencia de fichas. Esta opción, tras pruebas, también se ve muy bien, igual que la anterior pero sin la columna central llena, aunque no me termina de convencer, es difícil de explicar pero no me cuadra como juega respecto a como pienso que debería de hacerlo. Para mejorar ésto he optado por priorizar tener menos fichas propias ante que el oponente tenga más, para ello he dejado los valores tal que una ficha propia vale -4 y una ficha del oponente vale 2. En la imagen podemos ver un ejemplo de la jugada 24 donde podemos ver que apenas hay fichas, lo cual es muy bueno. También he hecho esta modificación en la heurística que penaliza el centro y también funciona bastante mejor, así que probaré ambas contra los ninjas para ver cual es mejor.

5. DESCRIPCIÓN DE LA SOLUCIÓN PLANTEADA



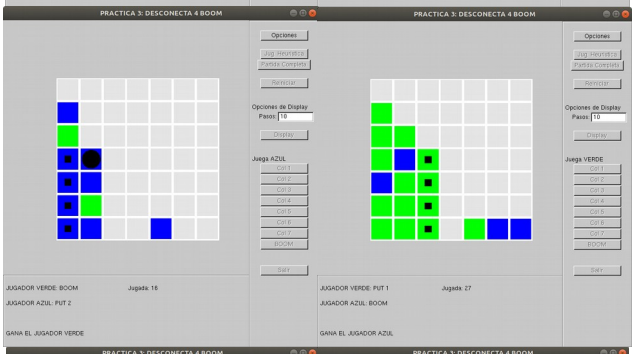
Se ha comprobado que ambas heurísticas funcionan bien, de hecho las he enfrentado una contra otra abriendo dos aplicaciones, una con cada heurística, y he usado el servidor proporcionado para enfrentarlas. Básicamente acaban haciendo un ciclo, como se puede ver en la captura(jugada 50), por tanto voy a mantener la más simple, aunque la otra quedará comentada en el código por si es de interés.

Por recapitular un poco: se ha implementado un poda alfa-beta con una heurística basada en minimizar tus fichas en el tablero y maximizar las del oponente, priorizando las primeras. También hay otra versión que, a parte de esto, penaliza las fichas del centro, que funciona prácticamente igual de bien que la otra y que queda añadida al código comentada.

A continuación, para terminar, vamos a ver que tal le va contra los ninjas.

Contra el ninja 1 vemos como gana en la jugada 31 como jugador 1 y en la jugada 37 como jugador 2, nada mal.

Contra el ninja 2 gana en la jugada 16 como jugador 1 y en la 27 como jugador 2, el mejor de todos los resultados.



Contra el ninja 3 gana en la jugada 21 como jugador 1 y en la jugada 32 como jugador 2, curiosamente antes que el primero.

Probablemente gana más rápido al ninja 3 que al 1 porque la heurística juega como si el oponente hiciera las mejores jugadas posibles(según ella misma). El problema es que si la otra heurística es mala(al ninja 1 le ganaba con la heurística por defecto) puede que nuestra heurística se haga un lío y vaya cambiando de estrategia según los movimientos del oponente. Aunque esto es solo una hipótesis, reforzada porque el ninja 2 es al que más rápido ganamos y, tras verlo jugar, es posiblemente el que juega más parecido a nuestra heurística.