Práctica 3: Parcheckers

Pablo Linari Pérez

Junio 2025

Contents

1 Introducción

En esta práctica veremos la implementación de varias mejoras del algoritmo minmax proporcionado y buscaremos una heurística adecuada para intentar ganar a los ninjas propuestos .

2 Mejoras del algoritmo MINMAX

2.1 Poda Alpha Beta

La primera mejora que implementaremos al algoritmo será la poda Alpha-Beta. Esta poda consiste en establecer un límite inferior, llamado alpha, y un límite superior, llamado beta. A medida que se explora el árbol de juego, se van acotando los valores que pueden tomar los nodos.

Cuando evaluamos los hijos de un nodo MAX, actualizamos el valor de alpha con el mayor de los valores de sus hijos; de esta manera, acotamos inferiormente el intervalo (alpha, beta). De forma análoga, cuando evaluamos un nodo MIN, actualizamos el valor de beta con el menor de los valores de sus hijos.

Los índices alpha y beta se propagan por cada nodo hasta llegar a la raíz. En caso de que, en algún momento, estos índices se crucen ,es decir, que alpha sea mayor o igual que beta, o que beta sea menor o igual que alpha, se podarán todos los hijos que aún no hayan sido explorados del nodo en el cual se cruzaron los límites.

De esta manera, conseguimos evitar evaluaciones innecesarias de nodos, lo que permite aumentar la profundidad de la exploración del árbol y realizarla de una manera más rápida y eficiente.

2.2 Poda Probabilística

En esta implementación del algoritmo alpha-beta se introduce un valor nuevo llamado tolerancia, el cual indicará cuándo consideramos que la distancia entre alpha y beta es lo suficientemente pequeña como para poder podar las ramas restantes. Este tipo de poda consigue que se realice un mayor número de podas, pero se sacrifican algunas jugadas que podrían tener un valor ligeramente superior a la elegida. Por esta razón, es de gran importancia elegir la tolerancia de manera precisa, para que lo que ganemos en eficiencia y velocidad no lo perdamos al podar jugadas que podrían ser demasiado buenas. El código de este algoritmo es muy parecido al alpha-beta normal, pero en vez de esperar a que se crucen los índices para podar, esperamos a que la distancia entre ellos sea menor que la tolerancia.

2.3 Ordenación de movimientos

En este caso, modificaremos el orden en el que se exploran los nodos antes de ser evaluados. De esta manera, si evaluamos primero los nodos con mayor valor

heurístico, será muy probable que podamos podar ramas más rápido, lo que hará que el algoritmo sea más eficiente y rápido.

Esto se ha conseguido añadiendo una segunda heurística destinada a evaluar los nodos para ordenarlos, siendo la jugada más prometedora la que se evaluará primero en el caso del nodo MAX y la menos prometedora primero en el caso del nodo MIN. Para ello, una vez desarrollada la lista de nodos hijos, se crea un vector de parejas al cual se le asignan elementos con valores (hijo, heurística). Posteriormente, ordenaremos el vector y aplicaremos el algoritmo con los nodos ordenados según la heurística indicada.

Se ha añadido un parámetro adicional con respecto a la poda básica para poder proporcionar una heurística distinta para ordenar y otra para podar.

Debido a que ordenar los nodos requiere un coste computacional, este paso se realiza solo en los primeros dos niveles del árbol de juego, ya que es donde más repercusión tiene, debido a que una buena elección como primera opción puede derivar en más podas posteriormente.

3 Heurística

En esta sección comentaremos las heurísticas implementadas, que en este caso han sido tres: dos para el juego en general y una heurística usada para la ordenación de los nodos en el algoritmo de Ordenación.

Como base comencé usando la heurística proporcionada en el tutorial, en la cual se recorren las piezas de ambos jugadores y en ambas heurísticas se procede de la misma manera. En el primer bloque se evalúan los casos favorables o desfavorables al jugador y en el segundo bloque se evalúan los casos favorables o desfavorables para el contrincante. Una vez calculado cómo influye cada acción, se devuelve la heurística calculada como puntuación jugador - puntuación oponente, para restar las acciones que vienen bien al oponente a la heurística de las acciones que vienen bien al jugador.

En ambas heurísticas se han considerado importantes las siguientes acciones:

- **Distancia a la meta**: Esta distancia se ha sumado a la heurística del jugador y del oponente considerando todas sus piezas disponibles respectivamente. El cálculo se ha hecho de manera que se le dé más importancia a las piezas que estén cerca de la meta y menos a las que estén más lejos. De esta manera, podemos ir moviendo todas las piezas juntas sin dejar ninguna atrás y pudiendo tener todas ellas defendidas.

- **Casilla Segura**: A esta acción no se le ha dado mucha importancia, solo la necesaria para que en caso de decidir entre dos jugadas del mismo valor, tenga más peso la que deje la pieza en una casilla segura, ya que como veremos más adelante, nos interesará que el mayor peso lo tengan otras acciones.

- **Pieza en casilla final**: Esta acción repercute en la heurística con el doble de valor que el de tener la pieza en una casilla segura, ya que nos interesa que cuando una pieza pueda entrar en la casilla final lo haga, para poder obtener el dado especial de 10 movimientos y tener una ficha a salvo.

- **Comer Pieza**: La manera en la que se ha implementado esta acción es teniendo en cuenta positivamente si el jugador se come una pieza del color del oponente y negativamente si el oponente se puede comer una pieza del jugador. De esta manera se consigue que se consideren mejores las jugadas en las cuales podemos comernos una pieza o evitamos que el rival nos pueda comer una pieza nuestra. Esta acción es una de las que más peso tiene en las heurísticas, ya que al comer piezas ganamos mucha ventaja sobre el rival, pudiendo recorrer mucho terreno y mandando una ficha suya a la casilla de inicio.

- **Piezas en la casilla de salida**: Esta acción también se valora con una ponderación muy alta para el jugador si el rival tiene piezas en su casa, y con una ponderación alta en contra del jugador si el propio jugador tiene sus piezas en casa. De esta manera se consigue que el jugador trate siempre de sacar sus piezas de casa en caso de que las haya y, si no, indirectamente se está valorando que hagamos jugadas que dejen a las piezas del oponente en su casa, como puede ser comer piezas o bloquear la salida con dos piezas en su casilla inicial. De igual forma influye en el jugador para no dejarse comer piezas por el contrincante.

3.1 Explicación Heurísticas

A continuación, explicaré cómo se han ponderado las acciones que se han descrito en el apartado anterior. La búsqueda de la heurística ha sido una mezcla de prueba y error, y el uso del sentido común sobre las acciones que nos hacen ganar el juego.

3.2 Heurística 2

El diseño de la heurística se ha pensado de tal forma que los movimientos mejor valorados sean aquellos que nos acerquen a la meta o que se coman una ficha enemiga, ya que esto nos hará avanzar con mucha facilidad. La manera en la que se usa la distancia a la meta es $100 - distanciameta$, de esta forma priorizamos que las fichas se muevan rápido hacia la meta.

La ponderación que tiene comer una ficha rival es de 60 puntos, ya que esto nos hará avanzar más rápido y perjudica al oponente. De la misma manera, también se valora positivamente los estados en los que el oponente tenga piezas en su casilla inicial, siendo la puntuación de 30 puntos por ficha en la casilla inicial. Con esto conseguimos que se incentive el comer piezas rivales y bloquearles la salida.

Por otra parte, también se valora positivamente con 15 puntos por pieza que consigamos meter en la casilla de meta. De esta manera, conseguimos meter las piezas que se encuentran en el pasillo de la casilla de meta lo antes posible.

Las valoraciones negativas hacia la puntuación de la posición son estados donde el rival nos come una pieza, ponderados con 40 puntos menos, y estados en los que tenemos piezas en la casilla de salida, ponderándolos con 40 puntos menos por pieza en casa. Así, hacemos que se elijan jugadas que mantengan las piezas a salvo y que, cuando haya piezas en la casilla de salida, se saquen lo antes posible. Como última valoración, también se tiene en cuenta que una pieza esté en una casilla segura y se pondera con 10 puntos, ya que no es una acción que cause una gran diferencia.

Los principales valores que se han ido modificando hasta obtener el resultado actual han sido las ponderaciones de comer pieza y las que valoran la penalización de piezas en casa. Al principio se le dio más importancia a la distancia a la meta, pero se desperdiciaban situaciones en las que podíamos comer una pieza para avanzar mucho más. Por tanto, se comenzó a subir la puntuación de comer una pieza para que la heurística fuese más agresiva. Esto hizo que las piezas quedasen más expuestas, por tanto se añadió la ponderación de ver si la pieza estaba en un lugar seguro y se subió la penalización por tener piezas metidas en la casilla inicial. De esta forma se evitaría que nuestras fichas fuesen comidas con tanta facilidad, pero se mantendría la opción de que se comiesen las piezas del rival cuando fuese posible. Tras modificar varias veces el valor que se le asigna a las piezas guardadas en casa tanto del oponente como del jugador se ha conseguido hacer que la heurística gane a todos los ninjas menos la partida de ninja 2 vs jugador 1.En este caso se ha usado la heurística de la clase Heuristic1 que si consigue ganar esa partida pero pierde la partida ninja 3 vs jugador 2.

3.3 Heurística de orden

Esta heurística se usa para ordenar los nodos generados en el algoritmo de poda con ordenación, y solo se tienen en cuenta las piezas en casa y la distancia a la meta, para que sea una heurística más rápida de evaluar y siga haciendo un orden coherente. No obstante usar la misma heurística para evaluar y ordenar en ocasiones da mejores resultados , esta heurística se desarrollo como una prueba.

4 Tablas de Partidas

A continuación, se especifica a qué ejecución se asocia cada ID y las heurísticas utilizadas en cada caso. Más abajo se presentan los resultados de las ejecuciones en una tabla.

- **AlphaBeta (Heurística dada en el guion):** ID 0
- **AlphaBetaH2 (Mejor ejecución, Heurística 2):** ID 1

- **AlphaBetaH1 (Heurística 1):** ID 2
- **PodaProbH2 (Poda probabilística, Heurística 2):** ID 3
- **OrdenacionH2 (Heurística 2, ordenación con heurística de orden):** ID 4
- **OrdenacionH1yH2 (Evaluación con Heurística 1, ordenación con Heurística 2):** ID 5
- **PodaProbH1 (Poda probabilística, Heurística 1):** ID 6

Estrategia	Ninja1 vs J1	Ninja1 vs J2	Ninja2 vs J1	Ninja2 vs J2	Ninja3 vs J1	Ninja3 vs J2
AlphaBetaH2	Victoria	Victoria	Derrota	Victoria	Victoria	Victoria
AlphaBetaH1	Victoria	Victoria	Victoria	Victoria	Victoria	Derrota
PodaProbH2	Victoria	Victoria	Derrota	Victoria	Victoria	Victoria
OrdenacionH2	Derrota	Victoria	Victoria	Victoria	Derrota	Derrota
OrdenacionH1yH2	Derrota	Derrota	Victoria	Victoria	Victoria	Victoria
PodaProbH1	Victoria	Victoria	Victoria	Victoria	Victoria	Derrota

Table 1: Resultados de las estrategias frente a los distintos rivales.

5 Reflexión

Tras completar todas las pruebas con las distintas variaciones de poda, podemos concluir que las mejores modificaciones corresponden a la poda probabilística y a la poda alpha-beta estándar.

En cuanto a las variantes con ordenación, observamos que el algoritmo funciona mejor cuando se evalúa con la heurística 1 y se ordena con la heurística 2. Sin embargo, este enfoque sigue resultando inferior al de la poda alpha-beta sin ordenación adicional.

Respecto a la comparación entre las heurísticas 1 y 2, se puede afirmar que la heurística 2 es superior. Ambas obtienen un número similar de victorias con poda alpha-beta estándar, pero en enfrentamientos directos, la heurística 2 vence a la 1 en escenarios con las mismas condiciones.

El proceso de diseño y ajuste de la heurística ha sido inicialmente ilustrativo, ya que obligó a analizar en profundidad las situaciones del juego para construir una estrategia representativa de cada estado. No obstante, en la última fase del desarrollo, resultó más tedioso ajustar los parámetros para obtener una tasa de victorias aceptable. Hasta que no se afinó la ponderación de la evaluación, las ejecuciones tendían a perder entre 2 y 4 partidas. Este ajuste se realizó mediante prueba y error, observando el comportamiento del algoritmo y afinando las ponderaciones en las situaciones en que la heurística fallaba.

Finalmente, algunas mejoras implementadas no ofrecieron los resultados esperados. El algoritmo minimax con poda alpha-beta ya realiza una muy buena ordenación de jugadas, lo que dificulta superar su rendimiento. Solo en el caso de usar la heurística 1 para evaluar y la 2 para ordenar se observaron algunos

resultados prometedores, venciendo a rivales que antes resultaban difíciles. No obstante, esta combinación sufrió derrotas en las dos particas con el Ninja 1. Por otro lado, la aplicación de poda probabilística tiene resultados más satisfactorios y consistentes.