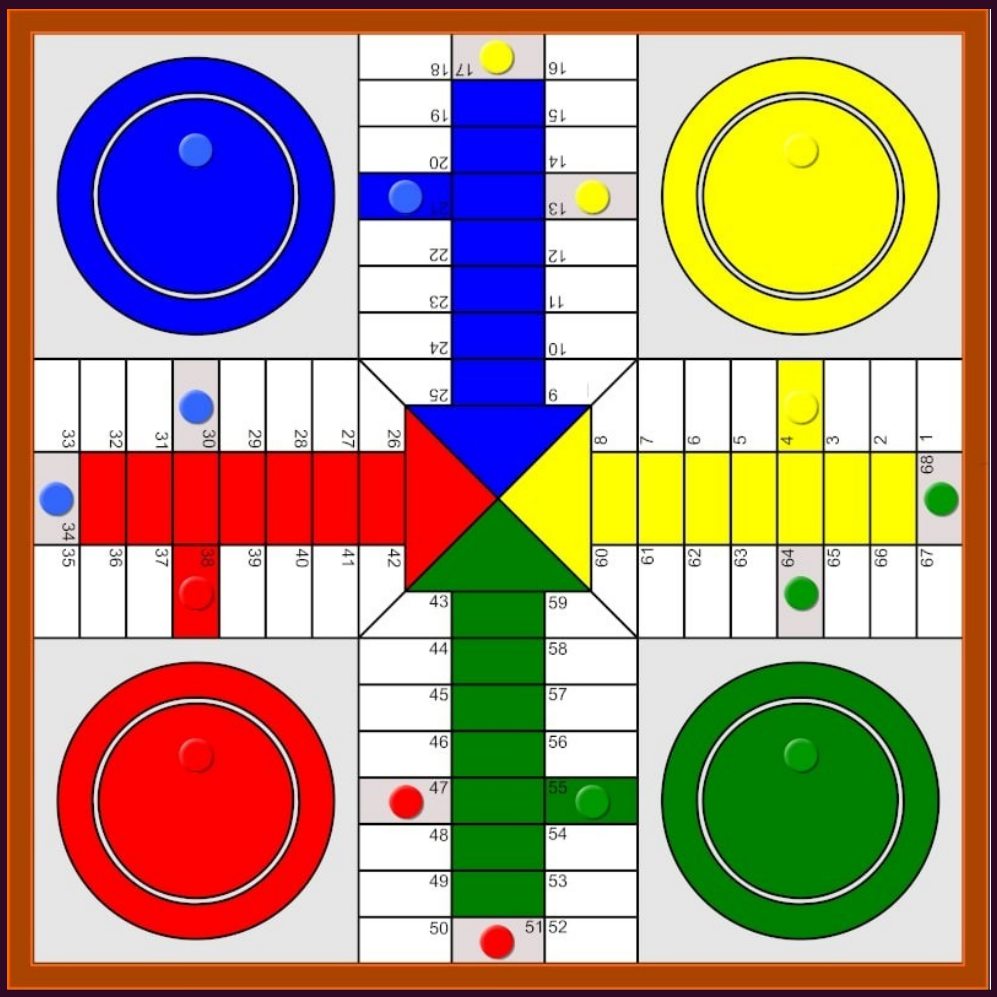
Inteligencia Artificial

Práctica 3

Andrés Piqueras Brück



Búsqueda con Adversario (Juegos)
El Parchís

1.Análisis del problema

El objetivo de la práctica consiste en implementar un algoritmo de decisión (ya sea Minmax o poda alfa-beta) que sea lo suficientemente bueno como para derrotar a varios rivales jugando a una versión modificada del parchís, que elimine elementos aleatorios.

Para ello, es necesario implementar dos partes claramente diferenciadas: El algoritmo de decisión, y la heurística.

El algoritmo de decisión tiene que navegar por todas las posibilidades tras un número determinado de turnos y seleccionar de entre ellas la más favorable. En cambio, la heurística es la parte encargada de asignar la puntuación a cada una de las posibilidades que el algoritmo de decisión le presente.

Una vez estén las dos partes correctamente implementadas, el problema estará solucionado.

2.Descripción de la solución planteada

2.1)Algoritmo de búsqueda

El algoritmo de búsqueda seleccionado ha sido el poda alfa-beta, ya que presenta una mayor facilidad a la hora de vencer a los ninjas. Esta facilidad es debida a las restricciones impuestas por el guión de prácticas: 6 contra 4.

Desde un punto de vista meramente numérico, es evidente que un programa capaz de ver todos los movimientos posibles con una profundidad de 6 es mucho mejor que uno que sólo sea capaz de ver los movimientos hasta profundidad 4. Además, puntúa favorablemente emplear la poda, por lo que decidí implementarla.

A la hora de hacer la poda me basé en el “esqueleto” proporcionado en la propia práctica.

En caso de llegar a la profundidad máxima o de finalizar el juego, devolvemos el valor de aplicar la heurística al estado actual. En caso contrario, generamos hijos del estado actual, y en función de si estamos jugando nosotros o nuestro oponente, nos encontramos en un nodo Max o Min.

En caso de encontrarnos en un nodo MAX, calculamos el valor de aplicar la poda a una profundidad mayor. Si el valor es más alto que alfa, sustituimos alfa por el valor. Una vez beta es mayor que alfa, podamos (es decir, salimos del bucle y por tanto no consideramos más opciones).

En caso de estar en un nodo MIN, aplicamos los mismos principios, pero guardamos el valor cuando beta sea mayor que el valor obtenido, y podaremos igual.

Generamos los hijos en orden descendente para mejorar sustancialmente la eficiencia del algoritmo, ya que se suelen elegir primero los valores del

dado más grande. De esta manera podamos muchas más posibilidades y facilitamos la ejecución del programa.

2.2)Heurística

Sólo he diseñado una heurística, que he ido mejorando poco a poco basándome en ValoracionTest, la heurística que ya incluía el programa.

Las primeras líneas están copiadas directamente, y sirven para identificar al ganador, al oponente, los colores del jugador y los colores del oponente.

La implementación general de la heurística está también basada en ValoracionTest. Hay dos bloques de código: uno que va sumando puntos para calcular la puntuación del jugado, y otro que va sumando puntos para calcular la puntuación del oponente. Cada bloque recorre cada pieza de cada color del jugador (el primer bloque) y del oponente (el segundo bloque).

Los dos bloques son muy parecidos. Para cada aspecto que sumaría puntos al jugador en el primer bloque existe una regla equivalente para el oponente en el segundo bloque. Por ejemplo, en caso de tener una ficha en la meta y ser el jugador, se suman 1000 puntos al jugador. En caso de tener una ficha y ser el oponente, se sumarían 1000 puntos al oponente.

Para explicar la heurística iré línea por línea en uno de los bloques, ya que el método de valoración es “simétrico”, como ya he explicado en el párrafo anterior.

Primero explicaré las condiciones que he tenido en cuenta una vez por ficha para el bloque jugador (ocho veces en total):

-Si la ficha está ubicada en el pasillo final, sumamos 900 puntos.

-Si la ficha está en una casilla segura, sumamos 150 puntos.

-Si la pieza forma parte de una barrera, sumamos 400 puntos (esta regla siempre suma 800 puntos porque las barreras requieren de 2 fichas).

-Por cada ficha del oponente que esté en su casa, aumentamos la puntuación 10000 puntos.

-Dependiendo de lo cerca que una ficha se encuentre de la meta, sumará más puntos.

La regla verde requiere una explicación:

El programa, al tener en cuenta muy favorablemente que el rival tenga fichas en casa, tratará por todos los medios de comerse las fichas rivales. Además, debido a que las condiciones son “simétricas”, el programa tendrá en cuenta muy negativamente tener fichas en nuestra propia casa, por lo que hará lo posible por mantener nuestras fichas fuera de peligro.

Ahora explicaré las condiciones que he tenido en cuenta una vez por color (dos veces en total):

-Si hay una ficha en la meta sumamos 1000 puntos, si hay dos sumamos 3000 y si hay 3 sumamos 9000.

Esta progresión no lineal se debe a que es más importante pasar de tener 2 fichas en meta a tener 3 de cara a ganar la partida. Esta regla viene particularmente bien para que el programa no decida comerse una ficha próxima a ganar la partida para avanzar otra de un color que no tiene ni una ficha en meta. De esta manera prioriza meter la ficha del color que más fichas tenga.

Como se puede apreciar, la heurística empleada es muy simple, y se apoya principalmente en sumar la puntuación adecuada para que el programa priorice unos movimientos sobre otros dadas las circunstancias.

Inicialmente mi heurística daba muchísimos puntos por unas cosas y poquísimos puntos por otras, haciendo que la heurística prácticamente no sirviera para nada, ya que si la diferencia de puntos es demasiado elevada es casi como tener un autómata reactivo, que responda a una única condición.

La clave está en darle el peso justo a todas las acciones relevantes. Comer y evitar que te coman es a mi juicio lo más importante del juego (más incluso que meter tres fichas en la meta), porque forzosamente llegarás a la meta si no te han comido, y dificultarás la victoria del oponente si le comes con asiduidad.

Así mismo, es más importante tener una ficha en el pasillo final que en barrera, y es mejor estar en una casilla segura que estar muy cerca de la meta.

Estoy muy orgullo del resultado final de la práctica, ya que he conseguido simplificar el código de la heurística lo máximo posible, y en consecuencia el código es legible y fácil de entender. Vence a los tres ninjas (a veces por los pelos, pero los vence) sin problemas y toma las decisiones con rapidez.