Explicación de Heurística Utilizada

Inteligencia Artificial

Integrantes

Sebastian Peñaranda
2041138-3743

Profesora
Oscar Bedoya

Universidad del Valle
Escuela de ingeniería de sistemas y computación
Programa de ingeniería de sistemas (3743)
Cali, Colombia
Junio,2023

Explicación de Heurística Utilizada

La heurística utilizada en el proyecto smart horses se trata de la diferencia entre el puntaje de la CPU y el puntaje del jugador:

$$Heuristica = puntajeCPU - puntajeJugador$$

El programa está hecho de tal manera que extrae estas dos informaciones de los nodos siguientes utilizando el método minimax con poda y calculando la heurística en cada nodo, la profundidad para el método minimax es importante y repercute en la heurística, ya que hay muchos casos cuando la dificultad es baja, donde los dos caballos cuando no tienen casillas cercanas con números, la heurística termina siendo la misma del nodo raíz.

Por esta razón he tomado libertad de modificar un poco como funciona la heurística, voy a ejemplificar el caso anterior:

$$heuristica = nodoRaiz.puntaje\_CPU - nodoRaiz.puntaje\_jugador$$

Cuando pasa esto se toma la decisión de que el nodo Raíz muestre sus hijos y los hijos de este se guardan en una lista, si y sólo si, los hijos cumplen la siguiente condición:

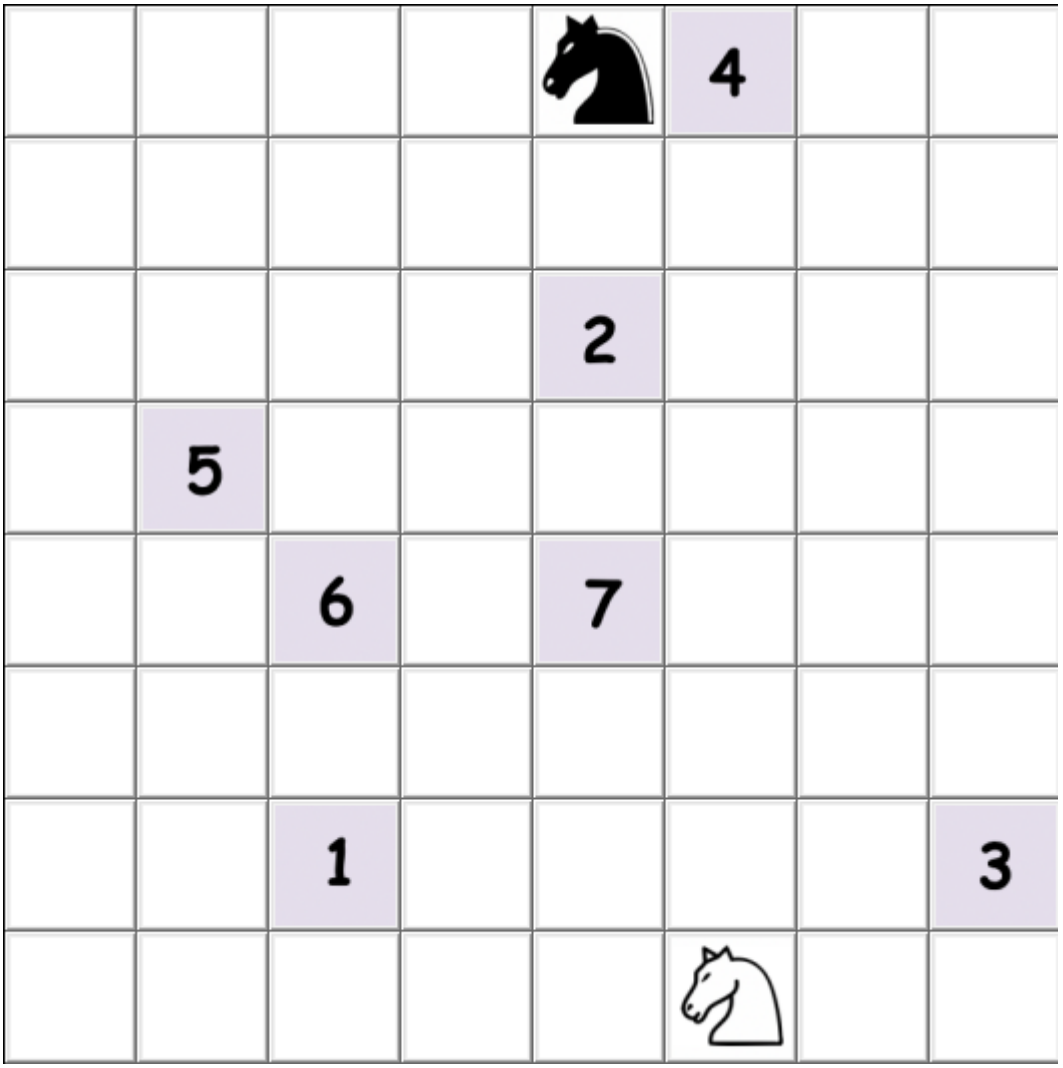
$$\text{heuristica} = nodoHijo.puntaje\_CPU - nodoHijo.puntaje\_jugador$$

Después se elige un numero aleatorio del rango de elementos de la lista, esto lo que hace es que cada vez que se escoja un camino diferente, no perjudique el puntaje de la CPU, y visualmente para el jugador sea más real los movimientos de la máquina, ya que esta si no tiene casillas con números cerca, formará un patrón visible después de cierta cantidad de movimientos

por otro lado otra modificación adicional que tiene la heurística utilizada es que cuando la dificultad del programa se haya elegido alta, estas heurísticas de los nodos tiende a aparecer en varios caminos, lo que hace que elegir cualquier camino sea viable para la CPU, pero visualmente para el jugador se vería como una opción no viable, ya que aunque ambos caminos tengan la misma heurística una se tarda 6 movimientos para tener toda la puntuación y otra solo 2.

Por esto se implementa que los nodos con números y más cerca de el nodo raíz estos tengan mayor puntaje, para que la CPU decida de ese camino antes que el otro.

En mi caso los puntajes de los nodos que siguen del nodo raíz se multiplican por 10, y el siguiente movimiento del caballo blanco sea por 5, la mitad, ejemplificando un caso se tiene lo siguiente:



Centrándonos en solo dos caminos que puede hacer el caballo blanco, uno es tomar el 3 que está al lado de él, o hacer dos movimientos, uno a la casilla izquierda-arriba y luego el 7, suponiendo que lo estamos viendo desde la profundidad del 7, la heurística del primer movimiento es (3*10) + 0 - movimiento del jugador = 30, y del otro movimiento es 0 + (7*5) - movimiento del jugador = 35 suponiendo que el jugador no decide coger fichas en ambas, el caballo blanco elegiría el camino del 7, pero en este caso podemos ver como él 3 por ser el nodo que repercute el siguiente movimiento

tiene mucho más valor, pero el movimiento que dirige hacia el 7 tiene mucha más heurística, haciendo una buena demostración de cómo el programa prioriza más las fichas cercanas pero respetando que los siguiente movimientos sean posiblemente mejores. Con esto terminamos la explicación de la heurística.