Memoria Práctica IA
Agente Deliberativo de Juego - 4 en Raya Crush

Adrián Portillo Sánchez

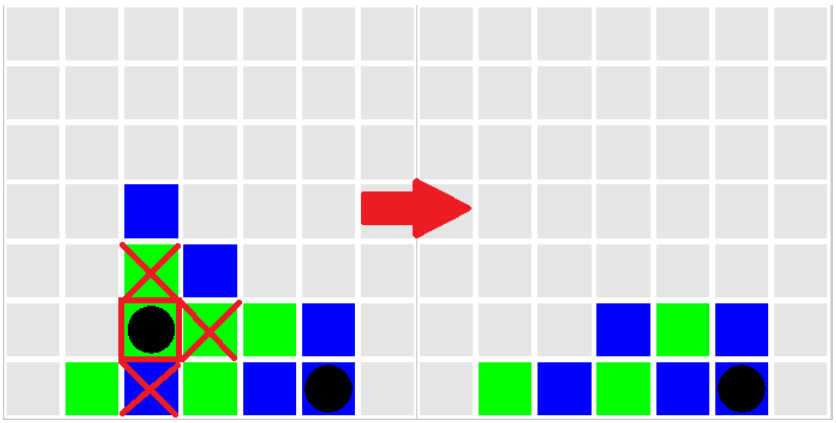
Mayo 2015

1. Análisis del Problema.

1.1. Análisis del Juego Planteado: 4 en Raya Crush.

El problema a tratar es una variante del tradicional 4 en raya, con algunas modificaciones para reimaginar el problema.
Las dos modificaciones fundamentales sobre el sistema son:

- En lugar del tradicional *7 ancho x 6 altura* el tablero de juego de 4 en raya crush será de *7 ancho x 7 altura*, evitando problemas como el método para ganar siempre empezando primero en 41 movimientos.
- Cada 4 movimientos de un jugador la ficha que coloque será una ficha bomba, la cual podrá estallar consumiendo uno de sus turnos en cualquier turno posterior, la ficha bomba explotará, desapareciendo y haciendo desaparecer a las 4 fichas que se encuentren en las posiciones adyacentes de la matriz.
  La ficha bomba será tratada como una ficha normal del jugador para contar la victoria con 4 en raya.



1.2. Árbol de Juego.

El árbol de juego del sistema será un arbol donde cada nodo será un estado del tablero de juego, y los arcos serán las posibles acciones a realizar, que serán, si no hay bomba, poner ficha en una de las columnas aún disponibles del tablero, y si hay bomba del jugador, hacerla explotar también será considerado una acción, para evaluar estas acciones se llama a GenerateAllMoves(), la cual creará los tableros de las posibles decisiones del juego.
Por ejemplo, en la anterior imagen, los dos nodos serán la imagen de la izquierda y la de la derecha y el arco entre ellos será la decisión "BOOM".

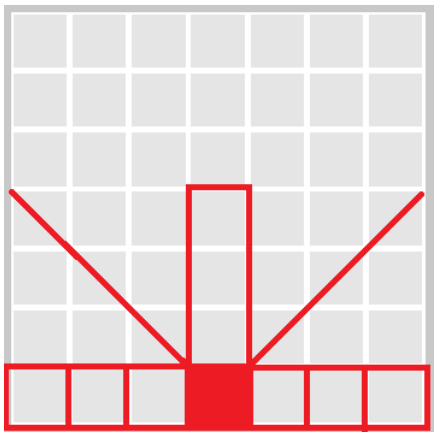
2. Descripción del Comportamiento de la Solución Aportada.

2.1. Minimax y Poda alfa-beta

El sistema utiliza la poda alfa-beta para elegir de entre las decisiones posibles la del nodo con una valoración más satisfactoria, que será aportada por la función Valoracion() la cual contendrá la heurística utilizada para nuestro problema, y valorará un estado del tablero según su grado de satisfacción para el jugador actual, que tendrá un rango de valores entre -99999999 y 99999999, donde estos dos valores serán la derrota y la victoria respectivamente.
La poda alfa-beta utiliza una profundidad 8, por lo que podrá planear hasta cuatro futuros movimientos de la forma más satisfactoria posible. La poda alfa-beta se implementa a partir de una función minimax, que en mi caso ha sido aportada por el profesor, a esta le he aplicado una serie de condiciones y mejoras, hasta llegar al estado de mi poda alfa-beta, eligiendo el conjunto de estados mejor y asignando los alfa y los beta hasta la profundidad que se le aporta al sistema.

2.2. Heurística utilizada: Función de Valoración

Mi función de valoración está basada en una lógica sencilla: tener un mayor número de posibilidades que el oponente, principalmente reduciendo las posibilidades del oponente, para ello contará el número de filas, columnas y diagonales del tablero que estén disponibles para el jugador y para el oponente y las restará, así considerará las posibilidades que le reste al oponente cada una de las posibles posiciones, como ejemplo, el primer movimiento siempre será la posición central, ya que elimina para el contrincante, 4 filas, 2 diagonales y 1 columna, de la siguiente forma:



Cabe decir que la valoración inicial será de $4 * 8$ columnas + $4 * 8$ filas + $4 * 8$ diagonales (16 hacia arriba y 16 hacia abajo).

Para mejorar esta heurística, se crea una función que cuenta las filas, columnas o diagonales de 3 elementos teniendo en cuenta también las distribuciones de la forma que se muestra aquí (2 fichas, un espacio y una ficha), para así darle una prioridad mayor a:

- Para el jugador, crear estructuras de 3 elementos, que darán paso a la victoria más facilmente.
- Para el rival, prioritizar el bloqueo de las posiciones que causen una victoria inmediata o adelantarse a esas estructuras bloqueándolas antes de que puedan ser creadas.

A estas estructuras de 3 elementos, se le da una mayor prioridad, definida como una constante entera prioridad, que tras varias pruebas los resultados más satisfactorios (probando a partir de potencias de 2), han sido con 2^{10}.

Esta prioridad se multiplicará directamente a la valoración para las columnas, filas y diagonales de 3 elementos del rival, dándole un valor 1024 veces superior a estas, de esta forma prioritizará el bloqueo de dichas estructuras por encima de todo lo demás, excepto una victoria inmediata, lo cual será la mayor prioridad del sistema, ya que el árbol le dará la valoración 99999999, que equivale en valores prácticos a infinito.

Sin embargo a la creación de estructuras de 3 elementos por parte del jugador, le dará una prioridad menor, aunque tras varias pruebas he descubierto que el método que mejor funciona es darle a las diagonales una prioridad de 1024 también mientras que a las filas y las columnas darle un valor 8 veces menor, aunque la razón de que dichos valores funcionen así de bien no la tengo muy clara, simplemente tras una serie de pruebas he notado que dicha valoración provoca un comportamiento espléndidamente mejor que sin dichos cambios, ya que crea jugadas mucho más complejas y que llevan a la victoria en más ocasiones.

Creo que esto sucede porque las diagonales, al ser las estructuras más complejas, también son las que provocan situaciones de victoria más a menudo y las que consiguen las estrategias ganadoras en los grandes jugadores de 4 en raya, ya que son las más complicadas de controlar, especialmente por una inteligencia artificial.

3. Resultados.

Para probar la efectividad de mi heurística, en primer lugar he jugado contra el ninja, hasta poder ganarle con esta heurística actual en las dos situaciones posibles, tanto empezando él, como empezando mi heurística, por lo que me encontraba muy satisfecho con los resultados.
Pero para asegurarme de que la heurística era realmente buena, le envié el programa a un amigo de la infancia con el cual mantengo contacto para que la probará, ya que él fue un gran jugador, y sabe mucho de 4 en raya, por lo que su juicio sería de gran valor; aunque mi heurística perdió en reiteradas ocasiones, ya que él utiliza jugadas de más alto nivel; consiguió ganarle en otro tanto de ocasiones, y mi amigo me dijo que mi programa era muy bueno y conseguía ponerle en situaciones realmente dificiles, por lo que consideré mi heurística bastante satisfactoria y mi programa de un nivel lo bastante bueno como para entregarlo así.

A continuación muestro algunas jugada que el algoritmo me ha conseguido realizar, que parecen sacadas de libro, en algunas pruebas que he realizado tras la escritura de esta memoria, para que se pueda ver la efectividad del algoritmo: