Integrantes: Federico Puentes Acosta, Juan Antonio Rodríguez
Curso: Introducción a la Inteligencia Artificial
Universidad Nacional de Colombia, Bogotá

Proyecto 2 - Multiagentes

Antecedentes - Definición del Problema

Contexto

2048 es un videojuego para un jugador creado en marzo de 2014 por el desarrollador italiano Gabriele Cirulli [1]. El juego se desarrolla sobre una grilla, tradicionalmente de 4×4, sobre la cual hay baldosas marcadas con una potencia entera y positiva de 2. El jugador debe mover las baldosas de la grilla en una de las cuatro direcciones básicas (arriba, abajo, izquierda, derecha) buscando combinar baldosas. Cuando el jugador escoge una dirección, todas las baldosas se mueven tanto como puedan en esta dirección:

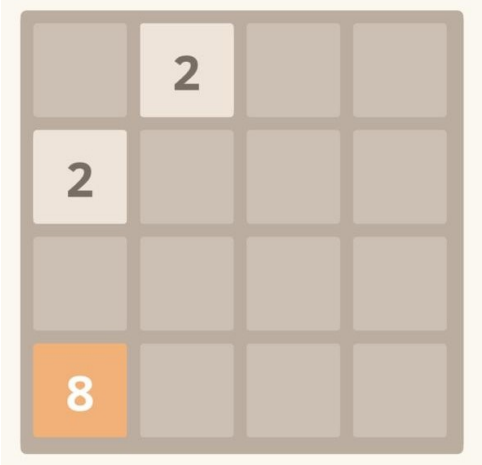

Figura 1: Antes de mover

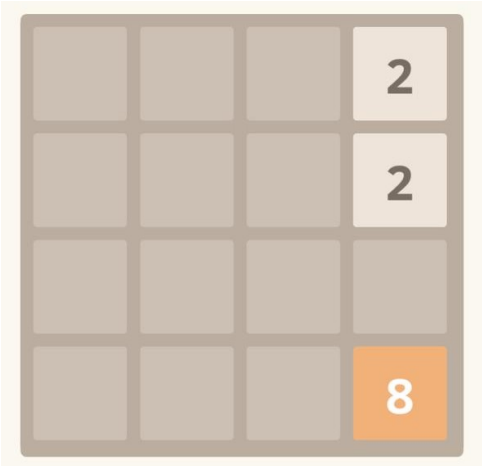

Figura 2: Después de mover a la derecha

Dos baldosas se combinan si ambas tienen el mismo valor y una de ellas se mueve en dirección a la otra sin que una tercera esté entre estas, cuando se combinan queda como resultado una sola baldosa con la suma de los valores de las dos baldosas iniciales, es decir el doble de su valor:

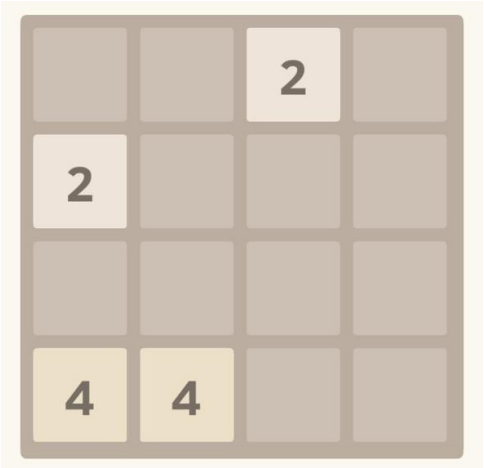

Figura 3: Antes de mover

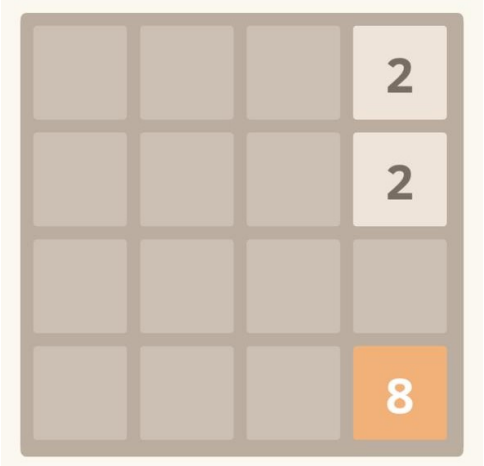

Figura 4: Después de mover a la derecha

Una acción del jugador, determinada por una de las direcciones, es válida solo si la configuración del tablero cambia con el movimiento en esa dirección, ya sea porque hay baldosas que se pueden mover en esa dirección o porque hay una combinación posible de baldosas en esa dirección.

Figura 5: El jugador no puede mover a la derecha

Después de cada movimiento del jugador, una nueva baldosa marcada con 2 o 4 en uno de los espacios vacíos.
El juego termina en uno de dos escenarios: El jugador logra hacer aparecer una baldosa de valor 2048 (en este escenario el jugador gana) o el tablero se llena sin que el jugador tenga más movimientos disponibles (en este caso el jugador pierde)

Definición del Problema

Podemos pensar el juego como como un entorno de dos agentes: el jugador, y el agente *oponente* que hace aparecer las baldosas nuevas en los espacios vacíos. En este caso podemos pensar dos tipos de juego: uno estocástico en el que el agente oponente ubica las baldosas nuevas de manera completamente aleatorio. Y uno en el que el agente es estratégico al colocar las baldosas intentando llenar el tablero antes de que el jugador gane.

Postulación Formal del Problema

Establecemos que el juego se desarrolla por turnos entre el agente jugador y el agente oponente.

Estados

Los estados del juego están determinados únicamente por los contenidos del tablero representados por una matriz 4×4 donde los espacios en blanco son representados por 0 y el valor en las casillas con baldosa corresponde con el valor en la correspondiente componente de la matriz.

Estado Inicial

El juego se inicializa con un tablero vacío al que se le agregan dos baldosas aleatorias con valores 2 o 4

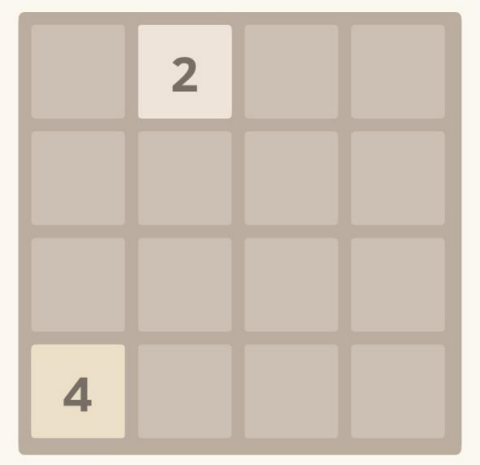

Figura 6: Inicialización del juego

Acciones

Las posibles acciones son completamente diferentes para ambos agentes. Por un lado, el agente jugador tiene cuatro posibles acciones, una por cada dirección, de las cuales sólo serán legales las que impliquen un cambio en el tablero, es decir, el estado sucesor a un estado dado, jamás es igual al anterior después de que el agente jugador ejecuta una acción legal.
Por otro lado, las acciones posibles del agente oponente, consisten en ubicar una baldosa de valor 2 o 4 en un espacio, y la acción es legal sólo si el espacio está vacío anteriormente.

Funciones isWin() isLose()

Se han programado pruebas para verificar que el juego termina en victoria o derrota. Para la victoria comprobamos si en el tablero hay alguna baldosa de valor 2048 y para la derrota verificamos que el tablero esté lleno y que no haya combinaciones posibles, es decir dos baldosas adyacentes del mismo valor.

Función de evaluación

Para poder implementar correctamente los algoritmos de Minimax y Expectimax debemos establecer una función que nos evalúe.
En este caso, primero establecimos que el puntaje sería dado por la suma de los valores de cada baldosa. También se decidió premiar la cantidad de espacios vacíos en el tablero para ir evitando que el tablero se llene. Y por último se premia fuertemente que aparezcan baldosas de valor alto. El valor de la función de evaluación para cada estado está dada por

$$score + 10maxtile + 5emptyCells$$

donde *score* es el puntaje, *maxTile* es el valor máximo de las baldosas en el tablero y *emptyCells* es la cantidad de espacios vacíos.

Resultados

Con esta función de evaluación se lograron resultados interesantes. En primer lugar no pudimos correr los algoritmos de Minimax o Expectimax con una profundidad mayor a dos, excepto con poda alfa-beta. Esto se debió a que el agente oponente tiene muchas posibles acciones: 2 por celda vacía, lo cual resulta en árboles que se expanden muy rápidamente. Creemos que algo de profundidad mejoraría mucho el algoritmo pero no pudimos comprobarlo tan detalladamente.

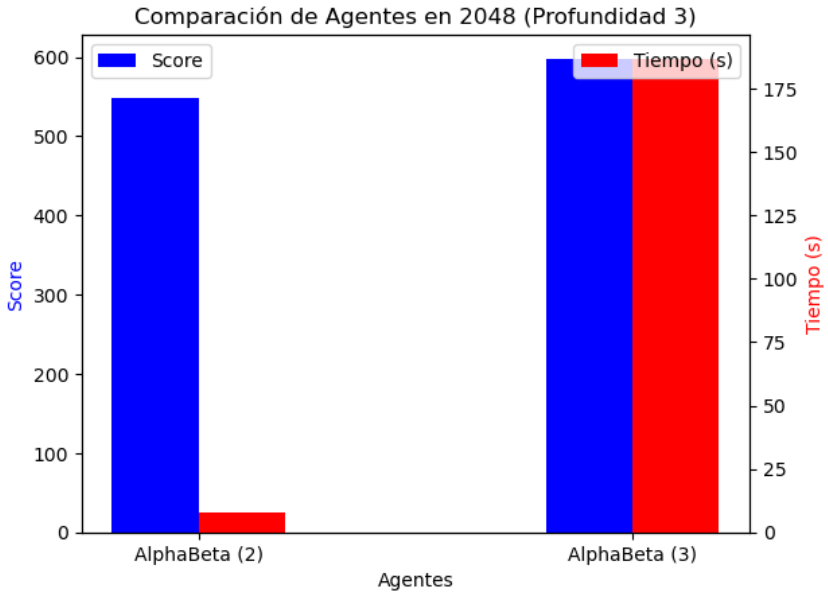

Figura 7: Agente Minimax con poda y distintas profundidades

Por otro lado es notorio que los puntajes con Expectimax son mucho mejores que con Minimax. Esto es muy seguramente debido a que el juego es mucho más fácil para el agente jugador si el oponente juega de manera aleatoria y no óptima como juega al correr el algoritmo Minimax.
La poda alfa-beta también implicó una mejora muy grande en los tiempos de ejecución del algoritmo Minimax. La diferencia puede verse reflejado debido, de nuevo, a la gran cantidad de posibles acciones del agente oponente.

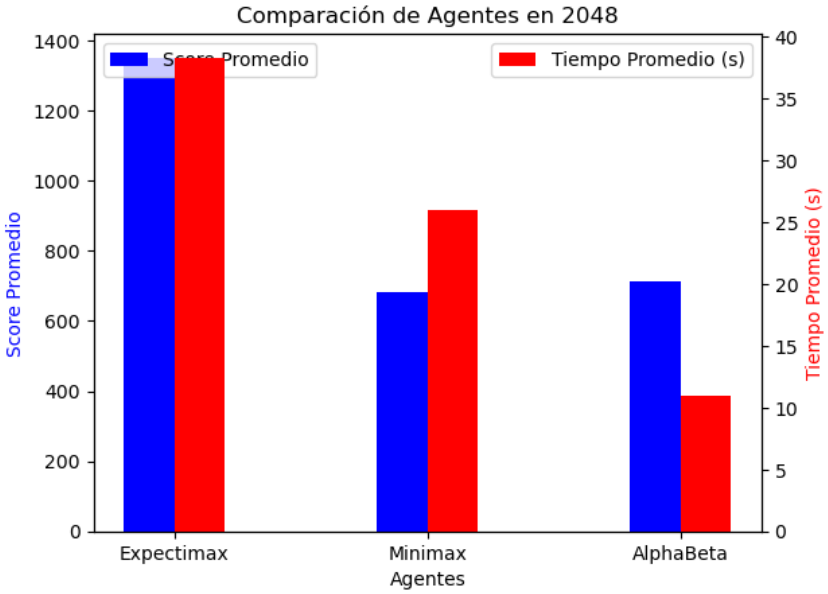

Figura 8: Comparación de los agentes

Por último debemos resaltar que son escasas las veces en las que se gana el juego corriendo el algoritmo. Generalmente se están alcanzando valores de 512 y 1024. Creemos que debemos explorar mejoras en la función de evaluación.

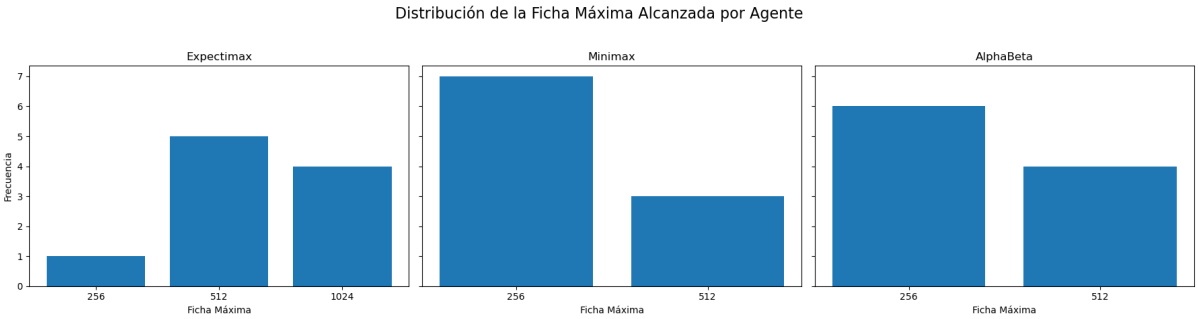

Figura 9: El jugador no puede mover a la derecha

Conclusiones

- Una cantidad grande de acciones posibles de alguno de los agentes implica dificultades grandes a la hora de buscar profundidad al momento de implementar los algoritmos Minimax y Expectimax
- La poda alfa-beta implica una mejor considerable con respecto al Minimax sencillo cuando la expansión de los árboles es grande
- Existe una alto oportunidad de mejora para la función de evaluación en futuras implementaciones

Posible trabajo futuro

La mayor oportunidad de mejora en el proyecto es seguramente el desarrollo de una mejor heurística que nos permita encontrar mejores estados de manera más óptima. Entre las ideas que nos surgen está tener en cuenta la cantidad de acciones legales que existen en un estado dado, con la dificultad que solo se pueden tomar en cuenta para las acciones del agente jugador. También podemos buscar heurísticas más similares a las estrategias usadas por jugadores humanos, que buscan ordenar los valores de las baldosas de manera ascendente en forma de serpiente.

Enlace al Repositorio

El código de este proyecto puede ser encontrado en el link: `https://github.com/jrodriguezru/2048_agent`

Referencias

[1] *2048 (video game)*. URL: `https://en.wikipedia.org/wiki/2048_(video_game)`.