Conecta 4

RESUMEN
Esta es una implementación en *AgentSpeak* del conecta 4. El clásico juego de conectar fichas en un tablero de 8 columnas y 8 filas (el más común es de 7 columnas y 6 filas). Está desarrollado en Jason, un intérprete de *AgentSpeak* construido en Java.

INTRODUCCIÓN

Aún no está claro cuando apareció el *Connect 4* por primera vez. Se estima que apareció a principios del s. XX (1901-1910). Su nombre fue acuñado por primera vez por *Milton Bradley Company* in 1974.

El conecta cuatro se engloba como un *connection game*, un tipo abstracto de juego en el cual los jugadores tienen que intentar conectar piezas. El juego tiene numerosas variaciones en lo que a tamaño nos referimos. El tablero más habitual tiene 7 columnas y 6 filas (7x6), sin embargo existen variaciones de 6x6, 8x7, 7x7, etc. Nuestro conecta cuatro se juega sobre un tablero de 8x8.

Este juego tiene un especial interés dentro del campo de la inteligencia artificial, dentro de la rama que engloba la de árboles de juego. Dentro del campo de *Game Theory*, este juego está clasificado como un juego de información perfecta, ya que los dos jugadores poseen la información necesaria en todo el momento desde el principio de la partida.

DESCRIPCIÓN

El conecta cuatro es un juego en el cual dos jugadores deben poner sus fichas por turnos en un tablero de 8x8 e intentar lograr hacer antes que el rival un cuatro en raya en vertical, diagonal o horizontal. A diferencia de otros juegos como el *TicTacToe*, en este la gravedad es un factor ya que no puedes colocar en cualquier parte del tablero, solo cada una de las columnas libres. Sin embargo, el conecta cuatro se jugará con dos estrategias diferentes: **jugar a ganar** y **jugar a perder**. Ambos jugadores tendrán la misma estrategia en una partida.

El agente implementado jugará el conecta cuatro y sus estrategias e información de la partida las proporcionará un tablero implementado en Java, que, además, se asegurará de que no se produzca movimientos ilegales. Sin embargo, no asegura que no se interfieran los agentes entre ellos y confundan al rival con información falsa. El agente está preparado para evitar comportamientos que desemboquen en movimientos ilegales.

ESTRATEGIA

El agente implementado llevará a cabo el algoritmo *minmax* con una profundidad máxima de 1 (es decir, su movimiento y el del contrario) en todas las columnas del tablero y seleccionará la que mejor puntuación le ofrezca.

Minmax

El *minmax* es una regla de decisión usada en numerosos campos como la inteligencia artificial, teoría de juegos, estadística, etc. Se basa en la minimización de las posible pérdida en el peor de los escenarios. Cuando nos referimos a “ganar” hablamos de máximo y para referirnos a perder hablamos de mínimo.

La mejor forma de entender como funciona el *minmax* es como un árbol de búsqueda de cierta profundidad a partir del cual se evalúa un estado mediante una heurística que devuelve una puntuación asociada. Esto permite maximizar o minimizar en el árbol según este definido. Entonces se evalúan todas las posibilidades y luego se escoge la máxima o mínima según corresponda.

En el caso de conecta 4 la profundidad del árbol de búsqueda es de 2 debido a problemas de eficiencia. Esto significa que solo evaluará un movimiento del jugador propio y uno posterior del contrincante antes de tomar la decisión.

Jugar a Ganar

Para entender bien el funcionamiento de este algoritmo es pertinente definir patrones que ayudarán al entendimiento en la creación de la heurística:

- Tres en raya con solo un espacio libre para hacer cuatro en raya: **TRIO**
- Tres en raya con dos espacios libres para hacer un cuatro en raya: **TRIO-GANADOR**
- Un pareja que pueda crear un TRIO-GANADOR: **PAREJA-GANADORA**

Para jugar a ganar el agente determinará que movimiento debe realizar mediante el árbol de búsqueda *minmax*. La heurística definida para determinar las puntuación se resumen en estos apartados:

GENERALES
- Si encuentra un cuatro en raya propio: 50
- Si encuentra un cuatro en raya del oponente: -50
- Si está en empate el tablero: 0
-

ESPECIFICOS
- Si encuentra un TRIO-GANADOR: 45
- Si encuentra más TRIO: 45
- Si encuentra solo un TRIO: 35
- Si encuentra una PAREJA-GANADORA: 30
- Si encuentra varias parejas determina en base a cuantas se forman la puntuación: 10 + número de parejas
- Si encuentra una ficha sola que puede crear 4 en raya en todas las direcciones: 9
- Si encuentra una ficha sola que puede crear 4 en raya en dos direcciones: 8
- Si encuentra una ficha sola que puede crear 4 en raya en una direcciones: 7
- Si encuentra una ficha sola que no crea nada: 4
- Si no ocurre ninguna de las anteriores: 0

Esta lógica va a ser que la permita evaluar el tablero y posteriormente permitirá al *minmax* maximizar o minimizar según corresponda.

Jugar a Perder

Para jugar a perder, debes lograr que el rival cree un cuatro en raya antes que tú. Para ello el agente tomará dos decisiones basadas en un principio básico determinado por la naturaleza del tablero debido a su tamaño: el que ponga pieza de segundo, gana siempre.

Si el jugar empieza poniendo ficha llevará a cabo el algoritmo *minmax* y elegirá el movimiento que otorgue más ventaja al contrincante. En el caso de que el agente tenga el segundo turno, lo que llevará a cabo será colocar su ficha por encima del rival hasta que éste no pueda realizar otro movimiento que no sea generar un cuatro en raya.

Los valores del *minmax* serán los mismo que definimos en el apartado anterior.

IMPLMENTACIÓN

El programa está implementado totalmente en Jason, un intérprete de *AgentSpeak* construido en Java que proporciona la plataforma para construir entornos multiagente. El agente es genérico y puede implementarse para jugar contra si mismo a partir del mismo código.

PROBLEMAS

El minmax no tiene mejoras, como la *poda alpha-beta* u otros similares, lo cual deja problemas de eficiencia muy grandes debido a la magnitud del problema. Esto además se agrava con el hecho de que la heurística requiere mucho más refinamiento ya que existen casos en los que el agente no toma la decisión correcta.

Se ha creado un paquete llamado *testing* que permite, mediante una regla interna, las pruebas de reglas y planes sin necesidad de detener la aplicación por medio del agente REPL. Sin embargo, la documentación y los comentarios en el código han dificultado la implementación de esta regla. Es muy complejo entender el funcionamiento interno solo mirando el código fuente.

CONCLUSIÓN

El trabajo abarca un problema ya estudiado por *Game Theory*, y nosotros lo abarcamos implementando el ya conocido *minmax* en Jason.

La potencia del lenguaje es muy grande, pero las reglas muestra a veces comportamientos no esperados y de difícil comprensión. Es muy complicado realizar *debugging* ya que las herramientas que proporciona Jason no son muy útiles ni lo suficientemente avanzadas.

En referencia al juego, hemos aprendido la complejidad que conlleva ideas muy simples de resolver para el humano y que son de una complejidad considerable para una máquina.

REFERENCES

Authored Book:
Bordini, R.H. ,Wooldridge, M. y Hübner, J.F. (2007).*Programming Multi-Agent Systems in AgentSpeak Using Jason*

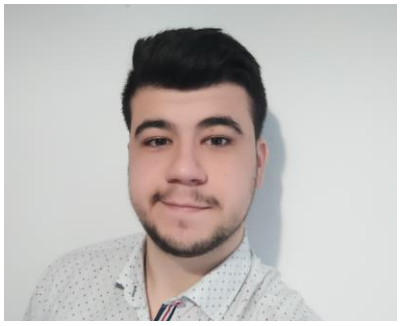
Authored Book:
Chalkiadakis, G. (2011).*Computational Aspects of Cooperative Game Theory*

Web site:
Wikipedia contributors. (2019, 10 mayo). Conecta cuatro. Recuperado 12 mayo, 2019, de https://en.wikipedia.org/wiki/Connect_Four

Web site:
Jason API. (s.f.). Recuperado 7 abril, 2019, de http://jason.sourceforge.net/api/index.html?help-doc.html

Paper presented at…:
Victor Allis (1988,October) . *A Knowledge-based Approach of Connect-Four*. Published in 1988 as Report IR-163 by the Faculty of Mathematics and Computer Science at the Vrije Universiteit Amsterdam

Alfonso Álvarez Pérez
apalfonso23@gmail.com
Estudiante de Ingeniería informática en la ESEI.
Amplios conocimientos de bases de datos.
Dominio de numerosos lenguajes de programación entre los que se encuentran: C, C#, Java, SQL, PHP y C++.
Especial interés en el campo de la ciberseguridad con conocimientos básicos de ingeniería inversa.



Mario Gayoso Pérez
mgperez6@esei.uvigo.es
Estudiante de Ingeniería Informática en la Universidad de Vigo. Ourense.
Conocimiento de lenguajes de programación: C#, Java y SQL.

José Ángel Durán Cerviño
jdcervinho@esei.uvigo.es
Estudiante de Ingeniería Informática.
Conocimientos básicos de Unity.
Conocimientos en C, C#, Java, JavaScript, PHP, HTML
Interesado en el sector de videojuegos y de la ciberseguridad.