4 EN RAYA

RESUMEN

El trabajo que se analiza a continuación consiste en un proyecto basado en AgentSpeak, utilizando Jason, con el propósito de crear un sistema inteligente que siga una estrategia para ganar o perder, según se le indique, a un juego conocido como "4 en raya", cuyo tablero ha sido implementado en Java y se utiliza como punto de referencia sobre el que realizar las jugadas.

INTRODUCCIÓN

En este proyecto se hará uso del entorno, conocido como "tablero", desarrollado en Java y facilitado por el profesor. Sobre este, el agente "jugador" tendrá que colocar una serie de fichas con el objetivo de conseguir colocar 4 seguidas en línea recta, si la estrategia a ejecutar busca ganar, sin embargo, si la estrategia busca perder, intentará evitar a toda costa colocar 4 fichas en línea recta, ya sea horizontal, vertical o diagonalmente.

El tablero tiene una dimensión de 8x8 cuadrículas, por lo que es posible colocar hasta 64 fichas, existiendo la probabilidad de terminar en empate.

El jugador se enfrentará a otro, utilizando un sistema de juego basado en turnos, durante los cuales se puede colocar única y exclusivamente 1 ficha, por lo que puede llegar a ser complejo lograr el objetivo de ganar o perder debido a las diversas posibilidades que dicho tablero permite.

DESARROLLO

INICIO DEL AGENTE

El jugador inicia por defecto con el plan "start", el cual obtiene su nombre como agente para así poder conocer cuales son sus fichas y cuales las del enemigo. Esto se consigue ejecutando conocerEquipo(player) y en función de si el nombre que se le pasa como argumento es player1 o player2, sus fichas serán las designadas con un 1 o un 2, respectivamente, y almacena dicha información.

A continuación, ejecuta el plan "esperar", el cual espera a que el tablero le conceda el turno para finalmente proceder a ejecutar el plan "jugar", que comprueba cual es la estrategia a seguir (ganar o perder) para así realizar unas acciones u otras.

ACCIONES

Para la colocación de una ficha en el tablero, el jugador utiliza put(X,Y), siendo X la posición horizontal e Y la posición vertical, que pueden tener valores entre 0 y 7.

En cuanto a la tolerancia a las trampas, existe un plan por defecto para toda la información que llegue de otras entidades que no sean el tablero o el propio jugador, mostrando un mensaje que indica que ignora a dicha entidad.

Por otra parte, también se asegura de que la información que obtiene del tablero, efectivamente proceda del tablero, como una capa de seguridad a mayores.

ESTRATEGIAS

Analizamos a continuación las dos estrategias implementadas, cuyos objetivos son totalmente opuestos debido a que una busca ganar y otra perder.

JUGAR A GANAR

El jugador comprueba si tiene que iniciar la partida, averiguando si están libres todas las posiciones del tablero, solicitándole dicha información al entorno. En caso afirmativo, coloca en la posición (4,3), que es una de las 4 posiciones más céntricas del tablero.

Para cualquier otro caso lo primero que hace es comprobar si puede hacer 4 en raya, es decir, si puede ganar en este turno.

Para comprobar si puede hacer el 4 en raya, llama a cuatroEnRaya(BestX.BestY), el cual buscará si se cumple que hay 3 en raya desde la posición de cada ficha aliada y la cuarta posición en línea esté libre para colocar una ficha. En caso afirmativo, almacena en BestX la posición horizontal de dicho hueco y en BestY la posición vertical.

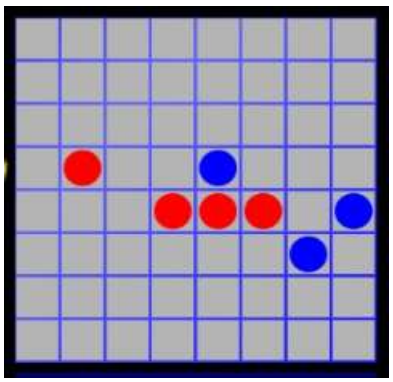

Figura 1.

Si no se cumple esta última condición tampoco, evalúa la siguiente, que es muy similar, ya que consiste en detectar si el jugador enemigo tiene 2 fichas alineadas, para así evitar que realice un posible 3 en raya "aislado" en su siguiente turno.

Esto se debe a que, si puede hacerlo, puede llegar a provocar que le queden dos posiciones que le otorguen la victoria simultáneamente, como podemos observar en la Figura 1 para el jugador rojo.

Las posteriores condiciones a evaluar se basan en buscar la mejor opción para constituir la línea de fichas aliadas más larga hasta el momento mediante tresEnRaya(NextX,NextY), dosEnRaya(NextX,NextY), unoEnRaya(NextX,NextY). Se ejecutan en ese orden hasta que alguna de ellas se cumpla.

Todos ellos realizan lo mismo que el predicado cuatroEnRaya(NextX,NextY) explicado anteriormente, pero buscando una línea con un menor número de fichas aliadas.

En cuanto se cumpla alguna de todas estas condiciones mencionadas, el jugador dejará de evaluar posibilidades y devuelve las posiciones más óptimas almacenadas, de manera que NextX contiene la posición horizontal y NextY la posición vertical para colocar en el tablero la ficha.

JUGAR A PERDER

La estrategia a seguir para perder tiene como objetivo básico evitar poner fichas aliadas continuadas y no bloquear rayas del enemigo. Para ello busca posiciones vacías y comprueba que a su alrededor no tengan una ficha aliada para posteriormente comprobar que al colocar la ficha aliada en dicha posición no se obstruye una posible raya de 4 o 3 fichas enemigas.

Cuando esa condición es imposible que se cumpla, procura colocar en la posición menos perjudicial para sí mismo, de manera que va cediendo ese objetivo básico comentado anteriormente a medida que evalúa condiciones.

Esto es, colocará 2 fichas aliadas en raya como máximo, si no obstruye una raya de 4 o 3 fichas enemigas, que entonces colocará hasta 3 fichas aliadas en línea sin obstruir al enemigo del mismo modo que antes.

Finalmente, si esta última condición tampoco es posible, obstruirá al enemigo procurando hacer la menor línea posible de fichas aliadas y, si esto tampoco es posible, colocará en cualquier posición que esté libre.

CONCLUSIÓN

Del trabajo aquí expuesto, es relevante mencionar que la capacidad del jugador, el cual tiene una estrategia a seguir implementada, permite detectar cuando el enemigo tiene posibilidades de crear una situación favorable para él, al mismo tiempo que analiza cual podría ser la mejor jugada favorable para sí mismo, priorizando debidamente para así alcanzar su objetivo (ganar o perder).

Al ser un juego con unas normas, es posible realizar trampas, así que el jugador también tiene unas directrices a seguir para cuando detecta una posible trampa, procurando trabajar exclusivamente con la información que le proporciona el tablero o él mismo.

También es importante mencionar posibles mejoras, ya que existen métodos como "Minimax", que permiten minimizar la posibilidad de perder, de una manera mucho más recursiva y efectiva que la que nos encontramos en este trabajo.

Finalmente, cabe destacar que para probar el jugador implementado se enfrentó contra otro agente con un código completamente diferente y basado en prioridades, de manera que así era fácil ver los puntos débiles del jugador, ya que, al ir modificando las prioridades de dicho agente para la colocación de fichas, permitía descubrir los errores del jugador, además de facilitar la comprensión de estos.

Este agente de pruebas utilizado también ha servido para demostrar que el jugador desarrollado es imperfecto, ya que no es capaz de ganar en todos los casos.

FUENTES

Beal, Donald F. (1999). *The Nature of Minimax Search*. Maastricht University: Institute of Knowledge and Agent Technology, UM.

A.Guerra-Hernández, J. M. Castro-Manzano, and A. El-Fallah-Seghrouchni (2009). *CTL AgentSpeak(L): a Specification Language for Agent Programs*. Journal of Algorithms, 64:31–40.

González Moreno, J.C. (2017, January). *Introducción a los sistemas inteligentes (Sistemas Inteligentes vs. Inteligencia Artificial)*. Paper presented at the first session of Intelligent Systems subject, Ourense, ESEI.

Dr. Alejandro Guerra-Hernández (2011). *Maestría en Inteligencia Artificial* from https://www.uv.mx/aguerra/documents/2011-mas-slides-06.pdf