Práctica 3- Conecta 4

**resumen**

El objetivo de este trabajo es la implementación de un sistema multiagente (SMA) que pueda jugar de dos maneras diferente al conocido juego Conecta 4. Nuestro sistema puede jugar a perder siempre (dejando que el adversario haga 4 en raya antes) o puede jugar a ganar siempre (siendo el primero en colocar 4 fichas alineadas). Las estrategias de juego se expondrán en los apartados siguientes pero adelantamos que este sistema emplea un algoritmo de decisión para minimizar la pérdida máxima esperada siguiendo algunas de las directrices del algoritmo minimax.

INTRODUCCIóN

Las reglas del Conecta 4 son bastante sencillas, se dispone de un tablero de 8x8 posiciones y las fichas se introducen desde la parte superior de las columnas. El primer jugador en alinear 4 fichas gana la partida.

El algoritmo implementado expande todas las ramas posibles derivadas de jugar la ficha en cualquiera de las posiciones posibles por el agente(max), simulando el turno del oponente hasta llegar a lo que consideramos nodos terminales, en este caso, el nivel de profundidad 1 .

En dicho nivel asignamos un valor a cada posicion con la función de utilidad correspondiente para max o el rival(min), valores positivos y negativos correspondientemente.

Consideraremos un nivel de profundidad 2, por criterios de velocidad, debido a la falta de una

poda que reduzca las ramas posibles.

Arranque del sistema

Arranque del sistema

El funcionamiento de nuestro sistema comienza con el plan *!start* que realiza una llamada al plan *!jugar*, que será ejecutado cada vez que el agente reciba un aviso *turno(player1)* proveniente del entorno.

El plan *!jugar* creará una copia auxiliar del tablero para realizar las predicciones del algoritmo minimax. Dado que el modo a jugar (o estrategia) es escogido por el entorno, el agente también recibirá del entorno un aviso *estrategia(Est),* tomando Est el valor jugarAGanar o jugarAPerder.

Como siguiente paso, el agente decide el movimiento que realizará mediante *?elegirEstrategia(X,Y,Est)* y una vez devueltas las coordenadas (X,Y) a jugar, realiza una acción *put(X)*  en el entorno para colocar ficha. Tras realizar el movimiento, el agente borra la tabla auxiliar creada para las predicciones y se mantiene a la espera de su próximo turno.

predicados esenciales para el sistema

Predicados para determinar opciones de juego

A la hora de escoger una buena jugada, es necesario considerar la totalidad (si se puede) de opciones posibles. Por ello se han implementado varios predicados con el fin de detectar qué casillas están disponibles para jugar, es decir, la posición en la que caería una ficha en una determinada columna.

El predicado *opciones(O)* devuelve una lista O con los posibles movimientos a realizar en el turno. Se apoya en el predicado *calcularOpciones(0,7,O)* que realiza un recorrido de la columna 0 a la columna 7 añadiendo a la lista O las posiciones donde caerá la ficha en cada columna.

Predicados para valorar las opciones a jugar

En este apartado trataremos los predicados necesarios para valorar la utilidad de una posición dada, es decir, determinar si colocar una ficha en esa posición da ventaja al jugador o si se la da al adversario.

Para determinar la utilidad de una posición para el jugador (también llamado Max) se emplea el predicado *asignarValor([X,Y],[V])* que comprobará diversas situaciones de cuatro, tres o dos fichas alineadas (como veremos más adelante en las funciones explicadas en este apartado) y asignará un valor numérico V positivo para representar dicha utilidad. A mayor valor, mayor el beneficio de la jugada.

Por la contra, disponemos de un predicado *asignarValorMin([X,Y],[V])* para comprobar la utilidad de la jugada para el adversario. La estructura es idéntica a *asignarValor* pero los valores que se le dan a V son negativos, cuanto más negativo es V, más beneficia al adversario dicha jugada.

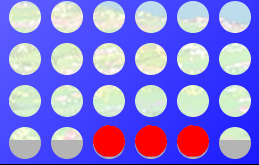
Ambos predicados emplean funciones para analizar el tablero y detectar diversas situaciones de interés que explicaremos de forma inmediata.

El predicado *hago4Raya(X,Y)* comprueba si el jugador alinea 4 fichas colocando una ficha en la posición (X,Y). Este predicado a su vez funciona llamando a los predicados que realmente analizan el tablero en todas las direcciones posibles, estos predicados mencionados son *hacer4RayaHor(X,Y,F)*, *hacer4RayaVer(X,Y,F)*, *hacer4RayaDiag1(X,Y,F)* y *hacer4RayaDiag2(X,Y,F)*, siendo F el valor de la ficha del jugador.

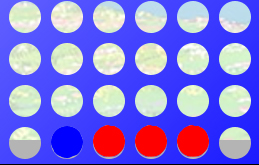
El predicado *hace4Raya(X,Y)* realiza la misma función que el predicado anterior pero realizando las comprobaciones en el papel del adversario, es decir, F tomaría el valor de la ficha del contrincante.

Pasamos a las situaciones de 3 fichas alineadas, y posteriormente a las situaciones de 2 fichas alineadas, que se comprueban mediante estos predicados:

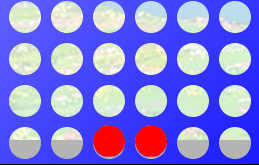
1. *hago3RayaUtil(X,Y)* comprueba si el jugador alinea 3 fichas con espacios a ambos extremos (véase Figura 1) al jugar la posición (X,Y). Utiliza los predicados *hacer3RayaHorUtil(X,Y,F)*,  *hacer3RayaVerUtil(X,Y,F)*,  *hacer3RayaDiag1Util(X,Y,F)* y *hacer3RayaDiag2Util(X,Y,F)* para analizar el tablero en las cuatro direcciones posibles desde la posición (X,Y), siendo F el valor de la ficha del jugador.

 *Figura 1.*

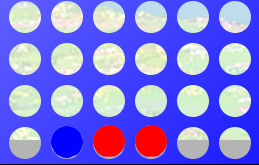
1. *hace3RayaUtil(X,Y)* realiza la misma comprobación que el predicado anterior pero en el papel del contrincante, o sea, tomando el valor de la ficha del rival para F.
2. *hago3Raya(X,Y)* comprueba si el jugador alinea 3 fichas con un espacio en alguno de los extremos (véase Figura 2) al jugar la posición (X,Y). Utiliza los predicados *hacer3RayaHor(X,Y,F)*,  *hacer3RayaVer(X,Y,F)*,  *hacer3RayaDiag1(X,Y,F)* y *hacer3RayaDiag2(X,Y,F)* para analizar el tablero en las cuatro direcciones posibles desde la posición (X,Y), siendo F el valor de la ficha del jugador.

 *Figura 2*

1. *hace3Raya(X,Y)* realiza la misma comprobación que el predicado anterior pero en el papel del rival, es decir, F toma el valor de la ficha del adversario.
2. *hago2RayaUtil(X,Y)* analiza si el jugador obtiene una situación de 2 fichas alineadas con ambos extremos vacíos (véase Figura 3) al jugar la posición (X,Y). Emplea los predicados *hacer2RayaHorUtil(X,Y,F)*,  *hacer2RayaVerUtil(X,Y,F)*, *hacer2RayaDiag1Util(X,Y,F)* y *hacer2RayaDiag2Util(X,Y,F)* para analizar el tablero en las cuatro direcciones posibles desde la posición (X,Y), siendo F el valor de la ficha del jugador.

 *Figura 3*

1. *hace2RayaUtil (X,Y)* realiza la misma comprobación que el predicado anterior pero en el papel del rival, es decir, F toma el valor de la ficha del adversario.
2. *hago2Raya(X,Y)* comprueba si el jugador alinearía 2 fichas con al menos un espacio en uno de los extremos (véase Figura 4) al colocar ficha en la posición (X,Y). Para esta comprobación utiliza los predicados *hacer2RayaHor(X,Y,F)*,  *hacer2RayaVer(X,Y,F)*,  *hacer2RayaDiag1(X,Y,F)* y *hacer2RayaDiag2(X,Y,F)* que analizan el tablero en las cuatro direcciones posibles desde dicha posición (X,Y), tomando F como el valor de la ficha del jugador.

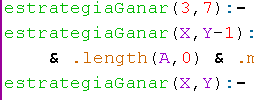
 *Figura 4*

1. *hace2Raya(X,Y)* realiza la misma comprobación que el predicado anterior pero en el papel del rival, es decir, F toma el valor de la ficha del adversario.

Estrategias de juego

JugarAGanar

Para esta estrategia empleamos un algoritmo que realiza un recorrido en profundidad inspirado en la idea del minimax, empleando una función de evaluación en los nodos terminales, en este caso los de profundidad 1.

En la llamada a la estrategia ganar, llegamos a las claúsulas:

las dos primeras determinan la primera jugada por defecto,

si empieza el agente colocará la ficha centrada en la posición

(3,7), maximizando sus oportunidades de victoria, o en caso

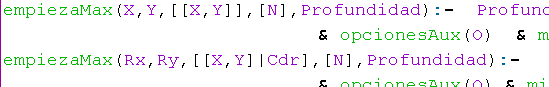
de realizar la segunda jugada, encima de la ficha del oponente,

minimizando sus oportunidades.

Las siguientes jugadas serán dirigidas únicamente por la estrategia implementada,

que asigna a cada nodo(posición jugable) un valor siendo el máximo obtenido de sus

nodos hijos(posiciones jugables por el rival) o el mínimo dependiendo del turno del jugador que se esté simulando.

 Las posiciones X e Y con la mejor jugada que ha determinado el agente unifican en la claúsula:

En ella se recorre la lista con todas las opciones de manera recursiva, estableciendo un valor a cada nodo dado por la función de utilidad correspondiente y para cada opción se actualiza el tablero auxiliar, colocando la ficha del jugador max, calculando las nuevas opciones y llamando a min(Opciones,Valor,Depth)

que nos dará el valor obtenido de la jugada que realizará el adversario en un proceso similar.

Una vez que finalizamos el recorrido de una rama, devolvemos el tablero auxiliar a su estado anterior a realizar el movimiento del jugador min/max, y realizamos la siguiente llamada donde este volverá a ser alterado durante el recorrido en profundidad.



El valor de cada nodo inicial se irá comparando con el resto de nodo del mismo nivel, seleccionando el que genera la mayor utilidad posible, obtenida esta como la suma entre el valor de la posición y el obtenido evaluando sus hijos. Max elegirá siempre aquel nodo con mayor utilidad, y min el nodo con el menor valor.

JugarAPerder

//TO DO

Conclusión

Utilizar esta sección para introducir **vuestros** comentarios sobre el trabajo y que utilidad pensáis puede tener los contenidos tratados. Se deben justificar todos los comentarios, ya sean personales o técnicos

**REFERENCES**

Las referencias que utilizadas deberán ajustarse al estilo que se ejemplifica a continuación.

**Authored book:**

Author, A. A. (1994). *Title of work.* Location/City, State: Publisher.

**Edited book:**

Zhao, F. (Ed.). (2006). *Maximize business profits through e-partnerships.* Hershey, PA: IRM Press.

**Chapter in an edited book:**

Jaques, P. A., & Viccari, R. M. (2006). Considering students’ emotions in computer-mediated learning environments. In Z. Ma (Ed.), *Web-based intelligent e-learning systems: Technologies and applications* (pp. 122-138). Hershey, PA: Information Science Publishing.

**Instance of publication in press:**

Junho, S. (in press). Roadmap for e-commerce standardization in Korea. *International Journal of IT Standards and Standardization Research.*

**Journal article:**

Sawyer, S., & Tapia, A. (2005). The sociotechnical nature of mobile computing work: Evidence from a study of policing in the United States. *International Journal of Technology and Human Interaction, 1*(3), 1-14.

**Unpublished doctoral dissertation or master’s theses:**

Wilfley, D. (1989). *Interpersonal analyses of bulimia: Normal-weight and obese.* Unpublished doctoral dissertation, University of Missouri, Columbia.

**Paper presented at … :**

Lanktree, C., & Briere, J. (1991, January). *Early data on the Trauma Symptom Checklist for Children (TSC-C).* Paper presented at the meeting of the American Professional Society on the Abuse of Children, San Diego, CA.

**Published proceedings:**

Deci, E. L., & Ryan, R. M. (1991). A motivational approach to self: Integration in personality. In R. Dienstbier (Ed.), *Nebraska Symposium on Motivation: Vol. 38. Perspectives on motivation* (pp. 237-288). Lincoln: University of Nebraska Press.

**Web site:**

VandenBos, G., Knapp, S., & Doe, J. (2001). Role of reference elements in the selection of resources by psychology undergraduates. *Journal of Bibliographic Research, 5,* 117-123. Retrieved October 13, 2001, from http://jbr.org/articles.html

****

**VÍCTOR OTERO CABALEIRO**

*vocabaleiro17@esei.uvigo.es, Campus as Lagoas- Edificio Politécnico- 32004 OURENSE*

***Formación:***

*-Estudiante de Grado en Ingeniería Informática(2017- actualidad)*

*-Bachillerato Científico-Tecnológico IES Politécnico de Vigo*

***Experiencia:***

*-Conocimientos básicos de programación en diversos idiomas, especialmente orientados a objetos.*

*-Opositor a bombero.*

***Aptitudes:***

*-Esfuerzo y perseverancia.*

-Empatizo en el lugar de trabajo.

****

**Inés Prieto Gonzalez**

*ipgonzalez2@esei.uvigo.es, Avenida de Santiago 154, 4ºC, 32001, Ourense (España)*

***Formación:***

*2014-2016 : Bachillerato, rama tecnológica - IES As Lagoas*

*2016 - Actualidad : Grado de Ingeniería Informática - ESEI*

*2018 -2019 : Curso de mentorización - ESEI*

*Carnet de conducir B*

*Idiomas : gallego, español e inglés.*



**Ignacio Quintas González**

*iqgonzalez17@esei.uvigo.es, Jenaro de la Fuente 27 - 2o B, 36205 Vigo (España)*

***Formación:***

*01/09/2016–presente 1o- 2o Curso de Grado de Ingeniería Informática, Actualmente*

*matriculado en 3o Curso de Grado de Ingeniería Informática UVIGO Campus Ourense*

*15/09/2014–20/06/2016 Técnico Superior, Desenvolvimiento de aplicaciones*

*multiplataforma.*

*CPR Daniel Castelao, Vigo (España)*

*Lengua materna español, gallego*

*Competencias relacionadas con*

*el empleo*

*- Carnet de conducir B*

**

***Rodrigo Currás Ferradás***

[*rcferradas@esei.uvigo.es*](mailto:rcferradas@esei.uvigo.es)*, Campus de Ourense - Edificio Politécnico - 32004 Ourense*

***Formación:***

*-Estudiante de Grado en Ingeniería Informática(2016-Actualidad)*

*-Bachillerato Científico-Tecnológico (2014-2016), IES As Barxas*

***Experiencia:***

*-Iniciado en programación de interfaces web empleando HTML5, CSS, Javascript y PhP.*

*-Iniciado en programación de objetos y programación concurrente empleando Java*

***Aptitudes:***

*-Puntualidad*

*-Abierto al trabajo en equipo*

*-Perseverancia ante la adversidad*