|  |
| --- |
| CCIA |
| Juego de fuerzas desiguales: El Buitre y los Cuervos |
| IA1 2010-2011 |

|  |
| --- |
| Jesús Fernández Sáez  David Russo Donoso |

JUEGO DE FUERZAS DESIGUALES

En este trabajo desarrollaremos el llamado *Juego de fuerzas desiguales* o, simplemente, *Juego del buitre y los cuervos*.

Dicho juego se desarrolla sobre un tablero en forma de estrella de 5 puntas, y en él participan 2 jugadores: el buitre y el cuervo.

El cuervo cuenta con 7 fichas que, una vez colocadas en el tablero, puede ir moviendo de casilla en casilla con la intención de atrapar al buitre en un lugar donde no tenga escapatoria posible, momento en que se declara vencedor.

Por otro lado, el buitre solo cuenta con una ficha, que puede mover de forma similar a los cuervos, con el añadido de que puede comer fichas enemigas saltando sobre ellas, de forma similar a como se haría en el juego de las [damas](http://es.wikipedia.org/wiki/Damas), pudiendo comer varios cuervos en un mismo movimiento. El buitre se considerará vencedor si logra devorar 4 o más cuervos.

Para Desarrollar el algoritmo del juego, recurriremos al algoritmo *minmax con poda alfa-beta.*

ALGORITMO MINMAX CON PODA ALFA-BETA

El algoritmo minmax se utiliza para que la máquina tome decisiones sobre qué movimiento realizar en su turno. Para ello se apoya en un árbol creado a partir de los posibles movimientos que puede llegar a realizar, llegando hasta una cierta profundidad dada (que en nuestro algoritmo irá incrementándose a medida que el juego avance) y usando una función de evaluación estática definida por nosotros para elegir la rama que considere más apropiada en un momento dado.

La poda alfa-beta se utiliza para no tener que evaluar todos los nodos, simplemente tomando el primero que tenga un valor realmente aceptable; es decir, en cuanto el valor de un nodo sea mejor que el de su nodo padre.

En las [transparencias del tema 7](http://www.cs.us.es/cursos/ia1/temas/tema-07.pdf) de la asignatura de Inteligencia Artificial 1 se puede encontrar más detalles sobre el algoritmo.

Representación del juego

Para representar el tablero de juego nos hemos basado en una lista de tamaño 12, siendo los 10 primeros valores las casillas del tablero, que toman la siguiente forma:

Cada casilla está numerada según la ilustración anterior, y marcada en juego según su ocupación:

-Si la casilla está vacía, su valor será 0

-Si la casilla está ocupada por un buitre, en lugar de su número estará marcada con una “B”

-Si la casilla está ocupada por un cuervo, en lugar de su número estará marcada con una “C”

Los dos últimos valores de la lista, que no representan a ninguna casilla, almacenarán el número de cuervos comidos y el contador de turnos, respectivamente.

## Matriz-movimientos y lista-movimientos

Para tener una idea clara de a que casillas puede moverse una ficha desde el lugar en el que se encuentra, hemos creado \*matriz-movimientos\*, que nos dice, para cada par de casillas, si se puede llegar de una a otra en un solo paso; es decir, si ir de una a otra es un movimiento legal.

\*lista-movimientos\* tiene la misma finalidad, pero en forma de lista, con las ventajas y desventajas frente al array que eso conlleva.

Construimos ambas para poder aprovechar las ventajas de una u otra según la situación nos convenga.

## Matriz-saltos y lista-saltos

De manera similar al punto anterior, aquí tenemos \*matriz-saltos\* y \*lista-saltos\*, que nos sirven para comprobar la validez de un posible salto por parte del buitre (el cuervo no las usa, al no poder saltar).

Para aclarar, un salto consiste en pasar de una casilla a otra pasando por encima de otra casilla que los separa. Esto solo puede hacerse cuando entre la casilla origen y la casilla destino hay tan solo un grado de separación, y únicamente en línea recta.

IMPLEMENTACION DEL JUEGO

Para la implementación de una partida del juego, hemos definido una serie de funciones que lo desarrollan, y que pasamos a comentar a continuación

(defun jugar-humano ())

Esta función se encarga de resolver el turno de un jugador humano. La función comprueba si el jugador maneja al buitre o a los cuervos y le da las opciones que puede realizar para que elija la que prefiera. Seguidamente lee la respuesta del jugador y devuelve el movimiento indicado.

(defun jugar-maquina ())

Esta función, de forma similar a la anterior, resuelve el turno de un jugador controlado por la máquina. Su cometido es calcular la profundidad con la que llamar al algoritmo minmax y luego llamarlo.

Además, si los dos jugadores son máquinas, establece la función de evaluación estática correspondiente al jugador actual.

(defun jugar ())

Esta función resuelve un turno de juego, llamando a las funciones jugar-humano() y jugar-maquina() según corresponda, aplicando los movimientos correspondientes (mediante la función aplica-movimiento(movimiento estado)) y aumentando el contador de turnos.

(defun juego())

Se encarga de ir llamando a jugar() hasta alcanzar un estado final o recibir una respuesta afirmativa a la opción de terminar la partida cada 50 turnos, momento en que imprime el ultimo nodo y da la partida por acabada.