CCIA

Juego de fuerzas desiguales: El Buitre y los Cuervos

IA1 2010-2011

Jesús Fernández Sáez
David Russo Donoso

INDICE DE CONTENIDOS

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, 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 *minimax con poda alfa-beta.*

ALGORITMO MINIMAX CON PODA ALFA-BETA

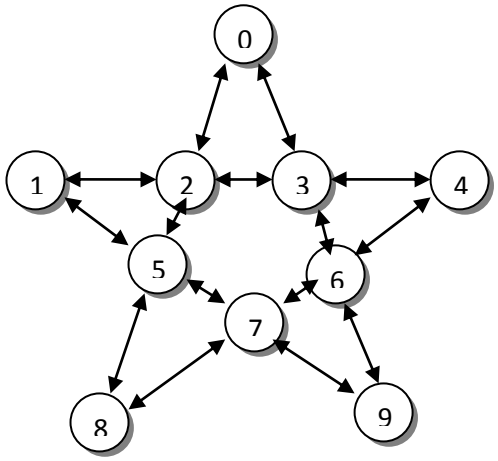
El algoritmo minimax se utiliza para que la máquina tome decisiones sobre qué movimiento realizar en su turno. Para ello se apoya en un árbol creado a partir de los posibles movimientos que puede llegar a realizar, llegando hasta una cierta profundidad dada (que en nuestro algoritmo irá incrementándose 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 de la asignatura de Inteligencia Artificial 1 se puede encontrar más detalles sobre el algoritmo.

REPRESENTACION 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). En la matriz, si el valor de una casilla no es nil, indica sobre qué casilla se está ejecutando el salto.

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 minimax 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.

IMPLEMENTACION DEL MINIMAX

Para la implementación del minimax nos basamos en el algoritmo enseñado en las transparencias del tema 7, pero agrupando la función sucesor dentro de la función sucesores, por la estructura de nuestro algoritmo.

(defun sucesores (nodo))

Esta función devuelve, para un nodo que se le pase como entrada, una lista de nodos sucesores con los posibles movimientos que se pueden realizar a partir de él.

La función determina si juega el buitre o el cuervo, y calcula los movimientos que puede realizar, construyendo un nodo sucesor para cada uno de ellos.

FUNCIONES DE EVALUACION ESTATICA

Para la implementación del juego hemos elaborado 5 funciones de evaluación estática.

Las 4 primeras son combinaciones entre simples estrategias para el buitre o los cuervos: ofensiva o defensiva.

Cuando el buitre se halla en la ofensiva, la función de evaluación estatica se basa únicamente en los saltos que puede realizar el buitre; es decir, en los cuervos que puede comer en un momento dado.

En la defensiva, sin embargo, el buitre se preocupa más por las posibilidades de huir que tiene, buscando un estado en el que le quede el mayor numero de casillas de huida posible. Los cuervos defensivos siguen un esquema similar, poniéndose fuera del alcance del buitre.

Los cuervos en modalidad ofensiva, sin embargo, buscan acorralar al buitre, limitando su posibilidad de movimiento para dejarlo sin posibilidad de huida.

La primera función trata el caso de que el buitre esté a la defensiva y los cuervos en la ofensiva.

En la segunda se da el caso contrario, el buitre toma una estrategia agresiva mientras que los cuervos se repliegan a la defensiva.

Las dos siguientes tratan los casos en que ambos son agresivos, o en el caso de que los dos estén a la defensiva.

Tenemos también una quinta función más exhaustiva, que va “dando puntos” según lo benigno que sea el estado:

· Para el buitre, tiene en cuenta cosas como la cercanía a los cuervos, la posición en el tablero según el momento del juego, el número de cuervos que puede comer desde su posición, etc.

· Para los cuervos se tiene en cuenta cosas como la posición en la que se encuentran (ya que los de las esquinas no pueden ser comidos, por ejemplo) o si un cuervo está al lado del buitre pero no puede ser comido.

Al final tomamos la puntuación y si es MAX la devolvemos tal cual, mientras que si es MIN la negamos antes de devolverla.

PROBLEMAS ENCONTRADOS

Durante la implementación del juego nos hemos encontrado algunos problemas que no hemos sabido resolver de forma satisfactoria.

Existe la posibilidad de que algunas partidas se atasquen y no terminen jamás, debido a que el cuervo siempre tiene alguna posibilidad de huida del buitre, pero al estar ocupado huyendo no puede acorralarlo. Esto ocurre cuando ya tres cuervos han sido devorados, momento en el que para el cuervo prima la supervivencia.

Se trata de un problema complejo, ya que en realidad las máquinas no juegan realmente mal, pues evitan perder, pero una partida infinita no es algo asumible, así que para paliar un poco este asunto, hemos implementado que cada 50 turnos de juego se pueda elegir si interrumpir la partida, dejándola en empate.

Para nuestras pruebas, hemos considerado empate una partida al llegar al turno 200.

El otro problema encontrado es que, probando a invertir los valores de máximo y mínimo valor de las funciones estáticas en los estados ganadores, a veces las máquinas hacen decisiones más inteligentes.

Lamentablemente, a pesar de haber repasado el código de forma metódica no hemos logrado encontrar el motivo de ese comportamiento, pero aun así el juego es 100% funcional y las máquinas hacen decisiones inteligentes basándose en los criterios que les marcamos mediantes las funciones de evaluación estática.

EJEMPLOS DE PARTIDAS: TABLA DE RESULTADOS

Partidas entre máquinas (ganador/número de turnos)

Buitre	Aleatoria	Estatica1	Estatica2	Estatica3	Estatica4	Estatica5
Cuervos						
Aleatoria	Buitre 72	Buitre 10	Buitre 40	Buitre 80	Buitre 58	Buitre 62
Estatica1	Cuervos 19	Cuervos 15	Cuervos 23	Cuervos 17	Empate 200	Empate 200
Estatica2	Empate 200	Cuervos 13	Cuervos 37	Cuervos 13	Cuervos 15	Cuervos 17
Estatica3	Buitre 26	Buitre 34	Buitre 20	Buitre 28	Buitre 24	Buitre 22
Estatica4	Empate 200	Buitre 20	Buitre 42	Buitre 68	Buitre 174	Buitre 20
Estatica5	Buitre 16	Buitre 144	Buitre 40	Buitre 52	Empate 200	Buitre 86

Partidas humano-máquina (ganador/número de turnos)

	Aleatoria	Estatica1	Estatica2	Estatica3	Estatica4	Estatica5
Buitre	Humano 22	Humano 18	Humano 22	Humano 16	Empate	Humano 22
Cuervos	Humano 28	Empate	Empate	Humano 23	Humano 11	Humano 13

Mediante estos resultados podemos afirmar con bastante seguridad que las funciones de evaluación estática creadas no son suficientemente inteligentes como para suponer un serio reto a un ser humano, pero incluso así son lo suficientemente inteligentes como para evitar ser derrotadas siempre.

También podemos concluir, viendo los resultados de los enfrentamientos entre máquinas, que la mayoría de las funciones favorecen al buitre, pero los resultados de la 1 y la 2 demuestran que no hay una descompensación real de bandos, pues los cuervos también pueden ganar con la evaluación adecuada.