ALGORISME MINIMAX

L'algorisme MiniMax es pot utilitzar en jocs on hi hagi dos adversaris, que vagin alternant-se el torn i persegueixin objectius oposats, és a dir, que si un moviment afavoreix un jugador implica que perjudica l'altre. Dels dos jugadors un serà el maximitzant i l'altre serà el minimitzant. Pel maximitzant, la millor jugada serà la que tingui un valor més alt, i pel minimitzant la millor jugada serà la que tingui un valor més baix.

Per tal de determinar el valor d'una jugada farem servir una funció d'avaluació heurística, que, donats un tauler i un color, determinarà el valor del tauler des del punt de vista del jugador que té aquell color. En el següent apartat s'explica amb més detall el funcionament de la funció heurística.

La implementació que s'ha fet de l'algorisme és la següent; primer de tot es generarà un array que contingui totes les posicions on, donats un tauler i un color, es podria col·locar una fitxa d'aquell color. Segons les normes de l'Othello, aquestes posicions es corresponen a totes les que implicarien una captura de fitxes.

Després s'iterarà sobre l'array i per cada una de les posicions que contingui, es crearà una còpia del tauler i s'hi afegirà una fitxa del color que pertoqui a la posició donada. Per tots els taulers resultants se'n calcularà el seu valor heurístic. La màquina triarà la posició on, posant-hi una fitxa, resulti en el tauler que obté el millor valor quan s'avalua amb la funció heurística.

Per tal d'avaluar el valor d'un tauler no necessàriament n'hi ha prou amb considerar el valor que té el tauler després d'haver-hi col·locat la fitxa, ja que una jugada que inicialment podria semblar avantatjosa, podria deixar de ser-ho uns torns més endavant segons les decisions que prengués el contrincant. Aquí és on entra el concepte de profunditat, que és el nombre de torns a què s'avançarà l'algorisme per tal de determinar el valor d'una jugada tenint en compte les possibles conseqüències que aquesta podria tenir.

Així doncs, quan es calcula el valor del tauler on s'ha col·locat una fitxa en una de les posicions possibles, es farà una crida a una funció recursiva que, si s'ha arribat a la profunditat adequada retornarà el valor del tauler resultant, i en cas contrari es cridarà a ella mateixa per continuar explorant les possibilitats del joc fins a arribar a la profunditat que pertoqui.

S'ha implementat dues d'aquestes funcions, una de les quals no explorarà totes les branques resultants ja que en farà podes quan es donin certes condicions. Les dues funcions implementen el mateix algorisme, però aquesta implementarà aquestes podes anomenades podes alfa-beta i s'espera que en descartar possibles jugades sigui més eficient, tot i que s'espera que el resultat de les dues pugui ser el mateix. Primer s'explicarà el cas general sense podes i més avall es detallarà les diferències que implica l'algorisme MiniMax amb podes alfa-beta.

El cas final de la funció MiniMax és clar; es retornarà el valor heurístic del tauler resultant. En el cas que no s'hagi arribat a la profunditat pertinent, es farà un procés similar a l'inici de l'algorisme; es generarà un array amb tots els moviments possibles i per cada un es

calcularà el valor del tauler resultant. Aquest valor no és més que el resultat de la crida a la mateixa funció canviant-ne el color i disminuint-ne la profunditat. De tots aquests valors, es retornarà el millor. Aquí és on entra en joc el concepte del jugador maximitzant i el jugador minimitzant. Per cada crida recursiva a aquesta funció, es canviarà el color de la fitxa que es col·loca (el torn) i el tipus de jugador (maximitzant o minimitzant). S'ha decidit que el jugador que decideix la jugada és el maximitzant, i el seu contrincant és el minimitzant.

Així doncs podem entendre que cada crida a la funció és en realitat una jugada, i cada conjunt de crides que es fan recursivament correspon a un dels diferents camins que podrien donar-se a partir d'un tauler inicial, on per cada jugada, si és el nostre torn posarem la fitxa on més ens convingui, però assumirem que el nostre contrincant també farà el mateix.

Podes alfa-beta

Les podes alfa-beta permeten arribar a una profunditat més gran intentant disminuir les ramificacions per tal de disminuir el temps d'exploració. A grans trets, deixarem d'explorar un possible camí quan veiem que la solució parcial que tenim és pitjor que una que ja és coneguda, ja que amb l'algorisme MiniMax estem fent una cerca en profunditat, i ja s'haurà arribat al node final d'un camí quan comencem a explorar-ne d'altres.

Per tal de decidir si continuem explorant un camí o no tindrem dos valors, anomenats alfa i beta. El valor alfa representarà la cota inferior que es pugui assignar a un node maximitzant, i el valor beta representarà la cota superior que es pugui assignar a un node minimitzant. Quan se surti d'aquest interval, considerarem que no val la pena continuar explorant aquell camí i no es faran més crides recursives.

HEURÍSTICA

Un factors bastant important i que hem de tenir en compte a l'hora d'utilitzar l'algorisme MiniMax o Minimax amb podes és la funció heurística.

Una funció heurística no és res més que un mètode per avaluar els diferents estats -aquí entenem un estat com una instancia determinada del taulell en un moment donat de la partida- computat per l'ordinador per determinar que tan bó és un estat de forma aproximada per tal d'aconseguir guanyar la partida.

En l'Othello, hi ha diversos factors que es poden tenir en compte per implementar una heurística. En el nostre cas, hem triat factors com la mobilitat, el nombre de fitxes i el domini de les posicions claus dins del joc. En el nostre cas hem implementat dos funcions heurístiques.

La primera heurística es una avaluadora estática, això vol dir basicament que utilitzem una matriu amb pesos associat a cada casella del taulell. Llavors el valor heuristic es calculat mitjançant la suma dels pesos on es troben les fitxes pròpies sostraient la suma dels pesos on es troben les fitxes de l'oponent.

Per realitzar aquesta operació, hem implementat un BFS clàssic per recórrer la matriu i anant sumant/ restant pesos. L'algorisme comença sempre des de la casella (3,3) i anirà mirant si els seus veïns estan visitats o no. Si no s'ha visitat, mirarà que la casella no sigui buit, ja que no volem visitar els buits, sumarà els pesos i aquesta posició s'afegirà a la pila. I així fins que pila quedi buida.

La segona heurística a diferencia de la primera, serà una funció heurística dinàmica calculat amb una combinació lineal dels factors mencionats, això és perquè aquests factors no són constants durant el transcurs de la partida i tindran una certa importància segons la fase de joc que es troba la partida, és a dir, si al començament de la partida, a la meitat de la partida o a les últimes rondes.

La mobilitat es refereix a la quantitat de moviments legals que pot fer un jugador. Una bona tàctica es reduir els moviments del nostre oponent i augmentar les pròpies. La mobilitat será quasi sempre relevant durant les tres fases de joc.

La quantitat de fites es refereix a la quantitat de fitxes que té el jugador en aquell moment. Aquesta tàctica és la més cobdiciosa ja que és l'objectiu principal del joc, qui té més fitxes guanya. En el nostre cas només hem tingut aquest factor a la meitat de la partida i cap al final de partida, agafant més importància al final.

Per últim tenim la posició que es basa en qui domina certs caselles del joc. Aquests són:

- Cantonades: són les caselles més importants, ja que aquest no es poden voltejar. Agafaran valors molt alts.
- Caselles X: són les caselles de la diagonal que estan adjacents a les cantonades. Aquestes caselles s'hauran d'evitar ja que ajudaran al nostre contrincant a accedir a les cantonades. Agafaran valors basat baixos.
- Caselles C: són les caselles que estan adjacents a les cantonades. Mateixa situació que les caselles X, per tant agafaran valors baixos.
- Primeres diagonals: són les diagonals que estan adjacents a les caselles X. Aquestes agafaran valors positius ja que ens ajuda a arribar a les cantonades.