Practica 3

José Manuel Pérez Léndinez

(Bot jmplz14)

Diseño de estado.

Como diseño de estado uso directamente el tipo Gamestate dado directamente con la practica.

La única nueva variable que añado al bot es un vector usado para los pesos de mi heurística de segundo jugador. Este vector se declara en la parte privada del bot y se inicializa en el constructor del mismo.

Diseño del algoritmo implementado.

El algoritmo costa de tres funciones principales para generar los estado y seleccionar el mejor hijo.

La primera función es **Move buscarConAlfaBeta(const GameState &state,int niveles):**

A esta función se le pasa como parámetro el estado inicial por el que se empezara la búsqueda. En este caso coincide con el tablero actual al iniciar el turno de juego para nuestro bot.

El segundo parámetro es el numero de niveles que bajaremos en la búsqueda.

Esta función crea los primeros seis estados hijos apartir del estado inicial. Se realiza la llamada a la función que se encarga de crear los nodos hijos y devuelve el valor de este hijo. Se va almacenando el valor del mejor de los hijos que sera el elegido para devolver el movimiento que se hara en este turno.

Para la exploración de los hijos se utilizan dos funciones recursivas.

-double estadoMax(const GameState &state, double alfa, double beta, int profundidad, const GameState &padre, int num_hijo);

-double estadoMin(const GameState &state, double alfa, double beta, int profundidad, const GameState &padre, int num_hijo);

Como el nombre de las funciones indica se utilizara estadoMin para crear un nodo min y estadoMax para crear un nodo max. Estas funciones crean los hijos del stado pasado como primer parámetro y después realiza una llamada a si misma si el jugador tiene otro turno o a la función contraria si el jugador no tiene turno extra. La ejecución recursiva de estas funciones paran cuando se llega a un nodo final o cuando se llega a el parámetro pasado com profundidad.

El parámetro profundidad se va pasando a los hijos restandole uno en cada llamada. También se les pasa los valores alfa y beta típicos para el algoritmo alpa-beta.

El parámetro padre representa al padre del nodo que se pasa como state y es utilizado a la hora de hacer la evaluación de una de mis heurísticas.

El entero num_hijo también es utilizado para el calculo de una de las heurísticas y representa el movimiento que se aplica al padre para obtener el nodo state que se pasa como parámetro.

El algoritmo usa la poda alpa-beta para ir podando ramas una vez que ya tenemos un valor de alpa-beta que se calculara con las funciones de evaluación.

Diseño de las heurísticas

Despues de probar varias heurísticas me he decidido por usar dos.

De jugador 1 utilizo un heurística mas simple y que me perminte sacar mas ventaja respecto a la heurística que utilizo como jugador 2.

La heurística utilizada de jugador 1 se basa en restarle a los puntos del mi jugador 1 los puntos del jugador 2. Después de muchas ejecuciones en la versión de la liga implementada por mis compañeros de clase, tenia un mayor porcentaje de victorias con esta heurística en las partidas en las que yo soy j1 ganando ademas por una cantidad de puntos bastante superior a la heurística que explicare a continuación para jugar como jugador 2.

La heuritica de jugador 2 es mas complicada y tiene varios valores.

Tengo 6 valores en total que van de h0..h5.

H0: El numero de semillas en el pozo mas a la izquierda del tablero,

H1: Cantidad de semillas en todos en todos los pozos de nuestro turno.

H2: Cantidad de pozos no vacíos,

H3: Almacenados en nuestro semillero.

H4: El ultimo movimiento fue el de mas a la izquierda

H5:Almacenados en el semillero del contrincante.

Luego se calcula la heurística multiplicándolas cada una por un peso especifico que esta almacenado en la variable privada del bot llamada pesos[6]. La formula para esta heurística seria la siguiente.

H0 * pesos[0]+ H1 * pesos[1] + H2 * pesos[2]+ H3 * pesos[3] + H4 * pesos[4] - H5 * pesos[5]

Esta heurística es mas probable que gane como segundo jugador que la anterior. Ademas esta heurística a la hora de jugarla como segundo jugador, hace que el resultado de la partida sea mas ajustado. Si se pierde perderemos por menos puntos que la heurística 1 en ese caso. Aumentado a la larga la cantidad de puntos totales para posibles empates en la liga.

A la hora de ganar una partida como segundo jugador en esta heuritica se suele ganar también mas ajustado. Pero con la heurística 1 de segundo jugador es casi imposible ganar una partida(Si el contrincante es bueno). Con esta hemos conseguido ganar unas cuantas mas que con la anterior.

Esta heurística esta basada en un proyecto final de un alumno de la universidad politécnica de milano, con algún cambio.(https://www.politesi.polimi.it/bitstream/10589/134455/3/Thesis.pdf).