Documentació A2 Conecta 4 - PROP

QMatic

Ixent Cornella, Roberto Lupu

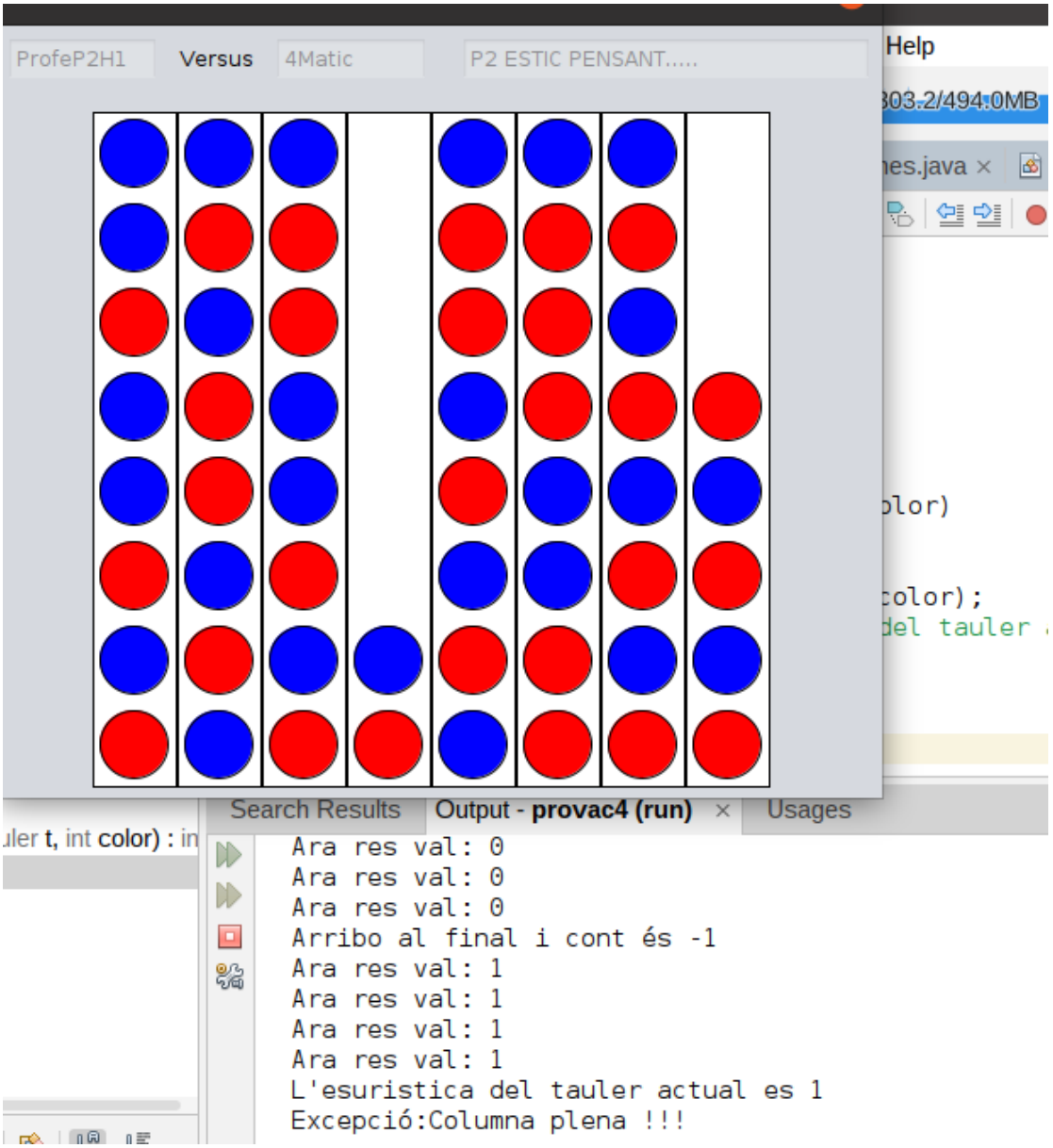
Explicació breu de l'algorisme

L'algorisme que hem és el mateix que el des les **transparències de PROP**, adaptat per a que funcioni amb el nostre cas (un 4 en ratlla). Per tant, és un algorisme de **Minmax amb poda alfa-beta**, i separat amb **funcions** de **Max-valor i Min-valor** que es criden recursivament fins que la profunditat és 0. Quan s'arriba al final de la recursivitat és calcula **l'heurística**, que en resum, és **aproximar** les **possibilitats** de guanyar i perdre vertical, horitzontal i diagonalment, i sumar aquestes possibilitats (amb ajustaments i ponderacions per a obtenir un resultat coherent). Tant l'algorisme com heurística estan més ben explicades avall.

Explicació de com vam arribar a l'algorisme

L'algorisme que hem implementat ha estat directament el **Minmax** amb p**oda alfa-beta** de les transparències de PROP, amb la base del pseudocodi i amb una adaptació inicial pel correcte funcionament de l'algorisme. Aquesta adaptació és l' anomenada funció Minmax, que és realment una copia de Max-Valor però amb una petita variació per a editar també col – variable que **retornarà la columna a la que s'ha de tirar la fitxa.**

És interessant comentar que tal i com vam trobar el pseudocodi, no es seleccionava una primera columna per a tirar a no ser que aquesta tingués una heurística superior a menys infinit, per aquest motiu en una de les partides ens vam trobar amb aquesta situació:



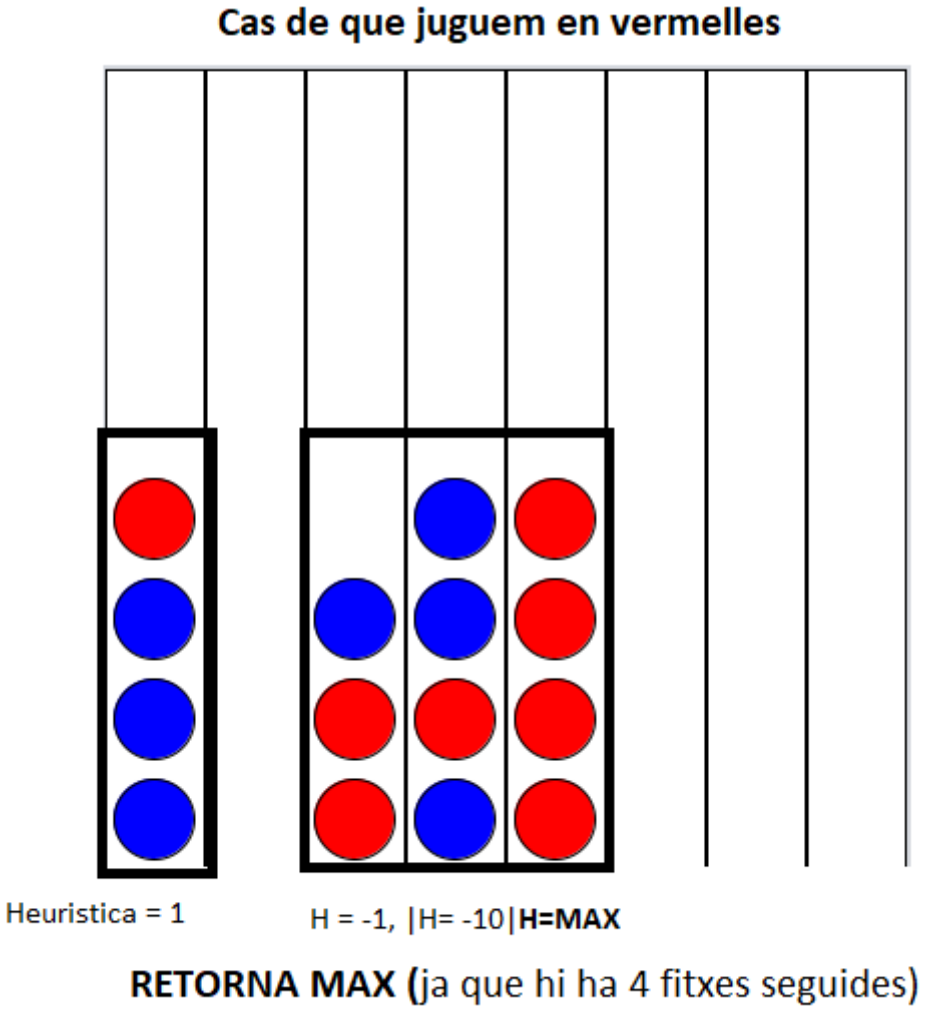
En aquest tauler es pot observar com hi ha 6 columnes plenes, i les altres dues columnes es tiri on es tiri fitxa, el nostre jugador (blaves) perdrà, per tant té una heurística de -MAX (degut a que MAX es una constant estàtica que mitjançant la qual representem heurística infinita), llavors, al no existir cap node amb una heurística major a -MAX no es canvia la variable valor i la fitxa es tira en la columna per defecte a l'inicialitzar, que és un 0, però com està ocupada es retorna un error.
La primera solució que s'ens va ocórrer va ser substituir la comparació VALOR<MIN per VALOR<=MIN, però d'aquesta manera ens carregavem la resta de l'algorisme, a més, en els casos de les funcions maxvalor i minvalor, al tractar-se de la funció min i max de *Math* no es podria controlar aquest igual, per tant vam trobar una altra solució.
Aquesta solució consisteix en assignar el valor -MAX-1 a valor, inicialment. Matemàticament no seria correcte assignar un nombre més petit que menys infinit, però és una petita trampeta per a diferenciar entre no haver escollit cap columna i haver escollit una columna amb valor menys infinit.

Un altre problema que ens vam trobar a l'hora d'analitzar el rendiment, és que el nostre **algorisme era molt lent**. El problema era que treballavem amb dues classes distintes (la principal i "Eines", per simplificar codi i tenir-lo més net), sense implementar funcions estàtiques, i al adonar-nos i voler canviar les funcions estàtiques ens vam adonar que també milloraria l'eficiència si tot el codi estigués en una sola classe, Qmatic. També feiem **System.out.println**, que retrassava significativament l'execució.

Finalment, pel que fa a la **poda alfa-beta**, nosaltres desde un bon principi ja vam incloure aquesta poda en el nostre codi, per aquest motiu no hem vist una gran diferència amb poda o sense. El que sí que hem fet és provar a treure la poda, i passem d'una mitja de +500.000 jugades explorades al principi a 1.6 milions (8^8). Això ens demostra que realment funciona la poda, i ens estalviem recorrer tot l'arbre de jugades possible.

L’Heurística

Tal i com s’ens va recomanar a classe, l’heurística amb la que vam començar va ser una bastant **senzilla**, en la que només miravem de forma **vertical** quantes fitxes d’un mateix color hi havien en una columna abans de trobar-se l’altre color de per mig (o que s'acabessin les posicions de la columna). De manera que si es trobava una única fitxa es sumava 1 o -1 (segons el color de la fitxa i el del nostre equip), si es trobava 2 sumava 10 o -10, si en trobava 3, 100 o - 100 i en cas de trobar-ne 4, MAX o -MAX. Sumant els resultats de cada columna (funció hCol) obteniem la heurística del taulell.



Representació de l’heurísitca simple en un cas concret

Com a segona millora vam introduir un algorisme una mica diferent per a les files, però amb la mateixa idea **d’heurístiques de 1/-1, 10/-10, 100/-100 i MAX/-MAX**. I

l'algorisme havia de ser diferent perquè en el cas de les files, és possible que una mateixa fila sumi un valor i en resti un altre. Pel que fa a les columnes només importa l'últim grup de colors, ja que si l'última fitxa és vermella i la partida encara no ha acabat és impossible que les blaves s'acostin més a la victòria per a aquella determinada columna. Però en el cas de les files, ens podem trobar amb una vermella, dues blaves, una lliure i una altra blava, i que l'avantatge sigui diferent segons l'ordre i les posicions que tinguin, per aquest motiu, hCol és un algorisme de cerca (cerca el nombre de fitxes del primer color que es troba començant desde dalt del tauler) mentre que hFil és més un algorisme de recorregut (ja que és necessari recórrer tota la fila per acabar-la d'analitzar correctament). L'únic problema és que el nostre algorisme no tenia en compte el cas que dues fitxes d'un mateix color estiguin separades per un sol espai en blanc, ni tampoc l'altura a la que es troba. Per aquest motiu vam introduir una millora: ponderació per nivells. En comptes de retornar directament l'heurística de cada fila, es retorna l'heurística de la fila multiplicada per 10 i dividida per la fila+1 (per a no tindre problemes amb la fila 0 i així cada fila té prioritat sobre una fila més alta), d'aquesta manera un 4 en ratlla horitzontal en files més baixes té més heurística que un en files altes.
Com a última millora, vam copiar el codi de les files i el vam adaptar per a que realitzés el recorregut en diagonals (tant diagonal creixent com decreixent), incloent ambdues tipus de **diagonals** en una funció anomenada hDiagonals.

Per últim, i analitzant profundament la nostra heurística, deixem a continuació diferents idees que podríen resultar interesants a l'hora de millorar-la. Veient que el nostre programa compleix els objectius guanyant els dos jugadors que se'ns han proporcionat no hem continuat, tot i així en cas de necessitar-ho, aquí deixem les possibles millores que buscaríem:

- Tenir en compte el cas quan hi ha un espai en blanc entre dos grups de colors per a les funcions hFil i Diagonals.
- Millorar l'operació simple que tenim per diferenciar dues possibilitats de guanyar horitzontalment per la seva altura. Ja que tot i que les possibilitats són petites, és possible que sigui més fàcil d'arribar a guanyar per 4 en ratlla horitzontal en una fila 6 que en una fila 2 per a un taulell determinat.
- Mateixa millora anterior per a diagonals.

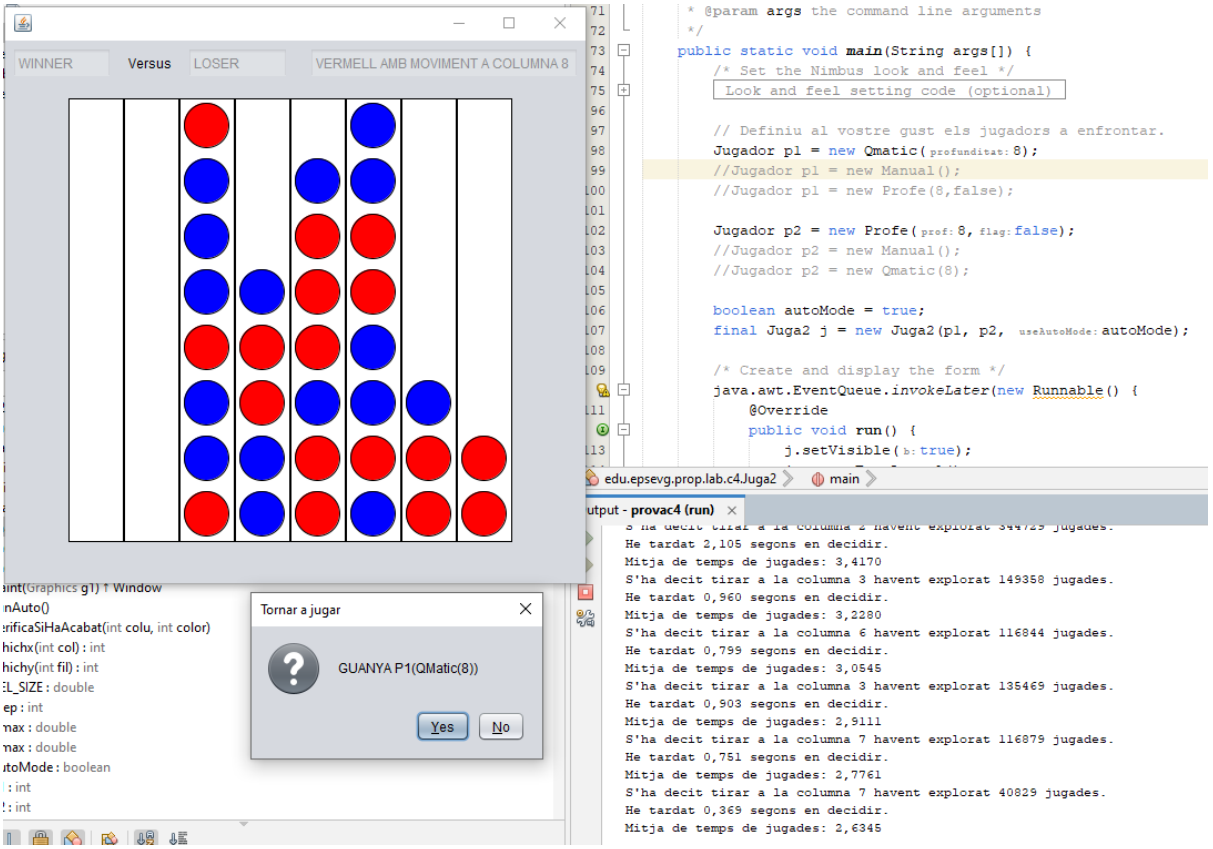
- Establir d'una forma més lògica els pesos de cada tipus de traça (està clar que 3 en ratlla en horitzontal és més difícil de parar que en vertical)
- Tenir en compte els interseccions entre diferents traces (tirar fitxa en una columna on completi tant una traça vertical com horitzontal és més valuós que tirar-ne una on només es traça linea en una direcció.

Resultats

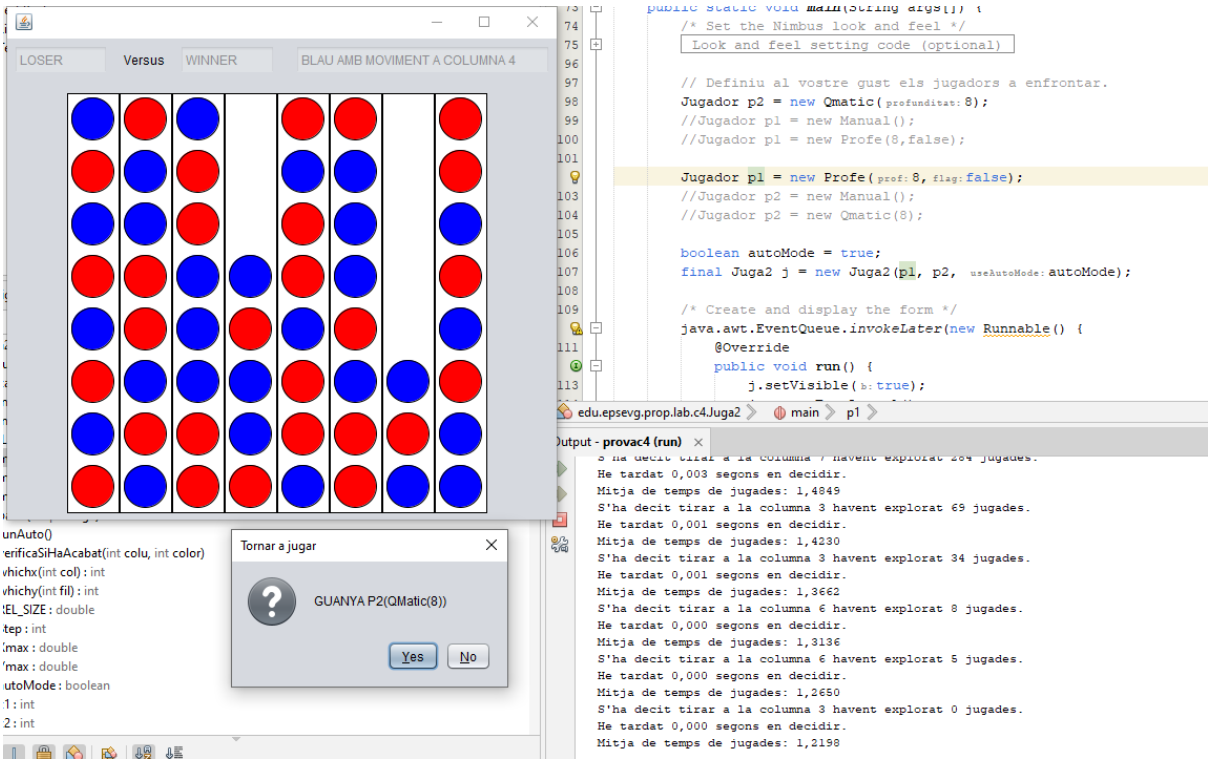
La nostre IA simple “**QMatic**” és capaç de jugar amb profunditat 10 i tot i així tardar menys d’un minut de mitja en triar la columna on tirar. Guanya al professor, aleatori, i fins i tot a humans (aka Roberto i Ixent) amb solvència, a partir de profunditat 6. Consta amb dos paràmetres a l’hora de crear l’objecte, profunditat desitjada i l’opció *printStats* que permet a l’usuari veure algunes estadístiques del algorisme (temps i jugades explorades). També es pot crear amb només la profunditat com a paràmetre (i *printStats* per defecte a true).

Per a crear un objecte QMatic per jugar, cal indicar-li sempre la profunditat a jugar, i opcionalment true o false per si es vol veure les estadístiques per consola.

Personalment, estem satisfets amb el resultat del projecte ja que QMatic ens guanya fins i tot als humans, que era més del que esperàvem.



QMatic(8) contra Profe(8), guanya QMatic amb mitja de temps de 2,63 segons.



Profe(8) contra QMatic(8) , guanya QMatic amb mitja de temps de 1,2 segons.