

Documentació A2 Conecta 4 - PROP



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 **poda 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:

The screenshot shows a Java Swing window titled "ProfesP2H1" with a "Versus" button and a "4Matic" button. The main area displays a Connect Four board with 7 columns and 6 rows. The board is filled with blue and red pieces. The console output window at the bottom shows the following text:

```

Ara res val: 0
Ara res val: 0
Ara res val: 0
Arribo al final i cont és -1
Ara res val: 1
Ara res val: 1
Ara res val: 1
Ara res val: 1
L'esuristica del tauler actual es 1
Excepció:Columna plena !!!

```

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 $VALOR \leq MIN$, 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.

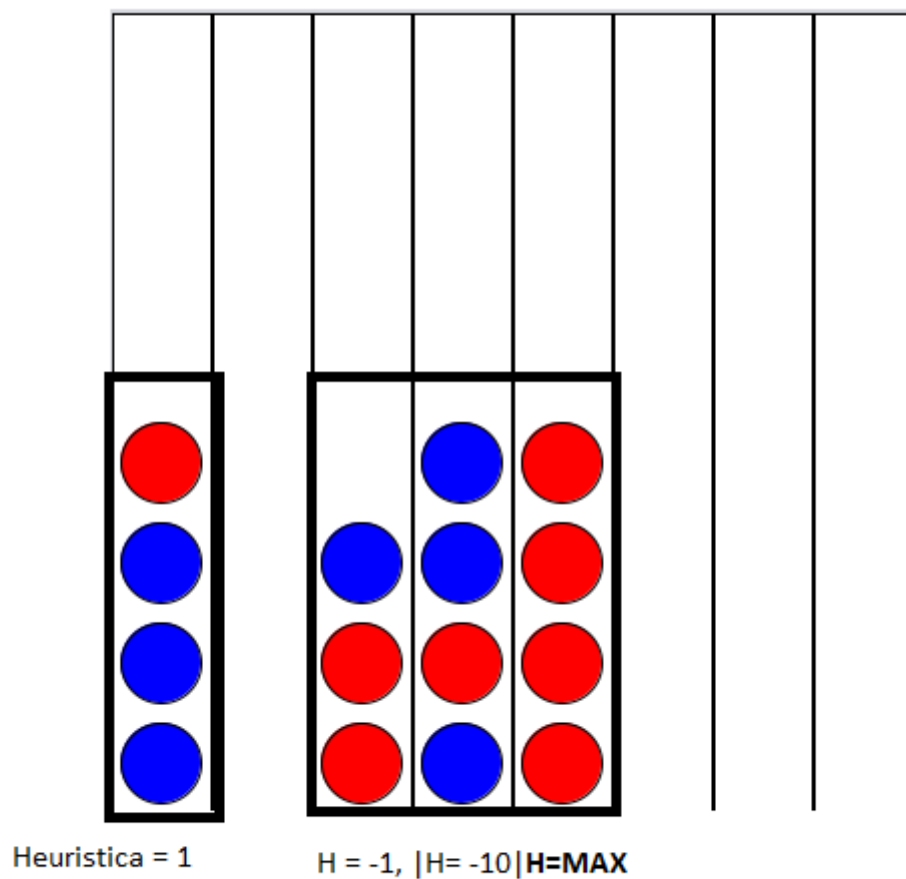
Un altre problema que ens vam trobar a l'hora d'analitzar el rendiment, és que el nostre **algorisme era molt lent**. El problema era que treballavem amb dues classes distintes (la principal i "Eines", per simplificar codi i tenir-lo més net), sense implementar funcions estàtiques, i al adonar-nos i 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 recórrer 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) obteníem la heurística del taulell.

Cas de que juguem en vermelles



RETORNA MAX (ja que hi ha 4 fitxes seguides)

Representació de l'heurística 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 podrien resultar interessants 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.

The screenshot displays a Java Swing application window titled "VERMELL AMB MOVIMENT A COLUMNA 8". The game board is a 7x7 grid with red and blue discs. A small dialog box titled "Tornar a jugar" (Return to play) with a question mark icon and the text "GUANYA P1(QMatic(8))" is shown, with "Yes" and "No" buttons. The console output on the right shows the game progress and statistics for a match between QMatic(8) and Profe(8).

```

71  * @param args the command line arguments
72  */
73  public static void main(String args[]) {
74      /* Set the Nimbus look and feel */
75      LookAndFeel lookAndFeel = new NimbusLookAndFeel();
76      lookAndFeel.installLookAndFeel();
77
78      // Definir al vostre gust els jugadors a enfrontar.
79      Jugador p1 = new QMatic(profunditat: 8);
80      // Jugador p1 = new Manual();
81      // Jugador p1 = new Profe(8, false);
82
83      Jugador p2 = new Profe(prof: 8, flag: false);
84      // Jugador p2 = new Manual();
85      // Jugador p2 = new QMatic(8);
86
87      boolean autoMode = true;
88      final Juga2 j = new Juga2(p1, p2, useAutoMode: autoMode);
89
90      /* Create and display the form */
91      java.awt.EventQueue.invokeLater(new Runnable() {
92          @Override
93          public void run() {
94              j.setVisible(true);
95          }
96      });
97  }
98
99  edu.epsevg.prop.lab.c4.Juga2 > main >
100
101  Output - provac4 (run) x
102  S'ha decidit tirar a la columna 2 havent explorat 34475 jugades.
103  He tardat 2,105 segons en decidir.
104  Mitja de temps de jugades: 3,4170
105  S'ha decidit tirar a la columna 3 havent explorat 149358 jugades.
106  He tardat 0,960 segons en decidir.
107  Mitja de temps de jugades: 3,2280
108  S'ha decidit tirar a la columna 6 havent explorat 116844 jugades.
109  He tardat 0,799 segons en decidir.
110  Mitja de temps de jugades: 3,0545
111  S'ha decidit tirar a la columna 3 havent explorat 135469 jugades.
112  He tardat 0,903 segons en decidir.
113  Mitja de temps de jugades: 2,9111
114  S'ha decidit tirar a la columna 7 havent explorat 116879 jugades.
115  He tardat 0,751 segons en decidir.
116  Mitja de temps de jugades: 2,7761
117  S'ha decidit tirar a la columna 7 havent explorat 40829 jugades.
118  He tardat 0,369 segons en decidir.
119  Mitja de temps de jugades: 2,6345
  
```

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

The screenshot displays a Java Swing application for a Connect Four game. The main window features a 7x10 grid with blue and red pieces. A 'Tornar a jugar' dialog box is open, asking 'GUANYA P2(QMatic(8))' with 'Yes' and 'No' buttons. The background code shows the game logic, including player initialization and move handling.

```

public static void main(String args[]) {
    /* Set the Nimbus look and feel */
    Look and feel setting code (optional)

    // Definir al vostre gust els jugadors a enfrontar.
    Jugador p2 = new QMatic(profunditat:8);
    //Jugador p1 = new Manual();
    //Jugador p1 = new Profe(8,false);

    Jugador p1 = new Profe(8, false);
    //Jugador p2 = new Manual();
    //Jugador p2 = new QMatic(8);

    boolean autoMode = true;
    final Juga2 j = new Juga2(p1, p2, useAutoMode: autoMode);

    /* Create and display the form */
    java.awt.EventQueue.invokeLater(new Runnable() {
        @Override
        public void run() {
            j.setVisible(true);
        }
    });
}

```

Output - provac4 (run) x

```

S'ha decidit tirar a la columna 7 havent explorat 284 jugades.
He tardat 0,003 segons en decidir.
Mitja de temps de jugades: 1,4849
S'ha decidit tirar a la columna 3 havent explorat 69 jugades.
He tardat 0,001 segons en decidir.
Mitja de temps de jugades: 1,4230
S'ha decidit tirar a la columna 3 havent explorat 34 jugades.
He tardat 0,001 segons en decidir.
Mitja de temps de jugades: 1,3662
S'ha decidit tirar a la columna 6 havent explorat 8 jugades.
He tardat 0,000 segons en decidir.
Mitja de temps de jugades: 1,3136
S'ha decidit tirar a la columna 6 havent explorat 5 jugades.
He tardat 0,000 segons en decidir.
Mitja de temps de jugades: 1,2650
S'ha decidit tirar a la columna 3 havent explorat 0 jugades.
He tardat 0,000 segons en decidir.
Mitja de temps de jugades: 1,2198

```

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