Terminal P4
a netrunner implentation
distributing Min-Max computation
with IBM Aglet

Romain Dequidt, Germain Le Chapelain

December 3, 2003

Contents

1 Introduction

We choose to implement a very trivial version of min-max to get familiar with IBM aglets framework.
Here is the minimalist draft for the report we released for our teacher.

1.1 IBM Aglets

Aglets stands for agent-applet. Actualy, an aglet is a quite standard application except that it can dispatch, clone and perform other sorts of operation on itself and other aglets over a network of *aglets servers* running on the set of hosts.

In code, the aglet is a simple class, that extends **Aglet** class of the frame work. This create bench of method for an application and other aglet to interacts with the aglet.

The IBM aglets framework permit developper to deploy very simply a distributed architecture to perform computation on several hosts. In addition, an aglet acces features of the actual host on wich it is running. As an instance.

1.2 Min-max

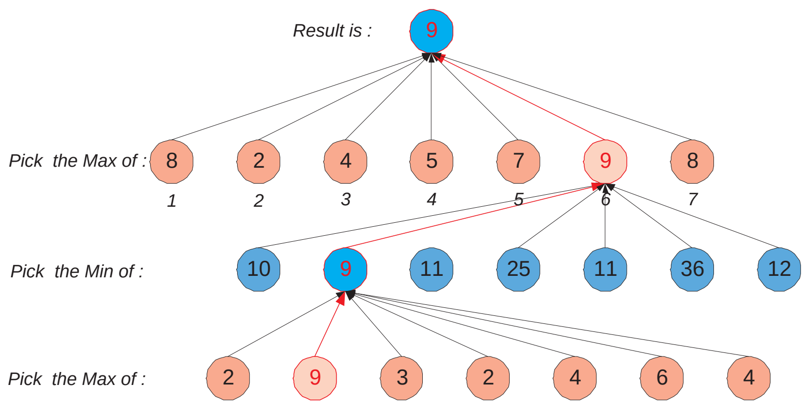
The Algorithm min-max is a very simple algorithm that apply to go-by-go games, in wich there is not much possible goes for player.

The basis idea is to rely on a function that releave the "goodness" of the game for a player. A game *good* for a player is *bad* for the oponent.

Here is the theorical steps of the game:

- You want to play the go that lead to the **Max** game, the game that is the *better* for you.
- The opponent will *obviously* perform a go that lead you to the worst game possible. This is the **Min**
- ... and so one

So choosing the best go is equivalent to choose the go that will lead to the **max** game, knowing that the following go, the one of your opponent, will lead for a **min** game. In such a vision, it is easy to construct the three of possible goes, and determine the better.



*Here, program gonna choose to play in **6** wich is valuate **9***

Alpha-beta

It is obvious that, in a indepth three evaluation, very much computation could be avoid since, when a minima or a maxima is found, leaves can be pruned. As a result, a very simple optimisation of the algorithm is to keep to variable :

$$\alpha and \beta$$

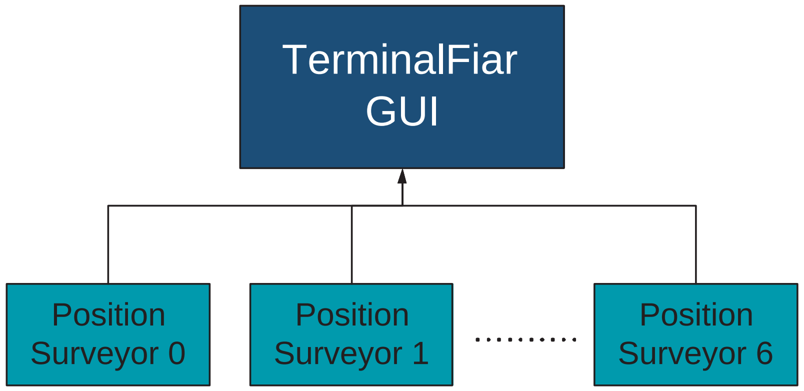
Those variable, maintained during the whole three evaluation permit to prune irevelant leaves. There is absolutely no disavantage in using this optimisation, thus, the trivial Min-max algorithm is only used for pedagogic purpose and never implemented in an actual solution, we decided to get rid of that unfairness by using that simple version rather than the regular Alpha-beta method.

2 Our application

We propose a *For in a row* game implementation, for the previous implementation done during our cursus in EPITA get stocked by the necessity of computing very deeply the possibles-goes three. Here the distributed computation enable seven powerfull Athlon-based EPITA computer to compute a solution parallely.

Our application involve a

- **Server-GUI**, that both interact with the human player throw events and a output windows.
- 7 **position surveyor** (one per columns) wich are probe that inform the server about the evaluation of the game in the case it would decide to play in **there** columns.

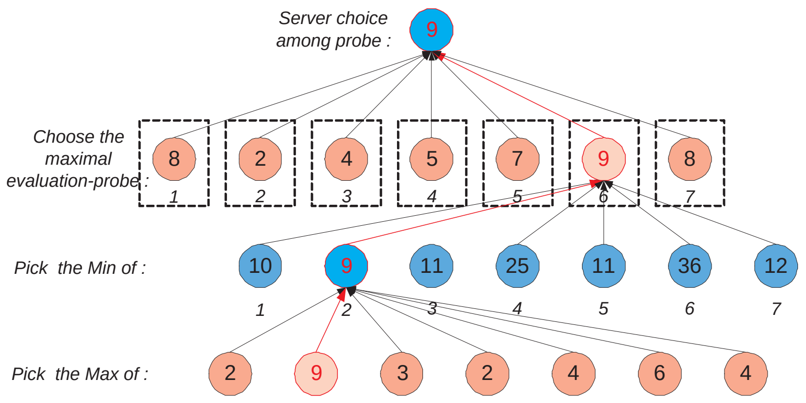


Seven surveyor inform server-GUI of there evaluations, one per columns

Each surveyor acts as a kind of probe that would test, every goes, the evaluation function for the cases server would play in there columns.

The very important distribution of the calculus (for it is the purpose for that project) is actualy the separation of the evaluation of the possible goes on each surveyor.

Actualy, it is just as gowing down of **one** level in the min-max algorithm, assuming that the actual precedent go is the one the probe is testing. Dunno if it is pretty clear, so here is the previous schema including the probe.



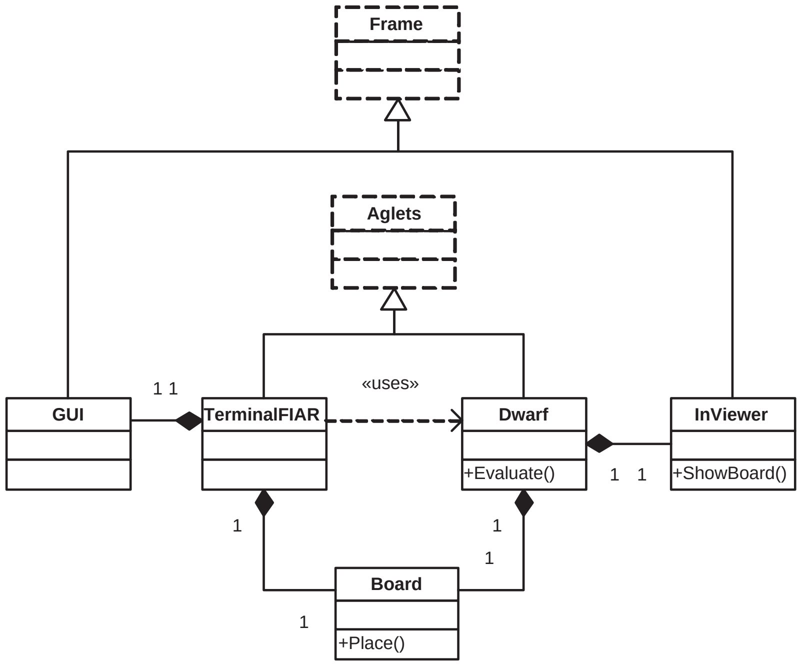
Distributing computation among computer
is just going down a level in Min-max

3 Implementation

The source code release is more to be see as a... kind of draft, cause we rely on structural problem about the aglet framework rather than a very clean, comprehensive modelisation. The software is not even completely finished, but remain a good start point for one who would like to get started in developping with aglets.

3.1 Model

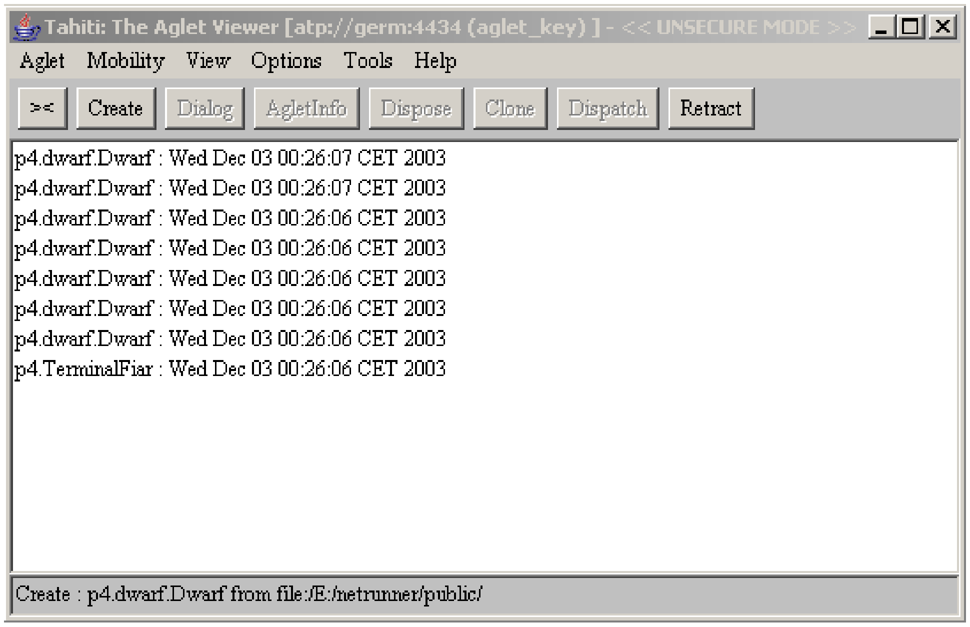
Here is a custom-UML schema of the class interaction :



Distributing computation among computer
is just going down a level in Min-max

3.2 How does it look likeand behaves ?

The main server, **p4.TerminalFiar** is launched from *tahiti*, the included simple aglets server runtime shipped in the whol aglet framework :

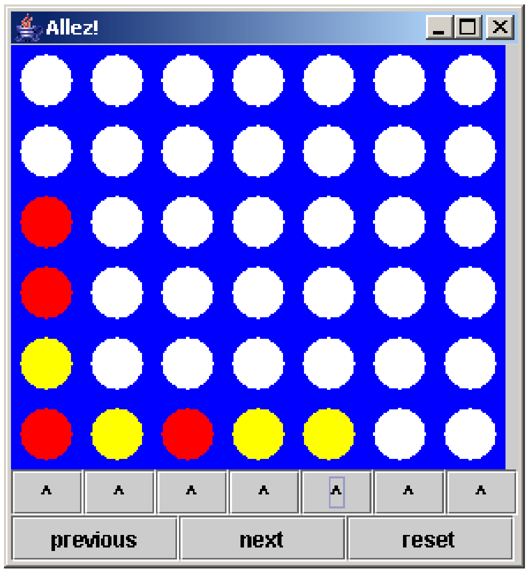

Tahiti: The Aglet Viewer [atp://germ:4434 (aglet_key)] - << UNSECURE MODE >>
Aglet Mobility View Options Tools Help
>< Create Dialog AgletInfo Dispose Clone Dispatch Retract

```
p4.dwarf.Dwarf : Wed Dec 03 00:26:07 CET 2003
p4.dwarf.Dwarf : Wed Dec 03 00:26:07 CET 2003
p4.dwarf.Dwarf : Wed Dec 03 00:26:06 CET 2003
p4.dwarf.Dwarf : Wed Dec 03 00:26:06 CET 2003
p4.dwarf.Dwarf : Wed Dec 03 00:26:06 CET 2003
p4.dwarf.Dwarf : Wed Dec 03 00:26:06 CET 2003
p4.dwarf.Dwarf : Wed Dec 03 00:26:06 CET 2003
p4.TerminalFiar : Wed Dec 03 00:26:06 CET 2003

Create : p4.dwarf.Dwarf from file:/E:/netrunner/public/
```

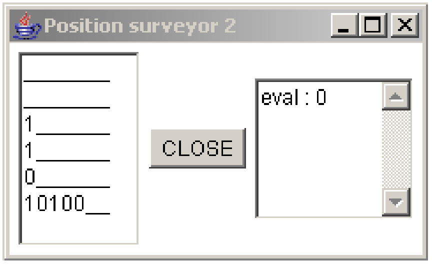
Tahiti sever is one of the few stuffs that work as expected.

Then, the GUI runned in TerminalFiar allow user to interacts with the internal representation of the game in Board.



a SWING GUI that desserve to be au louvre.

The evaluation is done thanks to the seven dwarves that compute evaluation function on each columns. Each dwarch has a minimalist AWT interface to show off is one actual representation of the game, and the evaluation it propose in the case the server should decided to play in its columns

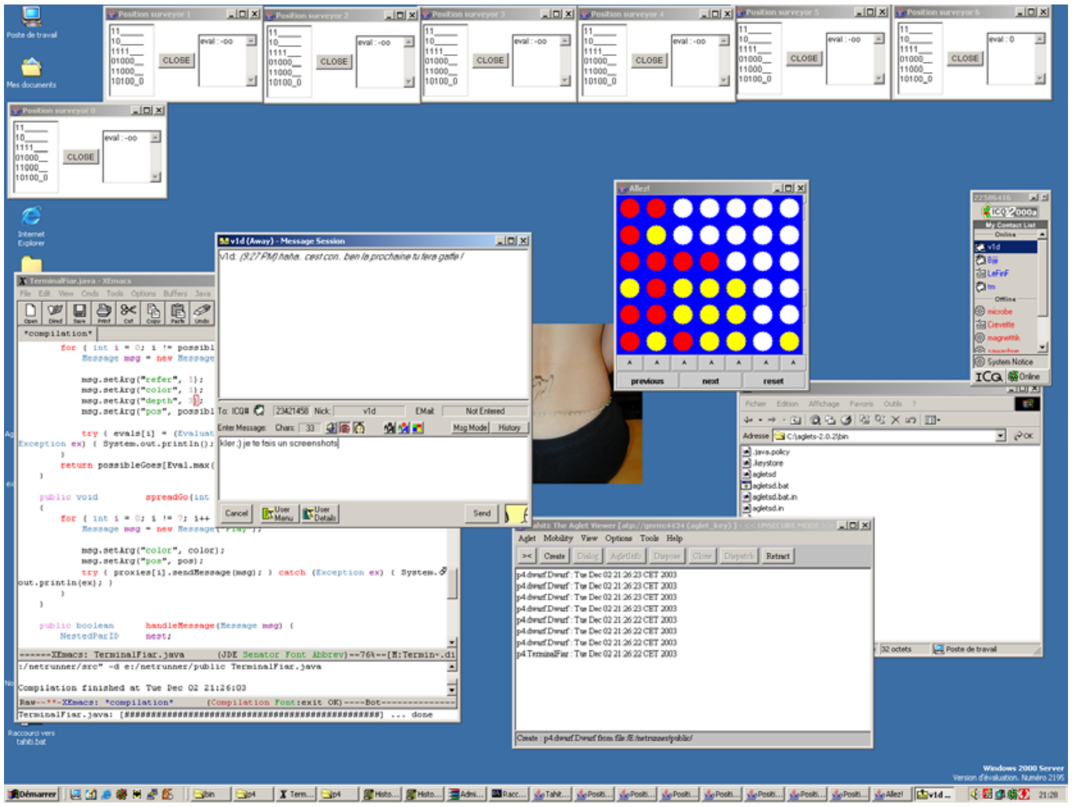


```
______
______
1______
1______
0______
10100__
```

```
eval : 0
```

The well-named dwarf.

And the whole stuff together



```
11____
10____
1111__
01000_
11000_
10100_0
```

```
eval : -oo
```

```
v1d (Away) - Message Session
v1d: (9:27 PM) haha. cest con. ben la prochaine tu fera gaffe !
To: ICQ# 23421458 Nick: v1d EMail: Not Entered
Enter Message: Chars: 33
kler ;) je te fais un screenshots
```

```
TerminalFiar.java - XEmacs
*compilation*
        for ( int i = 0; i != possibl
            Message msg = new Message

            msg.setArg("refer", 1);
            msg.setArg("color", 1);
            msg.setArg("depth", 3);
            msg.setArg("pos", possibl

            try { evals[i] = (Evaluat
Exception ex) { System.out.println();
        }
        return possibleGoes[Eval.max(
    }

    public void         spreadGo(int
    {
        for ( int i = 0; i != 7; i++
            Message msg = new Message("Play");

            msg.setArg("color", color);
            msg.setArg("pos", pos);
            try { proxies[i].sendMessage(msg); } catch (Exception ex) { System.
out.println(ex); }
        }
    }

    public boolean      handleMessage(Message msg) {
        NestedParID     nest;
------XEmacs: TerminalFiar.java      (JDE Senator Font Abbrev)--76%--[E:Termin~.di
:/netrunner/src" -d e:/netrunner/public TerminalFiar.java

Compilation finished at Tue Dec 02 21:26:03
Raw--**-XEmacs: *compilation*      (Compilation Font:exit OK)----Bot-------------
TerminalFiar.java: [##############################################] ... done
```

```
Adresse C:\aglets-2.0.2\bin
.java.policy
.keystore
agletsd
agletsd.bat
agletsd.bat.in
agletsd.in
```

```
Tahiti: The Aglet Viewer [atp://germs4434 (aglet_key)] - << UNSECURE MODE >>
p4.dwarf.Dwarf : Tue Dec 02 21:26:23 CET 2003
p4.dwarf.Dwarf : Tue Dec 02 21:26:23 CET 2003
p4.dwarf.Dwarf : Tue Dec 02 21:26:23 CET 2003
p4.dwarf.Dwarf : Tue Dec 02 21:26:23 CET 2003
p4.dwarf.Dwarf : Tue Dec 02 21:26:22 CET 2003
p4.dwarf.Dwarf : Tue Dec 02 21:26:22 CET 2003
p4.dwarf.Dwarf : Tue Dec 02 21:26:22 CET 2003
p4.TerminalFiar : Tue Dec 02 21:26:22 CET 2003
Create : p4.dwarf.Dwarf from file:/E:/netrunner/public/
```

This image file is called "aglets.png" on my hard-drive

Let's play !

3.3 Specific behavior due to our evaluation function (can look strange)

There is no random playing and no heuristic or cutt-of playing in our implementation. Thus, the AI will play the *very first* go that it can, since it consider it as the maximum go possible.

Since the $max()$ function evaluate position from the left to the right, if to position are equal in evaluation, the most on the left will be choosen, that the reason why the IA *allways* play on the left on its forst go. We could have implemented kind of an heuristic to correct that, but that beahvior doesn't disturb me personnaly, since I can explain it.

The evaluation function in our implementation of the evaluation function is pretty simple :

- A party won (four in a row for the IA) is valuate as $+\infty$
- A party lost (four in a row for the player) is valuate as $-\infty$
- When maximal depth for evaluation is reached : return 0 for evaluation.

We didn't had enough time to improve the evaluation, wich is very important to express, but was totaly irrelevant for the worked we had to do (computation distribution, for those who forgot it as soon.)

Very strange behavior can be observed, especialy when the user can win in two point :

In this particular case, every evaluation on the seven columns will return $-\infty$, because according to **minmax**, there is absolutely no way of preventing player to win.

$$Theplayermodelisedbyminmaxisperfect.$$

Therefore, IA will blow is mind on that case, and just play the first possible go on the left, wich is valuated $-infty$ just as all the others. We introduce the french verb *"Craquer"* for that special case.

A simple pass-throw tips would have made the computer choose at least a position that prevent player for winning on the next go, but again that behavior is more intersting to discuss about rather than to correct by a false method.

3.4 Honesty

We did not manage to use several features of the aglets :

- Impossibility of creating an stand-alone client application that create Aglets on a remote nor local server, even while following litteraly the documentation.
- Usation of Async message seems not to work when the same kind of message is received more than about five time. Therefore we were made to use sync message even in the computation loop, making the application kind of sequential even when deploying aglets over the network. At this point I am just praising for the teacher not to read that line and not to read the related code...
- Failure in our attempt to fully understand security policy of aglets, and especialy the server. Thus, we made or server ”unsecure” by default, that even worst than if mister aglet did default consistent security parameters.

4 Conclusion

Aglets are very intersting in the idea they introduce, but there is still so much technical disavantage in using it.

- Difficulty in installation
- Security philosphy impossible to understand for a regular / low EPITA student brain.
- Very few documentation, and even worth : rotten jokes in the few docs available for version 2.0
- Features simply not working, or with special magic capability.

By the way, netrunner remains a very intersting project, but desperately needs a whole course hour to install correctly the framework and explain the basis of Aglets usation, or, at the very least, a HOW-TO explaining the headlines. Why not giving bonus point for the student that would be in charge for that the next year ?