ORDO v0.7.1.1
Ratings for chess and other games

e-mail: mballicora (at gmail dot com)

August 2013

Abstract

Ordo is a program designed to calculate ratings of individual chess engines (or players) with similar goals as the ELO rating. However, it has different model and algorithm. Its main goal is to keep consistency among the ratings, and calculates them considering all the results at once. In that respect, it shares a similar philosophy than BayesELO[1]. Ordo program is distributed under the GPL license. It is available for GNU/Linux and Windows®, but It is very portable and could easily be compiled in other systems.

Files

In this distribution, you will find this file and and versions for GNU/Linux (32 and 64 bits) or Windows® (64 and 32 bits). For convenience, you can rename the proper file for your system to **ordo** (GNU/Linux) or **ordo.exe** (Windows®). As an input example, a publicly available file `games.pgn` is included. This was taken from *Ingo Bauer's* IPON rating list[2]. A batch file (`ordo_example.bat`) was kindly prepared by *Adam Hair*, and is included in the Windows® distribution. It is a quick and great start for users of that operating system

[1] `http://remi.coulom.free.fr/Bayesian-Elo/`
[2] `http://www.inwoba.de/individual.7z`

Usage

The input should be file that adheres to the PGN standard[3]. Based on the results of that file, Ordo automatically calculates a ranking . The output can be a plain text file and/or a *comma separated value*[4] (`.csv`) file. The `.csv` file is an interesting option because it can be opened/imported by most spreadsheet programs. Once imported, the user can choose to format the output externally. The simplest way to use Ordo is typing in the command line:

```
ordo -p games.pgn
```

which will take the results from `games.pgn` and output the text ranking on the screen. If you want to save the results in a file `ratings.txt`, you can run:

```
ordo -p games.pgn -o ratings.txt
```

By default, the average rating of all the individuals is 2300. If you want a different overall average, you can use the switch `-a` to set it. For instance to have and average of 2500, you can do:

```
ordo -a 2500 -p games.pgn -o ratings.txt
```

or if you want the results in `.csv` format, use the switch `-c`.

```
ordo -a 2500 -p games.pgn -c rating.csv
```

If you want both, you can use:

```
ordo -a 2500 -p games.pgn -o ratings.txt -c rating.csv
```

In addition, `-A` will fix the rating of a given player as a reference (*anchor*) for the whole pool of players.

```
ordo -a 2800 -A "Deep Shredder 12" -p games.pgn -o ratings.txt
```

That will calculate the ratings from `games.pgn`, save it in `ratings.txt`, and *anchor* the engine *Deep Shredder 12* to a rating of 2800. Names that contain spaces should be surrounded by quote marks as in this example.

[3] `http://en.wikipedia.org/wiki/Portable_Game_Notation`

[4] `http://en.wikipedia.org/wiki/Comma-separated_values`

White advantage

An important switch is `-w`, which sets the rating advantage for having white pieces in chess. Alternatively, the (highly recommended) switch `-W` lets Ordo calculate it automatically. With this switch we can complete the above example:

```
ordo -a 2800 -A "Deep Shredder 12" -p games.pgn -o ratings.txt -W
```

Simulation and errors

The swich `-s <n>` instructs Ordo to perform `<n>` simulations. The program will virtually *replay* the games `<n>` times. The results will be assigned randomly according to the probabilities previoulsy determined by the ratings. After running the simulations, and based on all those different results, Ordo will calculate standard deviations (*errors*) for the ratings. For this purpose, an important optional switch is `-F value`, in which `value` is the % *confidence level* used to calculate these errors. The default value is 95.0, which is roughly equivalent to $\pm$ 2 standard deviations. The errors displayed are the ones relative to the pool average. However, if one of the players is *anchored*, the rest of the errors will be relative to that *anchor*. In this case, the anchor error will be zero since it is the point of reference. To know the errors for rating differences between a given pair of engines, the swith `-e file.csv` will generate an error matrix and save it in `file.csv`. To run these simulations, a minimum reasonable number would be `-s 100`. Take into account that the more simulations, the longer it takes to calculate. But, the errors calculated will be more accurate. This is an example for the usage of these switches:

```
ordo -a 2800 -A "Deep Shredder 12" -p games.pgn -o ratings.txt -W
-s1000 -e errors.csv
```

Group connections

Sometimes, a data set contains players or groups/pools of players that did not play enough games against the rest. These *isolated* groups produce meaningless ratings when compared to the general pool. The `-g` switch saves a report of how many groups are in this situation. This information may help to to properly link those groups with extra games and stabilize the ranking.

Perfect winners and losers

Players who have won all games (*perfect winners*) or lost all of them (*perfect losers*) create problems in the rating calculation. It is impossible to estimate those rating accurately because winning all and losing all correspond to an $+\infty$ or $-\infty$ rating, respectively. In addition, the calculation slows down considerably because it cannot converge. Ordo removes these players automatically during the calculation, and place them back in after convergence. The rating assigned to them is a minimum for *perfect winners* and a maximum for *perfect losers*. This is indicated by a $>$ or $<$ symbol in the output text. These limits are established by calculating the rating they would have if one of the games was a draw.

Multiple anchors

When several players are known to have very accurate ratings, it is possible to assigned fixed values to them. In that case, they will behave like multiple anchors. An example will be:

```
ordo -p input.pgn <optional switches> -m anchors.csv
```

where `anchors.csv` is a file that contains lines like this

```
"spark", 2350.0
"glaurung-2.2", 2320
"crafty-23.4", 2275.5
```

telling Ordo to fix spark, glaurung-2.2, and crafty-23.4 to fix their ratings to 2350, 2320, and 2275.5, respectively. The name of the anchors should be present in `input.pgn`.

Switches

The list of the switches provided are:

```
usage: ordo [-OPTION]
 -h          print this help
 -H          print just the switches
 -v          print version number and exit
 -L          display the license information
 -q          quiet mode (no screen progress updates)
```

```
-a <avg>    set rating for the pool average
-A <player> anchor: rating given by '-a' is fixed for <player>, if provided
-m <file>   multiple anchors: file contains rows of "AnchorName",AnchorRating
-w <value>  white advantage value (default=0.0)
-W          white advantage, automatically adjusted
-z <value>  scaling: set rating for winning expectancy of 76% (default=202)
-T          display winning expectancy table
-p <file>   input file in PGN format
-c <file>   output file (comma separated value format)
-o <file>   output file (text format), goes to the screen if not present
-g <file>   output file with group connection info (no rating output on screen)
-s  #       perform # simulations to calculate errors
-e <file>   saves an error matrix, if -s was used
-F <value>  confidence (%) to estimate error margins. Default is 95.0
```

Memory Limits

Currently, the program can handle 3 million games, 50,000 players, and one megabyte of memory for names of the players.

Ordoprep

A tool is included to shrink the PGN file to contain only the game results. In addition, it could discard players that won all games, or lost all games. Other switches allow the exclusion of players that do not have a minimum performance or played too few games.

Typical usage is:

```
ordoprep -p raw.pgn -o shrunk.pgn
```

Which saves in `shrunk.pgn` a pgn file with only the results. You can add switches like this:

```
ordoprep -p raw.pgn -o shrunk.pgn -d -m 5 -g 20
```

where `-d` tells Ordoprep to discard players with 100% or 0% performance, `-m 5` will exclude players who did not reach a 5% performance, and `-g 20` will exclude players with less than 20 games. After all this, `shrunk.pgn` could be used as input for Ordo

Model for rating calculation

The model assumes that differences in strength are analogous to differences in levels of energy. A lower (more stable) level of energy would represent a stronger player. The analogy is that a valley is better at attracting water than a mountain top. In physics and chemistry, a particle or a molecule that can be in two different states can be predicted to be in one or the other with a certain probability.

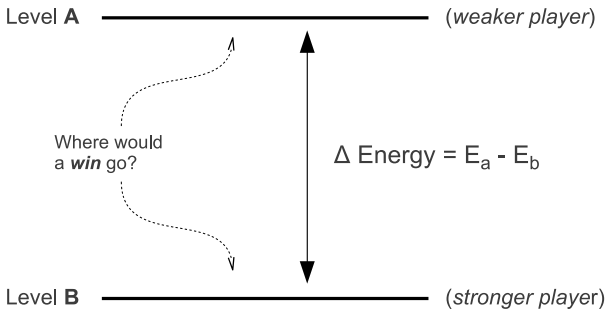

Figure 1: Energetic levels as strength levels

The probability to be found at each level is proportional to the *Boltzmann factor*[5] $e^{-\beta E_i}$. If N_a is the number of particles in level A, and N_b is the number of particles in level B, their ratio is:

$$\frac{N_a}{N_b} = \frac{e^{-\beta E_a}}{e^{-\beta E_b}} = e^{-\beta(E_a - E_b)} \tag{1}$$

β is a constant of the system (that depends on the temperature, which is noise in our case). The analogy is that we treat the probabilities of a *win* to land in level A or B as the probability of a particle to be in A or B. Therefore, after reordering equation 1, the *fraction of wins* ($f_{b,a}$) of player B in a match vs. A will be:

$$f_{b,a} = \frac{N_b}{N_a + N_b} = \frac{1}{1 + e^{-\beta(E_a - E_b)}} \tag{2}$$

if we define strength S as the negative value of *energy*, then, $S_a = -E_a$.

[5] `https://en.wikipedia.org/wiki/Boltzmann_factor`

$$f_{b,a} = \frac{1}{1 + e^{-\beta(S_b - S_a)}} \tag{3}$$

This equation has the same form as the logistic function[6]. With this equation we can calculate the predicted fraction of wins between two players. The predicted performance P_x, or number of wins of player x among a pool of other players will be the summation of each of the predicted fractions f for each game[7].

$$P_x = f_{x,opp(1)} + f_{x,opp(2)} + ... + f_{x,opp(n)} = \sum_{i=1}^{n} f_{x,opp(i)} \tag{4}$$

.

where n is the total number of games played by x and $opp(i)$ is the opponent it faced in the game i. Then:

$$P_x = \sum_{i=1}^{n} \frac{1}{1 + e^{-\beta(S_x - S_{opp(i)})}} \tag{5}$$

The most likely strength values (S) for each player are ones that satisfy that each *predicted* performance P_x equals the respective *observed* performance (O_x) of player x (actual number of games won by x). Therefore, the goal is to find S values so the following unfitness (U) score equals zero, where m is the total number of players, and j is each individual player.

$$U = \sum_{j=1}^{m} (P_j - O_j)^2 \tag{6}$$

Finding an adequate procedure to minimize U until reaches zero is critical for a proper convergence towards the optimal solution. The way Ordo fits it is in discrete steps (similar to *hill climbing*[8]), and making those steps smaller and smaller once the convergence was reached. However, those steps

[6] `http://en.wikipedia.org/wiki/Logistic_function`

[7] This is analogous to the number of particles/molecules that will be in state x in when there are other other energy levels or states

[8] `http://en.wikipedia.org/wiki/Hill_climbing`

are constrained to certain values to avoid big swings during the calculation. After many different tests, this procedure was found to be safe and fast.

Scale

Chess players are accustomed to the ELO rating, which is based on a Gaussian distribution[9]. The default value of β was chosen to make sure that Ordo's scale resembles the ELO scale. For that purpose, the rating difference when the winning expectancy is 76% has been set to 202 rating points. This parameter could be modified with the switch `-z`, and the overall scale can be displayed with switch `-T`.

The model is valid if the strength assigned to the individual players is additive like energy. If we know the strength differences between A→B and B→C, we should be able to calculate A→C as A→B + B→C. Then, the difference A→C should accurately predict the results of a match between A and C. Empirical observations seem to suggests that those estimations are reasonable, at least within a certain range.

It is important to note that the theoretical model does not take into account the existance of draws. Basically, this means that the performance of two draws equals one win and loss. If enough games are played and the draw rate for similar ratings remains similar for different players, this assumption should not alter the calculation. Empirically, this is a reasonable assumption to make.

Acknowledgments

Adam Hair has been testing Ordo extensively and suggested valuables ideas that certainly improved the program.

License

```
ordo v0.7.1.1

   Copyright (c) 2013 Miguel A. Ballicora
   Ordo is program for calculating ratings of engine or chess players
```

[9] `http://en.wikipedia.org/wiki/Elo_rating_system`