Rapid chess: A massive-scale experiment

Diego Fernández Slezak[1], Pablo Etchemendy[2], and Mariano Sigman[2]

[1] Departamento de Computación, FCEyN, UBA, Pabellón 1, Ciudad Universitaria, (C1428EGA) Buenos Aires, Argentina
[2] Departamento de Física, FCEyN, UBA, Pabellón 1, Ciudad Universitaria, (C1428EGA) Buenos Aires Argentina

Abstract. The proliferation of chess servers on the Internet has turned active chess, blitz and lightning, into a vast cognitive phenomenon involving engaged participants. Here we use this large database of human decision making (rapid chess) as a privileged window to understand human cognition. FICS (Free Internet Chess Server), `http://www.freechess.org/` is a free ICS-compatible server for playing chess games through Internet, with more than 300.000 registered users. Using this available chess server in the Internet, we constructed a massive decision-making database. This data includes thousands of million moves of chess games, with the estimated time of each one of them. In order to evaluate the goodness of moves, we used Crafty (an open-source chess engine) to analyse the score of the move. This process is compute expensive, so we parallelized the analysis on a Beowulf cluster. We studied the structure of the time players take to make a move during a game, and using parallelization we were able to analyse a huge amount of moves obtaining a quantification of the quality of the decision made in *millions* of instances. This approach allowed us to identify a number of statistical fingerprints that uniquely characterize the emergent structure of the game.

1 Introduction

Rapid chess (blitz and lightning) provides an unparalleled laboratory to understand decision making in a natural environment. In a game of chess players make around 40 movements, each comprising a decision. The time budget is finite and hence players need to adopt a policy of time usage. The outcome of each individual decision, i.e. to what extent it changes the value of the position, as well as the quality of the players, can be both measured accurately. Web-based chess is a natural laboratory that produces vast amounts of data, about 100K decisions a day, incommensurable with laboratory based experiments. This large database of human decision making can be used as a privileged window to understand human cognition. Computer scientist have been recently embarked in a project to direct voluntary use of human computing cycles in a coherent and productive direction [14–17]. For instance, in GWAP (Games With a Purpose) people play a game in which they determine the contents of images by providing meaningful labels for them. Thus, a computationally intractable problem (image labeling) is solved by encouraging people to do the work by taking advantage of their desire

to be entertained. Here, we use in a similar vein a leisurely cognitive activity, rapid chess, as a window into cognition.

Chess has long been a model system to study complex thought processes [2, 4–6, 8, 9, 11]. In particular, a consensus has emerged in that chess expertise comes in two forms: the ability to calculate variations (search) and the ability to recognize and remember meaningful patterns on the board (pattern recognition). The prevalent view is that expert players, as opposed to weaker ones, excel specifically at rapid object recognition abilities [1, 3, 4].

We hope that our work may prompt other large-scale studies in chess as well as similar decision-making activities. As Gary Kasparov suggests in his book "How Life Imitates Chess: Making the Right Moves, from the Board to the Boardroom" chess is more than a metaphor: it makes the case for using chess as a model for understanding and improving human decision-making everywhere else.

FICS (Free Internet Chess Server), `http://www.freechess.org/` is a free ICS-compatible server for playing chess games through Internet. This server is on-line since 1995, and has more than 300.000 registered users. Each registered user has associated a rating that indicates the chess skills strength of the player, represented by a number typically between 1000 and 3000 points. The rating is a dynamic variable which is updated after each game played according to the Glicko method (`http://www.glicko.net/glicko.html`). Also, a rating deviation (RD) is used to determine the stability of the rating measure and hence, how much a player's current rating should be trusted. A high RD indicates that the player may not be competing frequently or that the player has not played many games yet at the current rating level. A low RD indicates that the player's rating is fairly well established. For this work, we did not take into account the rating deviation.

Registered players may be humans or computers. These two types of players are distinguished in the server. Only 5% of our database players are computers: 2067 out of a total of 44069 players. For this work we discarded all games played by at least one computer.

Users connect to FICS using graphical interfaces, e.g. BabasChess (`http://www.babaschess.net/`) or Xboard (`http://www.gnu.org/software/xboard/`), or command line clients, e.g. telnet. Once connected, users can create, play and observe games.

1.1 Score

An ideal evaluation function would assign to each position three possible values according to the result following best play from both sides: 1, if white is won, 0 is the result is a draw and -1 if black is won. An ideal evaluation function exists for other type of games, as checkers, which is known to result in a draw with perfect play [10]. However, such ideal evaluation function does not exist for chess and most likely will never be computed according to many theoretical thinkers such as Claude Shannon [12].

An evaluation function in chess approximates an ideal one considering material value along with other factors affecting the strength of each side. When counting up the material for each side, typical values for pieces are 1 point for a pawn, 3 points for a knight or bishop, 5 points for a rook, and 9 points for a queen. The king is sometimes given an arbitrary high value such as 200 points [12], or any other value which adds more than all the remaining factors. Evaluation functions also consider factors such as pawn structure, the fact that a pair of bishops are usually worth more, centralized pieces are worth more, and so on. All these factors are collapsed on a single scalar, the score, typically measured in hundredths of a pawn, which provides an integral measure of the goodness of a position. Then, the evaluation is a continuous function which assigns a score (often also referred as value) to each position, i.e. an estimate of the likelihood of the final result. Conventionally, positive values indicate that the most probable outcome is a win for white.

If after a white move, the score drops abruptly, white winning chances decrease correspondingly. It is then said, in the chess jargon, that white has blundered. Seemingly, if the result of a black move is that the score goes up abruptly, he has lost winning chances, committing a blunder. Hence, the measure $\Delta S = (S(i+1) - S(i)) \cdot C$ where C is the color function (-1 for black moves and $+1$ for white moves) provides a measure of the goodness of the move. Negative (positive) values of ΔS indicate that the moving side has lost (increase) its winning chances.

2 Creating the database

We developed an application that connects to the FICS server every 30 minutes and downloads stored games. The server stores only the last 10 games of each player. We chose to download only the games played by players in the 15% ratings percentile of logged-in players. The application consists of a Python script that connects to the server using TCP sockets and downloads the stored games into a PostgreSQL (`http://www.postgresql.org/`) database using the standard ICS-compatible instructions (`history` and `smoves`).

First, the history of logged-in players is queried, checking for repeated games. Once a new game has been found, the detail of game is queried and server answer is parsed and converted into the PGN file format; finally, the PGN is stored in the database. We store the nickname of the logged-in players, the game information (total time, increment, white and black nicknames, players' rating, date, opening variant and result) and the moves of the game with the corresponding time between moves (in milliseconds precision).

2.1 Calculating moves score

We quantify the goodness of a move by calculating each move score. The score is a number between -999 and $+999$, that show the valuation of a certain move; 0 meaning this moves is not good for white nor black, positive and negative values

give advantage to white or black, respectively. For the analysis of the results, we saturated the score larger than 10 and smaller than −10, as these extreme values are not interesting in the current analysis.

We used crafty (`http://www.craftychess.com/`), an open source chess engine written by Robert Hyatt (`http://www.cis.uab.edu/hyatt/`), to analyze the moves and calculate the score. The analysis consists of evaluating the decision tree from a given board position, up to a predefined depth of move number; we used analysis with 8 moves of depth. A chess game consisting of approximately 100 moves would take at least 15 seconds to be analyzed on a Intel XEON 2.2GHz, 2GB RAM. Due to the amount of downloaded games and moves, calculation of score would be impossible to obtain using standard computers.

In this sense, the calculation of score was parallelized using a Beowulf [13] cluster. Every 3 hours, new games downloaded are selected, writen into PGN files and sent to the cluster. Then game analysis application is queued in the cluster. Using the basic MPI functions [7], the files are divided into equal-sized groups and sent automatically to the available nodes (`http://cecar.fcen.uba.ar`) to be analyzed in parallel.

The complete architecture is shown in figure 1.

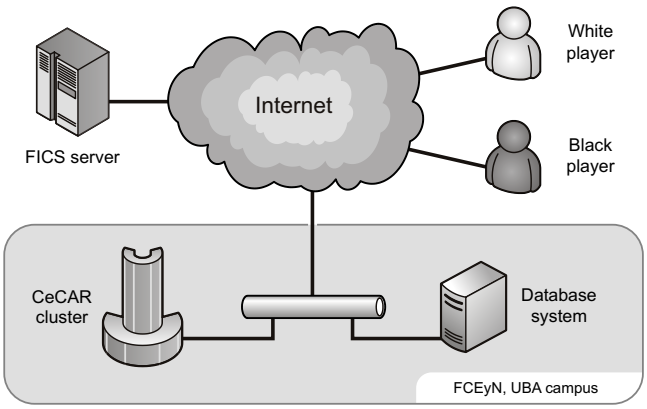

Fig. 1. Architecture of the complete system. Players play chess through the internet in the FICS server; our database system downloads new games, and sends them to the CeCAR cluster to analyze the score of each move.

3 Results

Our robot started functioning in May, 2009, downloading only lightning and blitz games, which means total times going from 1 minute to 15 minutes. On January 2010 the database consists of more that 2.8M games (downloading between 10K and 20K games per day), resulting in more than 200M total moves. This is

equivalent to a person who played 3 minutes games for 27 years without leaving the computer.

As we detailed before, each user has a rating, which indicates theirs game capacity, represented as an integer tipically 1.000 y 3.000; as larger this number, better the player. In other words, a player with 1000 rating points less than the opponent is very unlikely to win. We can imagine a sigmoide curve, which would describe this characteristic, where we express the probability of winning vs. the Δ rating. In figure 2, we calculated this curve. For each Δ rating we count how many games were won (or draw) by each player. A value of 1 means that white has a probability of 1 for winning the game; on the other hand a value of -1 means that black has a probability of 1 for winning the game.

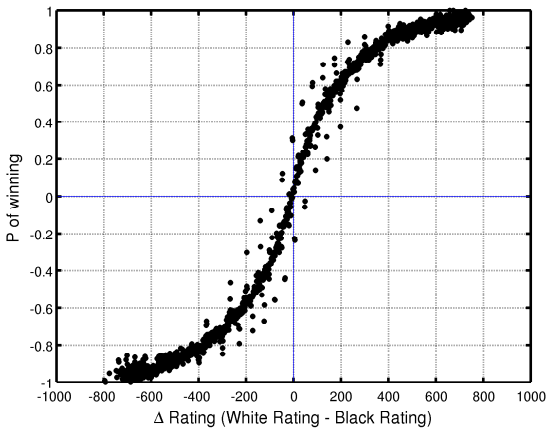

Fig. 2. Probability of winning vs. the Δ rating. For each Δ rating we count how many games were won (or draw) by each player, and divides for the total games played. A value of 1 means that white has a probability of 1 for winning the game; on the other hand a value of -1 means that black has a probability of 1 for winning the game.

As expected, figure 2 showed a sigmoid curve representing the chance of winning for each Δ rating. Obviously, when Δ rating is 0 (both players have the same rating), the expected result is a draw; or more precisely, after playing many games, both players will have won the approximately the same number of games. This values change very rapidly, showing that with a Δ rating of 150 (white has 150 more points than black player) white player has 3 times more chances to win the game than black one.

In the following table we show the number of games analyzed so far for each total time (without increment):

Total Time (min)	#
2	23872
4	38551
10	49865
5	69189
3	127635
1	465483

3.1 Game stages

As a very first look to the data collected, the three stages of a chess game are a very interesting aspect to analyze. Chess may be separated into three different stages: opening, middle-game and end-game. Each one of them has very different tactics and strategies of playing, including the use of time. In figure 3 we selected 1000 random games of 3 minutes of total time per player without increment, and plotted the duration of each move for each game. Games are sorted with length

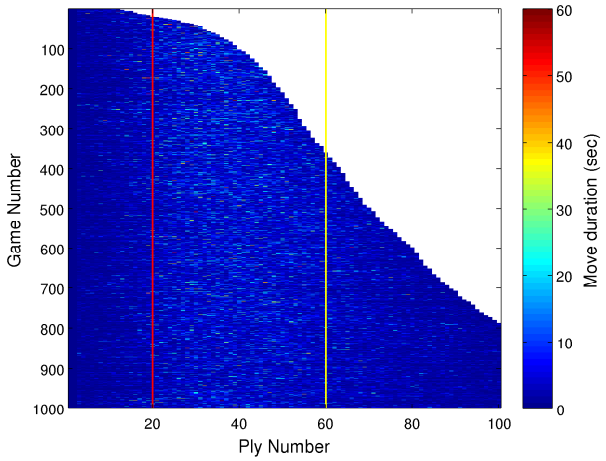

Fig. 3. Move duration of 1000 random games. Red line shows separation a possible division between opening and middle-games stages; yellow line, between middle-game and end-game.

(in number of moves). In this figure, we can distinguish the three different stages (marked with red and yellow lines). In the first stage, opening, players tend to play using opening books, hence playing very fast. As middle-game approached, more copmlex game situations appear and time spent to move is longer. Finally, on the endgame, the time constraint make players to play fast again.

3.2 Scores

One interesting aspect to look at is the evolution of score during the game, depending of the player's rating. The database, along calculating the score of each move, also has the Δscore, being the difference of score between 2 consecutive plies. In figure 4 we show 4 examples of games chosen at random, for different player's ratings.

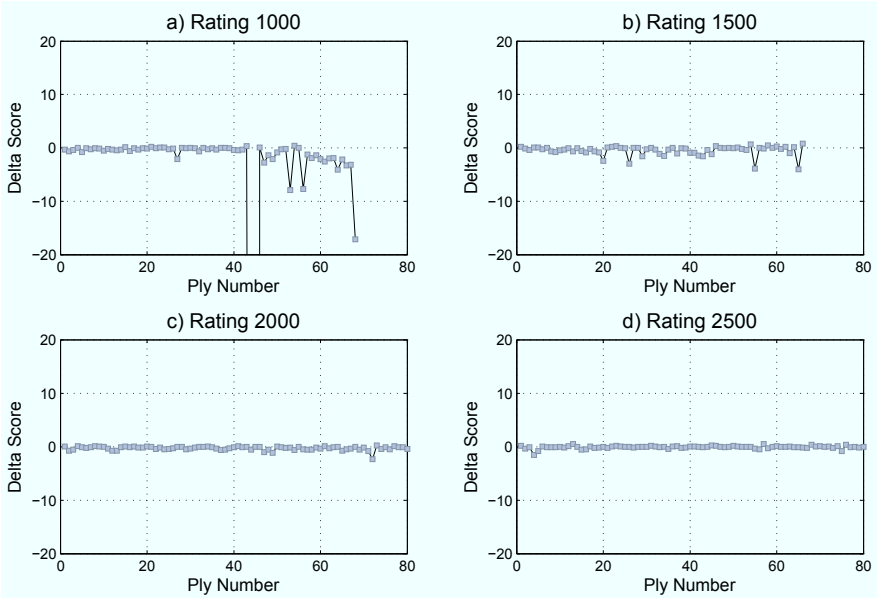

Fig. 4. Delta Score vs Ply number for 4 games chosen at random. Each subfigure shows games of players with rating approximately: a) 1000, b) 1500, c) 2000 and d) 2500.

These figures, though being just a single example, are representative of the behavior of the full database. Games between players with high rating tend to have very small deviations in the score. On the other hand, as ratings decrease, Δscores are larger and larger. This is evident as weak player use to make evident blunders[3]. Strong players also make blunders, but very sutil and these give a result far away from the current move. As our analyser, Crafty, is configured to search an 8 depth tree, if the blunder results in a bad move in more than 8 moves, it will be omitted, and score will not reflect it.

[3] A blunder is a very bad move; it is usually caused by some oversight, increased in the case of time pressure.

4 Conclusions

Gaming servers have widely spread throughtout the Internet in the past years. One of the games offered is Chess, which has been longly used as a path into the understanding of human cognition. One of the most famous servers fro playing chess is FICS with more than 300.000 registered users. This server saves every move played with their response-times. We decided to create a massive-scale human decision-making database based on the information provided by FICS.

Also, the evaluation of goodness of each move was performed. This analysis was done using Crafty, an open source chess engine. As this analysis is very compute-expensive, it was parallelized using a Beowulf cluster, using the basic MPI library routines.

We found that, as expected, the rating used to evaluate the player strength is very reliable, implying that two players with the same rating, has the same probability to win the game. On the other hand, only 150 of difference is enough to have 3 times more probility of winning than the opponent.

Game stages have been widely studied in chess, chracterized by different tactics and strategies. We performed an statistical study of millions of games that show that this stages may be differentiated looking at the response-times of the players.

Finally, the score show very different patterns depending on player's ratings. Weak player tend to make mistakes (blunders) and in consecuence score variability is significantly higher than for strong players. Even though storng players have an extremely smooth score pattern, they still make blunders, but this blunders are sutil and very difficult to detect.

Acknowledgements

We thank Robert Hyatt for developing Crafty as an open source project and for having promptly responded to many questions which have greatly contributed to the development of this work. We also thank the developers and administrators of FICS for continuously supporting a free chess server of the highest quality and completely open to experimentation. This work was supported by the Human Frontiers Science Program.

The computing power was partially provided by Centro de Computación de Alto Rendimiento (CeCAR), `http://pme84.exp.dc.uba.ar/`.

References

1. Burns, B.: The effects of speed on skilled chess performance. Psychological Science 15(7), 442–447 (2004)
2. Charness, N.: Visual short-term memory and aging in chess players. The Journal of Gerontology 36(5), 615 (1981)
3. Gobet, F., Simon, H.: Recall of random and distorted chess positions: implications for the theory of expertise. Memory & cognition 24(4), 493 (1996)

4. Gobet, F., Simon, H.: Templates in Chess Memory: A Mechanism for Recalling Several Boards. Cognitive Psychology 31(1), 1–40 (1996)
5. Groot, A.: Thought and choice in chess. The Hague, Mouton (1965)
6. Holding, D., Reynolds, R.: Recall or evaluation of chess positions as determinants of chess skill. Memory & Cognition 10(3), 237–242 (1982)
7. Pacheco, P.: Parallel programming with MPI. Morgan Kaufmann (1997)
8. Reingold, E., Charness, N., Pomplun, M., Stampe, D.: Visual span in expert chess players: Evidence from eye movements. Psychological Science pp. 48–55 (2001)
9. Reingold, E., Charness, N., Schultetus, R., Stampe, D.: Perceptual automaticity in expert chess players: parallel encoding of chess relations. Psychonomic bulletin & review 8(3), 504 (2001)
10. Schaeffer, J., Burch, N., Bjornsson, Y., Kishimoto, A., Muller, M., Lake, R., Lu, P., Sutphen, S.: Checkers is solved. Science 317(5844), 1518 (2007)
11. Schultetus, R., Charness, N.: Recall or evaluation of chess positions revisited: The relationship between memory and evaluation in chess skill. The American journal of psychology 112(4), 555–569 (1999)
12. Shannon, C.: Programming a computer for playing chess. Philosophical magazine 41(7), 256–275 (1950)
13. Sterling, T.: Beowulf cluster computing with Linux. The MIT Press (2001)
14. Von Ahn, L.: Games with a purpose. Computer 39(6), 92–94 (2006)
15. Von Ahn, L., Dabbish, L.: Labeling images with a computer game. In: Proceedings of the SIGCHI conference on Human factors in computing systems. pp. 319–326. ACM (2004)
16. Von Ahn, L., Liu, R., Blum, M.: Peekaboom: a game for locating objects in images. In: Proceedings of the SIGCHI conference on Human Factors in computing systems. p. 64. ACM (2006)
17. Von Ahn, L., Maurer, B., McMillen, C., Abraham, D., Blum, M.: recaptcha: Human-based character recognition via web security measures. Science 321(5895), 1465 (2008)