Advances in Computer Games: 13th Intl. Conf. ACG 2011; Tilburg, Netherlands, Nov 2011, H.J. van den Herik and A. Plaat (eds.), Springer-Verlag Berlin LNCS 7168, 2012, pp 220-229

Blunder Cost in Go and Hex

Henry Brausen, Ryan B. Hayward, Martin Müller, Abdul Qadir, David Spies

Computing Science, University of Alberta
{hbrausen,hayward,mmueller}@ualberta.ca

Abstract. In Go and Hex, we examine the effect of a blunder — here, a random move — at various stages of a game. For each fixed move number, we run a self-play tournament to determine the expected blunder cost at that point.

1 Introduction

To *blunder* — literally, to stumble blindly [9] — is to move stupidly, with apparent ignorance of one's situation. In the context of games, a blunder is usually defined as a serious mistake, for example a losing move when a winning move is available. But a mindless move need not be catastrophic. In the endgame, a blunder usually leads to a quick loss, but in opening play, a blunder might not even be noticed. So, what is the average cost of a *blunder* — namely, a random move — at various stages of a game?

Games that are interesting to play are typically hard to solve, so usually this question can be answered only experimentally rather than exactly. In this paper we present experimental blunder cost data for the Go player Fuego and the Hex player MoHex.

For a particular move number, blunder cost is a function of both player strength (the stronger the player, the higher the cost) and position difficulty (the greater the number of losing moves, the higher the cost). Thus *blunder analysis*, in which a player is compared with a blundering version of itself, can be useful as a diagnostic tool both for players and for games.

2 A simple model of two-player no-draw games

To explain how blunder analysis can be useful, we first outline a simple model of two-player no-draw games. (For other approaches of modeling more general games, see for example the work of Haworth and Regan [10, 20].)

Our model's parameters are as follows:

- T: maximum number of moves in a game,
- t: moves made so far, in {0,...,T},

- e_t: computational ease of solving a state after t moves, from 0 (intractable) to 1 (trivial),
- w_t: fraction of available moves that are winning, from -1 (all losing) to 1 (all winning), with $w_t = -x < 0$ indicating that all player moves are losing, and furthermore that after the best move the opponent will have w_{t+1} close to x,
- m_t: score of move t, from -1 (weakest) to 1 (strongest),
- r_t: rank of move t, from 1 (weakest) to k_t (strongest), where k_t is the number of available moves,
- s_p: strength of player p, from -1 (anti-perfect, always likely to play the weakest move) to 0 (uniform random) to 1 (perfect, always likely to play the strongest move).

We assume that games are non-pathological, in the sense defined by Dana Nau [19], so that the smaller the search space, the greater the tendency to make a strong move. Thus e_t, which approximates the size of the search space, increases smoothly with t.

With these parameters, a game can be modelled as follows:

- for each move number t, compute m_t by sampling from a distribution with mean $s_p \times e_t$,
- then compute r_t from m_t,
- then make the move with rank r_t,
- then compute w_{t+1} from w_t and r_t, say by sampling from a distribution, with the sampling formula ensuring that the strongest move from a winning position always leaves the opponent with no winning moves.

So, in this model, what is the expected cost of a blunder? The probability of making a winning move depends on w_t, the fraction of winning moves available, and s_p, the player's strength. We assume that both players try to win and can perform some useful computation, and so $s_p > 0$ for both players. A blunder is a uniform random move: in our model, this corresponds to a move whose strength is uniformly sampled from the interval [-1, 1]. Since non-blunder moves are sampled from a distribution D with expected value $s_p \times e_t$, a blunder is similar to (and, if D is uniform, identical to) a move made by a player whose strength temporarily drops to 0. Thus the expected cost of a blunder is the drop in win rate caused by this temporary strength loss. So this form of blunder analysis gives an indirect measurement of $s_p \times e_t$, namely playing strength times ease of solving the game at move t.

In this paper, we consider what happens when the "blunder player" makes exactly one blunder per game. We also considered experiments where the blunder player makes two consecutive blunders, but this resulted in the blunder player's win rate being extremely low and difficult to measure, and so we do not include any of this data.

3 Blunder Analysis of Fuego

Fuego is the open source Monte Carlo tree search Go program originally developed by the University of Alberta Computer Go group, led by Martin Müller and including Markus Enzenberger, Broderick Arneson [17, 8, 16]. In 2009 Fuego became the first computer Go program to win a 9×9 game without handicap against a top professional player, 9-dan Chou Chun-Hsun. Fuego has won a number of computer Go tournaments, including the 2010 UEC Cup [15, 7].

In our experiments, the default player never deliberately blunders. Baseline data, which shows the respective black (first player) and white (second player) win rates, is generated from a tournament between two default players. Each move-k win rate is an average of win rates from a trial between a blunder-player, who blunders only at that move, and a default player. The blunder-player data values are drawn as two curves, one for each possible blunder-player color. A third curve shows the fraction of games still active, namely unfinished, at that point. Each data value and each baseline value is computed from a 500 game trial.

Error bars show a binomial proportion confidence interval of $2\sqrt{(p(1-p)/n)}$, where p is the proportion and n is the number of trials, yielding a confidence of slightly more than 95%. Each move-k datum is computed only on games still active at move k, so error bars enlarge as the number of active games decreases.

In these Go experiments, white receives a komi of 7.5 points. Unless otherwise noted, Fuego runs 10000 MCTS simulations per move, and the resign threshold is 0.05.

3.1 9×9 Go

Figure 1 shows the effect of Fuego blunders on the 9×9 board. The baselines show black/white win rates of about 47%/53%, suggesting that on the 9×9 board the komi of 7.5 is in white's favour in games between programs. The move 1 (black) blunder-player has a 33% win rate, so the blunder cost of this move is $47 - 33 = 14\%$. This win rate indicates that, for roughly 1/3 of the 49 possible 9×9 opening moves, Fuego wins in self-play from that opening. At move 56 the active game rate is below 0.5, so for this move more than half of the 500 trial games finished before the blunder-player had a chance to make its move-56 blunder; as mentioned earlier, each data value is computed only on the games still active at that the time of the scheduled blunder, so the error bar here is larger than at moves made when all 500 trial games were active.

In the early game, the 9×9 blunder-player win rate is relatively high, suggesting that Fuego's play here is relatively weak and/or that the number of available winning moves is relatively high. By move 11 the blunder cost is about $47 - 18 = 29\%$, about double the move-1 blunder cost.

3.2 Go on other board sizes

Compare the 9×9 data with the 7×7 data in Figure 2 and the 13×13 data in Figure 3. For the 7×7 data, the number of simulations per move is the same

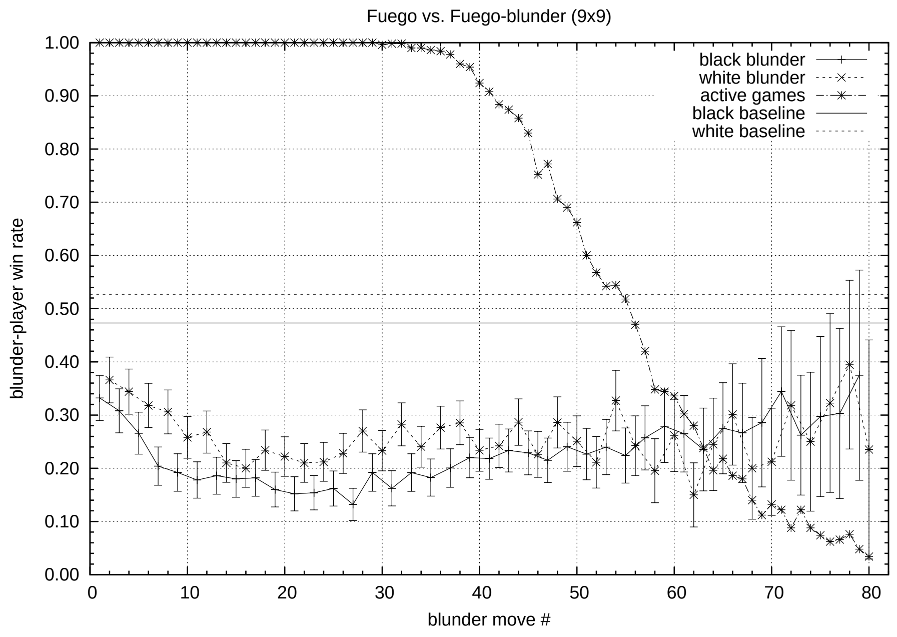

Fig. 1. Fuego-blunder performance, 1000 sim./move, 9×9 board

but the number of available moves per position is smaller (than for 9×9). Thus we expect Fuego to be stronger here, and this seems to be the case, as blunder costs are higher than for 9×9 Go. Fuego may play 7×7 Go close to perfectly, in which case the move-k blunder-player win rate is close to the move-k available-winning-move rate.

By contrast, for the 13×13 data, with a larger number of moves per position, we expect Fuego to be weaker. Again, this appears to be the case, as blunder costs are lower. The move-1 blunder cost is under 4%, and it is only by about move 50 that blunder cost is about 25%. For some reason, even though black and white baseline win rates are within 1% of each other, black blunders are typically more costly than white blunders. This is a topic for further study.

4 Blunder Analysis of MoHex

Hex is the classic connection game created by Piet Hein in 1942 [13] and John Nash around 1948 [18]. The players alternate turns, trying to connect their two sides. Figure 4 shows a game won by black.

Hex is simpler than Go in some aspects: stones never move once played, checking the winning condition is easy, and the game cannot end in a draw. On $n{\times}n$ boards the first player has a winning strategy, but no explicit such strategy is known for any $n > 9$, and solving arbitrary positions is Pspace-complete. (For

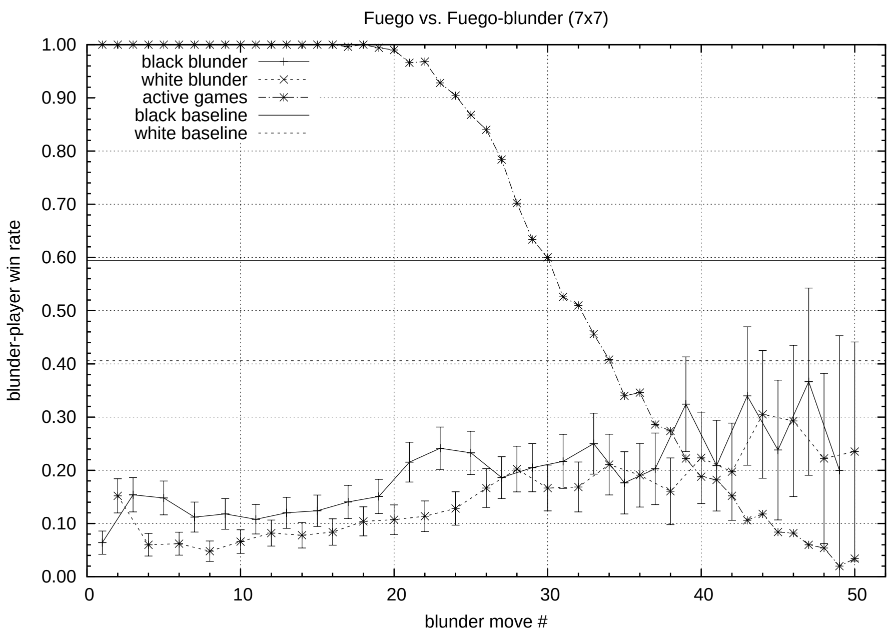

Fig. 2. Fuego-blunder performance, 1000 sim./move, 7×7 board.

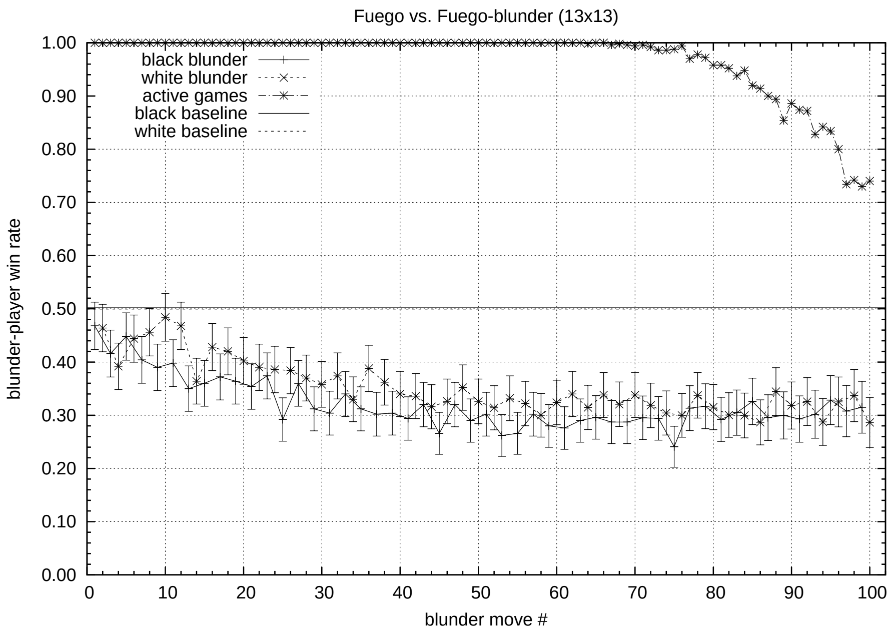

Fig. 3. Fuego-blunder performance, 1000 sim./move, 13×13 board.

more on Hex, including more on these results, see [6] or [11].) For this reason, Hex is often used as a test bed for algorithmic development.

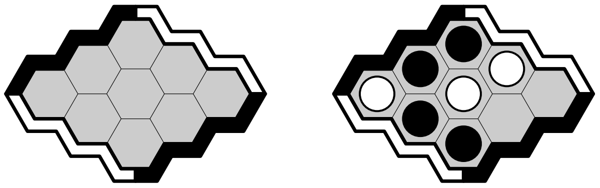

Fig. 4. An empty 3×3 Hex board and a game won by Black.

MoHex is the open source MCTS Hex program originally developed by the University of Alberta Computer Hex group, led by Ryan Hayward and including Broderick Arneson and Philip Henderson [5, 3]. MoHex is built on top of the Fuego Monte Carlo tree search framework. Its main game-specific ingredients are a virtual connection[1] engine and an inferior cell[2] engine [3]. MoHex won the gold medal for (11×11) Hex at the 2009 and 2010 ICGA Computer Games Olympiads [1, 2].

The parameters for Hex experiments were in general the same as for the Go experiments. Unless otherwise noted, MoHex ran 1000 MCTS simulations per move.

4.1 11×11 Hex

Figure 5 shows the effect of MoHex blunders on the 11×11 board. The baselines show first player (black) and second player (white) win rates of 62.8% and 37.2% respectively.[3] For some reason the white move-$k+1$ blunder cost is for small k slightly more than the black move-k blunder cost, but becomes less as k increases. This is a topic for further study.

4.2 7×7 Hex and available-winning-move rate

Although $n \times n$ Hex is a first player win, the baseline black win rate here is only 79%, rather than the 100% achievable by a perfect player. So, with 1000 simulations per move, MoHex is far from perfect, even on this small board.

[1] A virtual connection is a point-to-point 2nd-player connection strategy, i.e. the player can force the connection even if the opponent moves first.

[2] An inferior cell is one that can be pruned in the search for a winning move.

[3] This is perhaps partly due to MoHex arbitrarily assigning so-called dead cells (cells which are not in a minimal winning path-completion for either player) always to the black player in the inferior cell engine.

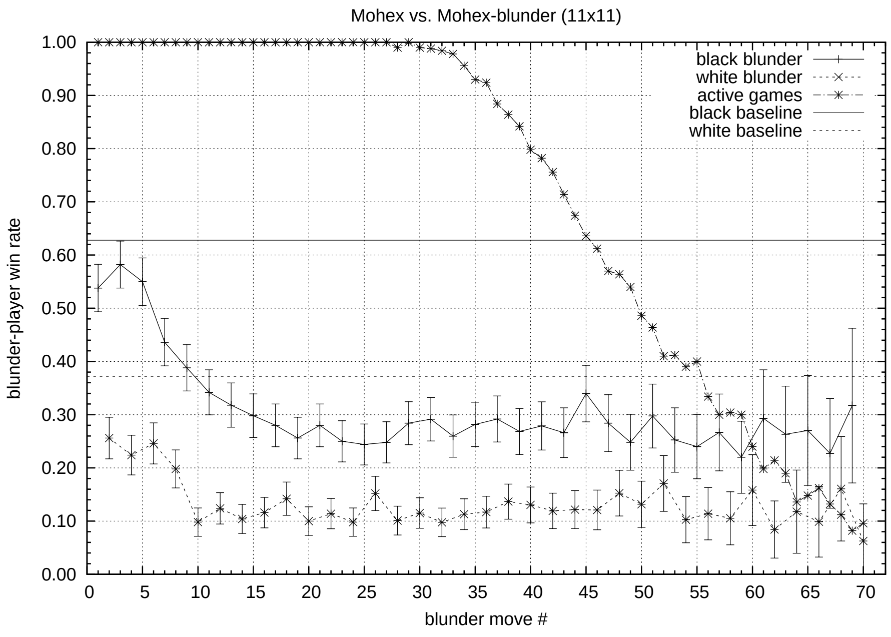

Fig. 5. MoHex-blunder performance, 1000 sim./move, 11×11 board.

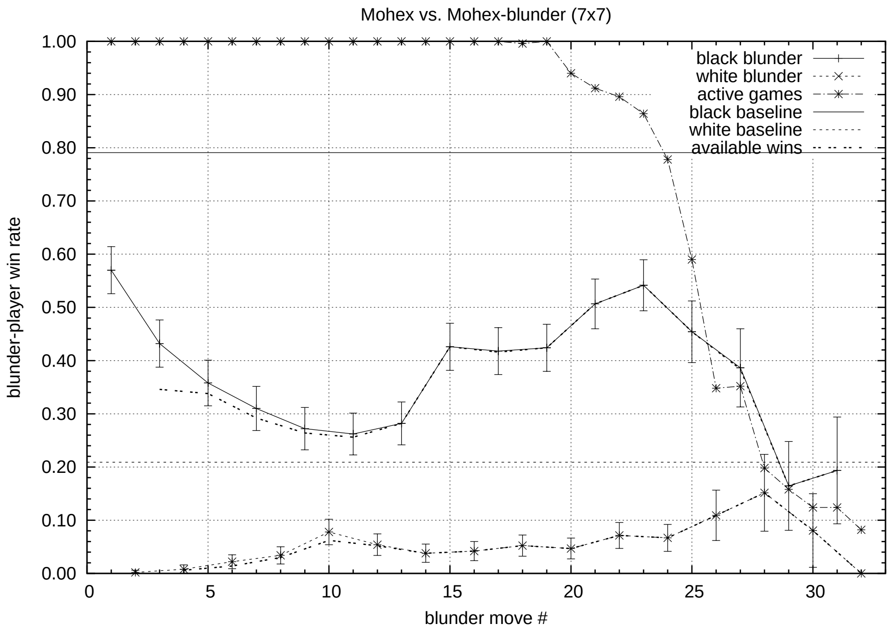

Fig. 6. Blunder-MoHex performance, 1000 sim./move, 7×7 board, also showing available-winning-move rate.

However, the move-1 blunder win rate is about 57±4%. This is within error of what is expected from a perfect player, since exactly 27/49≃0.55 of the possible 7×7 opening moves are winning [12], so any errors MoHex is making in these games are not having much effect.

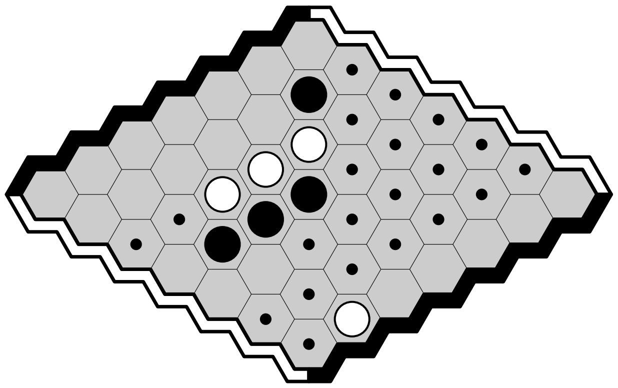

Fig. 7. A 7×7 Hex position with all winning black moves.

State-of-the-art Hex solvers can easily solve arbitrary 7×7 positions [14, 4]. So, in our 7×7 Hex experiment, for each position where the blunder player was about to move, we ran an exact solver to find the number of available winning moves at that point. For example, Figure 7 shows an 8-stone 7×7 position and all 21 winning moves. If black blunders in this position, black has a 21/41 chance of selecting a winning move.

In addition to the usual blunder data, Figure 6 shows the available-winning-move rates as two curves, one for each color. These curves are difficult to see, as from move 12 they coincide almost exactly with the respective blunder rates, suggesting that MoHex is playing most of these positions perfectly.

4.3 9×9 Hex and playing strength

Figures 8 and 9 show data on the same board size with two players of slightly different strength. We expect that in general the stronger player will have higher blunder cost, but in fact the opposite seems to occur here. This is a topic for further study.

5 Conclusions

We have shown that blunder analysis can offer insight into player performance and game difficulty. Our results with Fuego and MoHex suggest that when the

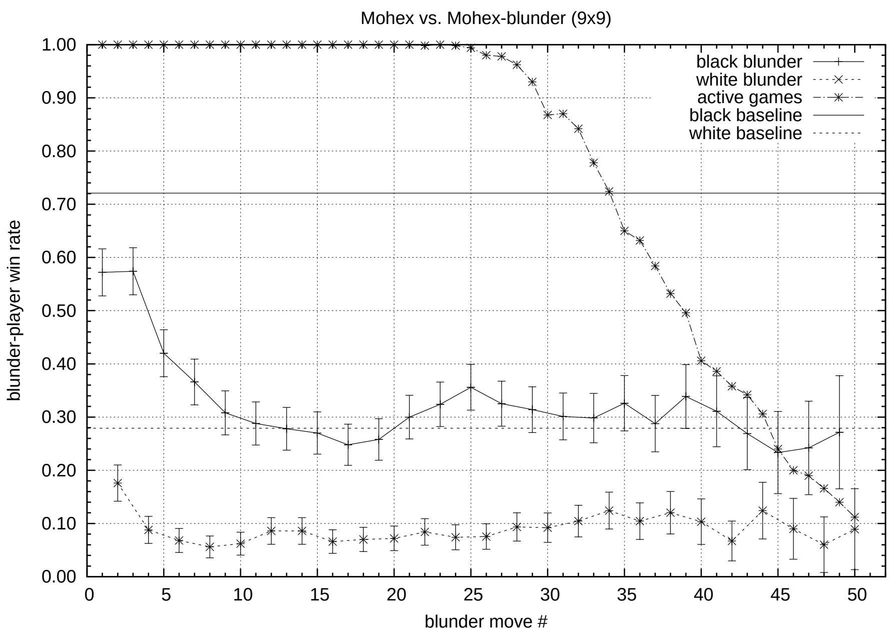

Fig. 8. MoHex-blunder performance, 1000 sim./move, 9×9 board.

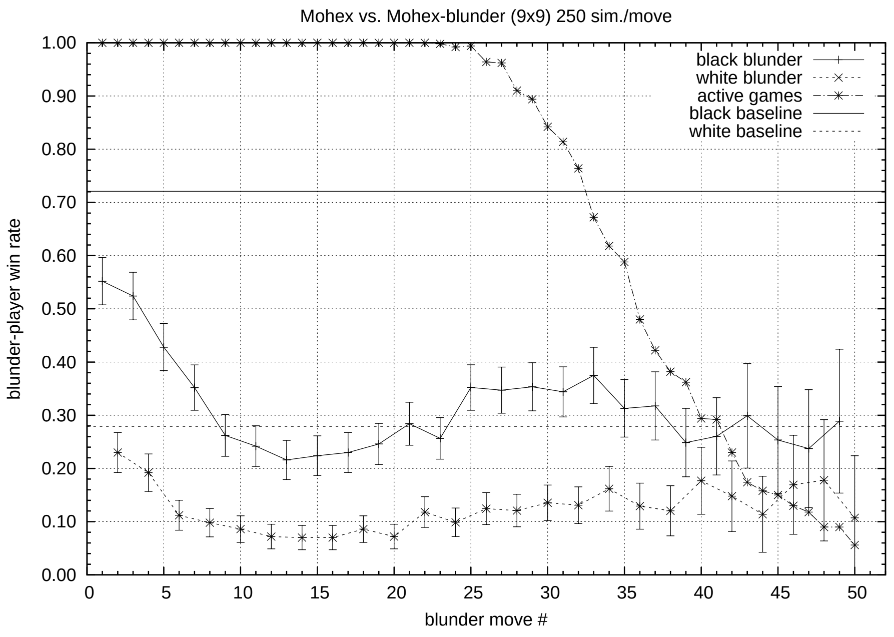

Fig. 9. MoHex-blunder performance, 250 sim./move, 9×9 board.

players are strong and the board size is relatively small, the blunder rate corresponds closely with the fraction of available winning moves. We conjecture that this holds for other games and players. We propose blunder analysis as a reasonable measure of perfect play in general.

One possible use of blunder analysis is as a tool for aiding in game time management, i.e. deciding how much processing time to spend on move selection at various points in a game. We leave this as a topic for further study.

Acknowledgements

We thank the referees for their helpful comments. We thank NSERC Discovery (Hayward, Müller), NSERC USRA (Brausen), Prof. Jonathan Schaeffer and Alberta iCORE (Spies), UofA India Internship (Qadir), and the UofA GAMES group for their support of this project.

References

1. Arneson, B., Hayward, R.B., Henderson, P.: MoHex Wins Hex Tournament. ICGA 32(2), 114–116 (June 2009)
2. Arneson, B., Hayward, R.B., Henderson, P.: MoHex Wins Hex Tournament. ICGA 33(2), 181–186 (Sept 2010)
3. Arneson, B., Hayward, R.B., Henderson, P.: Monte Carlo Tree Search in Hex. IEEE Transactions on Computational Intelligence and AI in Games. Special issue on Monte Carlo Techniques and Computer Go 2(4), 251–257 (Dec 2010), www.cs.ualberta.ca/˜hayward/publications.html
4. Arneson, B., Hayward, R.B., Henderson, P.: Solving Hex: Beyond Humans. In: van den Herik, H.J., Iida, H., Plaat, A. (eds.) Computers and Games, 7th International Conference, CG 2010. Lecture Notes in Computer Science, vol. 6515, pp. 1–10. Springer (2011), www.cs.ualberta.ca/˜hayward/publications.html
5. Arneson, B., Henderson, P., Hayward, R.B.: Benzene (2009–2011), http://benzene.sourceforge.net/
6. Browne, C.: Connection Games: Variations on a Theme. A.K. Peters, Wellesley, Massachusetts (2005)
7. University of Electro-Communications, J.: The Fourth Computer Go UEC Cup (2010), http://jsb.cs.uec.ac.jp/ igo/past/2010/eng/
8. Enzenberger, M., Müller, M., Arneson, B., Segal, R.: Fuego – an Open-Source Framework for Board Games and Go Engine based on Monte Carlo Tree Search. IEEE Transactions on Computational Intelligence and AI in Games. Special issue on Monte Carlo Techniques and Computer Go 2(4), 259–270 (Dec 2010)
9. Harper, D.: Online Etymology Dictionary (2001-2011), http://www.etymonline.com/
10. Haworth, G.M.: Reference fallible endgame play. ICGA 26(2), 81–91 (2003)
11. Hayward, R., van Rijswijck, J.: Hex and Combinatorics. Discrete Mathematics 306, 2515–2528 (2006)
12. Hayward, R.B., Björnsson, Y., Johanson, M., Kan, M., Po, N., van Rijswijck, J.: Solving 7 × 7 Hex: Virtual Connections and Game-state Reduction. In: van den Herik, H.J., Iida, H., Heinz, E.A. (eds.) Advances in Computer Games, IFIP International Federation for Information Processing, vol. 263, pp. 261–278. Kluwer Academic Publishers, Boston (2003)

13. Hein, P.: Vil de laere Polygon? Politiken (December 26 1942)
14. Henderson, P., Arneson, B., Hayward, R.B.: Solving 8×8 Hex. In: Proc. IJCAI. pp. 505–510 (2009)
15. International Computer Games Association: ICGA Tournaments (2011), http://www.grappa.univ-lille3.fr/icga/
16. Kroeker, K.L.: A New Benchmark for Artificial Intelligence. Communications of the ACM 54(8), 13–15 (Aug 2011)
17. Müller, M., Enzenberger, M., Arneson, B.: Fuego (2008–2011), http://fuego.sourceforge.net/
18. Nash, J.: Some games and machines for playing them. Tech. Rep. D-1164, RAND (February 1952)
19. Nau, D.S.: An investigation of the causes of pathology in games. Artificial Intelligence 19(3), 257–278 (Nov 1982)
20. Regan, K.W., Haworth, G.M.: Intrinsic chess rating. In: AAAI-11: 25th AAAI Conf. on AI. pp. 834–839 (2011)