Higher Order Statistics in Play-out Analysis

Tapani Raiko[⋆]

[⋆]Adaptive Informatics Research Centre
Helsinki University of Technology
tapani.raiko@tkk.fi

Abstract

Playing out the game from the current state to the end many times randomly, provides statistics that can be used for selecting the best move. This play-out analysis has proved to work well in games such as Backgammon, Bridge, and (miniature) Go. This paper introduces a method that selects relevant patterns of moves to collect higher order statistics. Play-out analysis avoids the horizon effect of regular game-tree search. The proposed method is especially effective when the game can be decomposed into a number of subgames. Preliminary experiments on the board games of Hex and Y are reported.

1 Introduction

Historically, chess has been thought to be a good testbed for artificial intelligence. In the last 10 years, though, computers have risen above human level. This is largely due to fast computers being able to evaluate a huge number of possibilities. One can argue, whether these programs show much intelligence after all.

There are still games that are similar to chess in some sence but have proved to be difficult for computer players. These games include Go, Hex, Y, and Havannah. The games have some common factors: First, the number of available moves at a time is large, which prevents the use of brute-force search to explore all the possibilities as the search tree grows exponentially with respect to depth. Second, there are lots of forcing moves which boost the so called horizon effect. If the program sees a limited number of moves ahead (search depth), there is often a way to postpone the interesting events beyound that horizon.

This paper introduces a method that avoids these two pitfalls. The search is performed by playing out games from the current situation to the end several times randomly. This is known as play-out analysis or Monte Carlo playing. The number of explored possibilities is constant with respect to search depth. Also, as the games are played out all the way to the end, there is no need to evaluate a game in progress, thus avoiding the horizon effect.

1.1 Game of Hex

Hex is a two-player boardgame with no chance. It is played on an $n \times n$ rhombus of hexagons. The players have colours red and blue and the edges of the board are coloured red and blue such that parallel edges have the same colour. The game is started with an empty board and the players alternately fill a cell with a piece of ones own colour. Red player wins if the red edges are connected with a path of red cells and vice versa. When the board fills up, exactly one of the players has formed a connecting path, see Figure 5 for an example game.

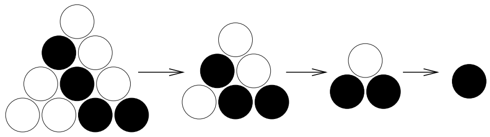

Figure 1: A small filled Y board is shown to be won by black by micro reductions.

1.2 Game of Y

The game of Y is a two-player boardgame with no chance and a relative to the more popular Hex. Y is played on a triangular board formed of hexagonal cells that are empty in the beginning. Two players alternately fill empty cells with pieces of their own colour. The player who creates an unbroken chain of pieces that connect all three edges, wins. Note that corners belong to both edges. The standard board in Figure 2 is slightly bent with three of the hexagons replaced by pentagons.

The fact that Y cannot end in a draw on a straight board can be proven using so called micro reductions described in (van Rijswijck, 2002). Size n board is reduced to a size $n-1$ board where each cell on the

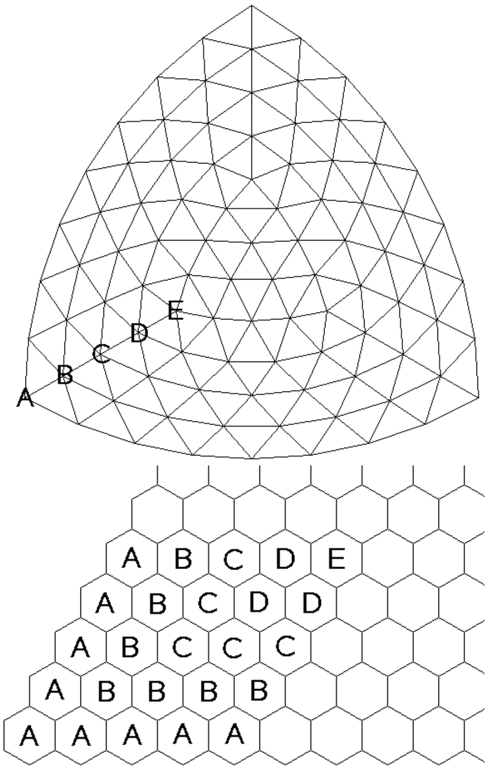

Figure 2: Top: The game of Y is usually played on the bent board, on points rather than cells. Bottom: The bent board can be transformed into a straight one for analysis such as micro reduction. Filling a point in the bent board fills all the cells with the same letter in the straight board. Only one corner is shown.

reduced board gets the majority colour of the three nearest cells on the original board. An example is given in Figure 1. It turns out that if a chain touches a side of the board, it does so also on the reduced board. A winning chain will thus be a winning chain on all the smaller boards including the trivial board of size 1. Figure 2 shows how the same analysis can be done for the bent board by first transforming it into a straight one.

1.3 Related Work

Abramson (1990) proposed the expected-outcome heuristic for game state evaluation. The value of a game state is the expected outcome of the game given random play from that on. The value can be used to evaluate leaf nodes in a game-tree that has the current state as the root node and possible moves as children of the node. The expected-outcome heuristic applies to a large number of games with or without randomness or hidden information and with one or more players. It is best applicable to games that always end after a reasonable number of moves, such as the games of Hex and Y. Expected outcome heuristic is also known as roll-out analysis or play-out analysis.

In many planning problems and games, the order of moves or actions is not always important. This applies especially in games of Hex and Y where the order does not have any role as long as the same moves are made. In Monte Carlo playing, unimportance of the order means that the all-moves-as-first heuristic (Brügmann, 1993) works well: After a game has been played out, not only the evaluation of the first move is changed, but all the subsequent moves are handled as they were the first move. Thus, in the games of Hex and Y, instead of playing the game out, one can just fill the empty cells by pieces of random colour. Sampling colours for the empty cells then corresponds to Monte Carlo playing. Those cells that were filled by the winning player, are important.

All-moves-as-first heuristic is very good in reducing the number of random samples compared to basic expected outcome, but unfortunately it limits the lookahead into just one move in some sense. Bouzy (2004) proposed a hybrid between expected-outcome and all-moves-as-first heuristics. A shallow game tree is made where each leaf is evaluated using the all-moves-as-first heuristic.

Müller (1999) proposed decomposition search with an application to Go endgames. The idea is that the game can be decomposed into a sum of subgames which are somewhat independent of each other. Because the subgames are small, the full game tree can be constructed for each of them, including local pass moves. Then, many of the moves can be pruned for instance because they are dominated by other moves in the same subgame. The global search is thus sped up a lot, allowing perfect solution of end-games of dozens of moves. A similar approach is suggested to be used heuristically for middle-game where the division into subgames is not yet strict.

Already Brügmann (1993) suggested an extension of the all-moves-as-first heuristic into an higher order heuristic, where not just the value of each move is evaluated but the value of making a move after some other N moves have been made. Time was not ripe for higher order considerations for computational resources. If there are M moves available, one would have to collect enough statistics for M^N combinations of moves. Now, thirteen years later, it would still be a waste of resources to try to uniformly estimate all combinations.

Inductive logic programming (ILP) (Muggleton and De Raedt, 1994) involves finding patterns of increasing size in data. A pattern in this case is a set of moves (in addition to the ones that have been already made) and the data are the random games. One can estimate the value of a move given that the pattern has already been played. New patterns are created by refinement operators, in this case by adding a move to an existing pattern. By using ILP, we can make the higher-order heuristic selective, that is, we do not have to estimate all M^N combinations of moves but only MP where P is the number of patterns.

2 Higher order statistics in play-out analysis

Black and white player are playing a game with alternate moves. At some state s of the game, a play-out analysis means that k hypothetical continuations c of the game are played out to the end and statistics from these play outs are used to determine the next move. The play outs might be sampled by drawing moves from a uniform distribution over all legal moves, or by other means. To sample a semi-random move, one can add noise from a random number generator to a heuristic evaluation of each move and select the best one.

The expected-outcome heuristic h_e evaluates the move m_0 as

$$h_e(m_0) = \frac{\sum_{i=1}^{k} \chi(c_i \text{ is a win})\chi(m_0 \text{ first move of } c_i)}{\sum_{i=1}^{k} \chi(m_0 \text{ first move of } c_i)}, \tag{1}$$

where χ is the indicator function ($\chi(\text{true}) = 1$ and $\chi(\text{false}) = 0$). The best move is thus the one that lead to most wins in the play outs c.

Note that in Hex and Y the order of moves does not matter. One can play the game from current state all the way to filling the board. The winner of the game can be determined from the filled board without knowing the order of moves. One can think that the single play out represents all the possible play outs that lead to the same position. This leads to the all-moves-as-first heuristic h_a:

$$h_a(m_0) = \frac{\sum_{i=1}^{k} \chi(c_i \text{ is a win})\chi(m_0 \text{ is made in } c_i)}{\sum_{i=1}^{k} \chi(m_0 \text{ is made in } c_i)}. \tag{2}$$

The best move is the one that lead to wins when used at any point in the continuation. An example is shown in Figure 3. Often, play out is done using a semi-random move selection with the heuristic itself.

Higher order heuristic h_h evaluates the move m_0 after moves $m_1, m_2, \ldots, m_l$ are made in any order.

$$h_h(m_0 \mid \{m_j\}_{j=1}^{l}) = \tag{3}$$
$$\frac{\sum_{i=1}^{k} \chi(c_i \text{ is a win})\chi(m_0 \text{ after } \{m_j\}_{j=1}^{l} \text{ in } c_i)}{\sum_{i=1}^{k} \chi(m_0 \text{ after } \{m_j\}_{j=1}^{l} \text{ in } c_i)}.$$

The sets of moves $\{m_j\}_{j=1}^{l}$ are called patterns. Many different patterns apply in a future state when sampling a play out, and one must decide how to combine evaluations corresponding to different patterns. Also, one has to be selective which patterns are taken into consideration. Before going into these details, an example is shown in Figure 4.

2.1 Combining the evaluations by different patterns

Many patterns can apply to the same state, that is, the moves of more than one pattern have been made. It would be possible to construct complicated rules for combining their evaluations; e.g. the average of the evaluations of applying patterns weighted with a function of the pattern size. One should note that play out is the most time intensive part of the algorithm and combining happens at every play-out move so it has to be fast. The proposed combining rule is:

- The maximum of evaluations given by each pattern that applies but is not a subpattern of another applying pattern.

The motivation is that the more specific pattern gives more accurate information about a move than its subpattern. In Figure 4, patterns 1, 2, and 4 apply to the example state, but pattern 1 is a subpattern of the other two so it is not taken into account. Using the maximum helps in computational complexity. One can take the maximum first within a pattern and then over different patterns. The evaluations of a pattern can be stored in a heap to make the time complexity logarithmic with respect to the number of available moves.

2.2 Selection of patterns

An algorithm based on inductive logic programming is used to select patterns. The process starts with just the empty pattern (for which $l = 0$). Repeatedly, after a certain number of play outs, new patterns are added. An existing pattern generates candidates where each possible m_0 is added to the pattern, one at a time. The candidate is accepted if it has appeared often enough, and if m_0 is a relevant addition to the pattern. If the original pattern makes move m_0 more valuable than without the pattern, the moves seem to be related. The used criterion was based the mutual information between the made move and winning, given the pattern. If the move makes winning less likely, the criterion was set to zero. Then the maximum of mutual informations of the move and winning, given any of the subpatterns, is substracted from the score. When new patterns are added, statistics are copied from its parent with a small weight. Also, the statistics of the first play-outs are slowly forgotten by exponential decay.

2.3 Exploration and exploitation

One has to balance between exploring lots of truly different play outs and exploiting the known good moves and studying them more closely. Here it is done by using simulated annealing: In the beginning, the amount of noise is large, but towards the actual selection of the move, the amount of noise is decreased to zero linearly.

Another possibility would be to add a constant (say 1) to the numerator and the denominator of Equation 3 for emphasising exploration of unseen moves. This corresponds to an optimistic prior or pseudo-count. When the time comes to select the actual move, one wants to de-emphasise unseen moves, which is done by adding the constant only to the denominator.

2.4 Final move selection

To select the best move after play-out analysis is done, it is possible to simply pick the move m_0 with the highest heuristic value:

$$\arg\max_{m_0} h_h(m_0 \mid \{\}). \tag{4}$$

This is the only option with first-order heuristics, but higher order heuristics allow for more. One can make a min-max tree search in the tree formed by known patterns. For instance, using the second-order heuristics, one selects the move m_1, for which the best answer m_0 by the opponent is the worst:

$$\arg\min_{m_1} \max_{m_0} h_h(m_0 \mid \{m_1\}). \tag{5}$$

2.5 Summary of the algorithm

Given a state $s(0)$ select a move $m(0)$ by

1: Patterns $P = \{\{\}\}$, # of play out $k = 0$
2: Play-out depth $j = 0$
3: Move $m(j) = \arg\max_m h_h(m \mid p) + \text{noise}$
4: Make move $m(j)$ in state $s(j)$ to get new $s(j+1)$
5: Increase j by one
6: If $s(j)$ not finished, loop to 3
7: Increase k by one
8: Save moves $m(\cdot)$ and the result of $s(j)$ in c_k
9: Loop to 2 for some time
10: Add new patterns to P
11: Loop to 2 for some time
12: $m(0) = \arg\min_{m_1} \max_{m_0} h_h(m_0 \mid \{m_1\})$

On line 3, the moves of the pattern $p \in P$ and $p \subset \{m(0), \ldots, m(i-1)\}$ and there must be no $q \in P$ such that $p \subset q \subset \{m(0), \ldots, m(i-1)\}$. On line 10, the candidate patterns are all patterns in P extended by one move, and they are accepted if the particular move has been selected by that pattern more than some threshold number of times on line 3.

3 Experiments

The proposed method was applied to the games of Hex and Y. An implementation is available at http://www.cis.hut.fi/praiko/hex/. Figure 5 shows a game of Hex played by the proposed method against itself. The red player won as can be seen from the path starting from moves 43, 33, 21. The level of play is not very high, but taking into consideration that the system is not game-specific and that the implementation is preliminary, the level is acceptable.

Figure 6 shows the position before move 31. Because of symmetry of the criterion for selecting new patterns, they are generated in pairs, for instance, the patterns red at a and blue at a were the first two. Patterns in the order of appearance $\{\}$, $\{a\}$, $\{c\}$, $\{d\}$, $\{h\}$, {blue c, f}, {red a, b}, {red c, d}, {red d, c}, {blue a, g}, and so forth. The patterns are clearly concentrated on areas of interest and they are local.

A tournament between 8 different computer players was held in the game of Y such that each player met every opponent 10 times as black and 10 times as white, 640 games in total. The first player A made just random moves chosen uniformly from all the available moves. Three players B, C, and D used the all-moves-as-first heuristic with 100, 1000, and 10000 fully random play-outs per move, accordingly. Players E and F used second-order heuristics, that is, considering all patterns where a single move is made by the player in turn. The final two players G and H used selective patterns, whose number varied from 0 to 99 and size from 0 to 7. The players E–H used

player	order	# playouts	time/ms	win %
A	-	0	5	0
B	1	100	8	16
C	1	1000	36	66
D	1	10000	276	65
E	2	1000	125	65
F	2	10000	1155	61
G	N	1000	240	64
H	N	10000	14735	63

Table 1: The number of playouts, the order of used statistics, the average thinking time in milliseconds per move, and the winning percentage against other players is given for each player.

	A	B	C	D	E	F	G	H
A	7	0	0	0	0	0	0	0
B	10	6	0	1	1	0	0	0
C	10	10	7	3	7	6	6	6
D	10	10	5	2	5	5	5	8
E	10	10	7	7	7	6	3	6
F	10	10	6	6	4	4	6	3
G	10	10	4	5	5	6	8	6
H	10	9	4	5	6	7	6	6

Table 2: Wins out of ten for the black player (label on the left) against the white player (label on the top).

the second-order move selection in Equation (5). The number of play-outs and the average time per move is shown in table 1.

The results are shown in Table 2. Black won 53% of the matches since the first move gives an advantage. Player A lost all games againt other players. Player B was also clearly worse than the others. The differences between the players C–H are not clear from the limited amount of data.

4 Discussion

The experiments did not show improvement over first-order heuristics. This might be due to a larger need for samples, or perhaps the criteria for selecting new patterns were unoptimal. Baum and Smith (1997) propose a well-founded measure of relevance of expanding a leaf in a search tree when evaluations are probabilistic. This measure could be applied here as well. There is a lot of room for other improvements in general and in reducing computational complexity as well as taking into use application-specific heuristics.

The applicability of the proposed approach is limited to games where the order of moves is not very important as long as all of them are made. In Hex and Y this applies exactly, whereas in games of Havannah and Go, it applies approximatively. Perhaps patterns with more complicated structure could be introduced, such as any boolean formula over moves and order relations among moves.

The random filling heuristic in the game of Hex corresponds to communication reliability in graph theory. Two-terminal network reliablity (Brecht and Colbourn, 1988), or probabilistic connectedness, is the probability that two nodes in a network can communicate. Edges are assumed to have statistically independent probabilities of failing but nodes are assumed reliable. The studies in graph theory have brought results such as that the exact calculation of two-terminal reliability requires exponential computations.

The random filling heuristic for the game of Y is used by van Rijswijck (2002). He uses implicit random colouring of empty cells but instead of sampling from the distribution, he uses micro reduction repeatedly until the single value on the size-1 board can be directly used as a heuristic. At each step of the reduction, all the probabilities are assumed to be independent, which makes the heuristic quick and dirty. The transformation in Figure 2 allows the usage of this heuristic also on the bent board. It would be interesting to test this against the proposed approach.

The proposed method generalises trivially to all 1-or-more-player games and to other rewards than win/loss. The 1 player version has a connection to nonlinear planning (see book by McAllester and Rosenblitt, 1991). Planning aims at finding a sequence of actions that lead to some goal. Sometimes plan involves non-interacting steps that can be reordered. This is taken into account in a nonlinear plan which only contains a partial order among actions. The set of patterns and statistics can be interpreted as a nonlinear plan.

The idea of estimating statistics for growing patterns is also used in natural language processing by (Siivola and Pellom, 2005). The model probabilistically predicts the next word given a context or pattern of n previous words. New patterns are generated by adding a word to an existing pattern. For the final model, some of the patterns are pruned.

Currently all the patterns and statistics are forgotten after each move, while they could still be used. Also, the proposed system could be used with machine learning. Patterns reappear from game to game so life-long learning should be possible. Games such as Hex, Y, Go, and Havannah all have also limited translational invariance which could be used for generalisation. Machine learning in this setting is left as future work.

5 Conclusion

One can collect statistics from playing out the game randomly to the end many times. This paper proposed a method for a selective collection of higher order statistics, that is, evaluations of some combinations of moves. If the game can be divided into a number of subgames, the proposed system seems to be able to find relevant combinations concentrated on a single subgame at a time. The preliminary experiments did not yet show significant improvement over the first-order approach, but a door has been opened for further improvement.

The paper also gave some analysis on the game of Y. The proof of impossiblity of draws was extended to cover the bent board. A quantitative difference between the straight and the bent board with respect to the importance of the centre was shown. Also, playout analysis was applied to the games of Hex and Y for the first time.

Acknowledgements

The author wishes to thank Jaakko Peltonen and Teemu Hirsimäki for useful discussions. This work was supported in part by the Finnish Centre of Excellence Programme (2000-2005) under the project New Information Processing Principles and by the IST Programme of the European Community under the PASCAL Network of Excellence, IST-2002-506778. This publication only reflects the authors' views.

References

Bruce Abramson. Expected-outcome: A general model of static evaluation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(2): 182–193, February 1990.

Eric B. Baum and Warren D. Smith. A bayesian approach to relevance in game playing. *Artificial Intelligence*, 97(1–2):195–242, 1997.

Bruno Bouzy. Associating shallow and selective global tree search with monte carlo for 9x9 go. In *Proceedings of the 4th Computer and Games Conference (CG04)*, pages 67–80, Ramat-Gan, Israel, 2004.

Timothy B. Brecht and Charles J. Colbourn. Lower bounds on two-terminal network reliability. *DAMATH: Discrete Applied Mathematics and Combinatorial Operations Research and Computer Science*, 21:185–198, 1988.

Bernd Brügmann. Monte Carlo Go, 1993. Available at `http://www.cgl.ucsf.edu/go/Programs/`.

David McAllester and David Rosenblitt. Systematic nonlinear planning. In *Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI-91)*, volume 2, pages 634–639, Anaheim, California, USA, 1991. AAAI Press/MIT Press. ISBN 0-262-51059-6.

Stephen Muggleton and Luc De Raedt. Inductive logic programming: Theory and methods. *Journal of Logic Programming*, 19/20:629–679, 1994.

Martin Müller. Decomposition search: A combinatorial games approach to game tree search, with applications to solving Go endgames. In *Proceedings of the International Joined Conference on Articial Intelligence (IJCAI 1999)*, volume 1, pages 578–583, 1999.

Vesa Siivola and Bryan Pellom. Growing an n-gram language model. In *Proceedings of the 9th European Conference on Speech Communication and Technology (Interspeech'2005)*, Lisbon, Portugal, 2005.

Jack van Rijswijck. Search and evaluation in Hex. Technical report, Department of Computing Science, University of Alberta, 2002.

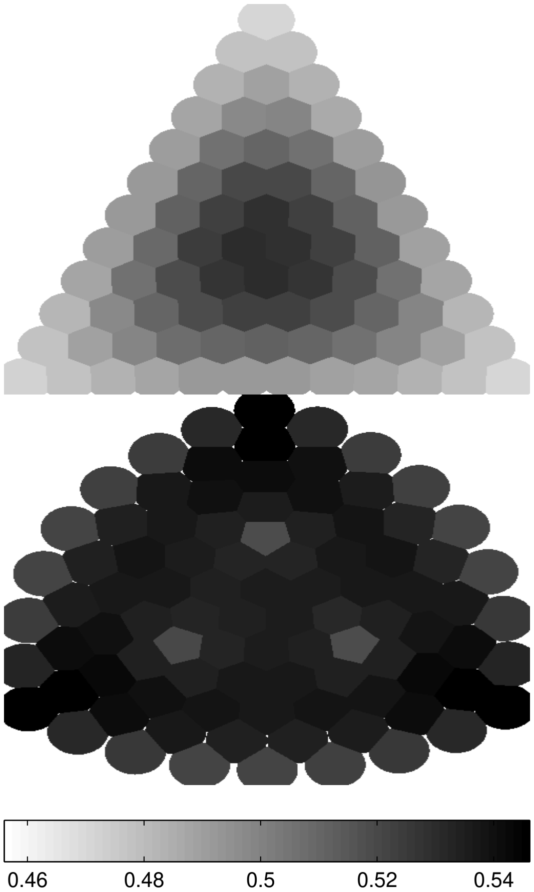

Figure 3: The expected-outcome, or equivalently in this case, the all-moves-as-first heuristic for the first move on a straight (top) and a bent (bottom) Y board. The colour shows the probability of a black win assuming random play after the first move. Note that the straight board gives a large emphasis on the center, which is the reason why the bent board is often used instead.

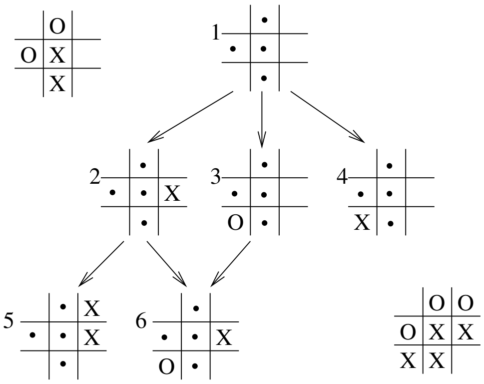

Figure 4: The current game state in tic-tac-toe is shown at the top left corner. Six patterns are numbered and shown in a graph. When playing out, a state at the bottom right corner is encountered. Patterns 1, 2, and 4 apply to it.

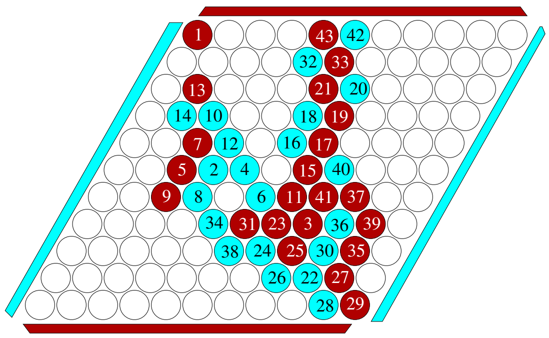

Figure 5: An example game of Hex by self-play of the proposed system.

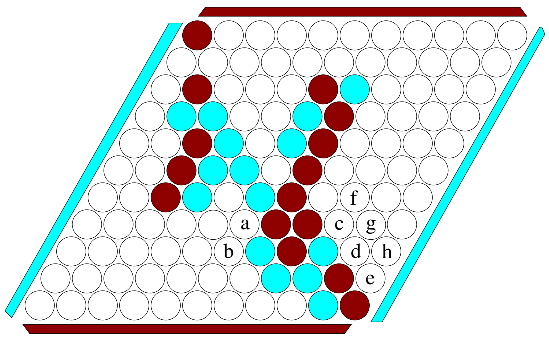

Figure 6: At move 31, the first patterns use the cells marked with $a, b, \ldots, h$. See text for explatations.