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 Go. This paper introduces a method that selects relevant patterns of moves to collect higher order statistics. This can be used to improve the quality of the play outs. Play-out analysis avoids the horizon effect of regular game-tree search. The proposed method should be especially effective when the game can be decomposed into a number of subgames. Game of Y is a two-player board game played on a graph with a task of connecting three edges of the graph together. Preliminary experiments on Y did not yet show significant improvement over the first-order approach, but a door has been opened for further improvement. The game of Y might prove to be a good testbed for machine learning.

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 [1] describes 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. 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. Recently, Gelly et al. (2006) made a noticable improvement in computer Go based on play-out analysis, being the best program on 9 by 9 and 13 by 13 miniature boards.

[1] A longer version of this paper is also available (Raiko, 2006).

1.1 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 graph in Figure 1. Starting from an empty board, two players alternately fill empty nodes with pieces of their own colour. The player who creates an unbroken chain of pieces that connect all three edges, wins. Corners belong to both edges. The board is slightly bent to balance the game by reducing the importance of the center.

The fact that Y cannot end in a draw on a straight board can be proven using so called micro reductions described by van Rijswijck (2002). Size n board is reduced to a size $n-1$ board where each node on the reduced board gets the majority colour of the three nearest nodes on the original board. An example is given in Figure 2. 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. The proof for the bent board is given by Raiko (2006).

1.2 Earlier Work

Abramson (1990) proposed the expected-outcome

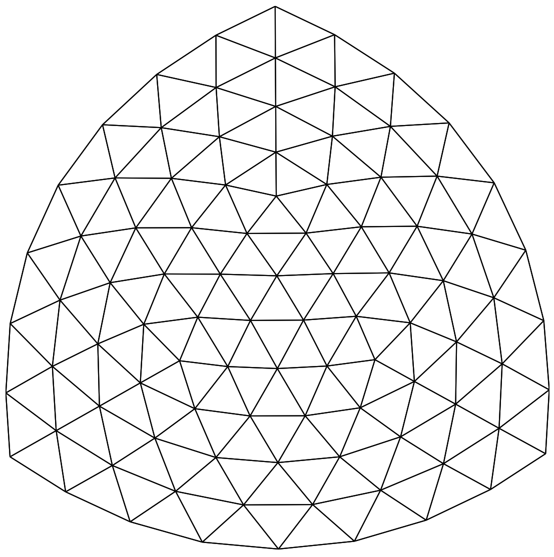

Figure 1: The game of Y is usually played on the bent board. Two players alternately fill empty nodes with pieces of their own colour. The player who creates an unbroken chain of pieces that connect all three edges, wins.

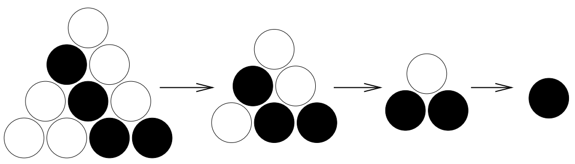

Figure 2: A small filled Y board is shown to be won by black by micro reductions.

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 play-out analysis, 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 nodes by pieces of random colour.

In Hex and Y, all-moves-as-first heuristic can be done by filling each node randomly. The random filling heuristic has a direct analogy 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.

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. Gelly et al. (2006) used a tree which grows by one node with every play-out. Leaves are evaluated by a single semi-random play-out.

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, fourteen 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 and the data are the random games. One can estimate the value of a move given a pattern. 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

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

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.

If the order of moves does not matter too much, one can adjust the opinion of all moves made in the play-out, not just the first one. This leads to the all-moves-as-first heuristic h_a:

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

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

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

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

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. Higher order heuristic is useful only if it is applied in the semi-random move selection of the playout.

The UCT approach used by Gelly et al. (2006) can be seen as a higher order heuristic that takes the order of moves into account. Also, their pattern applies only if the included moves are the only ones made. This specificity simplifies things, since only one pattern may apply at a time, but generalisation to other similar board states cannot be made.

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. 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 heuristic criterion for accepting candidates is given in (Raiko, 2006). 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.

Baum and Smith (1997) propose a well-founded measure of relevance of expanding a leaf in a search tree when evaluations are probabilistic. Gelly et al. (2006) uses the UCT (Upper bound Confidence for Tree) algorithm for choosing which leaf to expand. Either of these two measures could be applied here, which would be the most important direction for future work.

2.3 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:

$$m_0^* = \arg\max_{m_0} h_h(m_0 \mid \{\}). \tag{1}$$

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:

$$m_1^* = \arg\min_{m_1} \max_{m_0} h_h(m_0 \mid \{m_1\}). \tag{2}$$

A summary of the algorithm and more details are given in (Raiko, 2006).

3 Experiments

The proposed method was applied to the game Y. An implementation is available at [2]. A tournament between 8 different computer players was held 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 the second-order move selection in Equation (2) and the number of playouts was 1000 for E and G and 10000 for F and H.

Black won 53% of the matches because of first player's 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 were not clear. Details are given in (Raiko, 2006).

[2] `http://www.cis.hut.fi/praiko/hex/`

References

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

Sylvain Gelly, Yizao Wang, Rémi Munos, and Olivier Teytaud. Modification of UCT with patterns in Monte-Carlo Go. Technical Report RR-6062, 2006. `http://hal.inria.fr/inria-00117266`.

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.

Tapani Raiko. Higher order statistics in play-out analysis. In T. Honkela, T. Raiko, J. Kortela, and H. Valpola, editors, *Proceedings of the Ninth Scandinavian Conference on Artificial Intelligence (SCAI 2006)*, Espoo, Finland, 2006.

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