Toward Opening Book Learning

Michael Buro

NEC Research Institute
4 Independence Way
Princeton NJ 08540, USA

email: mic@research.nj.nec.com

Abstract

In this article an opening book framework for game–playing programs is presented. Motivated by basic requirements for successfully playing a sequence of games — such as avoiding losing games twice in the same way — it is shown how reasonable move alternatives can be found in order to deviate from previous lines of play. Variants of the algorithm are used by several of today's best Othello programs and allow them to extend their opening books automatically.

Keywords: opening book, game–tree search

1 Introduction

Over the years, programs playing perfect information games like Chess, Checkers, and Othello have become stronger mainly by increasing the search depth using faster hardware and improved game–tree search techniques. Moreover, better evaluation functions have been developed which estimate the winning chances more accurately while using less time than before. Unfortunately, in spite of these improvements, programs still show weaknesses in the opening phase stemming from a lack of strategic planning. In order to circumvent this problem, opening books are used in which move sequences or positions together with moves are stored. Their automatic generation was of little interest up to now, since move sequences can be taken from the literature, suited to one's own requirements — such as the striving for tactical complications — and manually updated if necessary. Today, many computer game–playing programs are attached to servers, playing against human players and other programs twenty–four hours a day. Thus, it has become necessary for the programs to update their opening books automatically without human intervention.

In this article, an opening book approach is presented which meets the minimal requirements of a skilled multi–game strategy that avoids losing games twice in the same way and looks for reasonable deviations. Furthermore, it enables a program with a reasonably good evaluation function to explore new promising opening lines by itself without the help of human experts, and to make use of games played by other players.

Since the new technique finds promising move alternatives globally with respect to all variations stored in the opening book, it goes beyond the usual procedures that simply choose random moves from the book, or use stored deep evaluations of previously encountered positions in subsequent game–tree searches [Samuel (1959), Scherzer et al. (1990)].

The basic idea of evaluating move alternatives and using minimax search to guide the opening book play was presented independently in [Delteil (1993)] and [Buro (1994)]. Today, several good Othello programs, including HANNIBAL[1] by Martin Piotte and Louis Geoffroy, ECLIPSE by Colin Springer and Michael Giles, and Mark Brockington's KEYANO[2], use variants of this technique. In what follows, the procedure currently used by LOGISTELLO[3] is described, which is among today's best Othello programs.

2 Basic Requirements

If a player wants to be successful not only in a single game against an unknown opponent but in a sequence of games, he might be faced with simple but effective playing strategies of the opponent which cannot be met only by the well–known game–tree search techniques. Perhaps the most obvious and simple one is the following: "If you have won a game, try it the same way next time." A program with no learning mechanism and no random component follows this strategy, but is also a victim of it, since it does not deviate and therefore can lose games twice in the same way. In order to avoid this,

[1] http://www.cam.org/~piotte/Hannibal.html

[2] http://www.cs.ualberta.ca/~brock/

[3] It placed first 17 times and second five times in the 23 tournaments it has played since October 1993. (http://www.neci.nj.nec.com/homepages/mic/log.html)

it is necessary to find reasonable move alternatives. This can be accomplished passively, as the following strategy shows: "Copy the opponent's winning moves next time when colors are reversed." The idea behind this elegant method is to let the opponent show you your own faults in order to play the opponent's winning moves next time by yourself. In this way, even an otherwise stronger opponent can be compromised, since — roughly speaking — eventually he is playing against himself. Thus, copying moves makes it necessary to come up with good move alternatives actively. In order to do so, a player must have an understanding of his winning chances after deviations from known lines.

3 Choosing Book–Moves

The mentioned basic requirements of a skilled match strategy lead directly to an algorithm for guiding opening book play based on negamax search. Suppose a game–tree is built from variations — starting with the initial game position — and its leaves are labelled as follows: The first component of the label indicates whether the corresponding position is a sure win, draw, or loss for the side to move (W,L,D). In cases where this classification is not yet known, a question–mark is used in the first component and the second component is the heuristic evaluation of the position computed, for instance by a deep negamax search (larger values indicate a higher winning chance for the side to move). Furthermore, in each interior node of the tree the heuristically best[4] deviation is added to the tree together with the reached position and its deep evaluation. Figure 1 shows examples. Here, solid lines mark variation moves, whereas long dashed lines represent the best move alternatives in each interior node. Short dashed lines indicate the existence of other deviations with lower evaluation which don't have to be considered for the moment because the best deviation would be preferred.

Given such a tree, it is easy to guide the opening book play in best–first manner: Find the node corresponding to the current position, propagate the heuristic evaluations from the leaves to that node by means of the negamax algorithm, and choose the move that leads to the successor position with lowest evaluation.[5] Before the move to be played can be determined using this algorithm, heuristic evaluations have to be assigned to the leaves for which the outcome is known. A natural choice is $+\infty$ for won positions, 0 for draws, and $-\infty$ in lost positions. Provided that heuristic evaluations are always greater than $-\infty$, and there are still unexplored variations, this setting ensures that games will not be lost twice in the same way.

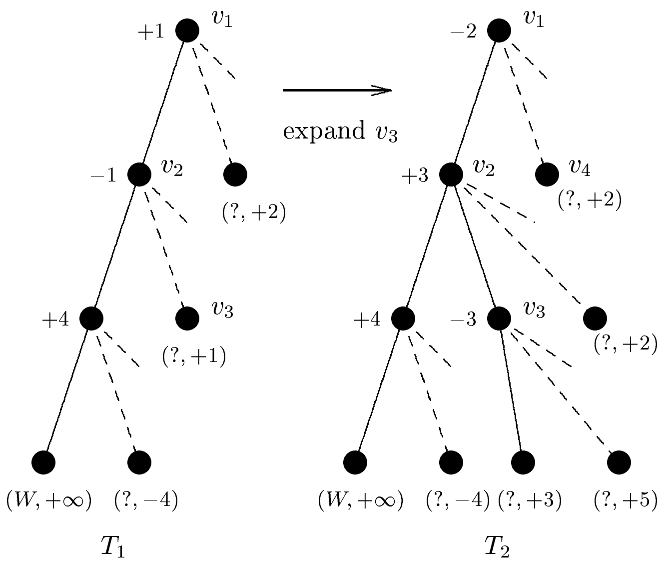

Figure 1: Example opening book trees. The principal leaf v_3 of T_1 is chosen for expansion. In the resulting tree T_2 leaf v_4 would be expanded next.

The algorithm applied to the root of the example–tree T_1 in Figure 1 yields the optimal path (v_1, v_2, v_3) which maximizes the winning chances of the root player against best counter-play (local to the tree).

4 Book Extension

Knowing how to select moves from an opening book immediately enables the automatic extension of the book by iterated expansion of the leaf at the end of the current principal variation. In each step, the best move and the best deviation are determined in the leaf position and added to the tree together with their evaluations. Furthermore, if the leaf position was reached by a deviation, the next best deviation in the predecessor position also has to be found and added. Thus, after each expansion, the tree is complete again consisting of a variation skeleton augmented by the best deviation in each interior node.

Figure 1 illustrates this process. In opening book tree T_1 the principal leaf v_3 will be expanded next. Since a deviation led to this position, the next best deviation at v_2 as well as the best and second best move at v_3 have to be determined by a search to complete T_2.

While extending the tree in the described fashion, it might happen that both sides are happy with the same drawing line, and therefore the principal variation doesn't change anymore. In order to prevent this scenario, draws can be alternately counted as a win resp. a loss for a fixed player by setting their heuristic evaluation

[4]In what follows, *best* means *heuristically best*.

[5]Since in this process heuristic evaluations of positions from different game phases are compared, it is recommended to use an evaluation function with a game–phase independent meaning, such as winning probability or expected result.

to $+\infty$ resp. $-\infty$ depending on the side to move.[6]

The described iterative node expansion is an inherently sequential process, because the evaluations of the positions added in the previous step directly influence the choice of the node to expand next. However, by allowing additional node expansions, it is possible to parallelize the book extension if several CPUs are available. One simple solution is to assign the subtrees beneath the most commonly played opening variations to the available processors, which then perform the sequential node expansion locally. This procedure has two disadvantages: First, it can lead to multiple expansions of nodes if the game allows transpositions, i.e. the variation 'tree' is in fact a directed acyclic graph (DAG) and the 'subtrees' given to the processors are sub–DAGs, which might not be disjoint. Second and more importantly, the algorithm generates a tree very different from the tree built by the sequential extension procedure if it turns out that only a few of the chosen opening lines are indeed worth exploring.

Another approach for a distributed book extension in a first pass collects a subset of leaves that are likely to be chosen by the sequential algorithm, and thereafter expands them efficiently in parallel. A simple method for collecting leaves is to iterate the selection and deletion of the current principal leaf from the tree.

Besides the just described iterative book extension, it is crucial to add games played in public together with the evaluated deviations, regardless of the game outcome. In case of a loss or draw, adding the game is necessary to avoid the repetition of the same line, whereas in case of a win, it helps to prevent following lines which looked good at the deviation point, but turned out to be bad later in the games in which the opponent only lost due to a mistake.

Finally, by including games of other competing players, it is possible to anticipate opponents' moves and to explore new opening lines, which otherwise would be considered by the described move–selection algorithm only after many iterations. Since this algorithm is driven by heuristic evaluations of deviations rather than solely by game outcome, it is not even necessary to correct the games before including them. It is sufficient to determine the best deviations along the new paths.

5 Implementation Aspects

Once the algorithms for opening book handling have been described the question arises how these can be implemented efficiently. Since potentially many variations and deviations are going to be stored, not only the access– and update–time has to be taken into account but also the space the book occupies on disk or in main memory. The crucial operation regarding the book is move retrieval since it must be fast to avoid delays during the game. Thus, either the implementation ensures a fast value propagation, so that negamax searches in the opening book tree can be performed during the game, or the values of interior nodes have to be determined in advance before the game begins. One advantage of a fast propagation is that the opening book behavior can easily be altered by command line options, for instance, telling the program to count draws as wins for one side or to randomize the move decision as described in the next section. On the other hand, performing fast negamax searches is only feasible with the entire opening book stored in main memory. While this might have been a problem in the past, today's main memory size of typical PCs allows the handling of megabyte data–structures.

For high–speed value propagation in memory a node can be represented as a record including moves and pointers to its successors in the tree. The node corresponding to the position for which a book move has to be determined can be found by using the sequence of moves played so far. Thereafter, the subtree below this node is searched by simply following pointer chains and backing–up evaluations. There is no need for making/undoing moves on a board while searching.

Opening book 'trees' for games that allow transpositions are actually directed acyclic graphs (DAGs) in which nodes may have more than one predecessor. In these games positions are not uniquely represented by move sequences which makes finding specific positions in the book harder. To solve this problem, the nodes can be stored in a hash table indexed by keys that are assigned to positions. Hash collisions are resolved by a probing algorithm that searches the entire hash table starting with the position key index until either the node with the given board id has been found or an unoccupied hash entry has been reached which indicates that the position is not stored in the table. After a successful search the described tree search routine can be used to quickly evaluate the sub–DAG beneath the node.

6 Discussion and Enhancements

Variations of the presented opening book algorithm are used by several of today's best Othello programs and have proved effective:

- Since each leaf position is evaluated by the program, surprises in tournament games caused by blindly following un–evaluated opening lines are no longer to be feared.

[6] This draw evaluation scheme also has a useful application in actual tournament play where it allows avoiding book–draws when a win is necessary or broadening the deviation opportunities when a draw suffices.

- Many programs connected to the Internet Othello Server[7] are now playing against human players and programs and improving their books autonomously all day long without the need of human interference.
- Extensive automatic book preparation is now possible which makes the difference at the top of the rating list. The best programs spend a lot of time in extending their books using the described method with time controls considerably longer than under normal tournament conditions. This allows playing a good game even against programs running on much faster machines.
- New sound opening lines and refutations of some openings frequently played by humans have been discovered by programs. Knowing these innovations gives humans an advantage in tournaments as Parsons (1995,1997) shows. By studying LOGISTELLO's openings played on the Internet Othello Server he recognised several improvements on human openings. These worked out in his last game against Hideshi Tamenori, who is one of the top Japanese players, and in several of his games at the 1996 Othello World–Championships.

However, there is still room for improvement. Recently, the move selection procedure has been extended in two ways: First, it now allows a basic form of move randomisation which makes the program's opening choice less predictable. The principle is simple: Moves are now selected after two passes through the subtree rooted at the current position. In the first step, the negamax–value v of the current position is determined with respect to the entire opening book — as before. Thereafter, in a second pass, all principal variations are counted which lead to values within $[v - \epsilon, v]$, where ϵ is a small tolerated evaluation difference. Finally, a move is selected randomly among the moves that lead to acceptable positions taking into account the just computed path counts. This procedure ensures that each path has the same probability to be chosen. Of course, it must not be iterated carelessly, since otherwise the evaluation could drop by ϵ after each move. Instead, the algorithm has to keep track of the maximal negamax–value — $v_{\max}$ — it could ever reach in the game so far, and to make sure that only paths are chosen with negamax–values in $[v_{\max} - \epsilon, v_{\max}]$. This goal can be accomplished by updating $v_{\max}$ if necessary before paths are counted. If v is greater than $v_{\max}$ (that is, the opponent has just made a mistake in view of the opening book), the current maximum value $v_{\max}$ has to be set to v. Then, paths having values in $[v_{\max} - \epsilon, v_{\max}]$ are counted and a move is selected as described above.

Second, the book algorithm is now able to distinguish two draw types: public and private, which broadens the deviation opportunities. As the notation suggests, positions in the opening book which occurred in drawn games played in public are marked as such. All other draws in the book were generated by private book extension (hopefully) unknown to the public. This distinction allows the program to avoid public draws while striving for wins and private draws (which can easily become wins if the opponent makes a small mistake) by using $+\infty$ resp. $-\infty$ as heuristic evaluations for drawn leaf–positions with respect to the color to move and the public/private flag.

[7] `telnet://external.nj.nec.com:5000`

7 Outlook

So far, it seems that move selection by means of negamax search would be ideal for guiding opening book play. However, the algorithm in the present form has a major weakness which clearly shows up when the book extension is pushed to the limit. Suppose that the game in question is a theoretical loss for the first player and book extension has reached a point where only a few deviations are left to explore. Here, the first player would desperately select one of these remaining deviations, even if its heuristic evaluation indicates that this plan is hopeless and — more importantly — even in case the player would know that the opponent is less informed. While this problem might not occur in common games like Chess or Othello because of their large search spaces, a similar behaviour can be observed much earlier: The described approach doesn't guarantee that a program equipped with a large opening book wins a tournament against a version with a smaller book. The reason behind this surprising insight is that the negamax algorithm implicitly assumes the opponent to have the same information. This prevents the program with the large book from playing lines which could be successful if the opponent doesn't know them.

Therefore, further investigations could focus on modeling the opponent's book in order to make full use of private analysis.

Acknowledgments

My thanks go to Igor Đurđanović and my wife Karen for many fruitful discussions. Furthermore, I am grateful to Mark Brockington, Warren Smith, and Judy Sonnenberg for their helpful comments on an earlier version of this article.

References

[Buro (1994)] M. Buro. *Techniken für die Bewertung von Spielsituationen anhand von Beispielen*, Ph.D. thesis (in German), University of Paderborn, Germany.

[Delteil (1993)] J. Delteil. *A propos des bibliotheques d'ouvertures*, Magazine de la Fèdèration Française d'Othello, FFORUM 29, pp. 18–19.

[Parsons (1995)] D. Parsons. *My Game with World Champion Hideshi Tamenori*, Othello Quarterly, Vol. 17, No. 4, pp. 16–17.

[Parsons (1997)] D. Parsons. *1996 World Championship: Parsons vs. Shaman*, Othello Quarterly, Vol. 19, No. 1, pp. 10–11.

[Samuel (1959)] A.L. Samuel. *Some Studies in Machine Learning Using the Game of Checkers*, IBM Journal of Research and Development 3, pp. 210–229.

[Scherzer et al. (1990)] T. Scherzer, L. Scherzer, D. Tjaden. *Learning in Bebe*, In: T.A. Marsland, J. Schaeffer (Eds.), Computers, Chess, and Cognition, Springer–Verlag New York, pp. 197–216.