Active Opening Book Application for Monte-Carlo Tree Search in 19×19 Go

Hendrik Baier and Mark H. M. Winands

Games and AI Group, Department of Knowledge Engineering
Maastricht University, Maastricht, The Netherlands

Abstract

The dominant approach for programs playing the Asian board game of Go is nowadays Monte-Carlo Tree Search (MCTS). However, MCTS does not perform well in the opening phase of the game, as the branching factor is high and consequences of moves can be far delayed. Human knowledge about Go openings is typically captured in *joseki*, local sequences of moves that are considered optimal for both players. The choice of the correct joseki in a given whole-board position, however, is difficult to formalize.

This paper presents an approach to successfully apply global as well as local opening moves, extracted from databases of high-level game records, in the MCTS framework. Instead of blindly playing moves that match local joseki patterns (*passive* opening book application), knowledge about these moves is integrated into the search algorithm by the techniques of move pruning and move biasing (*active* opening book application). Thus, the opening book serves to nudge the search into the direction of tried and tested local moves, while the search is able to filter out locally optimal, but globally problematic move choices.

In our experiments, active book application outperforms passive book application and plain MCTS in 19×19 Go.

1 Introduction

Since the early days of Artificial Intelligence (AI) as a scientific field, games have been used as testbed and benchmark for AI architectures and algorithms. The board game of *Go*, for example, has stimulated considerable research efforts over the last decades, second only to computer chess. Nevertheless, the strongest 19×19 Go programs are still only able to play at the level of an advanced amateur player [23]. Traditional game tree search methods like $\alpha\beta$ search, being quite successful in e.g. computer chess or checkers [5, 25], are impractical in Go; writing a master strength Go program stands as a grand challenge of AI [4].

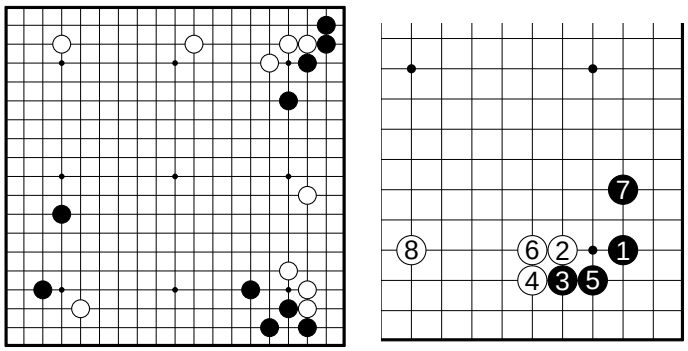
After its introduction in 2006, *Monte-Carlo Tree Search* (MCTS) [9, 19] has quickly become the dominant paradigm in computer Go [20]. MCTS has considerably advanced the strength of computer Go programs; however, it still has its shortcomings. Similar to other game tree search algorithms like $\alpha\beta$ search for example, it is difficult for MCTS to properly evaluate *opening positions* [2]. This weakness stems from the fact that MCTS decides on the best move in a given situation on the basis of statistical sampling. In an open position at the beginning of a Go game however, when the branching factor of the game tree is highest and the consequences of actions lie many moves in the future, MCTS is blind to the differences between good and bad opening moves. MCTS Go programs tend to play their first moves haphazardly [22], falling behind their human opponents who are familiar with opening theory.

In other games, similar problems have been approached by the creation of *opening books* [3, 5, 10, 16, 21, 24]—large databases of trusted moves that can be used to safely lead a game-playing program into the kind of middle-game positions that it is more comfortable with. However, little is known about how to build an opening book for 19×19 Go and how to combine it with MCTS. The concept of *joseki*, used to organize human knowledge about Go openings, bears considerable challenges regarding formalization and automated application. The goal of this paper is the construction of an opening book for 19×19 Go. The approach taken is *active* application of opening books, the integration of opening knowledge into the search process—as opposed to *passive* application, the replacement of search by a book lookup in the early phase of the game.

This paper is organized as follows. Section 2 describes the game of Go in more detail, followed by a presentation of MCTS in Section 3. Section 4 gives an overview of related work on opening books for game-playing programs in general and Go programs in particular. The specific difficulties with regard to Go openings are explained. Section 5 outlines the approach to opening books taken in this work, while Section 6 presents experimental results in 19×19 Go. Conclusions and future work follow in Section 7.

2 The Game of Go

Go is played on a grid board of typically 19×19 intersections, although smaller board sizes exist for quicker, informal games and for educational purposes. Starting with an empty board, two players alternatingly place white and black stones on an empty intersection of their choice (Fig. 1(a) shows a possible beginning of a game). If a player can surround any enemy stones completely with his own, the surrounded stones are removed. In the end of the game—after both players have passed—the player who occupies or surrounds more intersections on the board ("territory") than his opponent wins.



(a) The first moves of a 19×19 game. (b) A corner joseki.

Figure 1: Global and local view of Go openings.

Since the goal of the game lies in the surrounding of board area, players usually play in the corners of the board first. Here, two edges of the board serve as additional territory walls and make it possible to surround a given area with a smaller number of stones. After the corners have been claimed, players move on to the sides, where at least one edge of the board helps the player's stones in surrounding intersections. Only after this opening phase, stones are typically placed in the center of the board, where the building of territory is most difficult and fights are most complex.

Two Japanese terms are often used to describe opening play in Go. (1) *Fuseki* means the thinly spread out moves at the start of the game, when players claim corners and sides without immediately getting into fights. The term relates to the situation on the whole board. (2) *Joseki* means specific, established sequences of moves in a local area of the board—typically a single corner—which are judged to be locally optimal and balanced for both black and white sides. Fig. 1(b) shows an example joseki.

The largest part of Go opening theory deals with joseki. While joseki have some similarity to standard opening moves in other games like chess, there is an important difference. Joseki achieve locally optimal play, but they do not deal with the whole-board position. Players therefore have to choose wisely which joseki to play, based on their assessment of the board on a global scale and on their strategy for the game.

3 Monte-Carlo Tree Search

For each move decision in a game of Go, the Monte-Carlo Tree Search (MCTS) algorithm [9, 19] constructs a search tree to determine the best move. This tree starts from the current position and is selectively deepened into the direction of the most promising moves, which are chosen according to the success of simulated Go games starting with these moves. Initially, the tree contains only the root node, representing the current board position. Each further node added to the tree stands for one of its possible successor positions (or moves), and stores the current value estimate for this position. MCTS works by repeating the following four-phase loop until computation time runs out [7]. It can be interrupted after any number of iterations

to return the current result—the highest-valued move at the root node. Each loop represents one simulated game.

Phase one: *selection*. The tree is traversed from the root to one of the leaf nodes. At every step, MCTS utilizes a selection policy to choose the move to sample from this position. The selection policy tries to balance exploitation of positions with high value estimates and exploration of positions with uncertain value estimates.

Phase two: *expansion*. After a leaf node has been reached, a decision is made on whether to apply zero, one or more of its successors to the tree. A common expansion policy in Go is the addition of one newly sampled position per simulation [9].

Phase three: *simulation*. According to a simulation policy, moves are played in self-play until the game is finished. While uniformly random move choices are sufficient to achieve convergence of MCTS to the optimal move in the limit, more sophisticated ("quasi-random") playout policies have been found to improve convergence speed.

Phase four: *backpropagation*. After the end of the simulation has been reached and the winner of the simulated game has been determined, the result is used to update value estimates at all nodes traversed during the playout.

Two strategies that have been used amongst others to improve MCTS Go programs in the past are the (often temporary) *pruning* of moves which are judged inferior by expert heuristics [7, 8], and the *biasing* of the search tree into a direction chosen by expert heuristics [7, 13]. In Section 5, these two methods are considered to integrate opening book information into MCTS.

4 Opening Books

In this section, we review some of the literature on opening books and outline the obstacles to opening books for 19×19 Go. Section 4.1 focuses on opening books in other games than Go, using Lincke's [21] categories for opening book *construction* and newly introduced categories for opening book *application* to classify them. Section 4.2 reviews closely related work on opening books in Go. Finally, Section 4.3 specifies what we consider the main problem for opening books in 19×19 Go.

4.1 Classification of Opening Books

Opening theory was introduced to computer game-playing soon after the first game programs had been written. Opening books consisting of thousands of board positions and moves were compiled for some of the first chess [16] and checkers programs [24]. Typically, these books are constructed *passively* [21]: Information is gathered from strong players and entered into a database, usually in the form of expert games. This information is then used to modify opening play, in the most basic case by immediately playing a stored expert move whenever the current game position could be found in the database. More sophisticated ways of adapting expert knowledge to the needs of a game-playing program have been described e.g. in [17, 10] for chess or [25] for checkers.

Other research focused on *active* book learning [21], learning techniques that do not depend on expert information and instead construct an opening book automatically by actively playing and exploring the space of opening moves. Lincke [21] applied such methods to Awari and Othello, Buro [3] to Othello and Karapetyan and Lorentz [18] to Amazons.

Similar to how the *construction* methods of opening books can be divided into active and passive approaches, the information contained in opening books can be *applied* actively and passively as well. Here, we use the term *passive* application for methods that do not involve any search in the game tree. The current position is looked up in the book, and if it is found, the stored information is used to compute an optimal move. *Active* application uses book information to *modify* the search process, instead of *replacing* it.

Examples of active application—the integration of book information into a game tree search algorithm—can be found in [5, 10]. In both approaches, the "goodness" of a move, computed from the opening book, is used to shift the $\alpha\beta$ search window of the respective move. Thus, book information is not trusted blindly, but used to bias the search algorithm. This biasing is particularily effective in avoiding disastrous openings by faulty book moves, which appear with higher probability as opening book sizes grow. Active application is robust against "partially dirty statistical information" [10]. The famous chess machine DEEP BLUE [5] even combined a small passively applied book (a couple thousand moves hand-picked by an expert) with a bigger actively applied book (the first 30 moves of 700,000 grandmaster games).

4.2 Related Work in Go

In [6], a form of Meta-MCTS was used to construct a Go opening book. Thousands of complete games were played from the empty board, and the results of these games were used to build an opening book similar to how results of quasi-random games are used by MCTS to build a search tree. A drawback of this technique, while conceptually elegant, is that it is quite time-consuming—it has therefore only been applied to Go on the small 9×9 board instead of the regular 19×19 board. Consequently, the problem of joseki was not handled specifically, since there are no standardized corner sequences for 9×9 boards. Also, the authors found opening books handcrafted by experts to be more robust than Meta-MCTS books with respect to different time controls. It appears as if books constructed by self-play contain less universal Go knowledge and more information depending on the specific parameters, strengths and weaknesses of the program used for building them. We expect that by learning from expert games instead of self-play, a book of more lasting value can be built, at least until computers surpass humans in Go and are able to refine opening theory beyond the level currently known to human masters.

In [2], expert games were used in a novel way to improve opening play in 19×19 Go. Instead of building an opening book, the authors used a game database to tune a multivariate evaluation function for opening positions. This evaluation function was then applied to MCTS in two different ways: First to preselect a number of good opening moves for the MCTS search, and second to bias the nodes of the MCTS tree with opening knowledge. Only the first approach was experimentally tested. However, the improved openings were not tested by comparing their performance in complete games of Go, but by comparing positions reached after the opening through the evaluation of the strong MCTS Go program MoGo [20]. The authors wanted to avoid statistical noise in the remaining moves to the end of the game. This makes it difficult to compare the effectiveness of their method with the results of this paper.

In [22], fuseki and joseki books were constructed from expert games and applied passively, i.e. without combining their information with a search process. Whenever the board matched a whole-board position found in the opening book, the corresponding most frequent expert move was played (fuseki book), while joseki moves were found by matching individual corners of the board with corners extracted from the expert games, and then playing the most frequent next expert move stored for that corner. No significant improvements could be shown here. Nevertheless, we use a similar method for constructing opening books in this paper, combining it with an active technique for applying books in MCTS that is similar to the second technique proposed in [2].

4.3 The Problem of Joseki

One of the main difficulties in understanding Go openings, for humans as well as for computers, is the context-sensitiveness of joseki mentioned in Section 2. Failure to properly take subtle features of the global situation into account when choosing local exchanges can lead to worse results for a player than not knowing how to play locally optimal at all. Thus, a famous Go proverb says, "Learning joseki loses two stones strength—studying joseki gains four stones strength", meaning that rote memorization of joseki can be harmful, while understanding their principles and adapting them to the overall board situation as well as the own individual style increases playing strength.

From the point of view of pattern matching, joseki pose the following problem. If the opening book on the one hand does not recognize joseki at all and handles only whole-board positions, it has to find and store the same joseki in countless combinations with all other possible joseki in the other corners. The result could be a shallow and sparse book that has little effect on playing strength, as new games will often show new combinations of joseki that the book will be unable to match. If the book on the other hand does store individual corner sequences, it has to deal with challenges in the application of this knowledge. When a corner position allows for several possible joseki continuations, which one of them is most appropriate to the whole-board situation and to the playing style of the program?

Since computers still play Go on a level below that of the best humans, it is the goal of this paper to passively construct an opening book for Go, in order to fully exploit the knowledge contained in the thousands of professional or high-level amateur game records that are available on the internet. Furthermore, the opening book will be actively applied inside MCTS, to seamlessly integrate the information of an expert fuseki and joseki book with the strengths of the search algorithm. This integration is intended to mitigate the joseki problem as well: Search itself is tasked with choosing the optimal joseki in a given position. We hope that ineffective joseki will be filtered out by MCTS search just like problematic opening moves are filtered out by $\alpha\beta$ search in [5, 10].

5 An Opening Book for MCTS

In this section, we propose our two-step approach for combining opening books with MCTS.

1) The book is constructed passively from game records by professional or highly-ranked amateur players. Passive book construction is preferred to active construction due to its relative speed and the potential that is still untapped in human opening theory. Book information is collected in two ways:

   a) First, for each game all positions up to move 50 are stored together with the move that was made from the position. This is the *fuseki book*. Three parameters can be used to modify the content of this book: (1) The minimum number of times θ a position has to appear in the games in order to be stored, (2) the minimum number of times ρ a move has to appear in a given position in order to be stored, and (3) the minimum percentage of times σ a move has to appear in a given position in order to be stored.

   b) Second, for each game each corner is examined separately; up to move 100, every partial board position appearing in every corner is stored in a second book, together with the next move that was made in the same corner (not necessarily the next move in the original game, since players can switch back and forth between corners in the opening). This is the *joseki book*. The same parameters apply as for the fuseki book.

   All positions that can be produced by reflecting or rotating a position from an actual game record are taken into account as well. This procedure is similar to the book building described in [22]. Three modifications were made. (1) Instead of only the most frequent move for each book position, we store all moves that are "frequent enough" as defined by the parameters described above; (2) instead of the moves of both Black and White, only the moves of the winning player of each game are considered; and (3) a corner is defined as 10×10 intersections instead of 9×9 such that stones on the middle lines, especially on the side star points and the *tengen* (the middle of the board), are not ignored.

2) The combined fuseki and joseki book is used actively by integrating it into MCTS. Four techniques of using book information in MCTS have been implemented:

   a) The information can be used in the *selection phase* of MCTS by looking up the current whole-board position in the fuseki book and the current four corners of the board in the joseki book whenever a new tree node is created. The move choice at this game position is then simply restricted to the moves found in the books—all other moves are excluded (*book pruning*). Among all book moves, MCTS can choose the one it understands best.

   b) A second way the information can be used in the *selection phase* of MCTS is by not restricting move choice when a new node is created, but by giving each move found in the books a certain number of virtual wins (*book bias* through node priors) [13]. The strength of the bias can be uniform or depend on e.g. the number of appearances in the book. MCTS explores book moves first as their value estimates are better than those of other moves initially; however, if no book move is appropriate for the current position or if the search algorithm does not understand in time how to utilize them properly, the prior bias can be overcome and another move can be chosen.

   c) Similarly, book pruning can be used in the *simulation phase* of MCTS by looking up the current whole-board position in the fuseki book and the current four corners of the board in the joseki book whenever a move decision has to be made. Between all moves found in the various books—if any—one is then chosen at random and the simulation continues. If no book move is found, the algorithm falls back to its standard simulation policies. The application of book moves in simulations is introduced to ensure that book moves in the tree are not evaluated incorrectly because their move sequences are not played out fully after leaving the tree.

   d) And finally, the "soft" approach of giving book moves a bias can also be applied in simulations—by increasing the chances of book moves to be played as compared to non-book moves. Compared to the previous method, this could lead to more diversity in the simulations and to a better exploration of possible outcomes if the opponent would not continue with the expected joseki.

   The first two techniques apply book information in the tree, and the second two methods apply it in the simulations. From both groups, a choice can be made independently and combined with each other or with the default behavior of MCTS. The performance of four settings of techniques is evaluated in the next section.

6 Experimental Results

All experiments were run using OREGO [11] version 7.08 as the basic platform. The baseline version of OREGO uses a number of MCTS enhancements like a transposition table [16], RAVE [13], a simulation policy similar to that proposed in [14], and LGRF-2 [1]. The program was run on two CentOS Linux servers, each consisting of four AMD Twelve-Core OpteronT 6174 processors (2.2 GHz). Each experimental run involved 5632 19×19 games in total—2816 games as Black and 2816 games as White—either in self-play or against the classic (non-MCTS-based) program GNU GO 3.8 [12], using Chinese rules (area scoring), positional superko, and 7.5 komi. GNU GO ran at its default level of 10, with the capture-all-dead option turned on. The time limit for OREGO was 10 seconds per move, and it used a single thread.

The remainder of this section is structured as follows. In 6.1, the construction of our opening book is described. Next, 6.2 presents results of passively applying this book in games against GNU GO. Active application is tested in 6.3. Finally, 6.4 presents the results of active book application in self-play.

6.1 Construction of Opening Book

In all experiments presented in this section, the same opening book was used. It was built from about 50,000 game records as described under 1) in Section 5. All of the games are freely available on the internet (the majority on [15]). The parameters were set as follows. Whole-board moves were stored if they appeared at least 5 times in the game records and were used in at least 5% of the cases in their corresponding position. Corner moves were stored if they appeared at least 20 times and were used in at least 2% of the cases in their corresponding position. The minimum number of times a position has to appear in the game records in order to be considered was set to 0, so all positions were stored if at least one corresponding move fulfilled the constraints of the other parameters. This procedure resulted in a relatively small opening book of less than 3MB for fuseki and joseki book combined.

6.2 Passive Application of Opening Book

For the first experiment, a baseline was established by letting OREGO without any opening book play against GNU GO. Baseline OREGO only uses the following heuristic to roughly guide opening play: It prunes all moves that are neither on the third or fourth line from the edge of the board, nor near to another stone.

In the second experiment, the opening book was applied passively. As proposed for the "combined book" in [22], whenever the board matched a whole-board position in the fuseki book, the corresponding most frequent expert move was played. If there was no whole-board match, the joseki book was consulted, checking first the corner of the opponent's last move and then in random order all other corners for a matching partial board position. Again, the most frequent expert move was played in case of a match. Until no corner of the board could be found in the book anymore, no search was involved in the process of choosing a move. Table 1 shows the results.

Table 1: Performance of passive application of opening book.

Player	Win rate against GNU Go	95% confidence interval
Baseline OREGO	39.3%	38.0%–40.5%
OREGO with passively applied opening book	40.2%	38.9%–41.5%

Passive application of the book did not result in significantly stronger play. This agrees with the findings of [22]. After the fuseki book is left, joseki in the four corners are chosen without any relation to each other; what is gained by the correct local responses is probably lost by the lack of overall strategy.

6.3 Active Application of Opening Book

In the next experiment, the opening book was applied actively. Four settings of the techniques described under 2) in Section 5 were tested. The first setting in this group tested the combination of book pruning in the selection phase with default simulations that do not use book information (OREGO *prune-default*). In the second setting, book pruning in the selection phase was combined with book pruning in the simulation phase (OREGO *prune-prune*). The third setting tested the combination of book bias in the selection phase with default simulations (OREGO *bias-default*). Whenever a new node was added to the search tree, every corresponding book move retrieved from either the fuseki or the joseki book was given a prior bias of 40

Table 2: Performance of active application of opening book.

Player	Win rate against GNU Go	95% confidence interval
Baseline OREGO	39.3%	38.0%–40.5%
OREGO *prune-default*	41.7%	40.4%–43.0%
OREGO *prune-prune*	41.6%	40.3%–42.9%
OREGO *bias-default*	44.2%	42.9%–45.5%
OREGO *bias-prune*	42.5%	41.2%-43.8%

virtual won simulations. In the fourth setting, book bias in the selection phase was combined with book pruning in the simulation phase (OREGO *bias-prune*). Again, the bias was 40 wins for every book move. All four settings were compared to the baseline evaluated above. Table 2 shows the results.

All four settings are significantly stronger than baseline OREGO ($p<0.001$ for *bias-default* and *bias-prune*, $p<0.01$ for *prune-default* and *prune-prune*), although the effect of the opening book, influencing only the first few moves of a game, is rather small. OREGO *bias-default* is significantly stronger than *prune-default* and *prune-prune* ($p<0.01$) and almost significantly stronger than *bias-prune* ($p<0.1$). A possible hypothesis for the ineffectiveness of pruning is the lack of diversity that might come with it. Too much determinism is known to hurt MCTS performance [13].

6.4 Active Application in Self-Play

In the last experiment described in this paper, the opening book was applied actively in self-play. The goal of this experiment was to test how the opening book influences the performance against MCTS-based players. The best-performing algorithm variant in games against GNU GO, OREGO *bias-default*, was pitted against baseline OREGO without any opening book. Table 3 shows the results.

Table 3: Performance of active application of opening book in self-play.

Player	Win rate against baseline OREGO	95% confidence interval
OREGO *bias-default*	50.6%	49.3%–51.9%

The addition of the opening book does not significantly improve OREGO's play against itself. The probable reason is that baseline OREGO chooses non-standard moves in the opening, which forces its opponent quickly out of the book. If the book is left too quickly, OREGO *bias-default* does not gain a sufficient advantage—the book might even be harmful in cases when OREGO *bias-default* does not find the proper follow-up moves to the fuseki or joseki it started, and abandons advantageous positions.

7 Conclusion and Future Research

In this paper, we introduced the distinction between active and passive application of opening books in game-playing programs, and proposed several ways of active application in the framework of Monte-Carlo Tree Search. Special attention was given to the problem of joseki in 19×19 Go. Empirical results show that of the variants tested so far, biasing tree nodes with opening book information works best, significantly improving on the performance of the MCTS-based program OREGO as well as on the combination of the same opening book with a passive application method proposed in the literature.

Several directions appear promising for future work. First, we hope to integrate opening knowledge successfully into MCTS simulations beyond the tree, for example by only slightly increasing the playing probability of book moves and thus keeping more diversity in simulations. Second, the application of opening knowledge could be refined, e.g. by taking the frequency of moves into account when deciding on the strength of the prior bias for a given position. Third, it remains to be tested how active book application scales with computation time, and whether it can improve on the almost instant passive book application also in games with a per-game time limit (instead of per-move).

The performance of the proposed application methods depends on the quality of the opening book used, which could also be improved. An unsolved problem with the automatic extraction of joseki from high-level games—as shown by our self-play experiment—is that it is unclear how to learn optimal refutations to non-joseki moves from games in which non-joseki moves do not appear.

Finally, there is much room for optimization of the various parameters involved in the construction and application of opening knowledge according to our techniques.

Acknowledgment.

This work is funded by the Netherlands Organisation for Scientific Research (NWO) in the framework of the project Go4Nature, grant number 612.000.938.

References

[1] H. Baier and P. Drake. The Power of Forgetting: Improving the Last-Good-Reply Policy in Monte-Carlo Go. *IEEE Transactions on Computational Intelligence and AI in Games*, 2(4):303–309, 2010.

[2] J. Basaldúa, T.-N. Yang, and J. M. M. Vega. M-eval: A multivariate evaluation function for opening positions in computer Go. In *Workshop on Computer Games (IWCG 2010)*. IEEE conference, 2010.

[3] M. Buro. Toward Opening Book Learning. *ICCA Journal*, 22(2):98–102, 1999.

[4] X. Cai and D. C. Wunsch, II. Computer Go: A Grand Challenge to AI. In W. Duch and J. Mandziuk, editors, *Challenges for Computational Intelligence*, volume 63 of *Studies in Computational Intelligence*, pages 443–465. Springer, 2007.

[5] M. Campbell, A. J. Hoane Jr., and F.-h. Hsu. Deep Blue. *Artificial Intelligence*, 134(1-2):57–83, 2002.

[6] G. M. J.-B. Chaslot, J.-B. Hoock, J. Perez, A. Rimmel, O. Teytaud, and M. H. M. Winands. Meta Monte-Carlo Tree Search for Automatic Opening Book Generation. In *Proceedings of the IJCAI'09 Workshop on General Intelligence in Game Playing Agents*, pages 7–12, 2009.

[7] G. M. J.-B. Chaslot, M. H. M. Winands, H. J. van den Herik, J. W. H. M. Uiterwijk, and B. Bouzy. Progressive Strategies for Monte-Carlo Tree Search. *New Mathematics and Natural Computation*, 4(3):343–357, 2008.

[8] R. Coulom. Computing Elo Ratings of Move Patterns in the Game of Go. *ICGA Journal*, 30(4):198–208, 2007.

[9] R. Coulom. Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search. volume 4630 of *Lecture Notes in Computer Science*, pages 72–83. Springer, 2007.

[10] C. Donninger and U. Lorenz. Innovative Opening-Book Handling. In H. J. van den Herik, S.-C. Hsu, T.-S. Hsu, and H. H. L. M. Donkers, editors, *Advances in Computer Games, 11th International Conference (ACG 2005)*, volume 4250 of *Lecture Notes in Computer Science*, pages 1–10. Springer, 2006.

[11] P. Drake et al. Orego Go Program, 2011. Available online: http://legacy.lclark.edu/˜drake/Orego.html.

[12] Free Software Foundation. GNU Go 3.8, 2009. Available online: http://www.gnu.org/software/gnugo/, 2009.

[13] S. Gelly and D. Silver. Combining online and offline knowledge in UCT. In Z. Ghahramani, editor, *Proceedings of the Twenty-Fourth International Conference on Machine Learning (ICML 2007)*, volume 227 of *ACM International Conference Proceeding Series*, pages 273–280. ACM, 2007.

[14] S. Gelly, Y. Wang, R. Munos, and O. Teytaud. Modification of UCT with Patterns in Monte-Carlo Go. Technical report, HAL - CCSd - CNRS, 2006.

[15] U. Görtz and B. Shubert. KGS game records, 2011. Available online: http://www.u-go.net/gamerecords/.

[16] R. Greenblatt, D. Eastlake III, and S. D. Crocker. The Greenblatt Chess Program. In *Proceedings of the Fall Joint Computer Conference*, pages 801–810, 1967.

[17] R. M. Hyatt. Book Learning - A Methodology to Tune an Opening Book Automatically. *ICCA Journal*, 22(1):3–12, 1999.

[18] A. Karapetyan and R. J. Lorentz. Generating an Opening Book for Amazons. In H. J. van den Herik, Y. Björnsson, and N. S. Netanyahu, editors, *Computers and Games (CG 2004)*, volume 3846 of *Lecture Notes in Computer Science*, pages 161–174. Springer, 2006.

[19] L. Kocsis and C. Szepesvári. Bandit Based Monte-Carlo Planning. volume 4212 of *Lecture Notes in Computer Science*, pages 282–293. Springer, 2006.

[20] C.-S. Lee, M.-H. Wang, G. M. J.-B. Chaslot, J.-B. Hoock, A. Rimmel, O. Teytaud, S.-R. Tsai, S.-C. Hsu, and T.-P. Hong. The Computational Intelligence of MoGo Revealed in Taiwan's Computer Go Tournaments. *IEEE Transactions on Computational Intelligence and AI in Games*, 1(1):73–89, 2009.

[21] T. R. Lincke. Strategies for the Automatic Construction of Opening Books. In T. A. Marsland and I. Frank, editors, *Computers and Games*, volume 2063 of *Lecture Notes in Computer Science*, pages 74–86. Springer, 2000.

[22] J. Mullins and P. Drake. Using Human knowledge to Improve Opening Strategy in Computer Go. pages 730–734. CSREA Press, 2010.

[23] A. Rimmel, O. Teytaud, C.-S. Lee, S.-J. Yen, M.-H. Wang, and S.-R. Tsai. Current Frontiers in Computer Go. *IEEE Transactions on Computational Intelligence and AI in Games*, 2(4):229–238, 2010.

[24] A. L. Samuel. Some Studies in Machine Learning Using the Game of Checkers. *IBM Journal of Research and Development*, 3(3):211–229, 1959.

[25] J. Schaeffer, R. Lake, P. Lu, and M. Bryant. Chinook: The World Man-Machine Checkers Champion. *AI Magazine*, 17(1):21–29, 1996.