Solving Ponnuki-Go on Small Boards

Erik van der Werf Jos Uiterwijk Jaap van den Herik

Search and Games Group, IKAT, Department of Computer Science,
Universiteit Maastricht, P.O. Box 616, 6200 MD Maastricht
{e.vanderwerf,uiterwijk,herik}@cs.unimaas.nl

Abstract

This paper presents a search-based approach for the game of Ponnuki-Go. A novel evaluation function is presented that is used in an alpha-beta framework with several search enhancements. The search engine performs well on solving positions and on heuristic play. Optimal solutions were found for small empty boards up to 5×5, as well as some 6×6 variants. We believe that our system can also be applied to capture, life/death and connection problems in the game of Go.

1 Introduction

In the last decades, Go has received significant attention from AI research [2, 10]. Yet, despite all efforts, the best computer Go programs are still weak. Exemplary is the fact that the largest square board for which a computer proof has been published is only 4×4 [15]. Results based on human analysis exist for 5×5 and 6×6 but are exceedingly subtle and have not been confirmed by computers [8].

Ponnuki-Go (also known as Atari-Go) is a simplified version of Go that is often used to teach children the first principles of Go. The game is played by two players, black and white, who consecutively place stones of their colour on the intersections of a square grid. Black starts the game. During the game stones remain fixed. Adjacent stones of equal colour are connected, diagonal connections are not used. The goal of the game is to be the first to capture one or more of the opponent's stones. Stones are captured when they are completely surrounded and no longer connected to a free point on the board. Two rules distinguish Ponnuki-Go from Go. First, capturing directly ends the game. The game is won by the side that captured the first stone(s). And second, passing is not allowed (so there is always a winner).

Ponnuki-Go is simpler than Go because there are no ko-fights and sacrifices, and the end is well defined (capture). However, it does contain important aspects of Go such as capturing stones, determining life/death and making territory. From an AI perspective solving small-board Ponnuki-Go is interesting because the perfect play provides an absolute benchmark for testing the performance of learning algorithms. Furthermore, since capturing stones is an essential Go skill, any algorithm that performs well on this task, will also be of interest in computer Go.

In this paper we present a system that plays Ponnuki-Go using a search-based approach. The remainder of the paper is organized as follows: Section 2 presents

the search method, which is based on alpha-beta with several enhancements. Section 3 introduces our evaluation function. Then in section 4 we show the small-board solutions followed by some experimental results on the performance of the search enhancements and of the evaluation function. Section 5 presents some preliminary results on the performance of our program on larger boards. Finally, section 6 provides conclusions and some ideas for future work.

2 The search method

The standard framework for game-tree search is alpha-beta, which comes in many flavours. We selected an iterative deepening Principal Variation Search (PVS) with a minimal window in a negamax framework [9]. The efficiency of alpha-beta search usually improves several orders of magnitude by applying the right search enhancements. We selected the following: (1) transposition tables [11], (2) killer moves [1], (3) history heuristic [13] and (4) enhanced transposition cutoffs [12].

Transposition tables prevent searching the same position several times by storing best move, score, and depth of previously encountered positions. For the transposition tables we use the two-deep replacement scheme [3]. The move ordering is as follows: first the transposition move, then two killer moves, and finally the remainder of the moves are sorted by the history heuristic. Killer moves rely on the assumption that a good move in one branch of the tree is often good at another branch at the same depth. The history heuristic uses a similar idea but is not restricted to the depth at which moves are found. In our implementation the killer moves are stored (and tested) not only at their own depth but also one and two ply deeper. Further, our implementation of the history heuristic employs one table for both black and white moves, thus utilizing the Go proverb "the move of my opponent is my move".

Enhanced transposition cutoffs take extra advantage of the transposition table by looking at all successors of a node to find whether they contain transpositions that lead to a beta cutoff before a deeper search starts. Since enhanced transposition cutoffs are expensive they are only used three or more plies away from the leaves (there the amount of the tree that can be cutoff is sufficiently large).

3 The evaluation function

The evaluation function is an essential ingredient for guiding the search towards strong play. Unlike in chess, no good and especially no cheap evaluation functions exist for Go [2, 10]. Despite of this we tried to build an evaluation function for the game of Ponnuki-Go. The default for solving small games is to use a three-valued evaluation function with values [1 (win), 0 (unknown), -1 (loss)]. Such a three-valued evaluation function is quite efficient for solving games, due to the narrow window which generates many beta cutoffs, but becomes useless for strong play on large boards. Therefore we developed a heuristic evaluation function.

Our heuristic evaluation function is based on four principles: (1) maximizing liberties, (2) maximizing territory, (3) connecting stones, and (4) making eyes.

Naturally these four principles relate in negated form to the opponent's stones. The first principle follows directly from the goal of the game (capturing stones). Since the number of liberties is a lower bound on the number of moves that is needed to capture a stone, maximizing this number is a good defensive strategy whereas minimizing the opponent's liberties directly aims at winning the game. The second principle, maximizing territory, is a long-term goal since it allows one side to place more stones inside its own territory (before filling it completely). The third principle follows from the observation that a small number of large groups is easier to defend than a large number of small groups. Therefore, connecting stones, which strives toward a small number of large groups, is generally a good idea. The fourth principle is directly derived from normal Go, in which eyes are the essential ingredients for building living shapes. In Ponnuki-Go living shapes are only captured after one player has run out of alternative moves and is thus forced to fill his own eyes.

Since the evaluation function is used in tree search, and thus is called at many leaves, speed is essential. Therefore our implementation uses bit-boards for fast computation of the board features. Instead of calculating individual liberties per string, the sum of liberties is directly calculated for the full board. Territory is estimated by a weighted sum of the number of first-, second- and third-order liberties. (Liberties of order n are empty intersections at a Manhattan distance n from the stones). Liberties of higher order are not used since they appeared to slow down the evaluation without a significant contribution to the quality (especially on small boards). Since the exact size of the territory becomes quite meaningless when the difference between both sides is large the value can be clipped. (For solving the small boards we used a maximum difference of 3 points.)

Connections and eyes are more costly features to calculate than the liberties. Fortunately there is a trick that combines an estimate of the two in one cheaply computable number: the Euler number [7]. The Euler number of a binary image, is the number of objects minus the number of holes in those objects. Minimizing the Euler number thus connects stones as well as creates eyes. Since the Euler number can be computed per two rows using a lookup table, only a small number of operations is needed.

4 Experimental results

This section presents results obtained on a Pentium III 1.0 GHz computer, using a transposition table with 2^{25} double entries. We discuss: (1) small board solutions, (2) the impact of search enhancements, and (3) the power of our evaluation function.

4.1 Small board solutions

Our program solved the empty square boards up to 5×5. Table 1 shows the winner, the depth (in plies) of the shortest solution, the number of nodes, and the time (in seconds) needed to find the solution as well as the effective branching

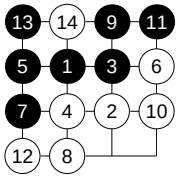

Figure 1: Solution for the 4×4 board.

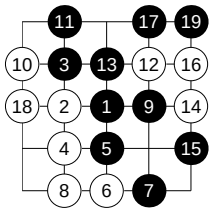

Figure 2: Solution for the 5×5 board.

	2×2	3×3	4×4	5×5	6×6
Winner	W	B	W	B	?
Depth	4	7	14	19	> 23
Nodes	68	1.7×10^3	5.0×10^5	2.4×10^8	$> 10^{12}$
Time (s)	0	0	1	395	$> 10^6$
b_{eff}	2.9	2.9	2.6	2.8	?

Table 1: Solving small empty boards.

factor for each board. In the figures 1 and 2 the principal variations are shown for the solutions of the 4×4 and 5×5 board.

We observed that small square boards with an even number of intersections (2×2 and 4×4) are won by the second player on zugzwang (after a sequence of moves that nearly fills the entire board the first player is forced to weaken his position because passing is not allowed). The boards with an odd number of intersections (3×3 and 5×5) are won by the first player, who uses the initiative to take control of the centre and dominate the board. It is known that in many board games the initiative is a clear advantage when the board is sufficiently large [16]. It is therefore an interesting question whether 6×6 is won by the first or the second player. We ran our search on the empty 6×6 board for a few weeks, until a power failure crashed our machine. The results indicated that the solution is at least 24 ply deep.

Since solving the empty 6×6 board turned out a bit too difficult, we tried making the first few moves by hand. The first four moves are normally played in the centre (for the reason of controlling most territory). Normally this leads to the stable centre of figure 3. An alternative starting position is the crosscut

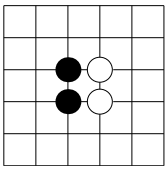

Figure 3: Stable starting position.

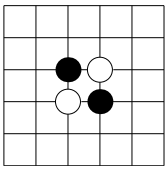

Figure 4: Crosscut starting position.

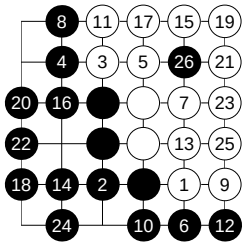

Figure 5: Solution for 6×6 starting with a stable centre.

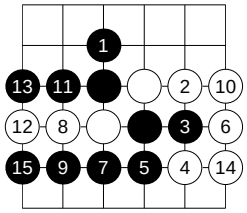

Figure 6: Solution for 6×6 starting with a crosscut.

	Stable	Crosscut
Winner	B	B
Depth	26 (+5)	15 (+4)
Nodes	4.0×10^{11}	1.0×10^{8}
Time (s)	8.3×10^{5}	185
b_{eff}	2.8	3.4

Table 2: Solutions for 6×6 with initial stones in the centre.

shown in figure 4. The crosscut creates an unstable centre with many forcing moves. Though the position is inferior to the stable centre, when reached from the empty board, it is generally considered an interesting starting position for teaching beginners (especially on larger boards).

Recently Cazenave [4] solved Ponnuki-Go on 6×6 starting with a crosscut in the centre. His Gradual Abstract Proof Search (GAPS) algorithm, which is an interesting combination of alpha-beta with a clever threat-extension scheme, proved a win at depth 17 in around 10 minutes. Cazenave concluded that a plain alpha-beta would spend years to solve this problem. We tested our algorithm on the same problem and found the shortest win at depth 15 in a comparable time frame. Figure 6 shows our solution for 6×6 with a crosscut. After implementing our selection of search enhancements into GAPS Cazenave was able to prove the win at depth 15 in 26 seconds on an Athlon 1.7 GHz [5, 6].

Unlike the crosscut, we were not able to find quick solutions for the stable centre (figure 3). (Estimates are that solving this position directly would have required around a month of computation time.) We did however prove that black wins this position by manually playing the first move. The solution is shown in figure 5. The stones without numbers were placed manually, the rest was found by our program. Details of this solution are shown in table 2. A number of alternative starting moves were also tested, all leading to a win for black at the same depth, thus indicating that if the first 4 moves in the centre are correct the solution of the empty 6×6 board is a win in 31 by the first player. This supports the idea that initiative takes over at 6×6.

4.2 The impact of search enhancements

The performance of the search enhancements was measured by comparing the number of nodes searched with all enhancements to that of the search with one enhancement left out, on the task of solving the various board sizes. Results are given in table 3. It is shown that on larger boards, with deeper searches, the enhancements become increasingly effective. The killer moves on the 4×4 board are an exception. The reason may be the relatively deep and narrow path leading to a win for the second player, resulting in a poor generalization of the killers to other parts of the tree.

	3×3	4×4	5×5
Transposition tables	42%	98%	>99%
Killer moves	19%	-6%	81%
History heuristic	6%	29%	86%
Enhanced Transposition Cutoffs	0%	6%	28%

Table 3: Reduction of nodes by the search enhancements.

4.3 The power of our evaluation function

The evaluation function was compared to the standard three-valued approach for solving small trees. Usually an evaluation function with a minimal range of values generates a large number of beta-cutoffs, and is therefore more efficient for solving small problems than the more fine-grained heuristic approaches that are needed to play on larger boards. In contrast, the results given in table 4 indicate that our heuristic evaluation function outperforms the minimal approach for solving Ponnuki-Go. The reason probably lies in the move ordering of which efficiency increases with the information provided by our evaluation function.

Table 4 further shows that our heuristic evaluation function is quite fast. Averaged over all nodes it requires only around 4% more time than the three-valued approach (which is always calculated). Even if we take into account that roughly 70% of all nodes are actually not directly evaluated (due to the fact that they represent illegal positions, final positions, transpositions, or are just internal nodes) this still amounts to a pure evaluation speed of roughly 5,000,000 nodes per second. Comparing this to the over-all speed of about 600,000 nodes per second indicates that there is still significant room for adding knowledge to the evaluation function.

	3×3		4×4		5×5	
Evaluation function	nodes	time	nodes	time	nodes	time (s)
Heuristic	1.7×10^3	0	5.0×10^5	1	2.4×10^8	395
Win/unknown/loss	1.7×10^3	0	8.0×10^5	1	6.1×10^8	968

Table 4: Performance of the evaluation function.

5 Performance on larger boards

We tested our program against Rainer Schütze's freeware program "AtariGo 1.0" [14]. This program plays on the 10×10 board with a choice of three initial starting positions, of which one is the crosscut in the centre. Our program was able to win most games, but occasionally lost when stones were trapped in a ladder. The reason for the loss was that our program used a fixed depth. It did not include any means of extending ladders (which is not essential for solving the small boards). After making an ad-hoc implementation to extend simple ladders our program convincingly won all games against "AtariGo 1.0".

We tested our program against some human players too (on the empty 9×9 board). In close combat it was sometimes able to defeat reasonably strong amateur Go players, including a retired Chinese first dan. Despite of this, most stronger players were able to win easily by playing quiet territorial games.

6 Conclusions and future work

We solved Ponnuki-Go on the 3×3, 4×4, 5×5 and some non-empty 6×6 boards. These results were obtained by a combination of standard search enhancements together with a novel evaluation function.

Cazenave and our group both solved 6×6 with a crosscut using different techniques. Combining our selection of search enhancements with Cazenave's GAPS can improve the performance even further. The next challenges in Ponnuki-Go are: solving the empty 6×6 board and solving the 8×8 board starting with a crosscut in the centre.

Future work

In the experiments it became evident that search extensions for ladders are essential for strong play on the larger boards. Future work will therefore focus on selective search-extensions.

Since capturing stones is an important sub-goal in the game of Go, we will test our search and evaluation function in a full Go-playing program. We expect that good results can be obtained for capture, life/death and connection problems.

References

[1] S.G. Akl and M.M. Newborn. The principal continuation and the killer heuristic. In *1977 ACM Annual Conference Proceedings*, pages 466–473. ACM, Seattle, 1977.

[2] B. Bouzy and T. Cazenave. Computer go: An AI oriented survey. *Artificial Intelligence*, 132(1):39–102, October 2001.

[3] D.M. Breuker, J.W.H.M. Uiterwijk, and H.J. van den Herik. Replacement schemes and two-level tables. *ICCA Journal*, 19(3):175–180, 1996.

[4] T. Cazenave. La recherche abstraite graduelle de preuve. 13ème Congrès Francophone AFRIF-AFIA de Reconnaissance des Formes et Intelligence Artificielle, 8 - 10 Janvier 2002, Centre des Congrès d'Angers. Paper available at http://www.ai.univ-paris8.fr/~cazenave/AGPS-RFIA.pdf.

[5] T. Cazenave, 2002. Personal communication.

[6] T. Cazenave. Gradual abstract proof search. *ICGA Journal*, 25(1):3–16, 2002.

[7] S.B. Gray. Local properties of binary images in two dimensions. *IEEE Transactions on Computers*, C-20(5):551–561, 1971.

[8] H.J. van den Herik, J.W.H.M. Uiterwijk, and J. van Rijswijck. Games solved: Now and in the future. *Artificial Intelligence*, 134(1-2):277–311, January 2002.

[9] T.A. Marsland. A review of game-tree pruning. *ICCA Journal*, 9(1):3–19, 1986.

[10] M. Müller. Computer go. *Artificial Intelligence*, 134(1-2):145–179, January 2002.

[11] H.L. Nelson. Hash tables in Cray Blitz. *ICCA Journal*, 8(1):3–13, 1985.

[12] A. Plaat, J. Schaeffer, W. Pijls, and A. de Bruin. Exploiting graph properties of game trees. In *Proceedings of the Thirteenth National Conference on Articial Intelligence (AAAI'96)*, volume 1, pages 234–239, 1996.

[13] J. Schaeffer. The history heuristic. *ICCA Journal*, 6(3):16–19, 1983.

[14] R. Schütze. Atarigo 1.0, 1998. Free to download at the site of the DGoB (Deutscher Go-Bund): http://www.dgob.de/down/index_down.htm.

[15] S. Sei and T. Kawashima. A solution of go on 4x4 board by game tree search program, fujitsu social science laboratory. The 4th Game Informatics Group Meeting in IPS Japan, pages 69-76 (in Japanese), translation available at http://homepage1.nifty.com/Ike/katsunari/paper/4x4e.txt, 2000.

[16] J.W.H.M. Uiterwijk and H.J. van den Herik. The advantage of the initiative. *Information Sciences*, 122(1):43–58, 2000.