A Phantom-Go Program

Tristan Cazenave

Labo IA,
Université Paris 8, St-Denis, France
cazenave@ai.univ-paris8.fr

Abstract. This paper discusses the intricacies of a Phantom-Go program. It is based on a Monte-Carlo approach. The program called Illusion plays Phantom Go at an intermediate level. The emphasis is on strategies, tactical search, and specialized knowledge. The paper provides a better understanding of the fundamentals of Monte-Carlo search in Go.

1 Introduction

Phantom Go is a variant of Go in which part of the information is hidden for the players. Since Phantom Go is therefore to be considered as a game with imperfect information[1], Monte-Carlo methods are expected to work well. This paper demonstrates that the expectation is true and shows the results of a Monte-Carlo based program that plays Phantom Go.

In Sect. 2 we present the game of Phantom Go. In Sect. 3 we recall previous work on Monte-Carlo Go. In Sect. 4 we detail how the Monte-Carlo method is adapted to Phantom Go. In Sect. 5 we give experimental results of our program Illusion. Section 6 provides a conclusion and outlines future work.

2 Phantom Go

Phantom Go is a two-player game. There are two players and a referee. It is played on three boards, one for each player and one for the referee. The board of the referee is called the reference board. It is usually played on 9×9 boards. The referee can see all three boards. Each player can only see his[2] own board. When it is a player's turn, he chooses a move and asks the referee if it is legal for him to play on the intersection intended by pointing at the intersection. The referee answers 'legal move' or 'illegal move' according to the reference board. If the move is illegal, the player chooses another move, and so on until he arrives at a legal move. When the player has indicated a legal move, it must be played on the reference board by the referee, and by the player on his own board. If

[1] Sometimes the term 'incomplete information' is also used for games with hidden information. In the case of Phantom Go, both players have complete information of the game rules, possible states, and possible outcomes.

[2] For brevity we use 'he' ('his') if 'he or she' ('his or her') is meant.

there is a capture, the referee announces the number of stones being captured and communicates to the other player which stones have been captured. After a move has been played, it is the other player's turn. The game ends when both players pass. Phantom Go (as a variant of Go) is the equivalent to Kriegspiel in Chess [2].

3 Monte-Carlo Go

Monte-Carlo methods compute statistics on a set of random games in order to find the best move. They have been used in games such as Bridge [7], Poker [1], Tarok [8], and Scrabble [9]. All these games have hidden information which make them particularly suited for Monte Carlo. However, Monte-Carlo methods have also proven to be useful in complete information games, and particularly in Go. Brügmann [5] was the first researcher to experiment with Monte Carlo in Go. Recently, other Go programs have started using it, and improved the method in many directions, among others by (1) simplifying the method and proposing basic improvements [4], (2) combining it with a knowledge-based program that selects a few number of moves that are evaluated by the Monte-Carlo method [3], and (3) by combining it with tactical search [6].

There are several slightly different ways to write a Monte-Carlo Go program [4]. In this paper, we use for Monte-Carlo Go the following algorithm: the program plays a large number (usually 1,000 to 10,000) of random games starting at the current position. The moves of the random games are chosen almost randomly among the legal moves, except that they must not fill the player's eyes. A player passes in a random game when his only legal moves are on his own eyes. The game ends when both players pass. In the end of each random game, the score of the game is computed using Chinese rules (in our case, it consists in counting one point for each stone and each eye of the player's color, and subtracting the opponent count from the player's count). The program computes for each intersection the mean results of (1) the random games in which the player starts with a move at that intersection, and (2) the random games in which the opponent starts at that intersection. The value of a move is the difference between the two means. The program plays the move with the highest value.

4 Monte-Carlo Phantom Go

Monte-Carlo Go is a game of complete information. Monte-Carlo Phantom Go has to deal with hidden information. In order to cope with this hidden information, the program has to guess where the stones of the opponent are, and the best move on average against different configurations of the opponent stones.

In essence, for Monte-Carlo Phantom Go the basic Monte-Carlo Go method is reused: the program plays many games randomly with the constraint of not filling its own eyes. However, all the random games do not start with the same position, as the program does not know exactly the real position. The program memorizes all the forbidden moves. It places an opponent stone on each of the forbidden moves. The program also knows the number of the opponent stones that are present on the reference board. Subtracting the number of forbidden moves from the number of opponent stones gives the number of stones with an unknown position.

From the beginning of each random game, the program places its own stones and places opponent stones on the forbidden intersections. Moreover, it randomly places the opponent stones on the empty intersections left. More precisely, it randomly places as many opponent stones as there are opponent stones with an unknown position. Once all the stones are placed, it plays a random game starting with a move of its color, and performs moves randomly on empty intersections, provided they are not a player's eye.

A player passes when his only moves left are his own eyes. When both players pass, the game is ended. At the end of a random game, the score is computed using Chinese rules. For each move played during the random game, the mean score of playing this move for all the random games is updated to take into account the result of the game. When 10,000 random games have been played, the program subtracts, for all the legal moves, the mean score of the move for the opponent's color from the mean score of the move for the player's color. The move that has the highest difference is tried.

If the move is announced to be illegal by the referee, the program memorizes it as a forbidden move, and starts its process again. It plays 10,000 new random games, taking into account the new information given by the referee on the forbidden move.

5 Experimental Results

The number of random games played at each move is set to 10,000. The program plays a move in a few seconds on a Pentium 3.0 GHz. Subsection 5.1 details an example game by Illusion. Subsection 5.2 gives results against different Go players.

5.1 An Example Game

The author is an European one-dan Go player. Playing games with the program Illusion usually results in a small win by the author. An example 9×9 game is given in Fig. 1. When a player tries an illegal move, it is reported in the game's notation, and therefore multiple moves by the same player follow each other. All but the last move by the same player are illegal, they are mentioned explicitly because they give information on the knowledge of the game by the player and are required to analyze and understand the game. The author is White and the program is Black.

In this game, the author followed the strategy of (1) dividing the board into two parts, and (2) trying to kill one side of the board after it has been divided. This strategy is also used by other experienced Phantom-Go players and admittedly it is quite efficient.

1 B(D4), W(E5); 2 B(E3), W(E6); 3 B(F5), W(E4); 4 B(E6-F6), W(E7); 5 B(F7), W(E3-D4-F4); 6 B(E5-F4-E4-G4), W(F3); 7 B(F3-E7-E8), W(F2); 8 B(G3), W(F1); 9 B(F2-D8), W(E8-D7); 10 B(F8), W(D8-C7); 11 B(G2), W(C8); 12 B(G5), W(C9); 13 B(D3), W(F7-D3-B7); 14 B(C7-D7-C8-F1-C2), W(A7); 15 B(B4), W(G4-G3-G2-B6); 16 B(B6-B3), W(B5); 17 B(C9-B5-E2), W(B4-A5); 18 B(C5), W(A4); 19 B(B8), W(A3); 20 B(B7-H6), W(B3-A2); 21 B(E1), W(B2); 22 B(A8), W(C2-B1); 23 B(G1), W(C1); 24 B(D5), W(D1); 25 B(A5-A7-H7), W(F6-F5-D9); 26 B(C3), W(E9); 27 B(B9), W(F8-F9); 28 B(G8), W(G9); 29 B(A4-A3-A2-C1-D1-B1-B2-D9-E9-F9-G9-H9), W(G8-H9-H8); 30 B(H8-J8), W(J9-J8-J7); 31 B(J7-H5), W(H8-G1-H7-H6-Pass); 32 B(J6).

Fig. 1. Example game: Illusion (B) – Cazenave (W)

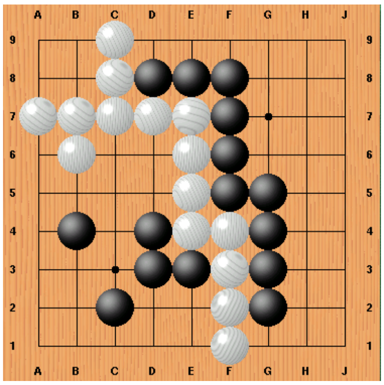

Fig. 2. The reference board after 15 moves of both sides

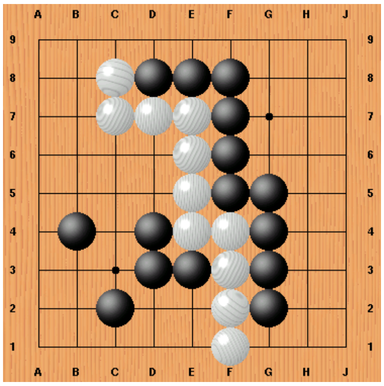

Fig. 3. The program board after 15 moves of both sides

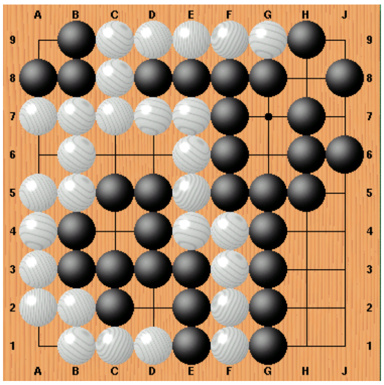

Fig. 4. The final position

Figure 2 gives the reference board after the first 15 moves (30 plies) of the game. Figure 3 gives the program board after the first 15 moves of the game. In the latter position, we see that Illusion has guessed most of the white stones, and is ahead of White. It has to close the borders of its right group, and to make two eyes with its left group. Figure 4 gives the reference board at the end of the game. The program failed to make its left group live and lost the game.

5.2 Results Against Go Players

In order to test the program Illusion adequately, it played 9×9 games against Go players of different levels. The results are in Table 1. For each game we give the level of the human player, the division of colors, and the result.

Table 1. Results of Illusion against Go players

Level	White	Black	Result
13 kyu	Nicolas	Illusion	W+8.5
13 kyu	Illusion	Nicolas	W+resign
5 dan	Bernard	Illusion	W+13.5
13 kyu	Illusion	Arpad	W+17.5
5 dan	Illusion	Bernard	W+27.5
5 dan	Bernard	Illusion	W+47.5

As the results show, the program can win by 27.5 points against a 5 dan Go player as well as lose by 8.5 points against a 13 kyu Go player. We may conclude that the program has the level of experienced Go players who only played a few number of Phantom-Go games. We do not know experienced (and even less ranked) Phantom-Go players to whom we could test the program.

6 Conclusion and Future Work

We presented our Phantom-Go program Illusion based on a Monte-Carlo approach. Illusion plays interesting Phantom-Go games. The peculiarity of the application of the Monte-Carlo method to Phantom Go is that unknown stones are placed at random at the beginning of each random game. From the results so far we may conclude that strategy, tactical search, and specialized knowledge plays an important role.

Below we suggest three essential improvements of the current program. A first improvement is to deal more accurately with the well-known Phantom-Go strategies in the random games. For example, the divide-and-kill strategy used by the author can be exploited in the random games to bias the move selection.

A second set of improvements are the improvements used in Monte-Carlo Go programs. For example, it is possible to combine the current search with a tactical search by computing the results of simple tactical searches at the beginning of the random games, so as to compute statistics in the random games on the goal to be achieved. Once the statistics on the goals are computed they give a better evaluation of the corresponding move than the basic statistics on arbitrary moves. Such an approach has worked in Monte-Carlo Go [6], and could well work too in Phantom Go.

A third improvement is to use patterns to bias the selection of moves in the random games so as to improve their quality as in [3]. Finally, we remark that the program also has problems dealing with semeais, and opponent eyes. This can be improved by specialized knowledge.

References

1. D. Billings, A. Davidson, J. Schaeffer, and D. Szafron. The Challenge of Poker. *Artificial Intelligence*, 134(1-2):210–240, 2002.
2. A. Bolognesi and P. Ciancarini. Computer Programming of Kriegspiel Endings: The Case of KR Versus K. In *Advances in Computer Games 10 (ACG10) Many Games, Many Challenges* (eds. H.J. van den Herik, H. Iida, and E.A. Heinz), pages 325–342, Kluwer Academic Publishers, Boston, 2004.
3. B. Bouzy. Associating Domain-Dependent Knowledge and Monte Carlo Approaches within a Go Program. In *Joint Conference on Information Sciences*, pages 505–508, Cary, 2003.
4. B. Bouzy and B. Helmstetter. Monte Carlo Go Developments. In *Advances in Computer Games 10 (ACG10) Many Games, Many Challenges* (eds. H.J. van den Herik, H. Iida, and E.A. Heinz), pages 159–174, Kluwer Academic Publishers, Boston, 2004.
5. B. Brügmann. Monte Carlo Go. ftp://ftp-igs.joyjoy.net/go/computer/mcgo.tex.z, 1993.
6. T. Cazenave and B. Helmstetter. Combining Tactical Search and Monte-Carlo in the Game of Go. In *IEEE Symposium on Computational Intelligence and Games (CIG'05)* (eds. G. Kendall and S. Lucas), Colchester, UK, 2005.
7. M.L. Ginsberg. GIB: Steps Toward an Expert-Level Bridge-Playing Program. In *IJCAI-99*, pages 584–589, Stockholm, Sweden, 1999.
8. M. Lustrek, M. Gams, and I. Bratko. A Program for Playing Tarok. *ICGA Journal*, 26(3):190–197, 2003.
9. B. Sheppard. Efficient Control of Selective Simulations. *ICGA Journal*, 27(2):67–80, 2004.