Golois Wins Phantom Go Tournament

Tristan Cazenave [1] *Joris Borsboom* [2]

1. PHANTOM GO

Phantom Go is a two-player game often played as an entertaining variation by regular Go players. It is the equivalent of Kriegspiel for Chess. Phantom Go is a variant of classic Go in which the players do not see the opponent's moves. Two players each have their own board, on which only their own stones are visible. A referee is needed. He has a reference board on which the actual state of the game is maintained. The players communicate their move decisions to the referee. There are a few variations to the rules of Phantom Go, in the rules used for the tournament, the referee has the following replies to a suggested move:

- Illegal (already occupied, suicide, ko); after which a new move may be suggested.
- Legal; the move is played. If stones are taken, they are removed from the boards.
- Stones taken; the surrounded stones are taken away on the reference board, and the positions of these stones are announced to the players

After both players pass, the game ends and the final score is computed on the reference board, according to Chinese rules. A part of the enjoyment of looking at Phantom Go games comes from the fact that spectators see the reference board when the players do not.

Monte-Carlo methods can produce Phantom Go players competitive with humans (Cazenave, 2006). The principle is to randomly place unknown opponent stones at the beginning of each playout.

2. THE TOURNAMENT

A Phantom Go tournament was held at the 2007 Computer Olympiad in Amsterdam. Two programs registered for the tournament. The first one, InTheDark (Borsboom, 2007a; Borsboom *et al.*, 2007b), is written by Joris Borsboom and is based on Monte-Carlo methods. The version used at the tournament used both all-as-first sampling and the UCT algorithm (Kocsis and Szepesvári, 2006) in combination with a heuristic for guessing unknown opponent stones. It tried to maximize its score. Scores were determined by random playouts. Joris wrote a referee program and used it to compare multiple Monte-Carlo algorithms.

The other one, Golois, is written by Tristan Cazenave and is also based on Monte-Carlo methods. It uses the percentage of wins as its evaluation function, and it performs a depth one search with Monte-Carlo evaluation. This version of Golois plays at equal strength with a one kyu player. Golois uses completely random playouts. Knowledge based playouts using Mogo's patterns (Wang and Gelly, 2007) and Crazy Stone urgencies (Coulom, 2006) result in weaker play due to overestimation of the position when few opponent stones are discovered. A possible solution to address this problem could be to put opponent stones less randomly at the beginning of each playout.

InTheDark ran on a single CPU and could simulate 80,000 playouts from the empty board, a number which decreased during the game. Golois ran on a 28 CPUs cluster and could simulate 300,000 playouts from the empty board. However the difference between the two programs does not only come from computing power. In

[1] L.I.A.S.D. Université Paris 8, Saint-Denis, France, email:cazenave@ai.univ-paris8.fr
[2] MICC-IKAT, Universiteit Maastricht, Maastricht, The Netherlands, email:J.Borsboom@student.unimaas.nl

the first two games, Golois had its non deterministic option switched off, therefore all the parallel programs where playing the same playouts, which was equivalent to have only one process and much less playouts, nevertheless Golois won these games more easily than some of the games at full computing power.

Six games were played. Golois won the tournament 6-0. InTheDark had winning position in some of the games, but as it used territory scoring it did not secure a win in these positions. Whereas Golois, using the percentage of wins as its evaluation function, successfully played risky moves when it was behind and caught up, and secured the win when it was ahead.

Some of the games in the tournament are described and commented in Borsboom (2007a). A case that deserves some special attention is the possibility of an illegal move because of ko or suicide. Both programs suffered from this, as they both handled the reply 'illegal' by placing an opponent stone at that position. This has a large impact on the evaluation value of moves since the Monte-Carlo algorithm starts with a position that is incorrect. As a result of this, mistakes were made about the life and death status of groups.

3. CONCLUSION

The combination of a difficult game like Go with uncertainty about the opponent's position presents an interesting research domain. While Monte-Carlo methods seem well suited to deal with the problems of this domain, how exactly to account for the uncertainty in board position and produce good play is not an easily answered question.

The first Computer Olympiad tournament on Phantom Go was an interesting experiment for both of the authors. We hope to do do more research on Phantom Go playing programs, and would welcome new entrants to a future competition.

4. REFERENCES

Borsboom, J. (2007a). Phantom Go. Master thesis, Universiteit Maastricht MICC-IKAT.

Borsboom, J., Saito, J.-T., Chaslot, G., and Uiterwijk, J. W. (2007b). A comparison of monte-carlo methods for phantom go. *BNAIC 2007*, Utrecht, The Netherlands.

Cazenave, T. (2006). A Phantom-Go program. *Advances in Computer Games 2005*, Vol. 4250 of *Lecture Notes in Computer Science*, pp. 120–125, Springer.

Coulom, R. (2006). Efficient Selectivity and Back-up Operators in Monte-Carlo Tree Search. *Computers and Games 2006*, Volume 4630 of LNCS, pp. 72–83, Springer, Torino, Italy.

Kocsis, L. and Szepesvári, C. (2006). Bandit based monte-carlo planning. *ECML*, Vol. 4212 of *Lecture Notes in Computer Science*, pp. 282–293, Springer.

Wang, Y. and Gelly, S. (2007). Modifications of UCT and sequence-like simulations for Monte-Carlo Go. *CIG 2007*, pp. 175–182, Honolulu, USA.