Monte-Carlo Tree Search in Crazy Stone

Rémi Coulom*

Abstract

Monte-Carlo tree search has recently revolutionized Go programming. Even on the large 19x19 board, the strongest Monte-Carlo programs are now stronger than the strongest classical programs. This paper is a summary of the principle of Monte-Carlo tree search, and a description of Crazy Stone, one of those new artificial players.

1 Introduction

The classical approach to game programming is based on alpha-beta tree search, combined with heuristic evaluation at the leaves. This approach has been very successful for games such as chess, but it is difficult to apply it to the game of Go, because of its high branching factor, and the difficulty to build a fast accurate evaluation function.

Instead of building an heuristic evaluation function, another possible approach to evaluating a position consists in averaging the outcome of several random continuation. This is the fundamental idea of Monte-Carlo search, that will be described in this paper.

2 History: The Monte-Carlo Revolution

This section presents a very brief time line of some of the most significant events in the development of Monte-Carlo programs.

1993 Bernd Brügmann [2] pioneers the application of Monte-Carlo ideas to the game of Go.

2003 Bernard Helmstetter, Bruno Bouzy, and Tristan Cazenave revive the idea [1]. They are later joined in their effort by Guillaume Chaslot (in 2004) and Rémi Coulom (in 2005).

2006 Crazy Stone wins gold at the Computer Olympiad [4] on the 9×9 board.

2007 Mogo [7, 6] and Crazy Stone [5] win gold and silver at the Computer Olympiad on the 19×19 board. Steenvreter, another Monte-Carlo program by Erik van der Werf wins gold ahead of Mogo and Crazy Stone on 9×9.

2007 Crazy Stone beats KCC Igo (winner of the Gifu challenge, four times in a row from 2003 to 2006) in a match, with a score of 15 wins and 4 losses, thus confirming the strength of Monte-Carlo programs on the 19×19 board[1].

3 Crazy Stone's Algorithm

Crazy Stone performs a global Monte-Carlo tree search of the position, with the UCT algorithm. Patterns are used to improve the quality of random simulations, and to prune bad moves in the search tree, with progressive widening. Details of these method can be found in other papers [4, 5, 8].

4 Playing Style

The playing style of Monte-Carlo programs is very different from the playing style of classical programs. This section summarizes some of the characteristics of the playing style of Crazy Stone, and tries to highlight the strengths and weakness of the current Monte-Carlo algorithms.

The most striking aspect of the playing style of Monte-Carlo programs is that they maximize their

*Université Charles de Gaulle, INRIA Sequel, CNRS GRAPPA, Lille, France

[1] This match was played on KGS and organized by Hiroshi Yamashita. Game records are available in the KGS archives of KCConGUI and CrazyStone.

probability of winning, not their amount of territory. As a consequence, they play very safe moves when they are ahead, and try very aggressive and dangerous moves when they are behind. Very few of the classical programs know how to do it that well. This aspect of Monte-Carlo programs is one of their major strengths.

Another strength of Monte-Carlo programs is their ability to have a global understanding of the position. This helps them a lot against classical Go programs who perform local searches focused on just one goal. Monte-Carlo programs can read fights that involve several high-level goals at the same time.

A major weakness of Crazy Stone is its frequent inability to determine correctly the life-or-death status of a group. Sometimes random playouts will incorrectly estimate that a dead group is alive, or, conversely, that a living group is dead. In some situations, search is not enough to compensate for the ignorance of playouts, and it leads to huge blunders.

5 Future of Monte-Carlo Search

Monte-Carlo search has already brought a lot of novelties to computer Go, but it is likely that the strength of the current strongest Monte-Carlo programs can be improved further. First, the addition of more knowledge in random playouts and selectivity algorithms should allow significant progress. Also, a more ambitious plan for further research would be to cure the weakness of life-and-death evaluation described in the previous section, by adaptively changing the playout policy. The current tree-search approach adapts the playout policy only near the root. The next breakthrough in Monte-Carlo search may come from algorithms that adapt the policy also far from the root. According to some discussions in the computer-go mailing list, some programmers are currently researching this direction.

Beyond computer Go, one may expect that techniques developed here could be applied to other domains. Other games, such has the game of Hex, have features that make them similar to Go, so they may be able to benefit from the same techniques. More generally, Monte-Carlo search can be applied to other planning and optimization problems [3].

References

[1] Bruno Bouzy and Bernard Helmstetter. Monte Carlo Go developments. In H. J. van den Herik, H. Iida, and E. A. Heinz, editors, *Proceedings of the 10th Advances in Computer Games Conference*, Graz, 2003.

[2] Bernd Brügmann. Monte Carlo Go, 1993. Unpublished technical report.

[3] Guillaume Chaslot, Steven De Jong, Jahn-Takeshi Saito, and Jos W. H. M. Uiterwijk. Monte-Carlo tree search in production management problems. In Pierre-Yves Schobbens, Wim Vanhoof, and Gabriel Schwanen, editors, *Proceedings of the 18th BeNeLux Conference on Artificial Intelligence*, pages 91–98, Namur, Belgium, 2006.

[4] Rémi Coulom. Efficient selectivity and backup operators in Monte-Carlo tree search. In P. Ciancarini and H. J. van den Herik, editors, *Proceedings of the 5th International Conference on Computer and Games*, Turin, Italy, 2006.

[5] Rémi Coulom. Computing Elo ratings of move patterns in the game of Go. In H. Jaap van den Herik, Mark Winands, Jos Uiterwijk, and Maarten Schadd, editors, *Proceedings of the Computer Games Workshop*, Amsterdam, The Netherlands, June 2007.

[6] Sylvain Gelly and David Silver. Combining online and offline knowledge in UCT. In *Proceedings of the 24th International Conference on Machine Learning*, pages 273–280, Corvallis Oregon USA, 2007.

[7] Sylvain Gelly, Yizao Wang, Rémi Munos, and Olivier Teytaud. Modification of UCT with patterns in Monte-Carlo Go. Technical Report RR-6062, INRIA, 2006.

[8] Levente Kocsis and Csaba Szepesvári. Bandit-based Monte-Carlo planning. In J. Fürnkranz, T. Scheffer, and M. Spiliopoulou, editors, *Proceedings of the 15th European Conference on Machine Learning*, Berlin, Germany, 2006.