MONTE-CARLO GALORE!

Computer and Games
6th International Conference, CG 2008
Beijing, China, September/October 2008
Edited by H. Jaap van den Herik, Xinhe Xu, Zongmin Ma & Mark H.M. Winands
Lecture Notes in Computer Science (LNCS) 5131
Springer, Berlin Heidelberg New York, 2008
ISSN 0302-9743; ISBN-13 978-3-540-87607-6

Reviewed by Dap Hartmann

After reading the first half dozen papers in the conference proceedings of the 6th Computers and Games Conference (CG 2008), I looked at the cover in search of the common theme for this conference. Surely it had to be Monte-Carlo simulations, because each paper addressed that subject. But alas, there was no theme or special topic. Maybe it is just a new phase in game-playing computers? After decades of computer games research which were dominated by tree searching and evaluation functions, it seems the time has come to add Monte-Carlo simulations to the repertoire of techniques to compute the best move in various games. Oh wait, ‘various games’? Maybe that is the more likely explanation. Computer games used to be mostly chess, checkers and draughts, and the ICGA used to be the ICCA. Games with a sufficiently modest branching factor can resort to full-width tree searching to reasonable depths. Add algorithms, heuristics and enhancements to prune most of the branches, top it off with a solid quiescence search and a static evaluation function, and you have yourself a decent competitive program. These programs got stronger and stronger until one day the strongest human chess player was beaten by a computer. And the strongest human checkers player passed away before his inevitable defeat by a computer program. Although a lot of very interesting research still remains in the field of computer chess, checkers and draughts, the main driver that fueled most of the research (beating the best human player) has disappeared. Mission accomplished, let us move on to the next challenge.

Enter games of higher complexity and with branching factors prohibitively large for using a traditional tree search, such as Go (361 opening moves) and Amazons (more than 2000 possible first moves). The similarities are obvious and the cross-pollination has started, as can be witnessed in these proceedings. In ‘Amazons Discover Monte-Carlo’, Richard Lorentz confirms this trend: “Monte-Carlo algorithms and their UCT-like successors have recently shown remarkable promise for Go-playing programs. We apply some of these same algorithms to an Amazons-playing program.” Although a pure MC/UCT algorithm does not seem to do the trick, Lorentz shows that a hybrid between MC/UCT and conventional minimax is superior to a pure minimax program. What works for one game may also work for another game, Winands, Björnsson and Saito point out in their paper ‘Monte-Carlo Tree Search Solver’: “Recently, Monte-Carlo Tree Search (MCTS) has advanced the field of computer Go substantially. In this article we investigate the application of MCTS for the game of Lines of Action (LOA).” The authors describe MCTS-Solver, a new MCTS variant, and show that it outperforms traditional MCTS.

While moving on and reading the next abstract, in Nathan Sturtevan’s ‘An Analysis of UCT in Multi-player Games’, it is not surprising that I suspected a common theme for this conference, and maybe even a template abstract that was supplied to every author: “The UCT algorithm has been exceedingly popular in Go, a two-player game, significantly increasing the playing strength of Go in a very short time. This paper provides an analysis of the UCT algorithm in multi-player games”. If MC and/or UCT works in *your* complex game (Go), it might also work in *my* complex game (Amazons, LOA). And if it works in *your* two-player game, it might also work in *my* multi-player games. Sturtevan analyses the application of UCT to 3-player Chinese Checkers and two card games (Spades and Hearts). Although no firm conclusions are presented (“These results are promising and suggest that UCT has the potential to richly benefit multi-player game-playing programs”) the tentative outcome of the experiments was that UCT performs at least as good as existing algorithms.

All bases were covered in this conference, and so there is also a paper on applying Monte-Carlo simulations to a one-player game. ‘Single-Player Monte-Carlo Tree Search’ by Schadd, Winands and Van den Herik is the opening paper of this book. Previously, single-player MCTS has only been applied to Sailing Domain, a one-player game that involves uncertainty. In this paper, MCTS is applied to SameGame, a one-person game with perfect information. The authors propose to refer to one-person games with perfect information as puzzles. SameGame appears deceptively simple. Basically, there is only one rule that governs which moves are allowed. A few more rules describe the scoring of points. The average branching factor is 20.7 and the average game length is 64.4 moves. Because the initial board configuration in SameGame is generated at random, the MCTS implementation was tested on a standardized set of 20 positions. The authors are happy to report that they achieved the highest score to date, using identical time restrictions (2-3 hours per position).

And when Monte-Carlo, which did miracles for Go, is beneficial to other games as well, including one-player and multi-player games, the next logical extension is… to apply it to multi-player Go! That is what Tristan Cazenave has done, and his contribution is simply titled 'Multi-player Go'. This variation of Go is sometimes played for fun at tournaments or at clubs. Cazenave focusses on the three-player version where Black, White and Red take turns playing their moves. The UCT implementation includes 'Coalition of Players' in which two or more players collaborate, and 'Paranoid UCT' where the coalition includes all but one player. It reminded me of a famous quote from the immortal BBC television series *Yes Minister*, in which the Minister of Administrative Affairs Jim Hacker (played by Paul Eddington) complains: "You'd be paranoid too, if everyone was plotting against you".

The next two papers are also about Monte-Carlo algorithms. In 'Parallel Monte-Carlo Tree Search', Chaslot, Winands, and Van den Herik describe three ways to parallelize MCTS: leaf parallelization, root parallelization, and tree parallelization. Experiments on 13x13 Go indicate that root parallelization gives the best results. However, in 9x9 Go tree parallelization showed promising results. Parallel Monte-Carlo is also the topic of the paper 'A Parallel Monte-Carlo Tree Search Algorithm' by Cazenave and Jouandeau. On a network of computers, the authors tested various configurations of the number of playouts (12,500 to 100,000), slaves (1-64) and computers (1-16). One of the conclusions is that 16 is the optimum number of slaves, achieving a speedup of a factor 14. One more paper deals exclusively with Monte-Carlo. In 'A Fast Indexing Method for Monte-Carlo Go', Chen, Du and Zhang describe how a building and using a 3x3 pattern library improved the performance of GO INTELLECT against GNU GO by 7.5% in 9x9 games.

So eight out of the 24 papers in these proceedings focus on Monte-Carlo Tree Search and/or UCT, and nine papers are mainly about Go. There is quite a bit of overlap between the two. Clearly, Monte-Carlo and Go are hot. Go is the new big challenge, and Monte-Carlo has helped computer Go to make a big leap forwards in performance. Still, there are plenty contributions in this book that are not about Go of Monte-Carlo. Two that I found particularly interesting are on proof-number search and on cognitive theory and perception.

In 'A New Proof-Number Calculation Technique for Proof-Number Search', Kazuki Yoshizoe explores a new technique for calculating proof numbers in Proof-Number Search. The idea is to sum the disproof number of only the best group of children in OR nodes. For AND nodes the algorithm is identical to existing methods. Yoshizoe tested this new implementation on a set of 434 capturing problems in 19x19 Go. He concludes that "this method is effective for capturing problems of Go if all legal moves are searched." It is about 30 times faster than normal Depth-First Proof-Number Search (df-pn). However, it is four times slower than df-pn with heuristic pruning, but the correctness of the search is guaranteed.

The ever productive Reijer Grimbergen wrote an interesting paper entitled "Cognitive Modeling of Knowledge-Guided Information Acquisition in Games". He uses the theory that Marvin Minsky outlined in his famous 1988 book *The Society of Mind* to explain the relationship between chunks and problem-solving tasks. The concept of chunks, of course, was already recognized by A.D. de Groot in his 1946 PhD thesis, later published as *Thought and Choice in Chess*. It is interesting to know that Herb Simon learned Dutch to be able to understand De Groot's PhD thesis. Together with Bill Chase, Simon refined the concept of chunking to explain the difference between beginners and expert players: the latter have a greater capability to chunk the problem space.

As a first step towards this framework, Grimbergen used a classic reproduction experiment in Shogi (while arguing that any board game will do – Shogi is just his preferred drosophila). The subjects were shown Shogi board positions for five seconds, and then had to reproduce them (with no time limit). The performances were used to test four hypotheses (see this issue pp. 12-22. – Ed.). Surprisingly enough, maybe, all four hypotheses had to be rejected. There was no evidence to support any one of them. Grimbergen concludes: "The experiment showed that perceptual clues in board and pieces (such as piece size) do not guide the knowledge stored in memory. This supports the assumption that perception is guided by knowledge in long-term memory and that perceptual clues are only used to trigger this knowledge. From these results we may conclude that the primitive agents in our model do not represent perceptual features directly."

Computers and Games 2008 is another great collection of papers describing research in a wide variety of computer games. I am already looking forward to the next one!