Appendix I

2010 Super Computer Go: Interviews with the Authors of MoGo, Many Faces of Go, Fuego and Zen

By Peter Shotwell
© 2010

Mogo

May 2010

I interviewed Dr. Olivier Teylaud again:

In 9 x 9, now, computers won against top pros. Fuego *(Univ. Alberta) did it as white with komi 7.5; Mogotw (in collaboration with NUTN, Taiwan) did it as black with the same* komi*, supposed to be disadvantageous for black. In 19 x 19,* Mogo *won with handicap 7 against a top pro and handicap 6 against a young pro. However, these performances have never been reproduced; I think H7 is still too hard for computers, against top pros.*

MoGo was running on the SuperComputer Huygens, using 25 out of the 104 nodes of the Supercomputer, i.e., 800 cores at 4.7GHz, with a floating point processing power of 15 Teraflop (more than 1000 times Deep-Blue). This was the most powerful supercomputer ever used by a board game. He also once quipped on the computer go e-group in answer to, 'What is the level of MoGo?' 'It all depends on the time you give to it. With very little time, it can be no better than random, and with infinite time, it plays perfectly! :-). And when asked why MoGo would sometimes lose big and often win by 1/2 point, he said, 'Winning is the only thing we are interested in. If MoGo is ahead, it plays safely as possible but when it is behind, it tries invasions and other risky moves that often fail.’

He continues:

Recent progress in computer go: By including extrapolation (technically speaking, supervised learning), it was possible to improve the design of Monte Carlo simulations by other means than just tedious optimization. However, these improvements are successful in other games, but not in go, probably because the Monte Carlo simulations in go are very strong and therefore more difficult to improve by automatic means.

By truncating Monte Carlo simulations, with replacement by value functions, it is possible to merge Monte Carlo Tree Search techniques with classical evaluation functions. This was in particular successful in amazons (a very interesting work by Richard Lorentz, in California).

On the industrial side, there are now applications in fundamental AI tasks (active learning, non-linear optimization), and direct applications (the tuning of the Spiral library). We have started an industrial collaboration around power management and I believe a lot in it; hopefully, we’ll start also a work on water resources for agriculture.

Main limitations: The MCTS technique used in all successful programs has trouble in case of multiple unfinished fights. The computer does not have the ability to estimate which part is the most important. This leads to bad choices in terms of [ignoring moves]. *Humans are very strong in situations in which many moves are equivalent; they know how to extrapolate between equivalent moves and situations. The computer does not have this ability, and is therefore very weak for in liberty races in which many moves are equivalent (compared to humans who efficiently count liberties instead of simulating all the orders for filling the liberties of a group).*

Big hopes: Extending the supervised learning briefly mentioned above, or using techniques which ‘merge’ several won simulations into better simulations (Peter Drake, in Portland, published interesting works on this).

Many Faces of Go

David Fotland's program, *The* Many Faces of Go, was a classic program that only evaluated a slow 100 positions a second until February 2008 . . .

I had been watching progress for a couple of years, thinking MCTS only worked for 9 x 9 but when CrazyStone won the 19 x 19 UEC Cup, it led me to rewrite Many Faces *to use UCT in early 2008. This led to my win for both 9 x 9 and 19 x 19 in the computer games Olympiad later that year. I tried more than 400 variations on the basic UCT algorithm or playout strategy during this half-year of intensive development. The engine was written in C and tuned from the start for performance. I couldn't have done this many experiments (often several per day), without a very fast engine, because I used 1000 game contests to see if there was improvement with statistically significant results. Part of the reason I won is because the basic UCT/MC is so fast that I can incorporate the slow* Many Faces *knowledge and still get over 10K playouts per second per CPU on* 9 x 9*.*

The Windows HPC server 2008 that I used, which Microsoft has just released, is a server operating system for high performance clusters and their MPI implementation is about 10% faster than Linux on the huge machines with thousands of cores. A video of the Demo is at http://www.youtube.com/watch?v=Qe0o-IvHOa0.

Fotland now has his own 16 core machine and Many Faces 1 at 10 seconds a move has been a strong 1-dan ever since becoming World Champion. His Monte Carlo program, Many Faces of Go 12, is available at www.smart-games.com.

Fuego

Martin Meuller also answered some questions in an email about Fuego, the open source program he is involved with. He wrote that it was written in C++ that is quite similar to other MCTS programs except that it is built above a game-independent library that has been successfully used for many games other than go. Some specific features are an efficient lock-free shared memory implementation for multithreading, playout rules for low liberty situations, prior knowledge to bias the exploration of the search tree, and an opening book. Also, BlueFuego, developed at IBM, is a (closed source) library for MPI connecting many copies of Fuego running on a distributed memory machine. This combination won the 9 x 9 tournament of the 2009 Computer Olympiad in Pamplona, Spain. Later in 2009, the program became the first computer program to win a 9 x 9 game on even against a top professional, It is running about 2-*kyu* on KGS.

Zen

Zen is a late-comer run by an anonymous programmer who calls himself 'Yamato.' He wrote that it is part MoGo with shape patterns that are generated by a minorization-maximization algorithm like Crazy Stone. But they are directly combined with UCT, without progressive widening. Probably the most original part of Zen, he says, is in the playout. The author does not think MC simulations must always be fast, so it has a lot of hard-coded go knowledge. Currently it is a 2-dan on KGS at 15 seconds a move.

* * * * *

To keep track of the assault on human thinking (or its advance, depending on how you look at it), go to
http://www.gokgs.com/graphPage.jsp?user