ICGA Journal 39 (2017) 44–46
DOI 10.3233/ICG-160003
IOS Press

Let's Catch the Train to Monte-Carlo[1]

Dap Hartmann*
Delft University of Technology, Delft, The Netherlands

While Monte-Carlo Tree Search (MCTS) has successfully been implemented in many games, its effectiveness appears to be greatest in the game of Go. In this thesis, Hendrik Baier even earmarks MCTS "the dominating paradigm in the challenging field of computer Go." Having mentioned Go, there is no escaping linking another statement from this thesis to the recent astonishing accomplishment by DeepMind's AlphaGo. It illustrates how fast computer game playing is currently improving – irrespective of whether you call that Artificial Intelligence or not. In Section 3.2.1, Baier writes: "the world's best human Go players are still superior to the best computer programs, and writing a master strength Go program stands as a grand challenge of AI." Four months after Hendrik Baier defended this PhD thesis, AlphaGo convincingly beat Lee Sedol 4-1. Lee is considered to be the best Go player in the world over the last decade.[2] A magnificent achievement indeed, but let us not forget that it is the culmination of all the hard work over the past decades of a great many computer games researchers who have contributed to this monumental victory. One of the three pillars on which rests the success of AlphaGo is MCTS, the topic of this thesis. It would be interesting to know whether AlphaGo contains any of the MCTS enhancements that Hendrik Baier describes in this thesis or in any of his scientific papers. The oldest paper I could find (Baier and Drake, 2010), was published in the *IEEE Transactions on Computational Intelligence* in 2010, and describes an improvement to the Last-Good-Reply Policy in Monte Carlo Go.

This thesis describes several enhancements to MCTS, which are tested in various one-player and two-player domains. The guiding problem statement is: How can the performance of Monte-Carlo Tree Search in a given one- or two-player domain be improved? To address this question, Baier formulates four research questions. Two questions apply to one-player domains and deal with the rollout phase and the selection phase of MCTS respectively. The other two questions apply to 'adversarial two-player domains', as Baier likes to call them, and deal with time management in MCTS and combining the strengths of minimax and MCTS respectively. The one-player test domains are *SameGame*, *Clickoma-*

[1] Monte-Carlo Tree Search Enhancements for One-Player and Two-Player Domains, *Hendrik Baier*, PhD Thesis, Maastricht University 2015, 290 pp. This thesis can be downloaded from: https://project.dke.maastrichtuniversity.nl/games/files/phd/Baier_thesis.pdf. This link also includes errata to the printed version (and the pdf version) of the thesis. I can add one more correction to this list: Figure 3.2a depicts the first moves of a 19 × 19 Go game, not a 13 × 13 game. Unfortunately, this pdf document does not include the 'statements' which traditionally accompany Dutch PhD theses. One though-provoking statement in Hendrik Baier's thesis is: "People talking about AI in public are either very enthusiastic because they don't know AI, or very afraid because they don't know AI".

[2] To emphasize the tremendous pace at which AI is progressing currently: during the editorial process of this contribution, the Go program Master(P) (an improved version of AlphaGo) convincingly defeated Ke Jie, the world's number one ranked Go player (http://www.nature.com/news/google-reveals-secret-test-of-ai-bot-to-beat-top-go-players-1.21253)

* E-mail: L.Hartmann@tudelft.nl.

nia, and *Bubble Breaker*, while the two-player games used to test the various enhancements are *Go*, *Connect-4*, *Breakthrough*, *Othello*, and *Catch the Lion*. To make life easy for the reader, the origin and the rules of each puzzle and game are described and its complexity is analyzed. I had not heard of *Catch the Lion* before (not surprising, as it was only invented in 2008), so this information was very useful and saved me a visit to Wikipedia[3].

During his research, Baier ran many hundreds of simulations in the one-player and two-player domains on each of the enhancements he designed. He ran these experiments not just on each individual enhancement, but also on various combinations of these enhancements. For example, while investigating time management enhancement strategies, he created five semi-dynamic strategies (EXP, OPEN, MID, KAPPA-EXP, and KAPPA-LM) and five dynamic strategies (BEHIND, UNST, CLOSE, STOP, and KAPPA-CM). He combined these in a multitude of ways, such as EXP-STONES with OPEN, EXP-STONES with KAPPA-EXP, and EXP-STONES with KAPPA-LM (for the semi-dynamic strategies) and EXP-STONES with BEHIND, EXP-STONES with BEHIND-L, EXP-STONES with UNST, EXP-STONES with CLOSE-L, and EXP-STONES with KAPPA-CM (for dynamic strategies), to name just a few, but enough to make you dazzle. And all these combinations were tested in each of the five two-player domains. If the expression 'big data' had not been coined yet, Hendrik Baier could have done it himself.

Needless to say, this thesis contains more findings, conclusions and recommendations than the two pages of this review allow me to convey. So let me mention just a few of Baier's main conclusions, and encourage you to read more of them when you download his thesis. The rollout quality of MCTS in one-player domains can be improved by introducing Nested Monte-Carlo Tree Search (NMCTS) which replace simple rollouts with nested MCTS searches. The selectivity of MCTS in one-player domains can be improved by a technique called Beam Monte-Carlo Tree Search (BMCTS) which is a combination of MCTS and beam search (the reduction of the number of nodes at each level to a constant number which renders the search effort linear in depth). According to Baier: "BMCTS expands a tree whose size is linear in the search depth, making MCTS more effective especially in domains with long solution lengths or short time limits."

Time management of MCTS in two-player domains can be improved by the STOP strategy, which estimates the remaining number of moves in the game and uses a proportional fraction of the remaining search time. The search is terminated when it becomes clear that a further search will not change the final move selection. The strength of the STOP strategy lies in the time saved by these search terminations, which can be used in later searches. STOP was the most successful strategy in the various two-player test domains. It won approximately 60% of the games against state-of-the-art time management in 13×13 and 19×19 Go, under various time controls. The tactical strength of MCTS in two-player domains can be improved by using a hybrid search strategy which combines MCTS with minimax searches. Optionally, such a hybrid strategy can contain relevant domain knowledge. Although it was not convincingly shown that this approach generally outperforms regular $\alpha\beta$-search, it did do very well in *Breakthrough*.

This thesis is very well structured and beautifully presented in crisp prose and with very nice illustrations. Future PhD students should take heed and read this thesis as a prime example of a thesis which is worthy and representative of all the hard work that went into the research project. And those

[3]Well, I did that anyway, and it taught me that the original Japanese name is *dōbutsu shōgi*, and not *doubutsu shogi* (without the accents), and that the game was marketed outside Japan as *Let's Catch the Lion* (not *Catch the Lion*). http://en.wikipedia.org/wiki/index.php?title=Dōbutsu_shōgi&redirect=no.

readers who already have a PhD, or have no intention of ever starting one, should also download a copy and enjoy this fine piece of work.

REFERENCE

Baier, H. and Drake, P. (2010). The Power of Forgetting: Improving the Last-Good-Reply Policy in Monte Carlo Go. *IEEE Transactions on Computational Intelligence and AI in Games*, 2(4):303–309.