Planing agents and their state space

One of the most important problems of planning agents in artificial intelligence is the size of the state space.

We can use this project as example. There are 3 problems that have been solved using different heuristics, each one with increased difficulty. All of them are easy to solve by hand and they have very few fluents. For example the second problem has only 27 fluents but that results in a state space bigger than 130 millions.

So when it comes to solve complex problems the state space becomes even more impossible to work with.

The problem of chemical synthesis

As a demonstration we can take a look at the problem of retrosynthesis. As Marwin, Mike and Marke states this is a plan technique used to synthesis chemical molecules by recursively dissecting them into simpler molecular blocs [1]. As they say the state space is almost intractable due to it's size.

One of the typical techniques used in this kind of situation is best first search along with some hand crafted heuristic. As we have seen in this project this technique leads to a good solution (but not optimal by definition) with a very reasonable computational time.

Monte Carlo Tree Search

One alternative that is gaining a lot of atention is the use of Monte Carlo tree search (MCTS) [2] for scoring nodes and deep learning for deciding which node to expand. This was the combination that AlphaGo used to become the first AI to beat a professional Go player [3] which was an event with huge repercussion in global media/news.

Monte Carlo tree search (MCTS) is a heuristic algorithm that focus on searching the most promising moves. It expands the search tree based on random sampling of the search space. It has 4 phases for expanding the tree (Figure 1).

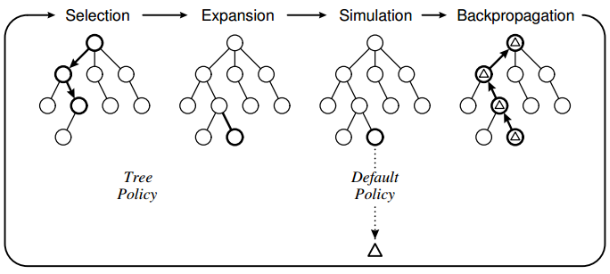

Figure 1: MCTS phases

Inspired with AlphaGo, Marwin, Mike and Marke successfully applied this wining combination to the retrosynthesis planning problem [1].

Other problems

So at the end the approach that the team of AlphaGo made seems to be a very good approach to solve planning and adversial problems. But it is not limited to that, Xiaoxiao, Satinder, Richard and Honglak proved that it can also be applied to play ATARI Games [4]. And it is possible that we still have to wait to all the possibilities that this approach offers

References

[1] Marwin Segler, Mike Preuß, Mark P. Waller. *Towards "AlphaChem": Chemical Synthesis Planning with Tree Search and Deep Neural Network Policies.* `https://arxiv.org/abs/1702.00020`

[2] Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, PhilippRohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton.
A survey of monte carlo tree search methods.
`http://ieeexplore.ieee.org/document/6145622/?reload=true`

[3] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel & Demis Hassabis.
Mastering the game of Go with deep neural networks and tree search
`https://storage.googleapis.com/deepmind-media/alphago/AlphaGoNaturePaper.pdf`

[4] Xiaoxiao Guo, Satinder Singh, Richard Lewis, Honglak Lee. *Deep Learning for Reward Design to Improve Monte Carlo Tree Search in ATARI Games.* `https://arxiv.org/pdf/1604.07095.pdf`