CS 486 – Assignment #6
Justin Franchetto
20375706

Paper selected: The Grand Challenge of Computer Go: Monte Carlo Tree Search and Extensions

1) **What are the motivations for this work?**
   Go has traditionally been a very difficult game for AI programs to play successfully. Brute force tree-search and reinforcement learning techniques are not as successful for playing Go as they are at playing games such as chess, checkers or backgammon. The motivation for building an AI that can play Go at a high level comes from the challenge it has posed in the past; namely due to the size of the search space and the difficulty in judging the strength of a state and potential moves. Additionally, an AI that is able to play Go at such a level can provide insight about how to play other two player games, as well as many other real-world problems with large state spaces.

2) **What is the proposed solution?**
   Instead of utilizing techniques such as reinforcement learning and brute-force search, the proposed solution is to use Monte-Carlo Tree Search (MCTS) algorithms.

   These algorithms are an improvement over previous solutions because they have little prior knowledge about the game itself and develop expertise by simulating random games in self-play (Monte-Carlo Simulation). Strength of particular moves are evaluated based on their success during the randomized play, overcoming one of the challenges faced by previous approaches. Further, a consistent version of MCTS can be used, called the Upper Confidence Bounds on Trees (UTC) algorithm. These types of algorithms will find the optimal value for each node of the tree if given sufficient time.

   Typically, MCTS repeats four phases: *descent, roll-out, update,* and *growth* until a set of stopping criteria has been met. In more detail, the four phases are:
   i. *Descent:* MCTS iteratively selects the best scoring action of the current state
   ii. *Roll*-out: A policy is used for selecting legal moves for both players until the game's end
   iii. *Update:* The statistics of each node visited during the descent phase are updated based on the results of the game
   iv. *Growth:* The first state visited by the roll-out phase is added to the tree and initialized

3) **What is the evaluation of the proposed solution?**
   There are several extensions to the basic MCTS which help to search more deeply, resist over-exploration, and speed up the search by sharing knowledge. These extensions provide further advantages over classical methods and are as follows:

   i. *Virtual experience:* This technique uses a database of expert Go plays to construct supervised estimates of the likelihood that an expert would make an action given the current game state. This is useful because the same state is unlikely to be seen twice due to the large size of the state space, and therefore an AI must create estimates by using similar, but not identical states. This seems to be a good solution to determining the strength of a particular move in a state.

ii. *Parallelism:* Unlike classic tree search methods, such as minimax with alpha-beta pruning, MCTS is highly parallelizable and can be run simultaneously on separate cores. However, the benefits of running MCTS in this way are not as great as expected.

iii. *Roll*-outs: A carefully chosen roll-out policy can greatly boost the performance over a random roll-out policy. However, if a policy is too complex, then it might be faster on individual simulations, but slow down the overall performance of MCTS. There are three approaches that attempt to find a suitable policy: *hand-designed policies, supervised learning of policies* and *simulation balancing.* In general, the idea of roll-outs are quite useful because they give a policy for choosing legal moves for each player, and the chosen policy can change the overall performance of the algorithm (for better or worse).

iv. *RAVE:* The Rapid Action Value Estimation method is a generalization of the All-Moves-As-First (AMAF) idea to search trees. RAVE shares action values among subtrees during MCTS on the assumption that the value of taking an action will be similar at any node of a subtree. This assumption provides a larger amount of data for each action, but this data can be misleading, especially in situation where nearby changes entirely change the value of a move. To overcome this issue, the MC-RAVE algorithm is used, interpolating between the AMAF value estimate and the Monte-Carlo estimate. This extension also supports the use of MCTS because it gives an AI the ability to judge the value and reward of making a particular move in a given state.

Despite some of their limitations, these extensions provide a convincing case for using MCTS over classical methods because they overcome many of the challenges that have made Go difficult for previous AI programs to play effectively.

4) What are the contributions?

The key contributions of this paper are the ideas of simulated balancing in roll-out policies and the Rapid Action Value Estimation (RAVE) method. Simulated balancing allows for the finding of a policy that is very good but still has a high level of variance, which is important because it allows for randomness. These two ideas are critical in helping AI systems to play Go as they directly address challenges that classical approaches have struggled with. The RAVE method enables the ability for a system to estimate the value of a particular move and simulated balancing in roll-out policies gives a good policy that still allows for variance and randomness.

5) What are future directions for this research?

MCTS has been applied to many other challenging games, such as Amazons, Lines of Action, Hex, Havannah and it is also dominant in the general game playing competition. MCTS is applicable to many real world decision making problems, particularly those with a large search space, a large branching factor, delayed consequences of actions and a difficult to construct evaluation function. Further, it has been applied to partially observable domains, classical planning, scheduling, biometrics and natural language processing.

In addition to the other applications discussed in the paper, these methods and algorithms could be useful in other domains, such as adversarial search, security, question generation and even games with more than two players, like Settlers of Catan.