Initial Plan Monte-Carlo Tree Search for Go

Author: *Thomas Ager* ***Module Number:*** *CM3203*

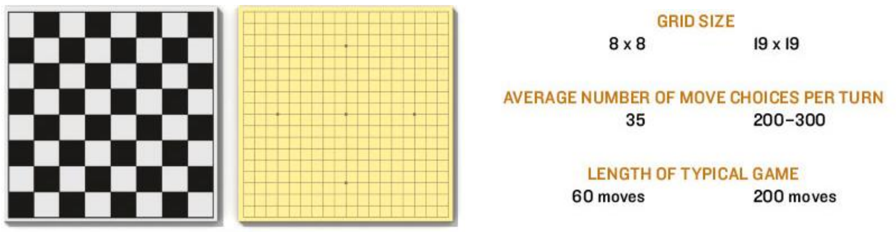
Supervisor: *Steven Schockaert* ***Module Title:*** *Individual Project*

Moderator: *Unassigned* ***Credits Due:*** *40*

Project Description

This project is about the implementation of a computer player for the board game Go using Monte-Carlo Tree Search (MCTS). Go is a game in which two players take turns putting black and white stones on the intersections of a grid of lines to surround areas.

In Artificial Intelligence, games like Chess have had success using a search-and-evaluate approach called Minimax. These kinds of techniques increase in complexity according to the number of children for each node of the search tree, called the branching factor. For chess, that factor is roughly 40, and a typical chess game lasts for about 50 moves. In Go, the branching factor can be more than 250, and a game goes on for about 350 moves. The amount of options in Go quickly becomes too much for a standard search algorithm, making it difficult for a program to defeat even an amateur with this approach. [1]



A comparison of Chess and Go. http://spectrum.ieee.org/computing/software/cracking-go/chess-vs-go

There are many proposed solutions for Go, and the one with the greatest success is Monte-Carlo Tree Search (MCTS). Instead of determining the best choice through the use of a decision-making heuristic, MCTS uses probabilities and randomly played out games to decide which move has the highest chance of resulting in a win. With this technique, computer players for Go are reaching ever higher ranks. In March 2014, a system called Crazy Stone defeated Norimoto Yoda, a professional Go player. [2]

This project will research the variants and performance of MCTS for Go, and provide an implementation of one of those variants. This will give an understanding of the MCTS algorithm, and the opportunity to evaluate the performance of a chosen variation of MCTS. The implementation will have its runtime measured and compared with other variants, and will have its rank determined playing against Go players. This will allow the quality of the implementation to be determined and the effect different design decisions can have, giving the implementation and its analysis a place in the field of MCTS Go research.

[1] http://spectrum.ieee.org/robotics/artificial-intelligence/ais-have-mastered-chess-will-go-be-next

[2] http://remi.coulom.free.fr/CrazyStone/

Project Aims and Objectives

Core Components

Background

- Compare the functionality, advantages, disadvantages and performance of the Monte-Carlo algorithm compared to other proposed solutions in Go.
- Compare the functionality, advantages, disadvantages and performance of different variants of Monte-Carlo in Go.
- Choose a variant of Monte-Carlo to implement for the solution.

Approach to Implementation

- Choose the language and Go GUI for the implementation.
- Research the functionality, advantages, disadvantages and performance of different data structures in the context of MCTS for Go.

Implementation

- Create pseudocode for the chosen Monte-Carlo variant.
- Implement the computer player for Go.

Analysis

- Analyse the performance of the implementation in millisecond time and Go rank.

Conclusion and Reflection

- Critically appraise the results of the analysis of the implementation compared to the performance of other solutions proposed for Go.

Optional Objectives

Implementation

- Implement different variants of the algorithm.

Analysis

- Analyse the performance of different variants of the algorithm in millisecond time and Go rank.

Conclusion and Reflection

- Critically appraise the results of the analysis of the implementation variants compared to each other and other solutions proposed for Go.

3 Work Plan

	02/02/2015	09/02/2015	16/02/2015	23/02/2015	02/03/2015	09/03/2015	16/03/2015	23/03/2015	30/03/2015	06/04/2015	13/04/2015	20/04/2015
Background	🟩	🟩	🟩	🟩								
Approach				🟦	🟦							
Implementation					🟧	🟧	🟧	🟧				
Extra features								■	■			
Analysis								■	■			
Conclusion									🟨	🟨		
Review										▒	▒	
Hand-In											⬛	⬛
Meet Supervisor	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪	🟪
Review Progress					🟪			🟪		🟪	🟪	