Monte-Carlo Tree Search for Robocode

J. Grooss (jcgr@itu.dk), J. Melnyk (jmel@itu.dk)

Abstract— **The abstract goes here. Please try to make it less than 150 words. We suggest that you read this document carefully before you begin preparing your manuscript.**

This template is for LaTeX users of the Advanced AI in games class. Authors should use this sample paper as a guide in the production of their report(s).

I. Introduction

What problem are you trying to solve?

Why is this important?

II. Background

Most games attempt to engage the player by presenting a number of challenges for the player to overcome. Sometimes these challenges consist of precision, timing, execution speed and reaction time, while in other cases the challenge consists of making a strategic choice. When making these strategic choices, a player must consider not only the present state of the game, but also the actions taken by the adversary (either another player, an artificial intelligence or the game itself).

A. Monte-Carlo Tree Search

Monte-Carlo Tree Search (MCTS)[1] is a searching algorithm that is based on the Monte Carlo method, which dates back to the 1940s. The idea behind the MCTS algorithm was explored in the 1980s and various implementations were written in the following years. It was not, however, until 2006 where a breakthough was made, which made AIs able to utilize MCTS to play games that had been considered too challenging until then, such as Go[2], [3].

MCTS is, as the name implies, a tree search algorithm. It analyses the moves that are deemed most promising and expands the search tree by choosing a random action from the set of possible actions at any given node.

In order to determine which move that is best from the root of the tree, MCTS runs a lot of *playouts*. *Playouts* are simulations of playing the game until an end condition is reached, with each move being chosen at random. The score of the game state at the end is based on the UCT and is used to update the weights of the tree in such a way that better nodes are more likely to be explored further.

The better the heuristic is at determining the value of a gamestate, the better MCTS will fare. An example of this is to factor in how many lives Pac-Man has left at an end state instead of only evaluating at the score.

B. MCTS in Partially Observable Games

MCTS works on the premises of information. The more it has access to in a game, the better it is at

The more information MCTS has access to, the better it will perform. This means that MCTS works well in fully observable games, such as chess or Pac-Man, as it has access to the full state of the game.

III. Game Mechanics

How does the game work that you are using?

Why do you need AI in this game?

A. Influence of Robocode mechanics on MCTS

IV. Methods

How does your algorithm work? Describe in as much detail as you can fit into the report.

Also, how did you interface it to the game?

V. Results

Did it work?

How well? Provide some figures, and a table or two.

How much time does it take?

Remember to include significance values (remember the t-test?), variance bars Reread some of the papers from class and compare how they report their results.

VI. Conclusions

The conclusion goes here.

What are the strengths and shortcomings of your method? Why did you choose method X instead of Y? How well would it generalize to other game genres? How would you develop it further, if you had time?

References

[1] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, "A survey of monte carlo tree search methods," *Computational Intelligence and AI in Games, IEEE Transactions on*, vol. 4, no. 1, pp. 1–43, 2012. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6145622

[2] S. Gelly and D. Silver, "Monte-Carlo tree search and rapid action value estimation in computer Go," *Artificial Intelligence*, vol. 175, no. 11, pp. 1856–1875, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S000437021100052X

[3] G. Chaslot, “Monte-carlo tree search,” Ph.D. dissertation, PhD thesis, Maastricht University, 2010. [Online]. Available: https://project.dke.maastrichtuniversity.nl/games/files/phd/Chaslot_thesis.pdf

[4] J. Orkin, “Three States and a Plan: The A.I. of F.E.A.R.” *Monolith Productions / M.I.T. Media Lab, Cognitive Machines Group*, 2006, [Online; accessed December 11, 2014]. [Online]. Available: http://alumni.media.mit.edu/ jorkin/gdc2006_orkin_jeff_fear.pdf