Desert Fox: a Simple Wargame AI

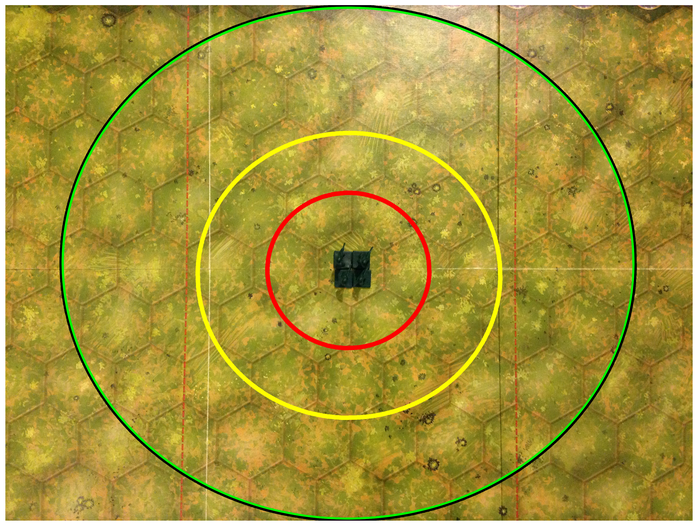
Parker Michaleson

Introduction

Playing wargames, ranging from classics like Chess to Go to modern consumer games like Total War, presents a historically interesting problem for AI researchers. Much progress has been made on the front of playing Chess and similar games with small search spaces and simple heuristics by using traditional search techniques. Games with large search spaces which lack good heuristics, like Go, however, have presented a major challenge for such techniques. Here we demonstrate the method of the *Monte Carlo Tree Search* for playing a simple game derived from the commercial game Memoir '44 called Desert Fox which, like Go, possesses a large search space and lacks search heuristics.

General Strategy

Desert Fox possesses a large search space compared to other games due to its simple, unrestrictive, rules, on any given turn, a player may order at least one and up to four of his/her units. An order to the unit consists of up to two tiles of movement and one combat instruction. In addition, Desert Fox uses hexagonal tiles, all of which are passable to the unit unless occupied by an enemy unit. Then for any given unit, the number of possible orders during a turn consists of at least 19 possible moves *without considering combat*. On the picture below, we can see the range for movement with combat in red, the range for maximum movement in yellow, and the maximum combat range in green.



To add to the complexity, the player must often choose units to order out of a pool of a dozen or more units based on cards in the player's hand. This leads to a swiftly growing combinatoric problem with an unreasonable size for a traditional unguided search and a lack of clear cut heuristics for implementing a guided search. This project works around traditional limitations by using the Monte Carlo Tree Search (MCTS), which uses a large sampling of child states to find a close-to-optimal action for a given parent state. MCTS can be further improved by adding a heuristic to opportunistically probe promising child states which are likely to be close-to-optimal. While there exists no simple heuristic rule for this game, we can use a learning technique such as Q-Learning to generate a heuristic by observing gameplay and adjusting the heuristic function accordingly.

Conclusions

References

Some major works useful to this project are given below:

References

[1] High-level Reinforcement Learning in Strategy Games. Christopher Amato and Guy Shanai. *Proc. of 9th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2010).*

[2] A Survey of Mente Carlo Tree Search Methods. Browne, Powley, *et. al.. IEEE Transactions of Computational Intelligence and AI in Games, Vol. 4, No. 1, MARCH 2012.*

Source Code

The source code is available as a Git repository at:
`github.com/pcm2718/psychic-pikeman.git`