Monte Carlo Real Time Strategy

Tristan Cazenave
Laboratoire d'Intelligence Artificielle
Université Paris 8, Saint Denis, France
cazenave@ai.univ-paris8.fr

1 Introduction

We present a simple real time strategy game and five algorithms that play this game. The algorithms are compared with head to head confrontations. The best one performs Monte-Carlo simulations associated to some goals of the game.

Monte carlo simulations have been used to select moves in strategic games such as Go [Bruegmann, 1993; Bouzy and Helmstetter, 2003].

2 A Simple Real Time Strategy Game

We have designed a simple real time strategy game for our experiments. It is played on a 50 by 20 map. The two opponents, Blue and Red, start each with a 5 by 3 base in the upper and lower middle part of the map. They also own 10 units each, aligned below the upper base for Blue, and aligned above the lower base for Red. The bases have a health of 100, and each unit has a health of 30 at the beginning of the game. Each unit occupies a one by one square on the map, and it can move in any of the eight directions provided the goal neighbor square is empty. Units can also shoot ennemies that are located in the eight neighboring squares.

3 Move Selection Algorithms

The random move algorithm picks randomly one move out of the nine possible moves (the eight directions and the stay in the same place move). This strategy mostly serves as an etalon for the other strategies. All strategies should perform better than the random move strategy.

The nearest enemy unit strategy consists in moving toward the nearest enemy unit and shooting at it as soon as it is in its neighborhood.

The evaluation function adds the health of the friend base and of all the friend units, and substracts the health of the enemy base and of all the enemy units.

The Monte-Carlo moves strategy consists in performing a given number of simulations, and to choose moves according to the statistics collected on the final results of the simulations. In each simulation, the first move of each unit is played randomly. The subsequent moves are played with the nearest enemy unit strategy for both sides.

The Monte-Carlo double moves strategy starts each simulation as the Monte-Carlo moves strategy by playing random moves for the units, and also continues the simulation using the nearest enemy unit strategy. But instead of recording the scores of each individual move, it records the scores of pairs of moves for pairs of units. A score is maintained for each possible pair of moves of each possible pair of units.

Table 1: Sum of results for each algorithm.

$Algorithm$	$Sum\ of\ results$
$random$	-960
$mc\_double(100)$	-530
$mc\_moves(100)$	-185
$nearest$	530
$mc\_goals(100)$	1145

The Monte-Carlo goals strategy does for the goals of the game what the Monte-Carlo moves strategy does for the moves. The only kind of goal we have currently tested is to attack an enemy unit or base. At the beginning of each simulation, a target enemy unit is set for each friend unit. During the simulation, the friend units will hunt for their target enemy unit. Once the target enemy unit is dead, the friend unit reverts to a nearest enemy unit strategy. The enemy units use the nearest enemy unit strategy during all the simulation.

A score is maintained for each possible goal of each possible unit. After a fixed number of simulations, the goal with the highest score is chosen for each unit.

Table 1 is created summing for each move selection algorithm its score against all the other algorithms. They have been tested with 100 simulations before each move, enabling a very fast move decision process.

One promising area for future work is to add higher level tactical goals such as protecting the base, protecting an area or helping another unit. Testing the strategies in a more complex real time strategy game is also appealing.

References

[Bouzy and Helmstetter, 2003] B. Bouzy and B. Helmstetter. Monte Carlo Go developments. In *Advances in computer games 10*, pages 159–174. Kluwer, 2003.

[Bruegmann, 1993] B. Bruegmann. Monte Carlo Go. ftp://ftp-igs.joyjoy.net/go/computer/mcgo.tex.z, 1993.