Automatic Heuristic Generation on Random Graphs

Niklas Ulvinge
Georgia Tech

Andrew Pruett
Georgia Tech

September 17, 2012

1 Problem

Trying an automatic heuristic generator algorithm on a random graph game is hopefully simpler and more successful that it was on the comparatively more highly structured grid based games like checkers, chess, and othello for which it was originally conceived.

2 Related Work

Minimax using a good heuristic results in a good player for grid based games. There has been some work on general game agents trying to find good heuristics automatically when playing based strictly on the rules of the game. Are these learning techniques generalizable to graph based games? Will the work for a simple game called Cops and Robbers? In this game, agents alternate single edge moves on a connected graph. When the cop catches the robber, that is a win for the cop. If the robber doesn't get caught, that is a win for the robber. It is a very simple game, and we plan to find out whether simple game rules input to the automatic algorithm get good evaluation functions.

2.1 Types of Graphs

A planar connected graph can be drawn in two dimensions with no edges intersecting. A non planar graph cannot be drawn on paper without an intersecting pair of edges. Non planar graphs occur in social networks, dependencies of systems, and VLSI planning. A critical non planar graph is such that if one vertex is removed it becomes a planar graph.

2.2 An obvious proof in planar Cops and Robbers

There is proof that 3 cops will always beat 1 robber on a planar graph. When allowing a multiplicity of cops, how many cops does it take to be unbeatable on a random non planar connected graph? Arbitrary many cops are needed to win on a nonplanar graph, but given enough cops, can a game agent group figure out a cheap way to locate and divide a random nonplanar graph into the fewest largest planar subgraphs so the cops can guarantee a cop win with the fewest cops? Probably not, because the maximal subgraph problem is NP-complete. The famous linear time algorithm may work well for gameplay though, since it has a random element. However, this random element may ruin the automatic heuristic generation algorithm. These are some other interesting side items we could encounter.

3 Implementation

Using the C language, we will implement pointer based graph data structures based on two possible implementations: adjacency lists and edge matrices. We will attempt a general play algorithm for Cops and Robbers with perfect information from the rules analysis technique of Kuhlmann and Stone. We intend to generate random graphs and evaluate the playing performance of a simple SLD heuristic against the automatically discovered heuristic against a baseline of randomly moving agents. Graph generation will be tunable for three parameters: vertices, edges, and max edges. This allows us to tune the branching factor for the usual effect using MINIMAX as in AIMA Russell and Norvig. It also lets us set the planar vs. nonplanar condition.

4 Evaluation

We will compare a randomly moving agent and a straight line distance heuristic and a automatically generated heuristic with each other in simulations on the same graphs for a variety of graph complexities. We will look for improvement above the random mover baseline.

References