TJHSST Computer Systems Lab Senior Research Project Proposal Design of a Simple Real Time Strategy Game with a genetic AI 2009-2010

Bharat Ponnaluri

October 30, 2009

1 Purpose

The goal of this project will be to use a simple RTS game I created and create a genetic AI. The genetic AI should be intelligent enough to win against a skilled human player. I also hope to make the genetic algorithm somewhat generic so it can be used for other games. The AI should also be able to mimic human emotions and be capable for diplomatic interation, otherwise playing against them would get boring.If I finish creating a genetic AI before the end of the year, I will rewrite the AI code for another computer game and try to apply my genetic algorithm. Currently, many computer games involve optimizing combination of heuristic evaluation functions and constants, and doing so is difficult and sometimes impossible. A genetic algorithm would be able to generate an optimal combination of evaluation functions and constants without significant user input. If enough computing computer can be obtained to run a genetic algorithm, game developers will be able to create more intelligent AIs. This will satisfy hardcore gamers. Also, game developers will not need to know good strategies in order to design an intelligent Ai since the genetic algorithm will determine the strategies. Basically, by coding a non-genetic AI, I am making an educated guess about what the optimal AI algorithm is.

2 Scope of Study

The first portion of my project will be creating a more intelligent AI by using better heuritstic evaluation functions and allowing the AI to be able to respond to other player's actions. Even if I run into problems and my code ends up being too complex, I will still be able to create a more intelligent AI. One of my goals is to use building block functions for my genetic algorithm that I will use later. If I run out of time, I will forget about the building block functions. During third quarter, I will code the genetic algorithm to optimize constants that determine AI behaviour and building block functions.

Genetic algorithms take up a signficant amount of time and computing power, even with networking. If networking ends up being too complex, I will make my game simply by reducing the size of the game map, remove the graphics, and run multiple instances of my game in parallel. If coding a genetic algorithm does not turn out to be feasible because of time or computing power constraints, I will focus on machine learning for 3rd and 4th quarters

3 Background

A genetic algorithm works by randomly determining a set of parameters and function combinations that are represented in chromosones. An algorithm is run once for each chromosome based on the data in the chromosome. Afterwards, based on the chromosome's performance during the algorithm, a fitness score is calculated for the chromosome. Then the chromosomes with the lower scores are eliminated. Then the chromosomes randomly mutate and have a small portion of their data randomly modified. Then the suriving chromosomes "mate" and swap data, then the algorithim runs again. Genetic algorithms have a tendency to have their chromosomes converge on locally optimal places. For my algorithm this will be a good thing is long as the local optima are not too far from the optima. This is because I would like a diverse set of AI's because they would have different personalities and make the game more exciting. Here, I propose the use of a genetic algorithm to optimize the heuristic evaluation functions for the AI of a strategy game.

Currently, there is not a significant amount of research on the use of genetic algorithms, but the research done so far is encouraging. Genetic algorithms apply the principles of natural selection and evolution to optimize certain parameters. These parameters are represented as a long string, which represents a chromosone. One site attempts to use genetic algorithms to create a mathematical equation consisting of the numbers 1-9 and /,+,*,-, that

adds up to 42. Here, the chromosones would be the mathematical equation. The algorithm then consists on a fitness score based on how close the chromosone's parameters are to a solution. In the case of a game AI, the fitness score is calculated based on how well the AI performs in a game. Then, the chromosones with a higher fitness score swap some of their parameters, and the process repeats. In order to prevent the genetic algorithm from converging on a local optima instead of a global one, the chromosones occasionally mutate like natural chromosones. Also, there is a genetic algorithm that finds the largest possible circle that can fit between a random collection of disks without overlap. Here, a genetic algorithm found a solution that the human did not know.

Another area of research involves using a genetic algorithm to optimize weights heuristics to generate the shortest possible degree spanning tree. A degree-constrained spanning tree is a tree structure that contains all verticies on a graph as nodes. The shortest possible degree spanning tree occurs when the total length of the connections between verticies is minimized. A genetic algorithm was able to find the shortest possible degree constrained spanning tree. Based on the sucess of the genetic algorithm the researchers suggest using genetic algorithms in other areas

3.1 Sources

- http://www.ai-junkie.com/ga/intro/gat1.html
- http://www.gamedev.net/reference/articles/article1175.asp
- http://portal.acm.org/

4 Procedure and Methodology

- 2nd quarter
  - 11-2 to 11-6
  - Rest of November
  - December
  - January
- 3rd quarter
  - February-Write genetic algorithm

  - March-Run genetic algorithm
  - March-While genetic algorithm is running, polish the rest of the game
  - April-Test out genetic algoritm against AI I coded 2nd quarter to see if it is more intelligent

- 4th quarter
  - Find another strategy game and design a genetic AI for it

4.1 Design

To design my AI, I will be programming in Java using Eclipse. Eclipse has an integrated debugger which I plan to use. The main game part of my program is stable and functioning without major bugs as long as the graphics algorithm is not taking too much time. I will not need any additional resources except additional computers that are not being used for 3rd quarter. Currently, the input part of my game functions and will be sufficient. The only input data are multiple text files describing the game maps, and I already have code to read such as text file.

4.2 Testing

My main debugging effort will be focuses on the AI algorithms. In order to make sure that the AI algorithms are more intelligent, I will try to exploit it using different methods such as only building troops when under attack, turtling and blitzing the mp, or being really agressive. I will consider the genetic algorithm intelligent if I win a free-for-all against 4 other AIs less than one times in five of the time. I will also have classmates and friends test out my game to make sure that my AI really is more intelligent. Also, I will text my genetic AI against the AI's I coded 2nd quarter. Graphs will be useful in order to display the win rates of my genetic AI against other AIs and players compared with a non-genetic AI.

5 Expected Results and Value to Others

Currently, the AI code of many strategy games is not created genetically. By creating a genetic AI, gamers of all skill levels will enjoy games more. In many cases, the AI algorithms of a game is basically composed of heuristic algorithms which decided on what to do based on the current state of the

game and a set of constant values. The problem is that optimizing those constant values can take a long time, and becomes impossible if you have too many values. Genetic algorithms sometimes have a tendency to converge around local optima, which is often a problem. For the AI I'm creating, this will actually be a good thing since it ensures that all the AI's do not act in the same way. Also, I can easily make my AI less intelligent by suboptimally modifying my genetic AI algorithm. This will make it more fun for beginners to play a game because a player is not handicapped. Also, the AI will be less likely to make arbritary decisions to attack/suicide on someone, which makes people annoyed.

The are many excellent solutions to problems such as the Travelling Salesman Problem that involve heuristic evalution functions and constants. If the designers of these solutions used genetic algorithms combined with their existing solutions, they could come up with more efficient solutions. Also, genetic algorithms would be useful to develop an intelligent AI for games such as Go, as current AI algorithms are not very intelligent compared to many top players