TJHSST Computer Systems Lab Senior Research Project Paper
Crowd Dynamics Artificial Intelligence with Particle Swarm Optimization
2006-2007

Keith Ainsworth

June 10, 2008

Abstract

Articial Intelligence has for long been an important aspect of computer science, but unfortunately articial intelligence is usually computed from a single agent perpective or with multiple, but highly omniscent agents. I have created an articial intelligence engine, which works by having multiple agents, each with a highly limited perspective. In order to solve tasks, they need to communicate their portions with each other through a network. Using that scheme, it will much more accurately simulate crowd dynamics, as seen in real life, using particle swarm optimization to optimize the calculations.
Keywords: AI, PSO, C++, OOP, SDL

1 Introduction

The AI engine I've programmed is implemented through C++'s object orientation. I have programmed several classes which interact in order to make a completed end project. As my engine is an agent based networking engine, naturally the first two classes are the agent, Person, and network, PNetwork classes. The main program must include an array of Persons, which is passed to the PNetwork class on instantiation. Then the main program only needs to talk to the PNetwork, as its managing the list of people from instantiation on, and will take care of the movement of the Persons.

The Person and PNetwork also utilize another class I've written, the weightlist class. This class is a set of two array based, fixed size, looping lists; one for data, the other for the data's relative weighting. The important feature of the weightlist that other prewritten container classes don't offer is a summation function. This function effectively averages the data list, based on the relative

weightings, and a decay weighting that favors the more recent entries in the list. This is crucial because the communication aspect of the PNetwork has to have a way of keeping track of each Person's communications. Therefore each Person in the PNetwork is assigned a weight_list.

The AI engine has many different variables which can be optimized and modified to change the effective efficiency of the simulation. The number of entries allowable within each person's weight_list determines how much each person can know at any given time. If they aren't allowed enough they won't be able to make comprehensive decisions, while if they're allowed too much they'll get confused by past data. The sightline distance determines how far each agent can see into the world (provided no obstructions). The greater the sighline is the more likely it is that more agents will see the target, and therefore the more efficient their seeking is. The final variable factor is the communication efficiency factor. As the distance between two communicating agents, the signal is distorted with some amount of noise. The effeciency factor is a compounded figure created by factoring a maximum communication distance with a decay function (by default, it's linear).

2 Background

Artificial Intelligence programming always requires a task at hand for functionality and relative significance. In my implementation, the initial task for the agents in my network based artificial intelligence engine will be target detection and convergence. The agents will have to locate and converge upon a random target (eventually to become a user controlled target) using methods streamlined with particle swarm optimization methods. Additionally, as they attempt to accomplish their motives, they should become increasingly more efficient at it. This would have to be as a result of the optimizations found in the paper by Kennedy and Eberhart

3 Procedures and Methods

3.1 Overview

This program relies heavily on object oriented programming and function pointers, two fairly involved programming tasks. Early on I ran into a roadblock attempting to create two simultaneously co-referencing classes. I solved that issue through template classes. I will expect to have to solve many similar issues in the future in the same manner. To program this engine, along with the accompanying game (for graphical output reasons) I've used C++ (and therefore the g++ compiler) along with the SDL (software digital layer) libraries, for keyboard input and graphical output, and I have used OOP programming (therefore the gcc compiler won't be sufficient) and PSO for optimization.

The way the engine works is by: instantiate a PNetwork with a list of Persons, and run the PNetwork in a non-terminating loop (except by exit). The

Figure 1: Regular PNetwork Simulation tester shell running with 212 agents.

PNetwork class has a step() method in which the communicator methods from each Person in its list are called. The communicator methods simply return whatever the Person can see that he wishes to inform the others about. Then those messages are added into weightlists associated with the other people in the list. When everyone has communicated what they have to communicate, the PNetwork calls the summation function on the weightlists and instructs the Persons the summation is relevant to, to head in that direction. That simple process can repeat endlessly, with obvious variations in the behavior created through different communicator methods (each Person only has a pointer to a function of the proper parameters and return type, it can be defined on instantiation to be whatever the programmer wishes) and different communication distances and message resilience (currently they decay in accuracy as a function of the distance they travel).

3.2 Testing

I've programmed this project so far with several debugging features and text based outputs for constant error checking. While for the final project these have been commented out for the final compilation, I plan to continue programming with those features to allow for ease of code writing and testing. Right now I'm using a series of testing shells to assess the resilience of my AI system. I have shells which print out pixels for each agent in the simulation, using SDL, which work for the regular PNetwork and ngon_world classes. These automated tests inform me whether the program is doing what it should be.

The testing done for this simulation has been collected through diagnostic output files, and a compounding system I made to measure the "efficieny" of the simulation. It weighs in the average number of communications each person receives before they have an accurate idea of the targets location, the ratio of extraneous to target-bound motion, and the average signal clarity of all communications made during the simulations run.

4 Results

This project has greatly expanded the field of Artificial Intelligence, with this much more modular and isolated AI engine. There are no hints given to the agents in this simulation, they must discover and solve their task themselves, with only their communications to guide them, which also get distorted. Essentially, by increasing the amount of communication, and decreasing the amount of intelligence of each agent, I have made a much more realistic search optimization, crowd dynamics AI engine.

Since time is a factor in the efficiency computations, all of these results are standardized through a rate-restricting class I wrote called rlimit. This down limits the simulations speed so that regardless of what computer it's run on the simulation will run at the same speed. If an iteration of the simulation runs too fast, the library will add a customized delay (accurate to +/-5 milliseconds) to regulate the speed. If more than .5% of the runs take more than the nominal iteration length, it will discard other statistical output.

With this complex AI system, there were many different variables which had significant results on the simulations efficiency, allowing for customization depending on circumstance. The variables I modified were: sightline distance and communication clarity. As it should be expected, when the sightline distance was decreased or the communication clarity decreases the efficiency of the simulation decreased, while when they increased, the simulation's efficiency increased.

5 References

Laird, John, and van Lent, Michael, "Interactive Computer Games: Human-Level AI's Killer Application," *National Conference on Artificial Intelligence*, AAAI, 2000.

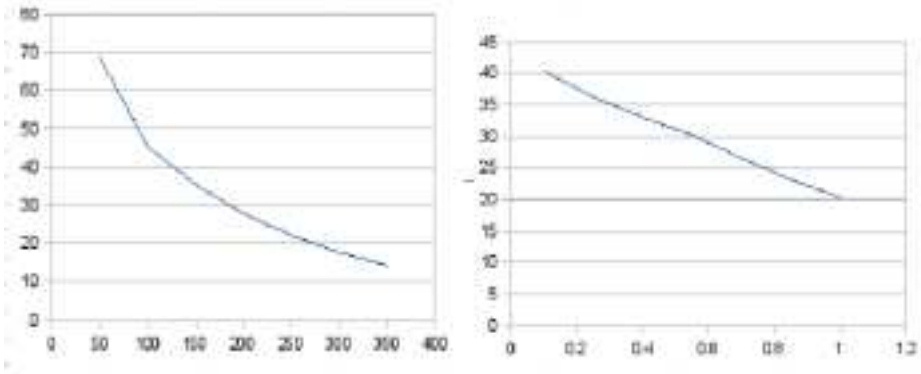
Ozcan, E., and Mohan, C.K., "Analysis of a simple particle swarm optimization system," *IEEE Transactions*, Piscataway, NJ, 1998.

Russel, Staurt J., Norvig, Peter, "Artificial Intelligence: A Modern Approach," *Prentice-Hall, Inc.*, Upper Saddle River, NJ, 1995.

Stanley, K. et al, "Real-Time Evolution of Neural Networks," *ISO Press*, Amsterdam, Summer 2006.

White, W., Demers, A., Kock, C., Gehrke, J., Rajmohan, R., "Scaling Games to Epic Proportions," *ACM SPecial Interest Group on Management of Data*, ACM, New York, NY, 2007.

Yasuda, K., Ide, A., Iwasaki, N., "Adaptive particle swarm optimization using velocity feedback," *International Journal of Innovative Computive*, Chicago, IL, 2003.



.7

Figure 2: These charts show the effect of diminished sightline (left) and diminished communication clarity (right) on the overall seek efficiency.