Complex Adaptive Behavior Exploration Simulator

Category A (competitive)

Final Report
April 10, 1995

Team 114
Los Alamos High School

Team Members:
Dave Bolme
Graham Booker
Gil Lundquist
Brian Olson
Eric Peterson

Sponsoring Teacher:
Lori Scott

Table of Contents

Executive Summary

Our project is to compare various learning algorithms and artificial life methodologies. The comparison is based on performance in controlling creatures in a simulated environment. We originally intended to compare S-systems, Classifier systems, Artificial Neural Nets, Fuzzy Cognitive Maps and a hard wired creature for reference. We didn't actually make the ANN's and FCM's. We found that the initial advantage of the hard wired creatures was so great that they couldn't be allowed in with the initial undeveloped versions of the other creatures. We haven't run the simulation for long enough to find significant differences between the other two.

Introduction

Our project was to test various learning algorithms by letting them control simulated creatures in an artificial world. There are many learning algorithms in use for various purposes, we wanted a comparison of strengths and weaknesses of several. Hopefully this could lead to more appropriate use of each of the algorithms. The algorithms we implemented are Classifier systems and S-systems. If we were to expand this we would add creatures based on Artificial Neural Nets (ANN) and Fuzzy Cognitive Maps (FCM).

The Environment

The environment is a square grid. There are dirt and food levels for all positions on the grid. The food grows only in places defined in a file. The file is set up so that the food grows only in patches. We decided that would be a good thing because it would give the creatures a chance to develop small communities. Also, the creatures would be given an advantage if they could somehow learn the position of these patches of food.

The dirt affects movement. Going from a place of low dirt to a place of higher dirt takes more energy than moving on level ground. We hoped that maybe a colony of bugs would wall off a patch of food and defend it. Also the creation of roads is then possible, the bugs would just level a section of the world. They could even make walled roads between two or more patches of food. We hoped that these creatures would eventually be very complex.

The world also has creatures in it. The generic creature has energy, health, and amounts of food and dirt being carried. A creature can move to the neighboring 8 places on the grid. It can do the following to the place it is at: eat food, pick up/drop food/dirt, attack another creature or reproduce. A minimum of health is required for reproduction.

Health and energy are interchangeable, but the exchange is not perfect. From energy to health, one energy point is worth two health points. From health to energy, four health points become one energy point. When energy is less than zero, a conversion is forced. When health is less than zero, the creature is dead.

Energy comes from food and is used in all actions except reproduction. Reproduction uses up half of the health of a creature if the minimum health is met. One fifth of that half is given to the child along with a small amount of energy.

To test the environment we wrote a simple hard wired creature that would go to food and eat it, reproducing when possible.

Classifier Systems

Classifier systems are another product of John Holland, the creator of the whole Genetic Algorithm paradigm.

Classifier systems are based on the buying and selling of messages. The messages are traded by a large set of IF-THEN rules. Each rule has a wealth with which it can buy messages. The rules check to see if it wants to buy a message or combination of messages. If the messages are desired, the rule will bid 10% of its' wealth. The sale goes to the highest bidder and the messages are removed from bidding. The seller of the message gets the money from the buyer, minus 10% sales tax.

Our program begins the cycle by selling sensory messages. Each individual rule may do any of the actions, but only the richest rule that can buy all of the messages that it wants is allowed to perform an action. Our program rewards the rules that bring about eating and reproducing by paying very large amounts for those messages. Eating gets payoff proportional to the amount of food eaten.

In a lecture by Holland, he stated that it may take a series of rules to get to an action that can be rewarded. For a while we weren't sure exactly how to make a chain of rules. The way we implemented it is to have it appear that all of the new sensor messages emanate from the previous turn's winner. Therefore a rule that moved to food would get paid off by the rich rule that actually eats. Because the rules pay 10% of their wealth, the moving rule would also get richer. This can extend on down several layers of rules, causing a set of profitable reactions to become well established.

It is possible to evolve a set of rules within a single creature. In our program the set of rules is constant within a creature. They can learn within their lifetimes by adjusting wealths. The only chance to actually change the set of rules comes at reproduction. The new creature can have a new or modified rule, or it can loose a rule.

S-Sytems

S-Systems work on the principle of tree-like chromosomes. These chromosomes are a lot like the chromosomes in living animals, but these chromosomes crossover more information during each reproduction. Contained in the chromosomes the the program for the S-Systems.

While I was coding the S-Systems I had to develop my own computer language for the S-Sytems to follow. This language included only the commands necessary for running the bug. The S-Systems are allowed to execute 50 commands per time step do to the fact that most of the command involve processing like if statements. The drawback to this it is possible the the S-Sytems to move 16 places away in a time step or do something else similar.

The commands are negative numbers in the tree and numbers that are not commands are put into a stack of 100 numbers. The stack is used when a command like move comes, move where? Two numbers are taken off the stack to tell the move command where to move to - x and y direction. When the stack goes over 100 the bug dies immediately, but if a number is needed off the stack and the stack is empty the number 0 is sent back in it place. There is one command that takes the next number and loads it into the stack in the case were an argument it a number that could be mistaken as a command. The commands are: picking up and dropping food, picking up and dropping dirt, eating the food it carries and the food on the ground, moving, loading arguments, looking around, asexual and sexual reproduction, if, attacking, loading the look variables, and random numbers.

S-System can be very powerful. Their creator, John Koza, first tested the S-Systems against UCLA’s learning algorithm. UCLA had a trail dubbed “The Santa-Fe Trail” that was broken into many pieces at the end. the bug had to get to square 89 in 200 time steps. a perfect bug could get to square 89. It took UCLA hundreds of generations to complete the trail but the S-Systems did the same trail in 7 generations. The S-Systems went on to figure out how to solve quadratic equations and learn that the square of the distance from a planet to the sun is proportional the the cube of it period all by itself!

When the S-Systems reproduce they take one of the command trees and duplicate it. Then one “branch” is broken off and the copy of a “branch” from the other tree is put in it’s place. This causes a lot of mutations as well as a lot of buggy code, but even the buggy code may do something right.

Results

When the hard wired creatures and the Classifiers were first run together, the Classifiers always died out almost immediately. It turned out that the hard wired creatures were much more efficient than the first attempts by the Classifiers. When the hard wired creatures were turned off, the Classifiers did eat and reproduce, but slowly. So far, all of the real test runs with Classifiers have been done at home on a 13 megaflop Power Macintosh. This is because Pi persists in giving a weird error that doesn't happen on the Mac. About a day of continuous running would be needed to see an improvement in the Classifiers.

Unfortunately, the S-Systems never got beyond debugging. We did still learn something from watching the hard wired creatures and the Classifier based creatures. It might be summed up as: "Automatic learning is great, but if the problem is small enough, hand written code will still do the job."