Candidate Topics for a Research/Programming Project

Description and title of two potential AI programming/research projects

Jacob Peck

CSC 466

Professor Craig Graci

Spring 2011

A machine learning approach to playing Mastermind, the code breaking game.

The board game Mastermind (a trademark of Invicta Plastics) is a game of code breaking based upon the earlier game Bulls and Cows. In the game of Mastermind, one player creates a code of length 4, from left to right, consisting of elements of 6 different colors. The colors may be repeated in the sequence any number of times to increase the possible amount of permutations to 6^4 or 1296. This code is hidden from the other player, who then has 10 chances to guess the code. After each guess, the code maker assigns a number of black and white pegs to the most previous guess, with a black peg signifying that there is a correct color in a correct space, and a white peg signifying that there is a correct color, but it is in an incorrect space. For example, if the hidden code was `(blue black white red)` and the guess was `(red green white black)` there would be a result of `(b w w)` because the white pin is the correct color in the correct spot (3^{rd} position) and the black and red pins are correct colors, but not in the correct spot. The green pin here doesn't match anything in the secret code, so it earns no pins. With this scheme, a perfect guess will result in `(b b b b)`, and a successful code break.

There is an algorithm to guess the code in as little as 5 tries reliably, however I would ignore that algorithm for anything other than a benchmark. The core of my project would be an implementation of a learning algorithm applied to this relatively simple game. A brute-force technique is out of the question here, as 1296 possibilities far outshadow the 10 guesses a player gets. So, a machine learning implementation of the game Mastermind would be an interesting approach to this (admittedly flawed, but still fun) problem.

Artificial Life Music – Algorithmic Composition with coupled Lindenmayer Systems and Cellular Automata

Algorithmic composition – the process of composing music with the assistance of well-defined processes – can benefit from certain subsets of artificial life. One of these, Lindenmayer Systems (henceforth L-systems) are systems that model organic growth. They are a subform of Chomsky's context free grammars, where the set of terminal symbols is the same as the set of non-terminal symbols, there is typically only one choice for each rewrite, and such rewrites happen as frequently as is desired. For an example, take the alphabet `{A, B}`, with the rules `{A->AB, B->A}`. Given the seed of "`A`", one sees a progression from `A` to `AB`, to `ABA`, to `ABAAB`, to `ABAABABA`, etc. As you can see, these are very self-similar, yet non-symmetrical, making them rather suitable for melody or rhythm generation.

Cellular Automata, on the other hand, are simulations of various natural phenomena. Perhaps the most famous example, Conway's Game of Life, simulates bacterium growth. In a CA system, the state of each cell in the next generation is determined by the states of its neighboring cells in the current generation. The rules for Conway's Life, for example, state that a live cell remains alive in the next generation if has exactly 2 or 3 live neighbors in the current generation, otherwise it dies (either of isolation or overpopulation). A dead cell comes alive in the next generation if it has exactly 3 live neighbors in the current generation, simulating reproduction. These are very useful as selection grids, if each cell is weighted according to some value or a history of its previous states.

The idea that I am considering is having a cellular automata selection grid (with weighted values based upon individual cell history) selecting various L-systems, one per cell. The sheer amount of L-systems this would require would be enormous without some doubling or tripling of assignments, and very tedious to input by hand. So to that end, I intend to allow the program to simply take in one single L-system, and mutate it as it will. I may or may not have a fitness test for these mutations, but the idea is that there will be a very small amount of genetic programming in this as well, to allow for crossover of rules and the like.