An Investigation of Genetic Algorithms Using Audio Output

Matthew Thompson

December 14, 2004

Abstract

This paper documents my work of researching and testing genetic algorithms.

1 Introduction

Genetic algorithms use feedback resulting from evaluating data sets to optimize these data sets for optimum performance, where optimum performance is defined by the user. The main data processing is done in LISP. The program has a simple shell script as its frontend. CSound is used to convert the data sets to audio files, which are heard and evaluated by the user.

2 Research

The first area of research was into various forms of genetic algorithms[1]. Topics covered included different methods of storing data, such as in a tree, list, or array. In a tree, mutation operators include subtree destructive, node swap, and subtree swap, and a single point subtree exchange as a crossover method. In a list, mutations can be generative, destructive, element flip, node swap, or sequence swap, and crossover can be single point or order based. In an array, mutations can be destructive, element flips, or element swaps, and crossovers can be single point or variable length. The second area of research was into music theory, to ensure that the program would, even with random data, produce something that sounded decent. To do this,

I wrote the program such that a melody will stay on key, and used a hash table of notes and frequencies[2] to accomplish this.

3 Program Development

The initial program was made to store data in lists, mutate using element flips, and crossover being single point. I used this type of genetic algorithm because at the time of starting the project, it was the type of genetic algorithm with which I was most familiar. After finishing a simple score processing function that would turn a list of numbers into a usable audio file, I integrated this with the genetic algorithm code so that the algorithm's evaluation function was user input telling what the user thought of the melody a specific population member created. Melodies were rated on a scale of 1 to 9, with higher numbers indicating a stronger like of the melody. A shell script was written to serve as a frontend for the algorithm.

4 Program Testing

Testing of the program involved running it over repeated trials, using different data storage, mutation, and crossover methods, and observing trends in the improvement of the melodies created by the program.

References

[1] Intro to Genetic Algorithms
http://lancet.mit.edu/ mbwall/presentations/IntroToGAs/index.html

[2] Frequencies of Musical Notes
http://www.phy.mtu.edu/ suits/notefreqs.html