Computer Systems Lab, 2006-2007
Applications of Genetic Algorithms

Harry Beddo

June 7, 2007

Abstract

My main areas of interest within Computer Science are machine learning and artificial intelligence.
Keywords: genetic algorithms

1 Introduction - Problem Statement and Purpose

The purpose of these programs is to apply genetic algorithms to find solutions to different problems. Genetic algorithms are a search technique that is used to find an approximate and sometimes correct answer. Usually the data being searched through is very large and the GA will be able to find a close answer. The GA imitates evolutionary biology by using generations of populations to create better generations until the optimal solution is found.

2 Genetic Algorithms

Genetic Algorithms are search techniques that find approximate solutions to search problems. The idea of genetic algorithms is inspired by evolutionary biology, "only the strong survive." Every individual in the population is made up of genes. Two individuals are matched in order to mate and exchange data. The idea is that the offspring of the two parents will be better than the parents. Through each generation, only the best individuals survive according to a cost function. The process continues until the population begins to converge.

3 Procedures

3.1 Creation

The initial population is put into an array consisting of 24 random chromosomes. Each chromosome is made up of fourteen random ones and zeros: the first seven digits refer to the x-coordinate, and the last seven refer to the y-coordinate. These values refer to discrete values of longitude and latitude on the map. There is also another array of costs. The values in the cost array

are the elevations at the locations of the corresponding chromosomes. The cost is negated in order to put it into the form of a minimizing algorithm. A large initial population will provide the genetic algorithm with a large search space. Usually, not all of the chromosomes from the initial population will survive the subsequent steps.

Next, the costs and their associated chromosomes are sorted from least to greatest. Only twelve chromosomes go through the iteration process. The top six are kept for each successive round and the bottom six are discarded. From the initial population, the top six from the original twenty-four are kept.

3.2 Pairing

The next step in the iteration process is to pair two chromosomes from the six remaining. There are many different ways the pairing process can be done. This is the step that I will have to experiment with to see which option is best. There are three options I will explore. First, is just simply pairing the first two, second two, and so on. Second, I could randomly select the pairs based on weights, with the lowest chromosome having the most weight to be selected as a parent. Third, I could use a tournament style selection. This style will select a small subset of any chromosome in the population and the chromosome with the lowest cost function of that subset is chosen.

3.3 Mating

Once the chromosome pairs are determined, the mating process begins. A crossover point is selected for the two parent chromosomes (p1, p2) to exchange bits and form two offspring (o1,o2). First, p1 passes its binary code from the left of the crossover point to o1. Next, p2 passes its binary code from the right of the crossover point to o1. Then, p1 passes its binary code from the right of the crossover point to o2 and p2 finally passes its code to o2. This process results in each offspring carrying portions of code from each parent. The population is doubled from the mating and goes to a size of twelve.

3.4 Mutation

Mutations then occur in the resulting population. A mutation will change a single bit from a 1 to a 0 and visa versa. Only .05 of all of the bits in the population and none from the best chromosome will undergo a mutation. A larger number of mutations will allow the population to search more outside the convergence path into new territory, while a smaller number of mutations converges the population quicker.

3.5 Checking

After the mutations take place, the new costs of the offspring are determined and the array is sorted again. This process is iterated again until a designated point. After a certain number of iterations, the population would not change if it were not for mutations. At this point, the search needs to be stopped and the current best chromosome should be the location of the highest elevation point.

3.6 Language and Algorithms

The main algorithms I will be researching is:

1. Genetic Algorithm

Computer language I'll use:

1. Python

4 Applications

Basic Applications using Genetic Algorithms:

1. "Highest Elevation"

   In order to find the solution without checking every point in a map, genetic algorithms are a useful tool. The individuals with the better characteristics will survive in a population, while those with the worst will die out. The solution should be found once the population begins to converge. The highest elevation is not always found but will usually find a close answer. The advantage to using a gentetic algorithm to find the solution is that a genetic algorithm will only check a fraction of the data, while a normal search algorithm has to check all of the data.

2. "Mary Had a Little Lamb"

   Genetic algorithms will be used to learn the first couple measures of the tune. This is a simple tune with only quarter and half notes. Each note and the hold will be represented by 3 bits:

000	hold
001	A
010	B
011	C
100	D
101	E
110	F
111	G

   For just the first sixteen notes, there are 2^48 possibilities (2.8147 * 10^14). The genetic algorithm will try to find the answer without having to check each of these possibilities. There are two different cost functions that can be used to find the solution. The first would be for the computer to count the number of errors between the current notes and the solution. The other solution would be to play the notes and have a human rank the notes by their ear.

3. "Word Guess"

   In the word guess program, genetic algorithms are used to guess a correct 8-letter word. A simple search would look at all the possibilities

(26^8 or 2.088*10^11). A GA will only search a fraction of those possibilities to find the solution. Another purpose of this program is to distinguish between the choosing of two different cost functions. One cost function is the sum of the squares of the distance between the letters and the true answer. The other cost function is to simply give a one to an incorrect letter and a zero to a correct letter. The latter cost function tends to have parents with more correct letters than the former cost function.

5 Discussion

These three programs written were small examples of how genetic algorithms could be used. Future explorations of genetic algorithms could include simulations of a virus spreading throughout a population or traffic flow in a city with traffic lights. Each of these examples have a large set of possiblities that could be searched through with a genetic algorithm easily. The overall purpose of these programs was to find the optimal solutions without evaluating all the pieces of data. If working correctly, each program should only have evaluated a fraction of the total data available and given a result exactly or close to the solution.

References

[1] Haupt, Randy L., and Sue Ellen Haupt. "The Binary Genetic Algorithm." Practical Genetic Algorithms. 2nd ed. Hoboken, New Jersey: Wiley-Interscience, 2004. 27-50.