Hello World in 10 Hours:
Optimizing Mutations in Genetic Programming

Ben Eysenbach

12/17/2012

1 Introduction

Genetic programming is well known to produce close to optimal solutions to many problems in short running time. However, the efficiency of genetic algorithms greatly depends on constants chosen to serve as biological analogues in the algorithms (e.g. population size, mutation rate).

This project attempts to find the optimal rate of mutations and number of mutations. That is to say, how many children out of a population should mutate, and how many mutations should occur for each individual.

2 Methodology

2.1 Algorithm

The genetic algorithm for printing 'hello world' works as follows:

1. A population of children is generated by randomly combining letters and spaces until the children are as long as 'hello world.
2. The 'fitness' of each child is tested by comparing the child to 'hello world.' For each child, if the fitness of the child is greater than or equal to the fitness of the next child, the becomes a parent for the next generation. Otherwise, the child is discarded.
3. To create the new child generation, two parents are chosen at random, and their letters are randomly combined.
4. A select number (depending on the mutation ratio) of children are mutated n times, where n is the number of mutations previously specified.
5. The process is repeated until the first parent equals 'hello world' within the accuracy bound.

2.2 Testing

For a given number of mutations, the mutation ratio was incrementally increased from 0 to 100, and the average (of 20 tests) number of generations required for the population to converge upon 'hello world' was recorded. The simulation was run for 1, 2, and 3 mutations.

The following constants were used for the simulation:

- Population size: 100
- Accuracy: 100
- Cutoff: 1000

3 Data

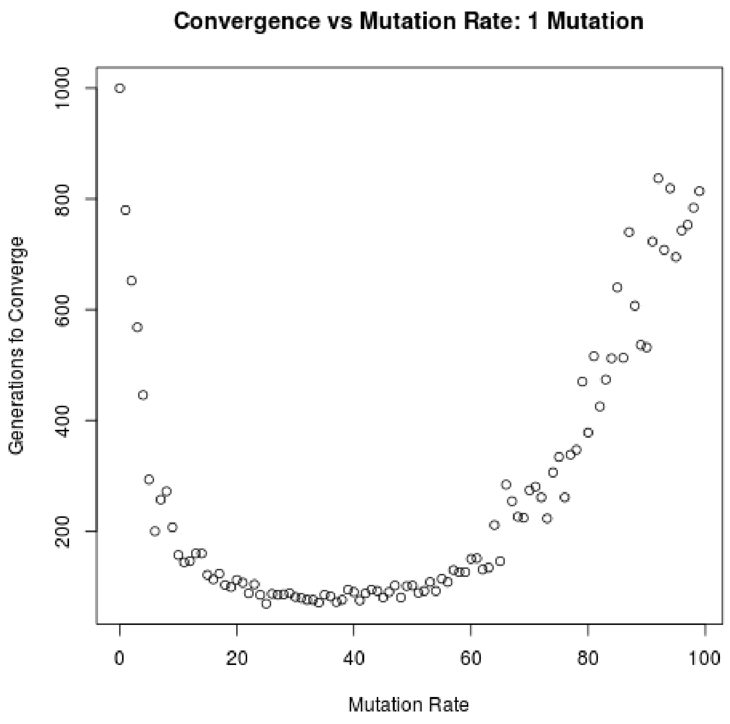

Figure 1: Average Convergence vs Mutation Rate with a single mutation

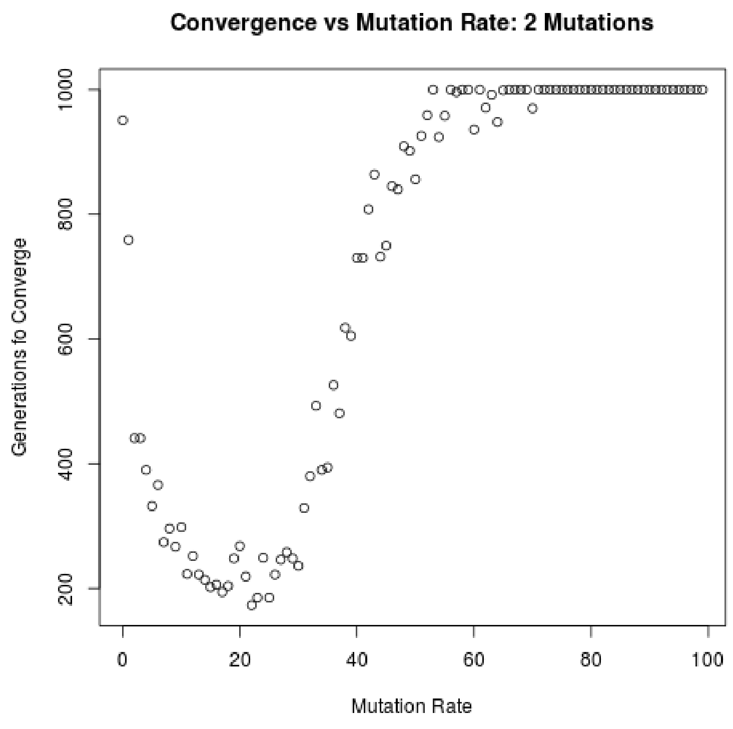

Figure 2: Average Convergence vs Mutation Rate with 2 mutations

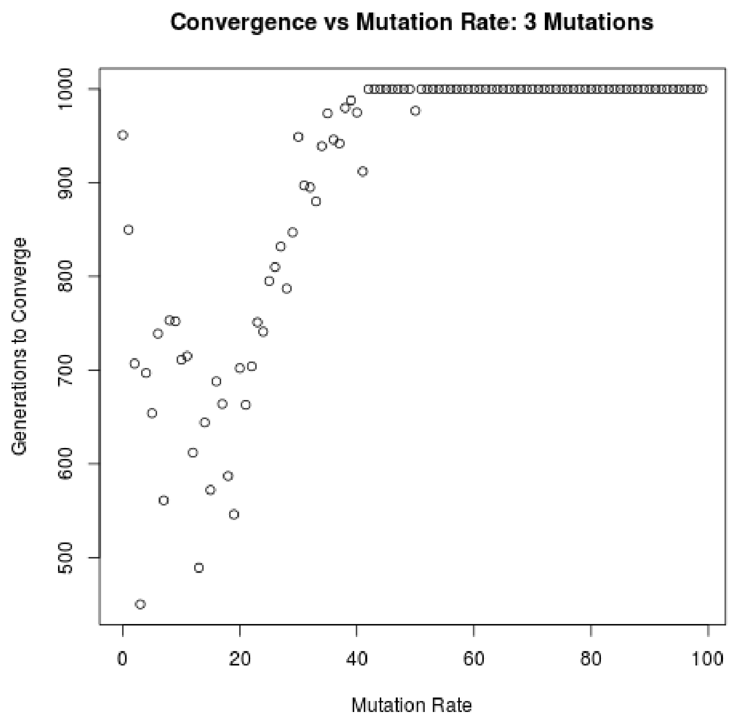

Figure 3: Average Convergence vs Mutation Rate with 3 mutations

4 Analysis

The data above makes clear two points: certain numbers of mutations cause the algorithm to converge faster than others, and certain mutation ratios cause the algorithm to converge faster than others.

This project tested convergence with between 0 and 3 mutations, inclusive. With no mutations, the average number of generations to converge was 1000, the cutoff. That is to say, the algorithm did not converge with no mutations. Choosing the optimal mutation rate, the minimum generations required to converge is about 180 with one mutation, 200 for 2 mutations, and 600 for 3 mutations. In short, the algorithm converges fastest with a single mutation.

The other variable tested was the mutation rate, the number of individuals out of the population who were mutated. The three graphs make clear that there are certain mutation rates that required significantly fewer generations to converge than others. More specifically, with a single mutation (Figure 1), the ideal mutation rate is between 30 and 40 percent. With 2 mutations (Figure 2), the ideal mutation rate is about 20 percent. With 3 mutations (Figure 3), the ideal mutation rate is also about 20 percent.

5 Conclusion

Overall, this project showed that a genetic algorithm can be made significantly more efficient by adjusting certain parameters, namely mutation rate and number of mutations.

One possible improvement to these tests would have been to specify the total number of mutations. Instead of multiplying the number of mutations by the mutation rate to calculate the total number of mutations, specifying the exact number of mutations would allow for the ideal number of mutations to be determined more accurately. Also, these tests could be repeated for strings of varying length.

Combined, both these improvements would probably allow for a relationship to be found between string length and and the ideal number of mutations. But alas, running these tests alone took nearly 11 hours to run (albeit on an Atom processor). Someday, when Moore's Law catches up, the most efficient way to print 'hello world' will finally be determined!