I Can't Believe It's Finally Over

By: Zachary Reichenbach

Final Comment:

I just want to start this report off by saying……. LORDY LUE THAT WAS AN INSANE PROJECT AND I HAVE NEVER BEEN MORE RELIEVED OR PROUD OF MYSELF FOR POWERING THROUGH SO MANY HOURS WORTH OF PAIN.

Now that that is out of my system, let's move on now, shall we?

Zak Reichenbach
12/8/2019

1.) **Changing the length of the target phrase:**

My program is currently set to run until 18 space long phrase "To be or not to be" is generated. The number of generations varies widely, sometimes being completed in some 200 generations to exceeding 2000.

To experiment, I changed the phrase to the 5 space long "Hello", and it was able to generate the phrase. However, it did not properly calculate the fitness when the word was achieved, so the loop did not stop. However, it generated the phrase so I am going to take that as a W. Surprisingly, it also seems that the shorter-phrase takes longer to generate, some 3000-4000 generations till the first "Hello" was made. Interesting.

2.) **Changing the population size:**

Now looking into the population size, I started out by initializing a population size of 200 people, but the larger that gets, the quicker the phrase is generated.

Experimenting, I made the population size 500 just to test. This made the phrase (instead of completing at a minimum of just over 200 generations) completed in just 46. Which is quite impressive if you ask me.

So one could safely assume that the speed in which a phrase is generated is inversely proportional to that of the population size.

3.) **Changing the mutation rate:**

As asked of in the project, I have made the standard mutation rate 4%, and decreasing it usually makes the program complete faster, or not at all as I have learned from trying to make this godforsaken program work in the first place.

Cranking the mutation rate up to 20%, the phrase was unable to be completed. I assume because the mutation of the child DNA happens so often, that it does more harm than help when generating the phrase at a certain point. The fitness plateaus around 0.1-0.8 and nothing more.

4.) **Changing the range of generated characters (including numbers):**

I'm going, to be honest, I'm not going to do this because it involves adding more to the array that generates the initial strings in the first place. I have neither the time nor energy to do it and I do not care enough even to feed myself right now let alone do that.

However, I will go through my theory of what would happen.

Thinking about all of the added characters, the final string will be much harder to generate since there are more individual characters to pull from that are not similar to the phrase we are looking for. So I can assume that it would take quite a bit longer to generate, and probably even break my code.

5.) **The two breeding methods:**

Through thorough testing, I have found that the midpoint breeding method works far faster and more frequently than that of the random section breeding.

The random section breading, much like having a high mutation rate, seems to hurt the process near the end when the fitness is approaching 1. It too often rids the DNA string of the needed matching characters, resulting in much longer processing time on average and it was also shown to sometimes never finish at all no matter how many generations I let it run through.

6.) **Changing the mating factor:**

It is plain to see that changing the mating factor will give an edge when computing the string that we desire. When creating the ticket pool the mating factor helps give more tickets to the fitter strings, resulting in the desired result being bred faster from the more fit members.

However, when I tested this by doubling the mating factor (from 10 to 20) it seemed to be the same type of results. Varying widely from lucky cases of 400 generations to well over 4000. Not what I anticipated, but interesting none the less.

7.) **Changing the maximum generations:**

Unfortunately, any attempt I made trying to implement a generation limit did not go so well. Also, the more I tested the more pointless it seemed since most of the time it struggled to make phrase at all. But I can tell you that all you do by limiting the number of generations is just stoping some attempts from ever reaching their goal. So the fitness in question will just sometimes stop at the highest of its generation whether it reaches the desired score of 1 or not.

8.) **Which function takes the longest:**

With all things considered, no. I haven't a clue how I would go about doing that, or why I would. Nor do I see any need to try and reduce the time it takes to run any of them. While I bet there are many coding practices I could adopt to make sure that the code runs more efficiently and smoothly, it already generates each generation faster than I can see. So I'm good thank you very much.

Comments to the whole cubing thing:

I'm just happy that my code is done. That is beyond excessive. But I did a report on what I can do in a timely manner without feeling more frustrated and depressed than I already am with all of the other things going on. This project was the longest and most frustrating thing I have ever worked on and some of these things tacked on to the end of the report seem downright ridiculous. However, they are cool to read about it consider.