Jackson Hacker

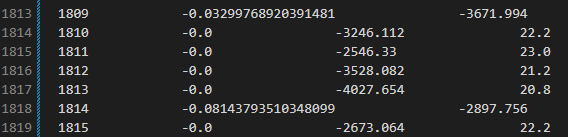
CS 5146

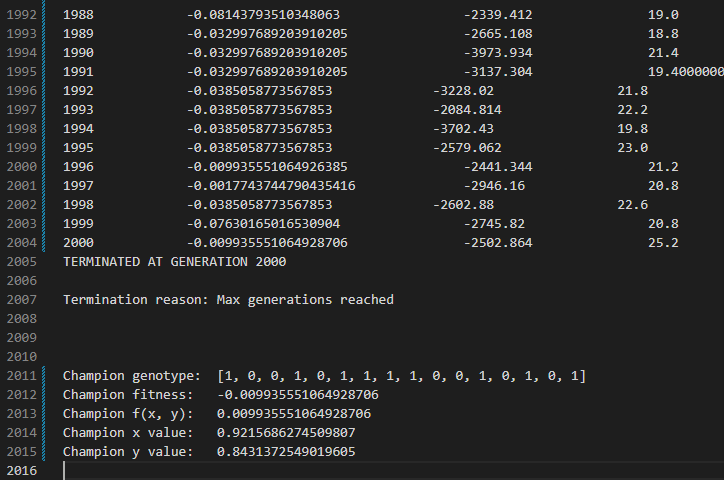
Professor Gallagher

23 January 2023

Why My Rosenbrock Program Kinda Works

**Below is a sample output of my Rosenbrock Program after 2000 generations:**

****



In order to minimize the function, I took the fitness function to be the negative value of the Rosenbrock function, and maximized that negative value to minimize the function. The program *is* definitely working (to an extent), as champions are getting stronger (fitness is generally increasing generation over generation), and, often, the answer my program gives is pretty close to the minimum of (1, 1). The program even produces the answer in some generations (champion fitness is 0, meaning we found the minimum; can be seen in one of the above screenshots). However, my program does not terminate at a reasonable point, meaning there is a flaw with my termination condition (or representation, or parent selection).

**Flaws with my Termination Condition:**

My program terminates if the average fitness does not change by more than 1% over 10 generations (or if 2000 generations have gone by without convergence). Although my program is finding the answer (champion has minimized the function), my algorithm is not halting because the average fitness is not converging quickly enough. This is likely the fault of my representation, parent selection, or a combination of the two. Alternatively, I could have selected a different termination condition that better recognizes when I’ve landed at an answer.

**Possible flaw with my Representation:**

I went with perhaps the easiest and most common form of representing these two numbers: a single bitstring, in which the front half represents a scale to be applied to a range for the x value, and the back half does the same for the y value. These scale values from the bitstring are converted into decimal, then divided to become a scale factor applied to a range (in this case -5 to 5 for each axis). The main issue I see with this representation is that it perhaps does not evolve or mutate particularly well over time. What I mean by this is that a single bit change can swing the x and y values quite a bit, so maybe something like gray coding, like we discussed in class, would be better to keep these values from swinging quite so much, allowing the algorithm to operate more predictably.

**Possible flaw with my Parent Selection:**

I used roulette wheel selection here (as per the instructions), which may be inherently flawed, but I think maybe my implementation was not ideal either. I went with a linear scale for my roulette wheel, as in the least fit member of the population is given 0 spots on the roulette wheel, and the most fit member is given a number of slots equal to the total size of the population. I think that the issue with this, in Rosenbrock, is that it gives the weakest members of the population *too much* of a chance to be selected as parents. Perhaps a better solution would be a roulette wheel that gives each fitter member of the population an exponentially better chance of being selected than the next worse individual. This would hopefully help the average fitness of the population to actually converge, and to converge more on good values, thus giving even better parents to select from generation over generation.