Christopher Ginac

[christopher.ginac@nevada.unr.edu](mailto:christopher.ginac@nevada.unr.edu)

Personal Webpage: <https://cwginac.github.io/>

Part 1:

My hillclimber starts at any given state and then evaluates the next state given back from an increment function. If that next state is greater than, or equal, it will continue to evaluate the next state and loop until the next state evaluates at a lower fitness. Unfortunately, that was the only part that was easy at first. I approached the problem as one binary number rather than 100 separate bits. Because I chose this poor approach, my increment function incremented the whole binary number and I was never able to find the best fitness. A friend suggested to treat them as separate bits, and I ignored this suggestion until it was said in class that his suggestion was the correct way to approach the problem. I threw away most of my work (you can see the github history through the link on my website), and rewrote both the init and the increment. The algorithm for the hillclimber didn’t really change (the implementation changed slightly to account for the new increment).

The reason my algorithm works well on this problem is because there aren’t local maximas, just the global maxima. Because I can start at a known state (all 0’s), and the maxima is just uphill from there, my hillclimber can finish with 50 increments. The same reasons for eval.o also hold true for eval1.o. This time the only catch is there is a shoulder (at least according to what was said in class), and my hillclimber already accounted for that by checking for equality in addition to if the new state had a higher fitness.

Part 2:

The eval I wrote gave a point for a 1 bit in even positions and a point for a 0 bit in odd positions. This trips up my hillclimber pretty quickly because the increment function doesn’t handle this change in scoring. In fact, this is a really important note in the modify/increment function. If you don’t know how the evaluation happens, it is really easy to assume the wrong way to proceed forward. That next step may not really be related to the previous step (as I found out early on in this assignment). Hillclimbers only work if you know how one state relates to the previous state.