Fernando Recinos

COSC 4368

Dr. Eick

02/13/2024

Randomized Hill Climbing Report

1. The results of the Randomized Hill Climbing Algorithm were obtained by writing an algorithm in python that checked each combination of p and z along with the starting point. The seed was originally set to 42 and a table was generated as a csv and then again for seed 43, another csv was generated.
2. The tables can be seen on RHCResults.xlsx. Download all on GitHub: https://github.com/nandoj28/AIAssignments/tree/main/Task1
3. After viewing all the results from the 8 different tables. It was shown that all 8 tables agreed on a minimum. Specifically, if the starting point was (-510, 400). The z and p values did not affect whether or not the minimum was found. The seed did affect the total calls of the functions. When the seed was set at 43, it found the minimum in a total of 723 calls. Which was a significant call difference compared to others. Thus, the parameters chosen for the 33rd run were:

   p = 120, z = 50, starting points = (-510, 400), seed = 43.

   The results were:

Run	Solution as (x, y)	Minimum of fFrog	Function Calls
Solution 1	(-499.044012874095, 404.0034464181656)	-492.827293	241
Solution 2	(-498.658231026949, 404.91245710024157)	-494.163148	241
Solution 3	(-498.545160263755, 404.8289746436277)	-494.164297	241
			Total: 723

4. For the parameters that were chosen, the algorithm performed well and efficiently. Compared to the rest of the results, it seems to have found the minimum. The starting point seemed to have the most significance when it came to finding the minimum. It just so happened that the starting points were already fairly close to the ending points. Also, an interesting insight was that setting the seed at 43 reduced the number of recursive calls needed to find the minimum. It shows that although it's random, it can be tricky to choose a random start point. Random number (seed setting) generators could also perform different on different systems.
5. The resampling done by the algorithm was the final touch to get closer to the minimum. The first sample was fairly close to the minimum but depending on the situation, the two-point difference could mean a lot. Choosing different p's and z's don't seem to hold much of a significance to finding the minimum because through all 8 tables, the minimum was found. RHCR2 did a great job at computing the minimum but only if the starting points were close to the solution. If the starting points aren't set correctly, the RHCR2 seems to get stuck on local minimums. There could be a higher global minimum but from the variety of starting points, it seems like the algorithm got fairly close to it or if not, found it.