Intro to AI: Hill Climbing Project

Daniel Eppinger

May 17, 2016

1 Three Search Algorithms

I implemented the three algorithms, hill climbing, hill climbing with random restarts, and simulated annealing, as functions in a python program. The program calls each of the algorithms and passes in the required parameters. One such parameter is a function and the algorithm's job is to find the global minimum of the two variable function given the maximum and minimum for x and y. In order to change the function used, the content of the function must be changed to calculate the z value of the desired function given an x and y value.

1.1 Hill Climbing

This algorithm worked about just as well as it would be expected to. On occasion, it would find the global minimum, but on average it would only locate a local minimum. For this algorithm, I used steepest ascent hill climbing. So instead of randomly choosing a neighboring point, it examined all neighbors within one step and chose the one with most decrease.

1.2 Hill Climbing with Random Restarts

This algorithm is extremely similar to hill climbing. It simply runs the hill climbing algorithm a given number of times and takes the best value given. This obviously takes longer than hill climbing; however, it should always return a better answer than standard hill climbing. In the tests with the given function, it always returned the global minimum when running the hill climbing algorithm a total of eleven times. The amount of restarts needed is based on the function and search space, so on some functions, this algorithm may need to run hill climbing a large number of times, and would most likely be fairly inefficient.

1.3 Simulated Annealing

Simulated annealing is different from both versions of hill climbing, but it uses a somewhat similar algorithm. This algorithm took the longest to write and run, and in the

given function it did not perform very well. Although it did better than hill climbing on average, this is not impressive as hill climbing is bad at finding the global minimum on a function like this. Despite this, simulated annealing would be better than the others if the function had more local minimums and a larger search space. So although simulated annealing did not do well with this function, it would perform better than either of the other algorithms if given far more complicated functions.

2 Conclusion

It seems that overall, no one algorithm is better than another. While one may appear to have a stronger performance, the other algorithms would be more optimal in different situations. It also seems that the more complicated the algorithm is, the longer it takes to run, but the better job it does at search more difficult functions. This observation may seem obvious, but it is also important to note that the simplest algorithm, hill climbing, would be better for searching a function with only minimum than simulated annealing or hill climbing with random restarts would. While the other two algorithms should also return similar answers, hill climbing would do so more efficiently and a lower run time.