K. Hohmeier

CSC 380

Homework 7 Part I

Due November 12, 2020

<u>Part 1: Hill Climbing for TSP</u>

<u>Part A: Empirical Estimation of the Big-O of Hill Climbing for TSP</u>

We wish to conduct experiments to determine how many guesses, on average, the hill climbing algorithm performs as the size of n, the number of nodes, grows for the traveling salesman problem. From that information, we will attempt to estimate the Big-O efficiency of the hill climbing algorithm when it is applied to the traveling salesman problem.

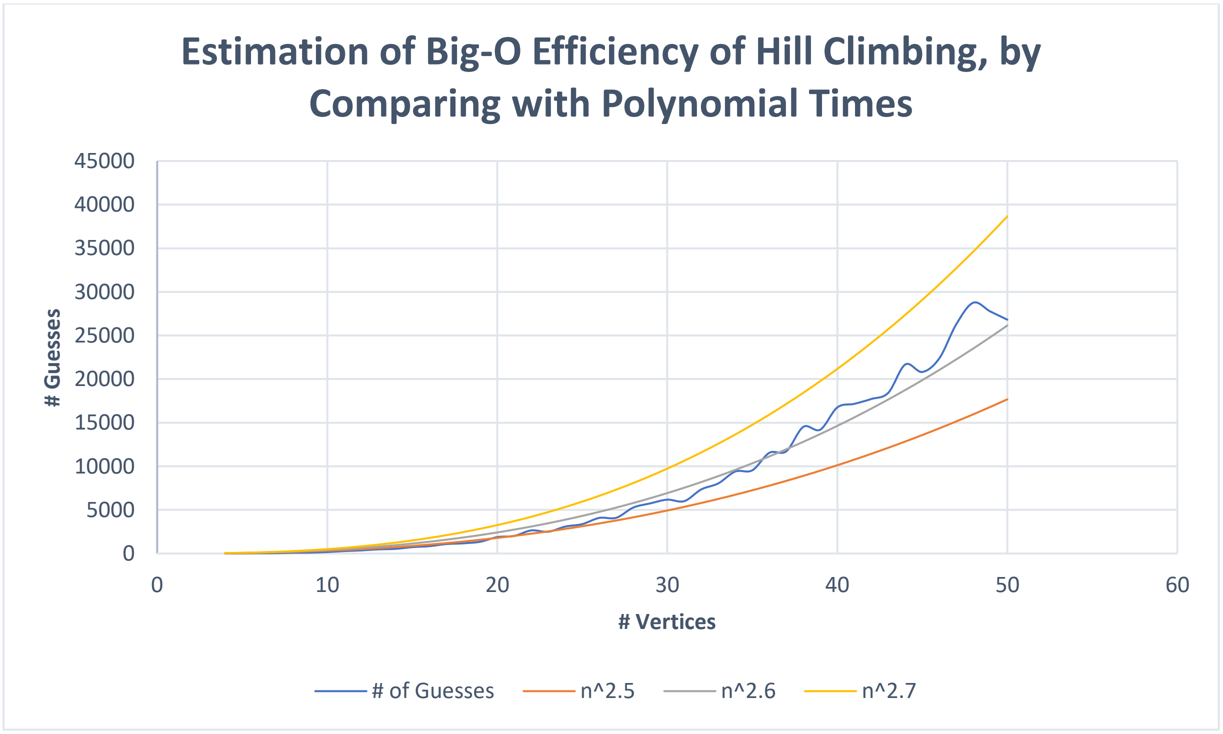

Figure 1. Graph of results of experiments with hill climbing applied to the traveling salesman problem. Three additional graphs of $n^{2.5}$, $n^{2.6}$, and $n^{2.7}$ are shown for comparison.

We will estimate the Big-O of this algorithm by graphing the number of vertices of the graph (that is, the number of cities visited) against the number of guesses made by the hill climbing algorithm. The resulting graph of these experiments is shown in Figure 1, along with three polynomial times ($n^{2.5}$, $n^{2.6}$, and $n^{2.7}$) for comparison. The $O(n^{2.6})$ graph seems to approximate the hill climbing algorithm graph very well. Certainly the Big-O of the hill climbing algorithm seems to be bounded between $O(n^{2.5})$ as a lower bound and $O(n^{2.7})$ as an upper bound. We claim that the efficiency of hill climbing in the context of the traveling salesman problem seems to be $O(n^{2.6})$.

With this estimation, we will try to extrapolate this function to predict the values for three other values of n (the number of vertices). Since for the experiment the values of n used were 4 to 45, values of 55, 65, and 70 will be used for predictions. Using the $O(n^{2.6})$ claim for efficiency, the predicted number of guesses for these values of n, as well as the actual values, are shown below in Table 1. Percent error was calculated using the following formula:

$$\%\ Error = \frac{experimental\ value - theoretical\ value}{theoretical\ value} \times 100\%$$

Value of n	*Predicted Number of Guesses*	*Actual Number of Guesses*	*Percent Error*
55	33392	49228	-31.97%
65	51710	67739	-23.66%
70	62698	80235	-21.86%

Table 1. Results of extrapolation of the function estimation for the Big-O of the hill climbing algorithm, compared with actual results.

From Table 1, the $O(n^{2.6})$ function appears to underestimate the actual number of guesses, but since the behavior of the experiments does seem to indicate an $O(n^{2.6})$ trend, perhaps an $O(n^{2.6})$ function multiplied by a constant would be a better approximation. For example, when using $1.3n^{2.6}$ as the function approximation for the hill climbing algorithm, this generated the results shown in Table 2. A constant multiple of the O($n^{2.6}$) function seemed to better approximate the results of the experiments on the hill climbing algorithm.

Value of n	Predicted Number of Guesses	Actual Number of Guesses	Percent Error
55	33392	43540	-11.55%
65	51710	67223	-0.76%
70	62698	80235	1.59%

Table 2. Updated results of extrapolation of the function estimation for the Big-O of the hill climbing algorithm using $1.3n^{2.6}$, compared with actual results.

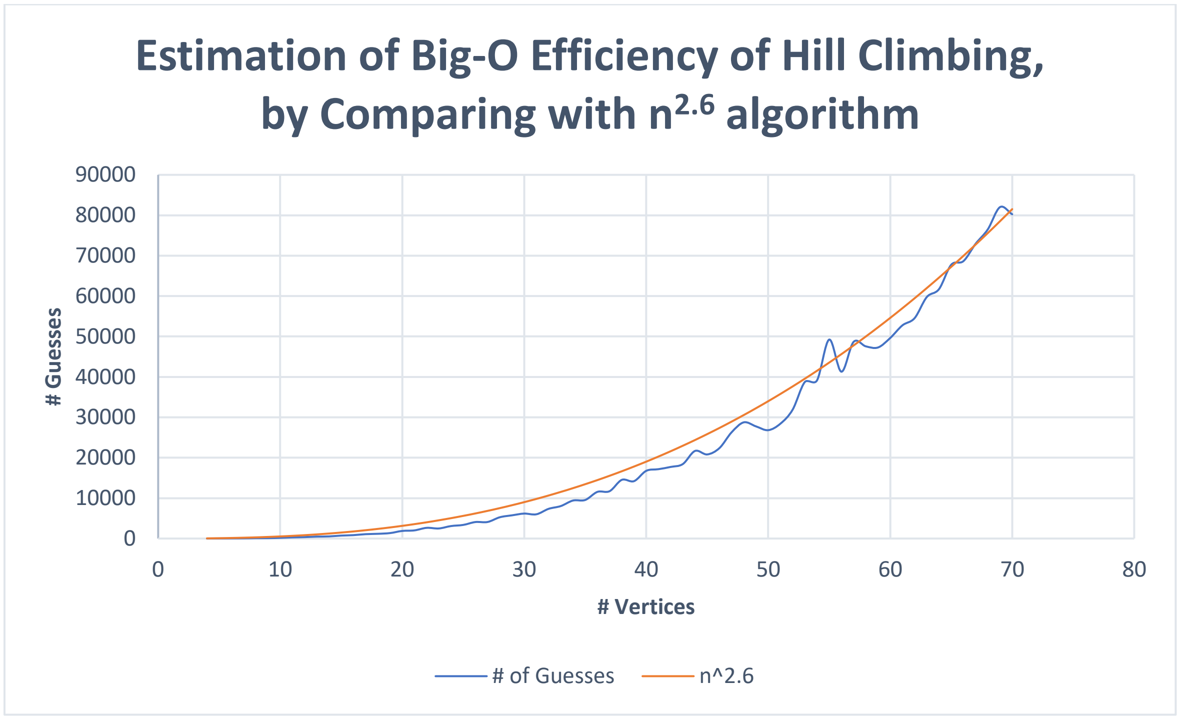

Figure 2. Estimation of Big-O efficiency of hill climbing as applied to the traveling salesman problem, compared with $1.3n^{2.6}$

<u>Part B: Empirical Estimation of Suboptimality of Hill Climbing without Random Restart for TSP</u>

To estimate the suboptimality of hill climbing without random restart, the cost of the solution found by hill climbing for the traveling salesman problem was compared against an optimal solution. This optimal solution was calculated by multiplying 10 by the number of vertices in the graph (which represents the number of cities visited) to generate the “ideal cost” of a path. From this information, percent error was calculated between the actual solution cost and the optimal solution cost for multiple values of n, the number of vertices. The formula used is that shown in part A. A graph of these results is shown in Figure 3.

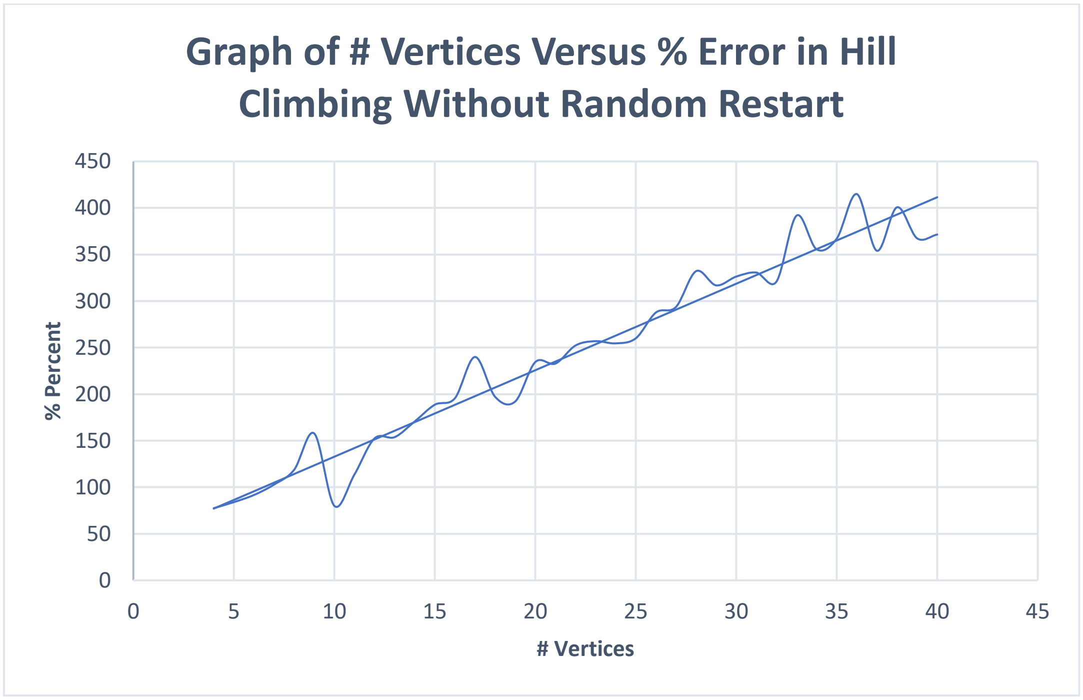

Figure 3. Graph of number of vertices versus percent error for the hill climbing algorithm without random restart.

Clearly, the optimality of hill climbing without random restart does change as the number of vertices increases, and also fairly consistently returns a path cost far greater than the optimal path cost. It appears to follow a roughly linear pattern (based on the trendline), with the percent error increasing as the number of vertices increases. This also indicates that the optimality of hill climbing is “good” but not very close to optimal when implemented without random restart. These results are summarized in Table 3 below.

# Vertices	Avg Cost	Optimal Path	Percent error
4	71	40	77.5
5	92	50	84
6	115	60	91.66667
7	142	70	102.8571
8	175	80	118.75
9	232	90	157.7778
10	180	100	80
11	235	110	113.6364
12	303	120	152.5
13	330	130	153.8462
14	379	140	170.7143
15	433	150	188.6667
16	473	160	195.625
17	578	170	240
18	535	180	197.2222
19	555	190	192.1053
20	669	200	234.5
21	699	210	232.8571
22	775	220	252.2727
23	821	230	256.9565
24	851	240	254.5833
25	900	250	260
26	1008	260	287.6923
27	1063	270	293.7037
28	1210	280	332.1429
29	1209	290	316.8966
30	1279	300	326.3333
31	1335	310	330.6452
32	1347	320	320.9375
33	1623	330	391.8182
34	1549	340	355.5882
35	1635	350	367.1429
36	1854	360	415
37	1680	370	354.0541
38	1903	380	400.7895
39	1823	390	367.4359
40	1886	400	371.5

Table 3. Results of optimality tests for the hill climbing algorithm as applied to TSP.