K. Hohmeier

CSC 380

Homework 7 Part 3

Due November 24, 2020

Part III – Genetic Algorithm for TSP

Part A: Big-O of Genetic Algorithm

In this report, we will be studying and analyzing the genetic algorithm as applied to the traveling salesman problem. For this section, we will first attempt to estimate the Big-O efficiency of the genetic algorithm by performing experiments to find the average number of guesses made by the algorithm while solving the traveling salesman problem. The number of vertices in a randomly-generated graph will then be graphed against the number of guesses made by the algorithm. The results of these experiments are shown below in Figure 1.

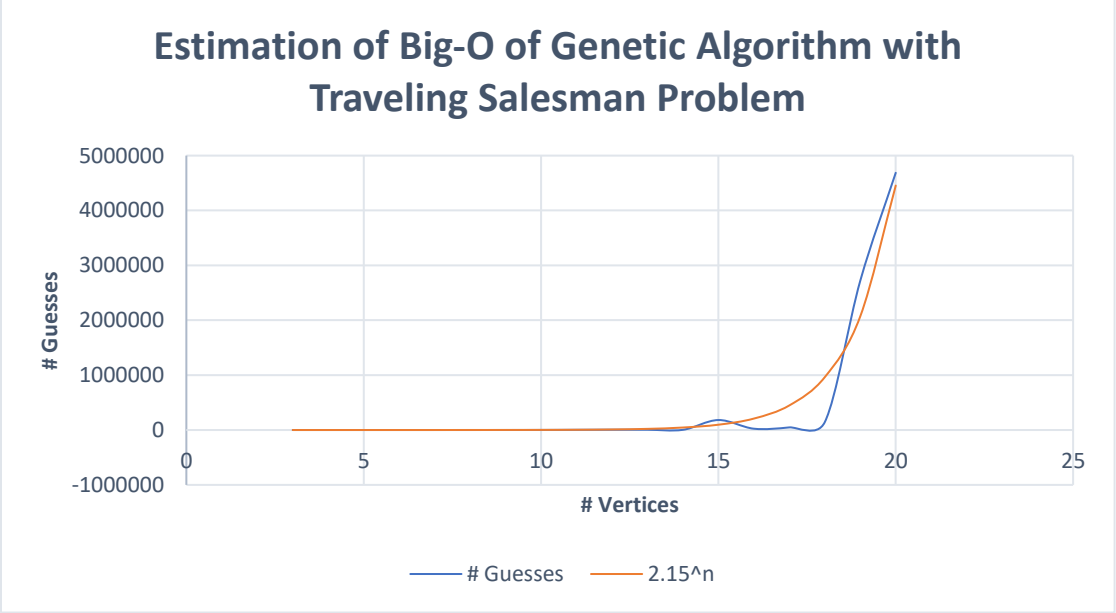

Figure 1. Graph of number vertices in graph versus number of guesses made by genetic algorithm as applied to the traveling salesman problem, along with a function estimation of the Big-O of the algorithm.

Also shown in Figure 1 is the curve of the function 2.15^n, where n is the number of vertices in the graph. When the graph for the genetic algorithm is compared against the function 2.15^n, the two graphs seem to match quite closely. It is claimed, then, that the Big-O of the genetic algorithm as applied to the traveling salesman problem is $O(2.15^n)$. To try to test this hypothesis, we will test the performance of this function against the performance of the genetic algorithm to see how the predicted and actual values compare. We will also measure percent error. These results are shown in Table 1. Percent error shown in Table 1 was calculated using the following formula:

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

The predicted number of guesses was treated as the theoretical value, while the actual number of guesses was considered as the experimental value.

Value of n	*Predicted Number of Guesses*	*Actual Number of Guesses*	*Percent Error*
21	9,576,519	9844501	2.80%
23	44,267,457	526901	-98.81%
25	205,626,322	27126501	-86.74%

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

Based on the results in Table 1, the function estimate of 2.15^n approximated the value of n = 21 well, but did not perform well with n = 23 and 25. This suggests that the guess for the Big-O efficiency, $O(2.15^n)$, was not a particularly good guess. The genetic algorithm does seem to have some highly variable performance, based on how difficult or easy the graph is. This could have affected the results obtained in Table 1 and thus affect an estimate for the Big-O efficiency of the algorithm.

Part B: Suboptimality of Genetic Algorithm

In this section, we wish to ascertain the suboptimality of the genetic algorithm when the number of generations is limited. To determine the degree of suboptimality based on the number of generations, the maximum number of generations will be adjusted and experiments will be performed with differing numbers of generations. The optimal cost to compare against is determined by 10 times the number of vertices in the graph. The suboptimality will be analyzed using percent error. Percent error will be calculated using the following formula:

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

The optimal cost was treated as the theoretical value, while the actual cost as measured from the experiments was considered as the experimental value. We will be testing the following maximum numbers of generations: 100, 250, and 500. The results of these different maximum generation values are shown below in Tables 2, 3, and 4, and the results in the tables are also graphically summarized via the graphs in Figures 2, 3, and 4.

# Vertices	Actual Cost	Optimal Cost	Percent Error
3	30	30	0
4	40	40	0
5	50	50	0
6	60	60	0
7	70	70	0
8	80	80	0
9	79	90	-12.22222222
10	100	100	0
11	110	110	0
12	129	120	7.5
13	126	130	-3.076923077
14	172	140	22.85714286
15	164	150	9.333333333
16	214	160	33.75
17	264	170	55.29411765
18	249	180	38.33333333
19	281	190	47.89473684
20	389	200	94.5
21	386	210	83.80952381
22	557	220	153.1818182
23	505	230	119.5652174
24	538	240	124.1666667
25	745	250	198

Table 2. Optimality results of genetic algorithm with maximum number of generations = 100.

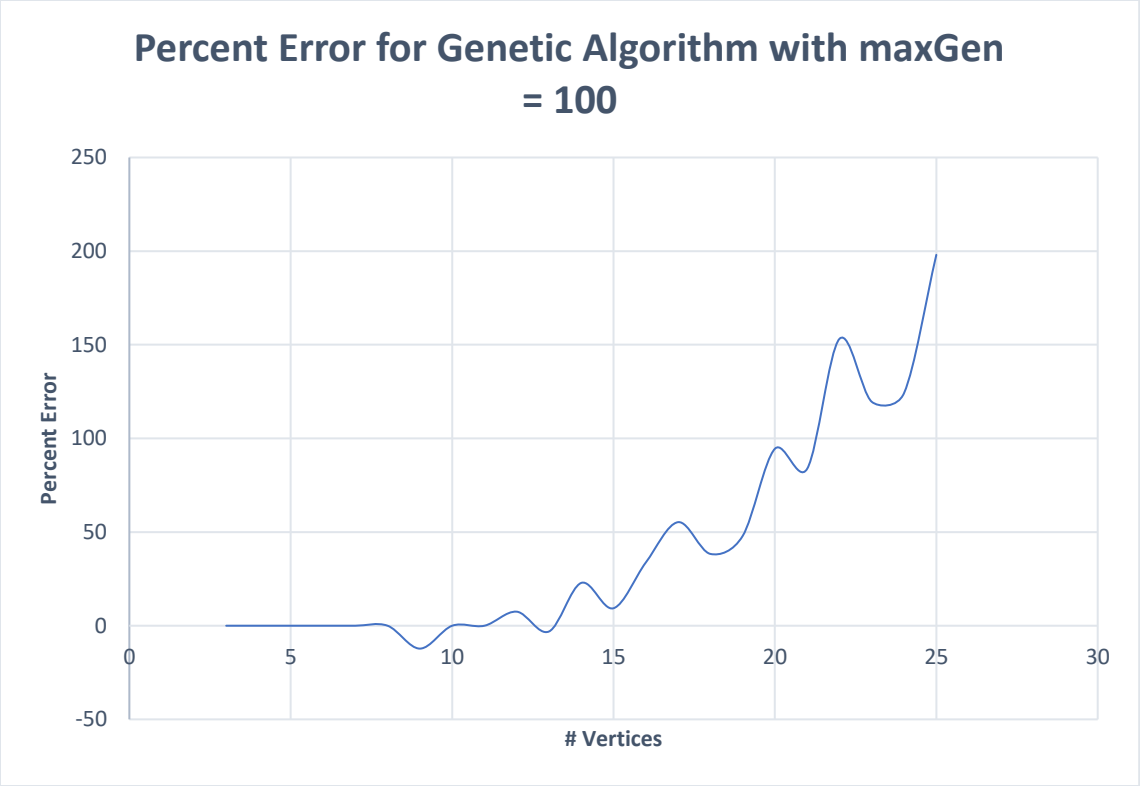

Figure 2. Graph of optimality results of genetic algorithm with maximum number of generations = 100.

# Vertices	Actual Cost	Optimal Cost	Percent Error
3	30	30	0
4	40	40	0
5	50	50	0
6	59	60	-1.666666667
7	65	70	-7.142857143
8	77	80	-3.75
9	90	90	0
10	100	100	0
11	106	110	-3.636363636
12	117	120	-2.5
13	128	130	-1.538461538
14	135	140	-3.571428571
15	148	150	-1.333333333
16	184	160	15
17	183	170	7.647058824
18	241	180	33.88888889
19	217	190	14.21052632
20	282	200	41
21	253	210	20.47619048
22	361	220	64.09090909
23	355	230	54.34782609
24	371	240	54.58333333
25	442	250	76.8

Table 3. Optimality results of genetic algorithm with maximum number of generations = 250.

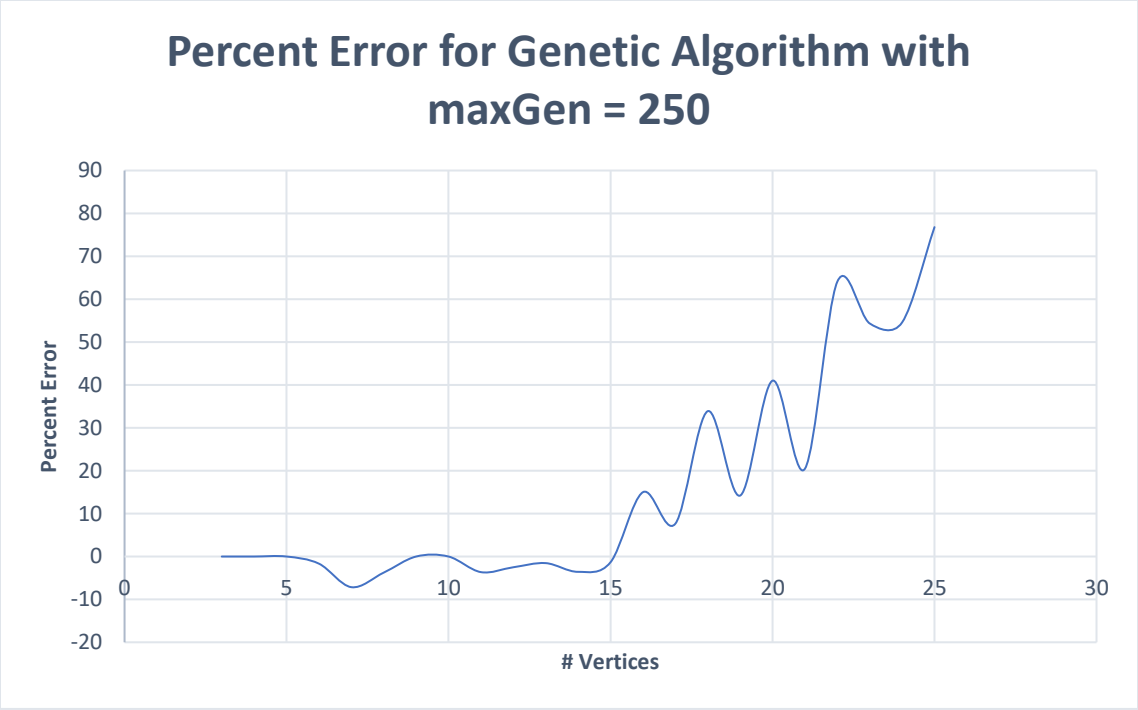

Figure 3. Graph of optimality results of genetic algorithm with maximum number of generations = 250.

# Vertices	Actual Cost	Optimal Cost	Percent Error
3	30	30	0
4	40	40	0
5	50	50	0
6	60	60	0
7	65	70	-7.142857143
8	80	80	0
9	90	90	0
10	93	100	-7
11	110	110	0
12	120	120	0
13	130	130	0
14	129	140	-7.857142857
15	148	150	-1.333333333
16	160	160	0
17	166	170	-2.352941176
18	175	180	-2.777777778
19	184	190	-3.157894737
20	315	200	57.5
21	205	210	-2.380952381
22	310	220	40.90909091
23	220	230	-4.347826087
24	518	240	115.8333333
25	545	250	118

Table 4. Optimality results of genetic algorithm with maximum number of generations = 500.

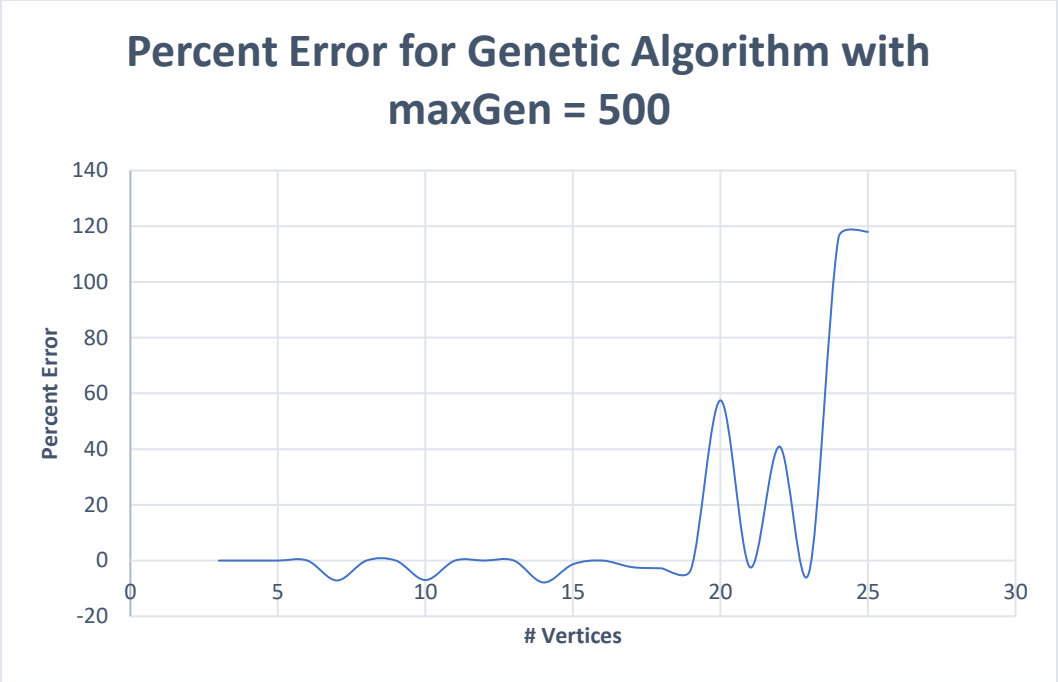

Figure 4. Graph of optimality results of genetic algorithm with maximum number of generations = 500.

As can be seen from all of these table and figures, the suboptimality of the genetic algorithm does appear to worsen as the maximum number of generations is increased. Furthermore, we can see that the suboptimality of the algorithm worsens as the number of vertices in the graph increases. The worst performance came from the experiments with the 100 maximum generations; the experiments with the 250 maximum generations was also rather poor, especially for graphs with a large number of vertices. The 500 maximum generation had relatively good performance until 20 vertices, after which there was a marked increase in the percent error, and the genetic algorithm's performance is highly suboptimal.