ECE650 - Methods and Tools for Software Engineering

University of Waterloo

Electrical and Computer Engineering

Final Course Project

Authors:
Zihao Liu (ID: 20750564)
Su Sun (ID: 20862738)

Date: December 3, 2019

1 Introduction:

This project aims to generate three different algorithms to solve minimum vertex cover problem in multi-thread way and compare the efficiency(Running time) and performance(Approximation Ratio) of each algorithm.

These three algorithms are:
CNF-SAT-VC : A Min-Vertex-Cover problem reduced to CNF-SAT problem used in Assignment 4.
APPROX-VC-1 : Highest degree vertex picking method.
APPROX-VC-2 : Edge picking method.

There is no random component for APPROX-VC-1 and APPROX-VC-2 algorithm, the selection of each highest degree vertex and edge in each graph is decided by the ascending order of vertex, which means that the ascending order of each graph is computed, then the vertex size with most occurrences is selected. These three algorithms are run in three threads to solve the minimum vertex cover problem and the other thread is implied for I/O. The input graphs were generated by a graph generator on *eceubuntu* contributed by Professor Arie Gurfinkel.

The graphs, Table 1, were generated in an ascending number of vertex to show the algorithm execution information.

Num of Vertex	5	7	10	12	13	15	20	25	30	35	40	45	50
Num of Graph	10	10	10	10	10	10	10	10	10	10	10	10	10
Running Times	10	10	10	10	10	10	10	10	10	10	10	10	10

Table 1: Table to show the vertex and graph information

When the size of vertex is larger than 20 (including 20), the result of CNF-SAT-VC algorithm shows **timeout.**, which means that no vertex cover result generated within given time, due to the extremely high running time. Thus 7, 12 and 13 were appended into the vertex number sequence. In conclusion, APPROX-VC-1 and APPROX-VC-2 algorithms were computed 1300 times respectively, however the CNF-SAT-VC algorithm was only computed 600 times totally.

The running CPU time of each algorithm in each execution was obtained by **pthread_getcpuclockid**, and the results were stored in a text file after each vertex computation. Moreover, the average CPU time, standard deviation and approximation ratio across 100 runs for each value of $|V|$ were also stored in the text file after execution. With the incremental size of vertex, CNF-SAT-VC algorithm cannot generate results under the given time, so the approximation ratio of APPROX-VC-1 and APPROX-VC-2 cannot be computed.

2 Analysis

This section analysis the Vertex Cover problem-solving efficiency and performance of each algorithm. In the following subsections, running time analysis of these three algorithms and approximation ratio analysis of APPROX-VC-1 and APPROX-VC-2 will be introduced in detail. All programs in this project were executed on eceubuntu server.

2.1 Running Time

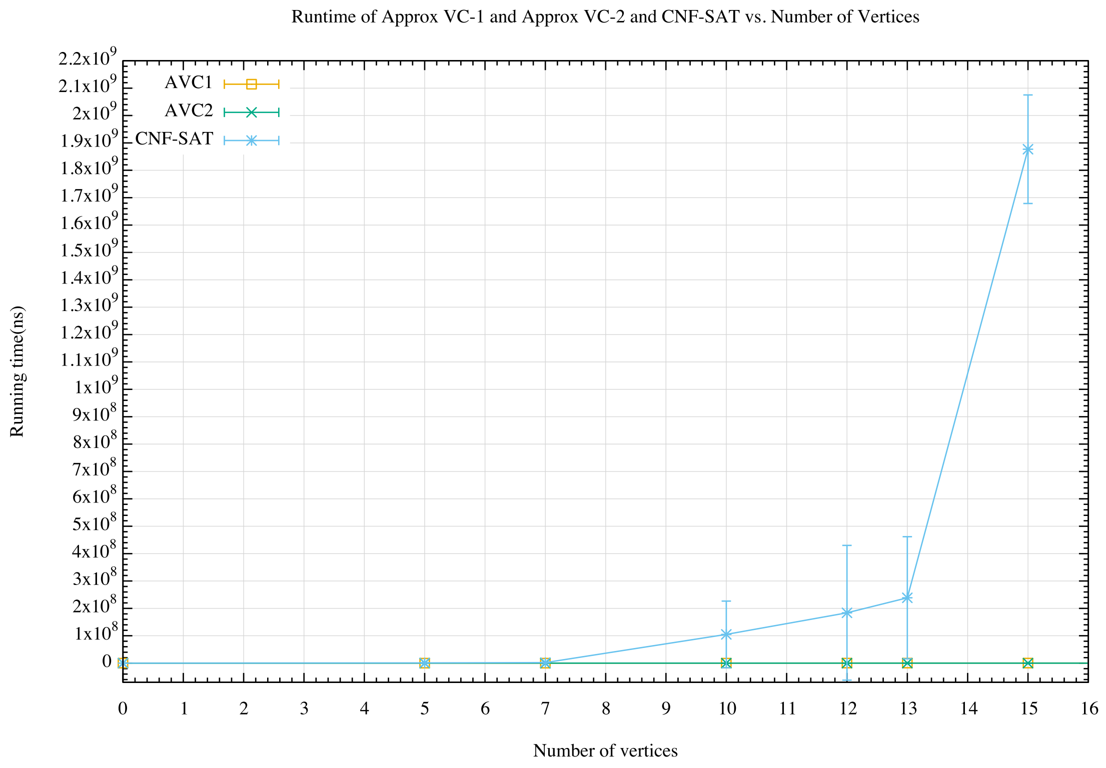

Figure 1: Runtime of Approx VC-1 and Approx VC-2 and CNF-SAT vs. Number of Vertices

Figure 1 shows the performance of the three algorithms. As it can be seen, the time cost of APPROX-VC-1 and APPROX-VC-2 are very low compared with CNF-SAT-VC . Even when vertex size goes up to 50, algorithm APPROX-VC-1 and APPROX-VC-2 can finish in far less than 0.1 seconds. However, CNF-SAT-VC costs much more time with the increasing number of vertices. Especially when the graph has 15 vertices, the running time of CNF-SAT-VC gets a surge and its average time is 1.88s. We also tried to test CNF-SAT-VC's running time when vertex size is 20 and 22, setting timeout value to 1000-second and 3600-second respectively, but only got a timeout in the end. Thus, we can infer that the running time of CNF-SAT-VC is more than 16.7min when vertex size is 20, and more than 60min when vertex size is 22. The high running time is caused by the polynomial-time reduction used in our algorithm, which is closely correlated with vertex size. The number of clauses in CNF-SAT-VC algorithm is $k + n\binom{k}{2} + k\binom{n}{2} + |E|$, where k is the size of vertex cover and n is the number of vertices of the graph. To better observe the relationship between running time and number of vertices, we generate a plot with log-scale on the vertical Axes for CNF-SAT-VC.

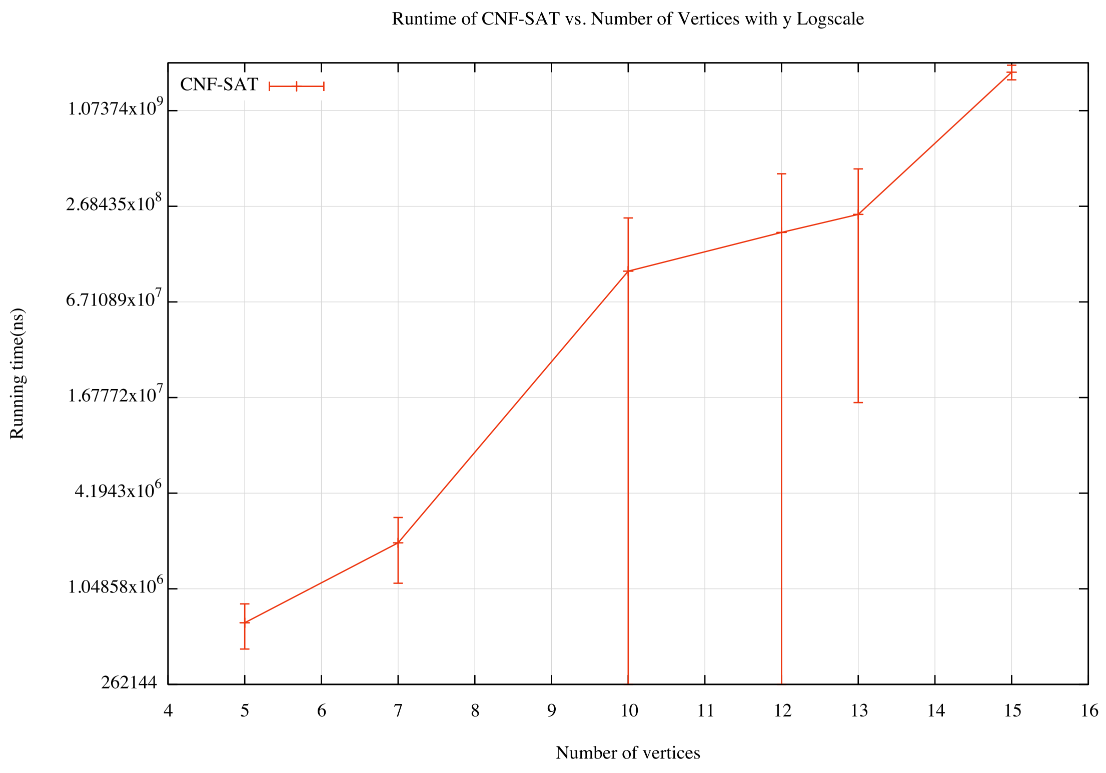

Figure 2: Runtime of CNF-SAT vs. Number of Vertices with y Logscale

According to figure 2, the running time of CNF-SAT-VC increases almost exponentially. During our testing, the first round ran at around 5-6 pm, $|V| \in [5, 50]$, in increments of 5, was included. And in the second round, we tested the remaining data at 12 am. Thus, we deduce that the time is a little bit lower when the vertex size is 7, 12, 13 is because at that time *eceubuntu* has a lower server usage, leading to a faster running time.

Figure 1&2 also show algorithm's stability through error bars. It is clear that CNF-SAT-VC is very unstable when vertex size is greater than 10. This is because the size of the minimum-sized vertex cover among different graphs differs a lot even the vertex number is the same.

In order to compare APPROX-VC-1 and APPROX-VC-2 more intuitively, we generated a plot excluding CNF-SAT-VC. Figure 3 shows the running time of APPROX-VC-1 and APPROX-VC-2. The performance of these two algorithms is very similar until around a vertex size of

10. When vertex number exceeds 10, APPROX-VC-1 usually has a higher running time than APPROX-VC-2 due to the reason that APPROX-VC-1 has to find a vertex of highest degree and then throw away all edges incident on that vertex to achieve the minimum-sized vertex cover. This step needs to traverse the whole graph. However, APPROX-VC-2 only needs to pick the smallest edge. The different step of "find a highest degree vertex" in APPROX-VC-1 or "pick a smallest edge" in APPROX-VC-2 causes the different running time between APPROX-VC-1 and APPROX-VC-2. For APPROX-VC-1, the running time increases linearly with $|V|$ and has some fluctuations when the vertex size is bigger than 25. However, the change of APPROX-VC-2 is not that obvious. And its error bars are not visible because the standard deviation is small.

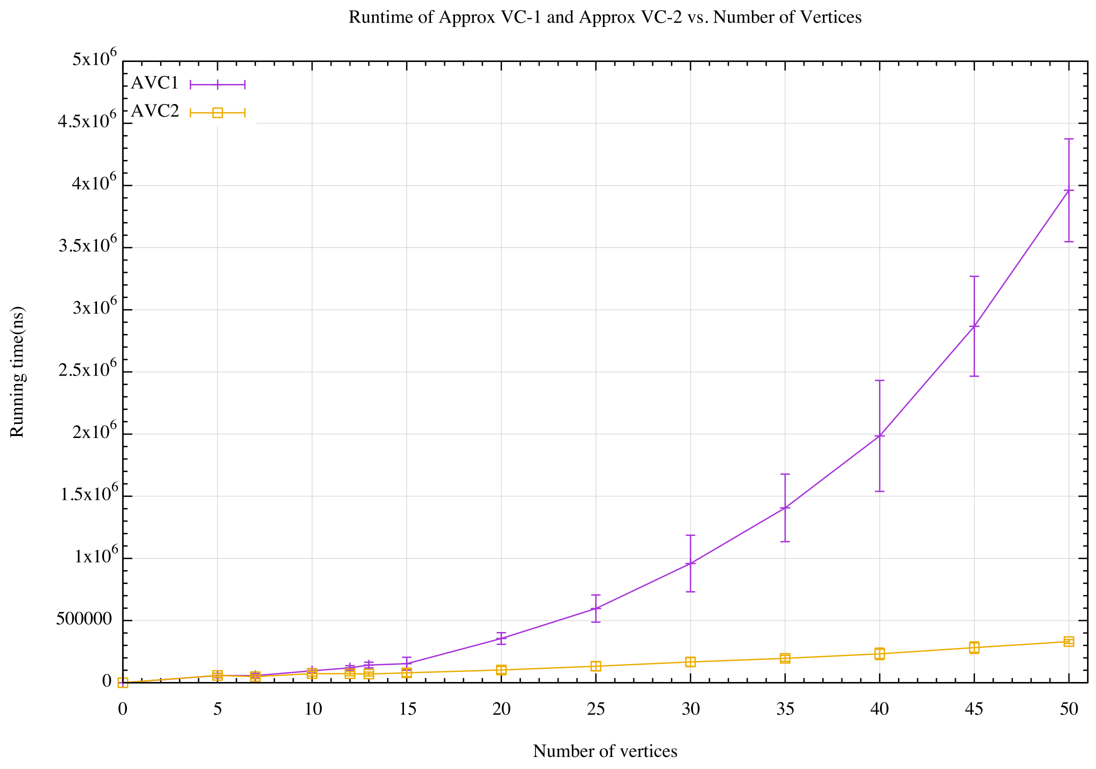

Figure 3: Runtime of Approx VC-1 and Approx VC-2 vs. Number of Vertices

2.2 Approximation Ratio

In general, the CNF-SAT-VC algorithm always generates the minimum vertex cover for any size graph input, therefore, the optimal vertex cover result obtained by CNF-SAT-VC algorithm was utilized as a standard data, which can be seen as the benchmark for other algorithms. As another important factor to measure the performance of an algorithm, the approximation ratio can be expressed by 1.

$$ApproxRatio = \frac{Approximate}{Optimal} \tag{1}$$

The *Approximate* indicates the number of vertex cover returned by APPROX-VC-1 or APPROX-VC-2, and the *Optimal* means that the optimal algorithm, CNF-SAT-VC algorithm. Therefore, the lower *ApproxRatio*, the better the Algorithm is.

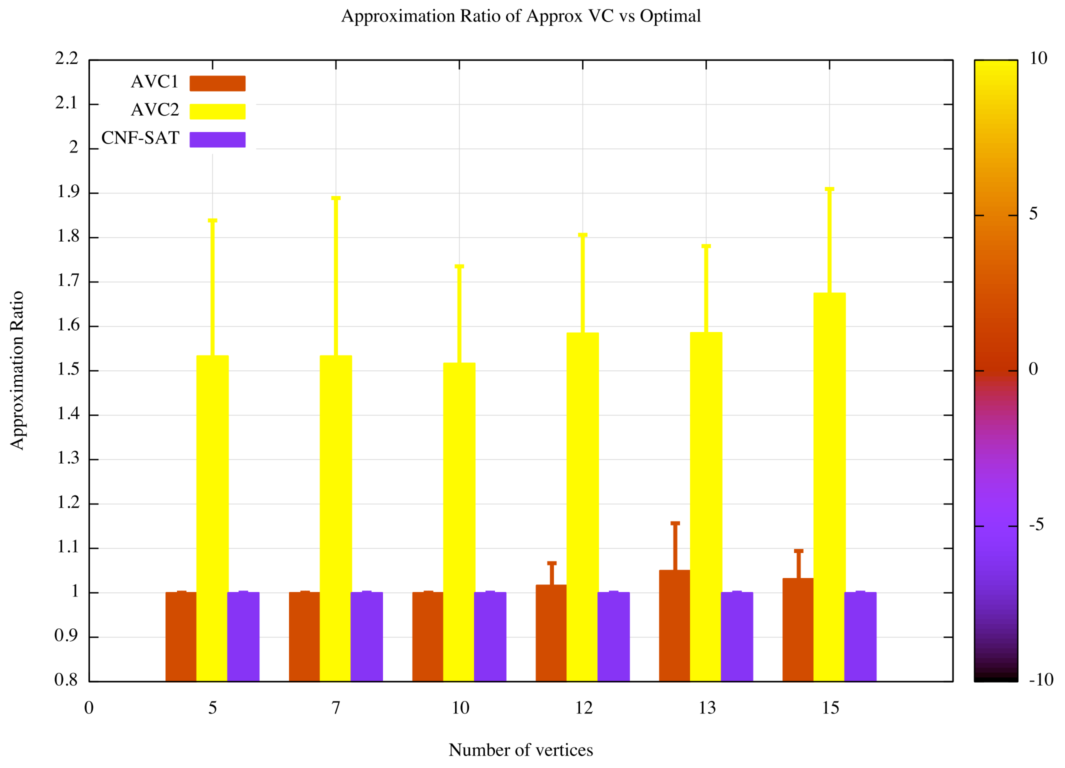

Figure 4: Approximation Ratio of APPROX-VC vs Optimal

Figure 4 shows the results of APPROX-VC-1 and APPROX-VC-2 algorithms' performance. The purple bar is CNF-SAT-VC algorithm, which is always 1 as benchmark. The red bar represents the average approximation ratio of APPROX-VC-1, which indicates a good performance when the size of vertex is 5, 7 and 10. Although the ratio of APPROX-VC-1 becomes larger when the size of vertex exceeds 10, its performance still shows well results. Moreover, standard deviation appears when the size of vertex is larger than 10, meaning that with the increment of vertex size, the vertex cover results will be different from CNF-SAT-VC algorithm. However, the yellow bar, which is the performance of APPROX-VC-2, does not as well as APPROX-VC-1, and always be 1.5 times as much as the optimal algorithm. With the number of vertices increases from 10 to 15, the ratio of APPROX-VC-2 becomes larger and larger. What should be noticed is that there is a slight ratio decrease from 7 to 10, and the reason might be that small size of vertex forcing the APPROX-VC-2 algorithm to pick more vertices. In contrast, the standard deviation shows a slightly decreasing trend. Therefore, an inference, with the vertex size increasing, the approximation ratio of each graph approaching the average value under a given vertex size could be concluded. The standard deviation of APPROX-VC-2 represents that the vertex cover result fluctuates greatly among different graph input under the same vertex size.

3 Conclusion

Regardless of time and computer hardware configuration, the CNF-SAT-VC algorithm shows the most accurate result. In this project, a proposed size for the vertex cover was input from one to a number where the SAT solver returns true value. This proposed size selection method might be a good choice when the vertex number is small, however, with the increasing of vertex size, this method may reduce efficiency. Consequently, if a binary search method was utilized, the running time of large-size vertex might be reduced. $O(2^n)$ is the time complexity of CNF-SAT-VC algorithm.

In terms of APPROX-VC-1, this algorithm represents a good performance, which reflects a good approximation ratio and low running time under the vertex size of 20. The reason why APPROX-VC-1's running time always longer than that of APPROX-VC-2 is that the highest degree vertex must be found before execution each loop. As a result, this retrieval method must iterate each vertex and notice how many edges do this vertex connect to all other vertices. The time complexity of this algorithm is $O(n^2)$.

The APPROX-VC-2 runs the fastest among all the algorithms, and the vertex cover results are generated by conducting only one iteration through all edges of the input graph. The time complexity is $O(n)$. However, the relatively poor approximation ratio is the price of high speed.

In conclusion, if there is enough time to execute and better computer hardware provided, or pursuing the most accurate result, the CNF-SAT-VC algorithm could provide the optimal result. If vertex cover problems are required to solve under large vertex size without too much time and good computing machine, APPROX-VC-1 and APPROX-VC-2 are also suitable choices. In terms of running time and approximation ratio, APPROX-VC-1 combines with efficiency and performance. Therefore, when vertex size is uncertain and time is not sufficient, APPROX-VC-1 is always a good choice.