Performance Analysis of SAT Solver for Vertex Cover Problem

ECE 650: Methods And Tools For Software Engineering

Masters Of Engineering- Electrical and Computer Engineering

Mohit Gupta: 20754709

Harshpal Singh: 20752839

1 Introduction

In this report, comparison of three vertex cover algorithms are analyzed. The goal of each algorithm is to deduce the minimum sized vertex cover of the given graph $G = \langle V, E \rangle$ Where $|V|$ represents the total number of vertices and $|E|$ denotes the number of edges. The first Algorithm, named CNF-SAT, relies on the conversion of polynomial-time reduction form vertex cover to CNF-SAT (Conjunctive normal form) problem. The second algorithm is called Approx-1, it picks up the vertex with highest degree (the vertex with highest number of incident edges) and remove all its connecting edges. This operation continues till $|E| = 0$. The third and the final algorithm, is called Approx-2, it picks up the first edge in the list and removes all the connecting edges and vertices to that edge, this will continue until no edges are left.

For the purpose of analysis, We are determining the following properties of the algorithm:

1) Running Time - Time taken by each algorithm to calculate the minimum vertex cover.

2) Approximation Ratio - Ratio of the size of the computed vertex cover to the size of optimal minimum vertex cover.

The graphs depicting the statistics were plotted in Matlab.

2 Analysis

The analysis program uses multi-threading. In total four threads were used which are:

I/O thread: For fetching input from the user in the form of vertices and edges.

CNF thread: It uses the CNF-sat algorithm to find the minimum vertex cover.

APPROX1 thread: It finds the optimal minimum vertex cover using the approx1 algorithm.

APPROX2 thread: It calculates the optimal minimum vertex cover using the approx2 algorithm.

The running time is of each thread was computed at the end of each thread using the pthread-getcpuclockid() function of C++. The program was tested on a ecetesla1.uwaterloo.ca server. The machine on which the analysis was carried out is Intel Core i5 Multicore-processor, 2.7GHz, 8G of RAM, and its OS is Ubuntu.

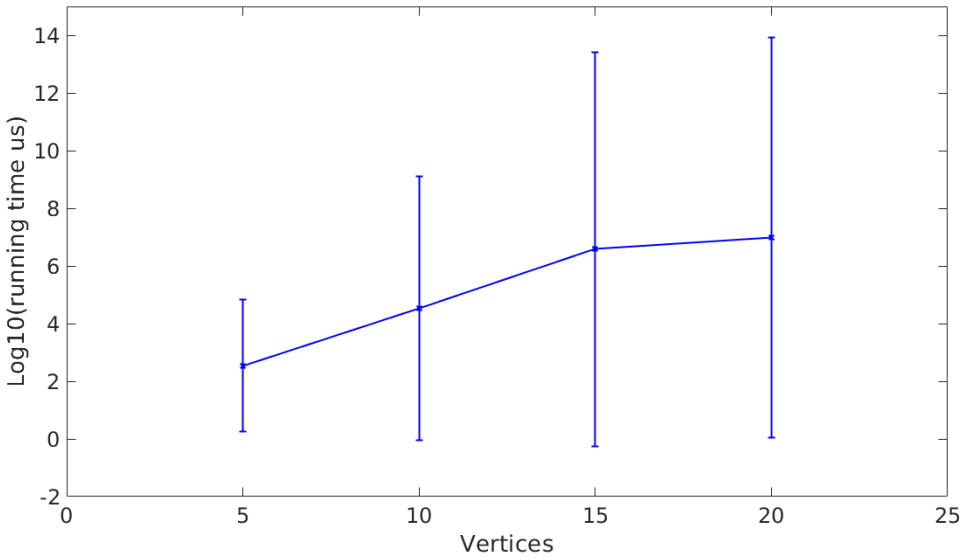

Figure 1: CNF-SAT running time analysis in interval [5,20].

2.1 Running Time Analysis

2.1.1 CNF-SAT - Run Time Analysis

According to the analysis, CNF-SAT algorithm takes the maximum running time as compared to all the three algorithms. As an input for the analysis the $|V|$ ranges [5,20] with the increment of 5 (i.e. 5, 10, 15, 20). Since, CNF-SAT takes a lot of time to calculate vertex cover for larger $|V|$, therefore, we have restricted the number of edges i.e.$|E| = 15$ for $|V| = 20$ On the other hand, CNF-SAT algorithm takes very less running time when number of vertices are less, specifically $\leq$ 10 but it shows exponential rise in the running time when number of vertices becomes greater than 10. This is because polynomial time reduction used in sat solver calculates the number of clauses based on the vertices of the input graph and is given by:

$$k + n(\frac{k}{2}) + k(\frac{n}{2}) + |E| \; where \; n = |V| \; in \; G \tag{1}$$

The standard deviation is almost as high as the mean which indicates that for different graphs generated with the same $|v|, |E|$ the running times vary drastically. As the running time increases exponentially for the larger vertices, therefore we can say that the standard deviation for larger input compresses the standard deviation for smaller input.

Vertex	Mean Time(us)	STDev
5	345	197
10	34108	37880
15	$3.7X10^6$	$6.7X10^6$
20	$9.7X10^6$	$8.7X10^6$

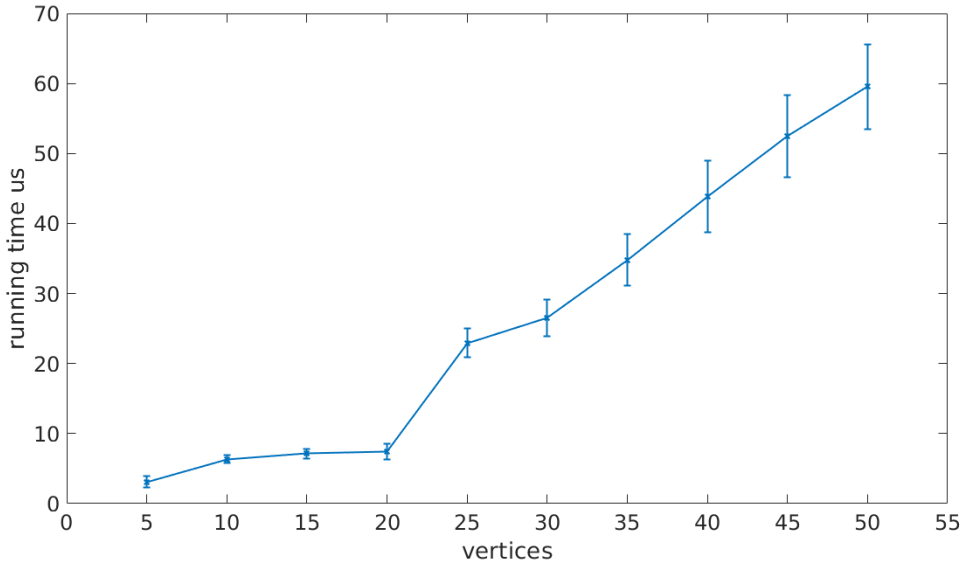

Figure 2: Approx-1 running time analysis in interval [5,50].

2.1.2 APPROX1 - Run Time Analysis

Below figure 2 depicts the running time of Approx-1 algorithm as a function of the number of vertices. The scale of vertices is taken from [5,50] with the increment of 5. As this is an approximate algorithm which may not give the minimum vertex cover but provides us with the optimal solution.From the figure 4 we can clearly say that Approx-1 takes less time as compared to the CNF-SAT discussed above for the same inputs. The reason behind this that Approx-1 will provide us with the efficient output but CNF-SAT will give us the correct output.

Vertex	Mean Time(us)	STDev
5	3.11	0.76
10	6.35	0.61
15	7.12	0.68
20	7.44	1.13
25	22.95	2.04
30	26.54	2.60
35	34.78	3.66
40	43.83	5.14
45	52.45	5.89
50	59.55	6.02

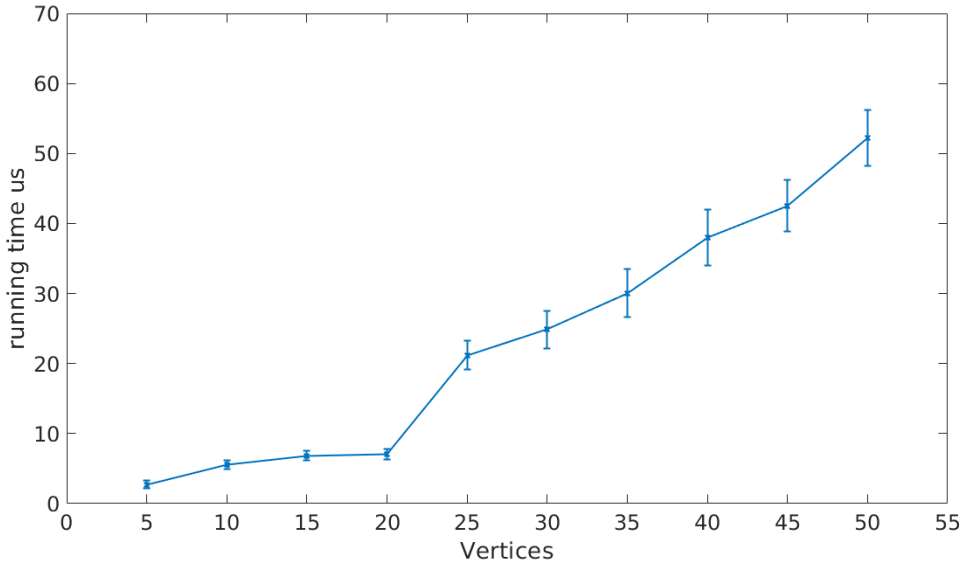

Figure 3: Approx-2 running time analysis in interval [5,50].

2.1.3 APPROX2 - Run Time Analysis

Figure 3 shows the running time of Approx-2 algorithm as a function of number of vertices. The range of vertices is taken from [5,50] with the increment of 5. It is also an approximate algorithm as Approx-1. From the figure 4 we can clearly state that Approx-2 takes less time as compared to the CNF-SAT. From figure 5 we see that Approx-2 takes less time than the Approx-1 algorithm for the same inputs even for inputs beyond vertex size of 20. Theoretically, Approx-1 should perform better asymptotically than Approx-2 because Approx-2 will pick edge at a time which is $O(|V|^2)$ than the vertices in the case of Approx-1 which is $O(|V|)$. This contradiction as depicted in the plots arises due to our implementation of Approx-1 where after picking a vertex of highest degree we remove it from graph, adjust degrees of all vertices ($O(|E|)$) and then find the vertex with highest degree ($O(|V|)$). This causes additional overheads and hence Approx-1 becomes less efficient that Approx-2 in our implementation.

Vertex	Mean Time(us)	STDev
5	2.73	0.54
10	5.51	0.62
15	6.84	0.67
20	7.02	0.72
25	21.18	2.08
30	24.87	2.67
35	30.07	3.45
40	37.98	4.01
45	42.52	3.85
50	52.23	3.96

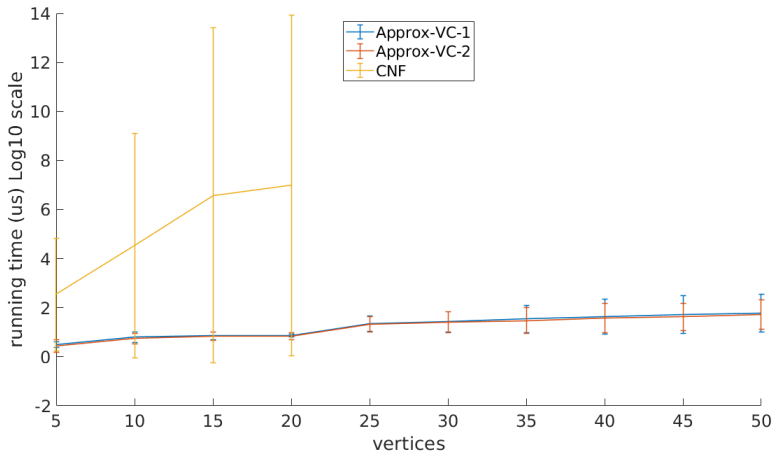

Figure 4: Comparing running time of all three algorithms in interval [5,50], cnf is only [5,20], with log_10-scale on Y-axis.

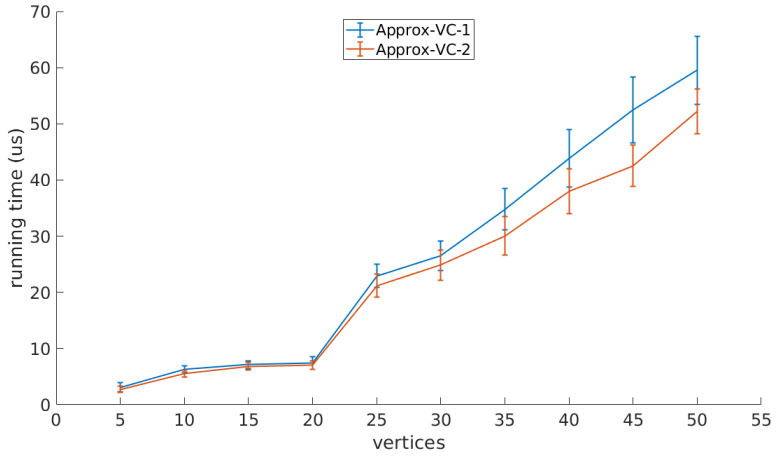

Figure 5: Comparing running time of Approx-VC1 and Approx-VC2 algorithms in interval [5,50]

2.1.4 Comparison of all Algorithms

To compare of all 3 algorithms, we have taken log scale on Y axis (i.e. the running time) and X axis as the number of vertices ranging from [5,50] with the increment of 5. Now from Figure4, we can observe that CNF-SAT takes much more time in comparison to Approx-1 and Approx-2. Moreover, CNF-SAT follows an exponential increase trend (as mentioned before) in running time with respect to increase in the number of vertices but this is not the case for other two algorithms. Also the CNF-SAT traverses the whole solution space and looks for all possible assignments to the given CNF in order to satisfy every clause for minimum vertex cover. The number of clauses and time taken by SAT solver increases exponentially with the increase in vertices. While in the case of other two algorithms, they have less time complexity as compared to CNF-SAT. Hence, they will never show exponential rise in their running time as compared to CNF-SAT.

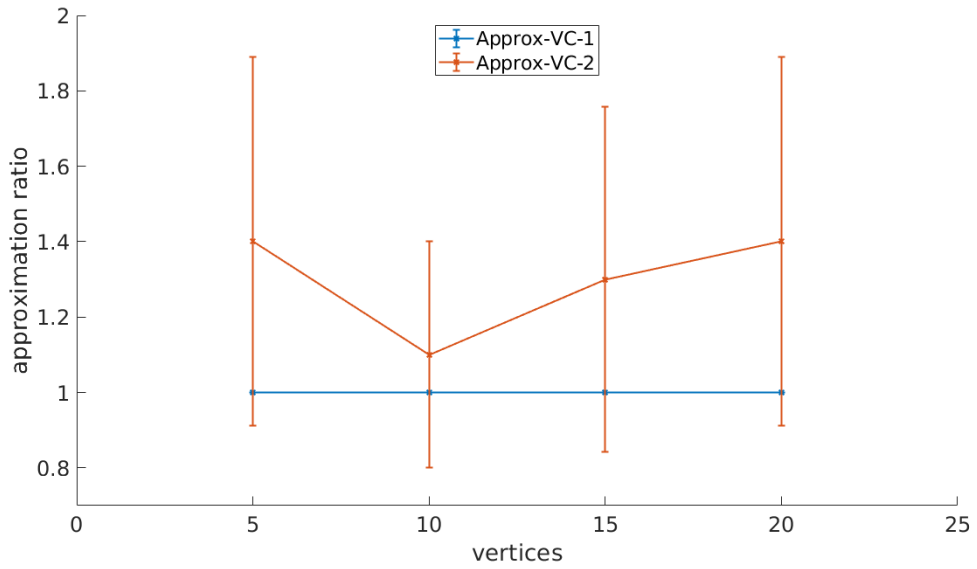

Figure 6: Comparing Approximation Ratio of Approx-1 and Approx-2 in interval[5,20].

2.2 Approximation Ratio Analysis

To Approximation ratio is defined as the below formula:

$$ApproximationRatio = \frac{SizeofComputedVertexCover}{SizeofMinimumVertexCover} \tag{2}$$

So smaller the approximation ratio, more optimal will be the output of the algorithm. As we can see from Figure 6 that the approximation ratio for all of the vertices is 1 for Approx-1. On the other hand, the approximation ratio $\in [1.1, 1.4]$ for Approx-2. This is because the Approx-1 algorithm picks up vertices with highest degree whereas Approx-2 picks up one edge at a time. So in Approx-1 often gives us the minimum vertex cover (especially for smaller —V—) which was found to be equivalent to the output of CNF-SAT. Approx-2 mostly provide vertex covers which is greater than the minimal vertex covers provided by the CNF-SAT algorithm. The standard deviation in Approx-2, corresponds to the randomness aspect of the algorithm, and may vary after each run.

Vertex	MeanAPR Aprx-1	STDevAPR Aprx-1	MeanAPR Aprx-2	STDevAPR Aprx-2
5	1	0	1.4	0.489
10	1	0	1.1	0.3
15	1	0	1.3	0.458
20	1	0	1.4	0.489

3 Conclusion

As deduced from the above plotted graphs, it is safe to say that all the three algorithms can be used to solve the vertex cover problem. But, on the basis of running time and approximation ratio, their efficiency may vary.

• In terms of running time, Approx-2 is better as compared to CNF-SAT, but it fails to get minimum vertex cover as we have seen that the approximate ration for Approx-2 was quite high.

• In terms of approximation ratio, CNF-SAT definitely gets the minimum vertex cover, but it fails for the larger inputs as the running time rises exponentially.

• In general, Approx-1 is a better approach to solve large scale vertex cover problem as its the running time is less than CNF-SAT and Approx-2 and also it can provide optimal minimum vertex cover.