ECE 650 Project Report

Performance Analysis of Various Algorithms for Vertex Cover problem

Chandra Mouli Saride : 20810347

Nidhin Easo Thomas : 20809980

1 Introduction

Our aim is to analyse the performance of the three given algorithms for finding out the vertex cover of a given graph. For a given graph G=(V,E), where $|V|$ indicates the number of vertices and $|E|$ indicates number of Edges. In our analysis, the first algorithm is CNF-SAT-VC based on minisat. This is based on the polynomial-time reduction from vertex cover to CNF-SAT problem, which transforms it into a formula F that can be given as input to SAT solver and reconstruct the minimal vertex-cover based on assignments from SAT solver. The second algorithm used in our analysis is APPROX-VC-1, Here we select the vertex with most incident edges, add it our vertex cover and delete all edges , we will repeat this process until no edges remain i.e, $|E|$=0. The final algorithm used is named APPROX-VC-2, here we select an edge, add both elements to the vertex-cover and then remove all edges where either of these elements exists, we will repeat this process until no edges remain.

For our analysis, we are taking into consideration the following properties of each algorithm:

1. Running Time - CPU time utilised by each algorithm for calculation of minimum vertex-cover.
2. Approximation Ratio- It is the ratio of the size of calculated vertex-cover to the size of minimum sized vertex cover, Here we consider CNF-SAT-VC to be giving minimum-sized vertex cover.

2 Analysis

For our analysis we are using multi-threading in our main program, with the following four thread:

Input/Outpt Thread: It is used for taking graph input from the user .
CNF-SAT-VC Thread: It is used for calculating the vertex cover using CNF-SAT-VC algorithm.
APPROX-VC-1 Thread: It is used for calculating the vertex cover using APPROX-VC-1 algorithm.
APPROX-VC-2 Thread: It is used for calculating the vertex cover using APPROX-VC-2 algorithm.

The CPU running time of each thread is determined by the help of function `pthread_getcpuclockid()` in C++. The analysis was held on a local machine with Intel Core i7-4720HQ @2.60GHz, 12GB RAM, OS: Ubuntu 18.04.1.

2.1 CPU Running Time Analysis

2.1.1 CNF-SAT-VC CPU Run Time Analysis

From analysis we observed that, CNF-SAT-VC algorithm has higher running time than the other two algorithms. In fact, we had to limit the number of vertices to 15, for CNF-SAT-VC . As, we have declared time-out to be 2-minutes, CNF-SAT-VC timedout for vertices greater than or equal to 20. But, for $|V| \leq 10$ CNF-SAT-VC has very short running time. As vertex number increase beyond 10, CNF-SAT-VC shows a sudden rise in the ruuning time. This can be explained, since the number of clauses a SAT solver uses depends on the number of vertices for a given graph.

The number of clauses in the reduction is $k + n(\frac{k}{2}) + k(\frac{n}{2}) + |E|$, Where $\mathbf{n} = |V|$ and $\mathbf{k}$ is number of vertices in vertex cover.

The standard deviation is very high ,which denotes that CPU running time is different, even for the same $|V|$ and $|E|$

Table 1: Run time analysis for CNF-SAT-VC algorithm

Vertex	Mean Running time(us)	stdev
5	1213	118.4
10	35910	25294
15	$5.15X10^6$	$6.31X10^6$

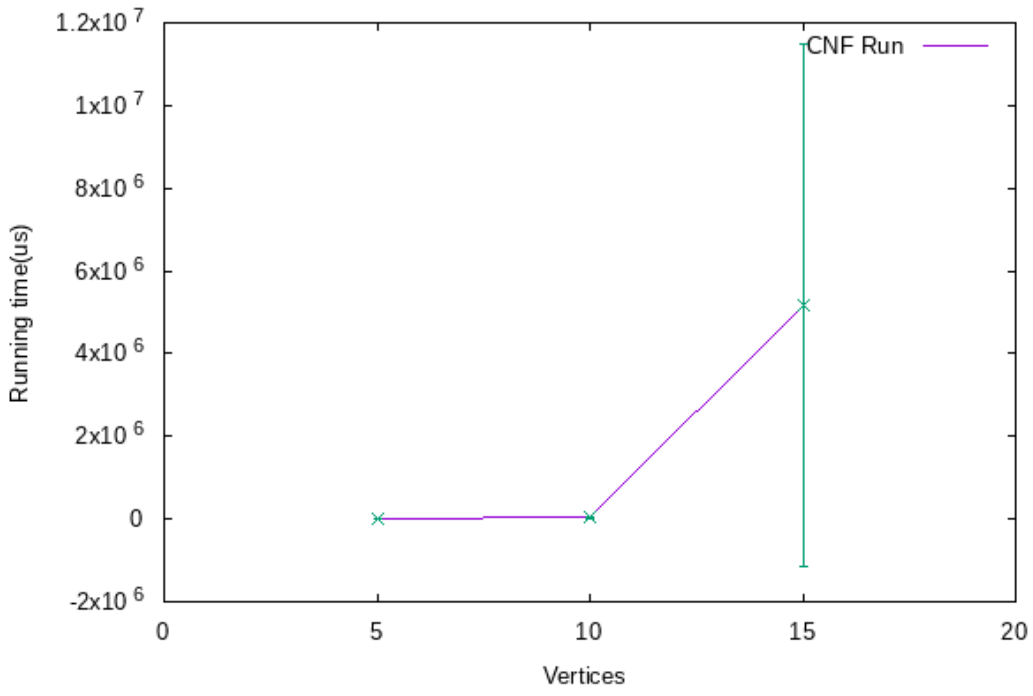

Figure 1: Run time analysis for CNF-SAT-VC algorithm

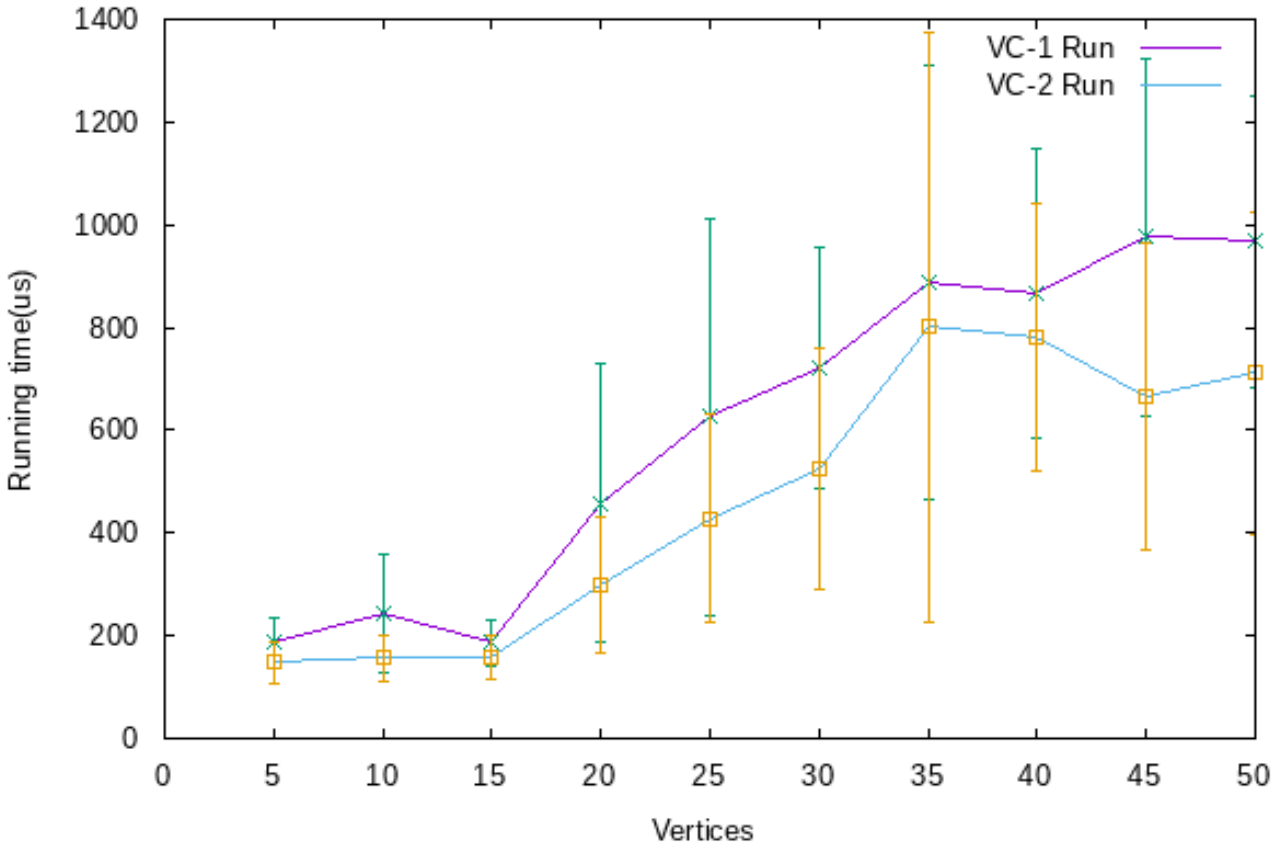

Figure 2: Run time analysis for APPROX-VC-1,APPROX-VC-2 algorithm

2.1.2 APPROX-VC-1 CPU Run Time Analysis

Figure 2 above shows the run time of APPROX-VC-1 algorithm based on the number of vertices in the provided input. The vertices range from [5,50] with a step size of 5. Even though APPROX-VC-1 algorithm may not give the minimum vertex cover, it provides the optimal solution. From the figure, it can be observed that APPROX-VC-1 algorithm takes less time for execution as compared to CNF-SAT-VC discussed above for the same set of inputs. This is because APPROX-VC-1 algorithm provides us the efficient output whereas the CNF-SAT-VC gives the correct minimum vertex cover

Table 2: Run time analysis for APPROX-VC-1 algorithm

Vertex	Mean Running time(us)	stdev
5	186.2	50.39
10	244.19	116.25
15	186.177	43.5444
20	457.776	271.249
25	625.975	387.417
30	721.927	235.684
35	888.345	421.467
40	866.063	281.738
45	975.632	348.763
50	966.897	284.122

2.1.3 APPROX-VC-2 CPU Run Time Analysis

Figure 2 above shows the run time of APPROX-VC-2 algorithm based on the number of vertices in the provided input. The vertices range from [5,50] with a step size of 5. APPROX-VC-2 algorithm is also similar to APPROX-VC-1 algorithm that it does not give the minimum vertex cover. It is clear from the above figure that APPROX-VC-2 takes less time to calculate vertex cover than CNF-SAT-VC. From the figure, we can also see that APPROX-VC-2 algorithm takes less time to compute vertex cover than APPROX-VC-1 algorithm even for vertex count more than 20. Theoretically, APPROX-VC-1 is expected to execute faster than APPROX-VC-2 as APPROX-VC-2 perform vertex cover computation by selecting an edge at a time which is of the order O($|V|^2$) instead of vertices in case of APPROX-VC-1 which is of the order O(V). This variation is due the implementation of APPROX-VC-1 in which we pick a vertex of highest degree, remove it from graph,adjust degree of all vertices O($|E|$) then find the vertex with the highest degree O($|V|$). This additional overhead causes APPROX-VC-1 algorithm to be less efficient than APPROX-VC-2 algorithm

Table 3: Run time analysis for APPROX-VC-2 algorithm

Vertex	Mean Running time(us)	stdev
5	148.54	40.0688
10	157.423	44.3998
15	158.902	43.5058
20	300.308	132.781
25	428.901	202.73
30	524.104	235.853
35	800.388	575.87
40	781.855	260.801
45	664.466	298.976
50	710.698	313.122

2.2 Approximation Ratio Analysis

The Approximation ratio is defined as :

$$Approximation\ ratio = \frac{Size\ of\ computed\ vertex\ cover}{Size\ of\ minimum\ vertex\ cover}$$

Smaller the approximation ratio,more optimal the calculated output of the algorithm is. From the below table it can observed that approximation ratio for APPROX-VC-1 algorithm is almost 1 for all the vertex counts considered. This is due to the implementation of APPROX-VC-1 algorithm which picks the highest degree vertices whereas APPROX-VC-2 picks one edge at a time and perform computation. It can be observed that APPROX-VC-1 algorithm provides minimum vertex cover especially for smaller values of vertices which is equivalent to output of CNF-SAT-VC. The standard deviation of APPROX-VC-2 corresponds to randomness in selecting the edges and may vary after each run.

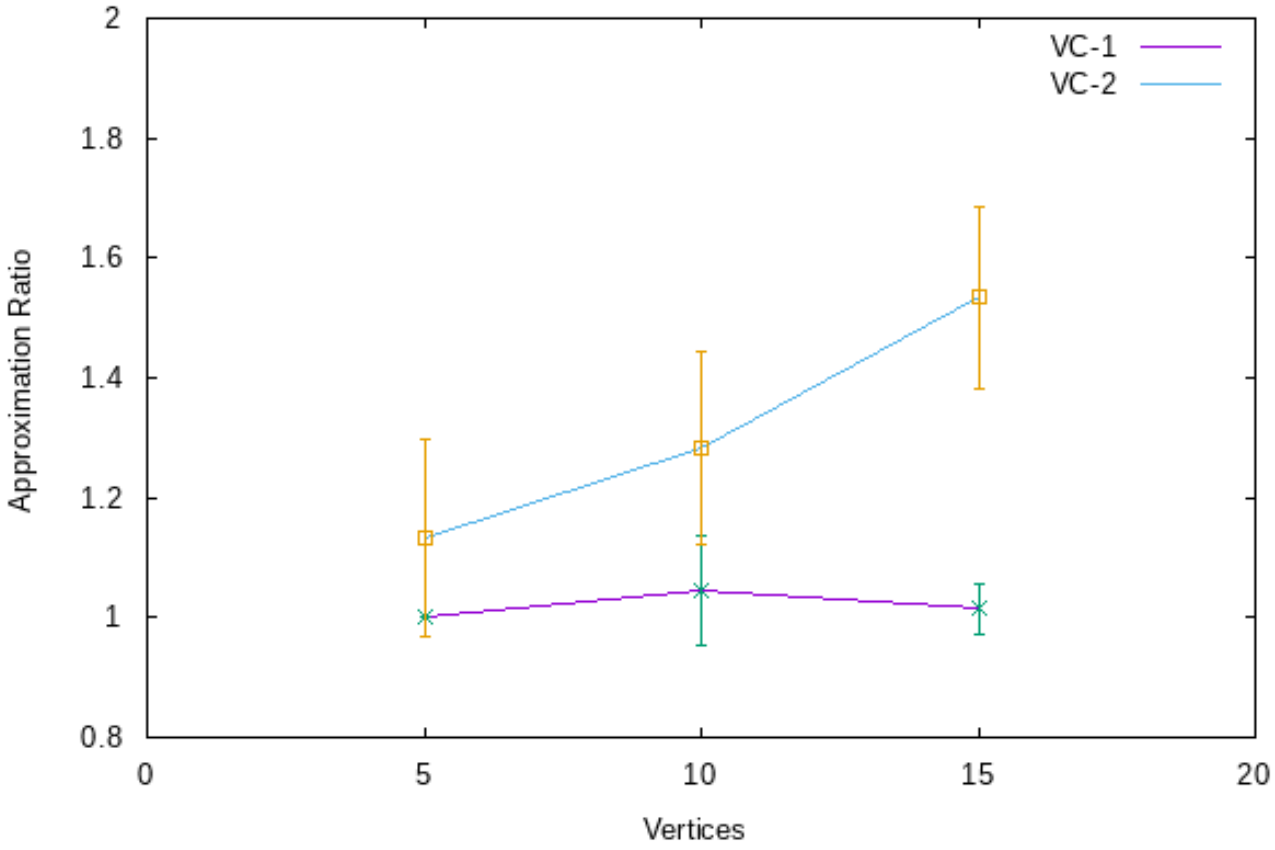

Figure 3: Approximation ratio for APPROX-VC-1 and APPROX-VC-2 algorithms

Table 4: Approximation ratio analysis for APPROX-VC-1,APPROX-VC-2 algorithm

vertex	mean APPROX-VC-1	stdev APPROX-VC-1	mean APPROX-VC-2	stdev APPROX-VC-2
5	1	0	1.13	0.16
10	1.045	0.09	1.28	0.16
15	1.014	0.04	1.53	0.15

3 Conclusion

⇒ From the above analysis based on plotted graphs, it can be observed that all algorithms can be used to determine the vertex cover even though their efficiency may vary based on running time and approximation ratio

⇒ APPROX-VC-2 algorithm has the minimum execution time but it fails to generate minimum vertex cover which can be observed from its high approximation ratio

⇒ CNF-SAT-VC has the minimum approximation ratio but its running time rises exponentially when the number of vertices increase

⇒ APPROX-VC-1 algorithm can be considered an optimal approach to solve large vertex cover problems as its running time is less than CNF-SAT-VC and it computes near optimal minimum vertex cover