UNIVERSITY OF WATERLOO

Department of Electrical & Computer Engineering

Report for Efficiency Analysis of Vertex Cover

Prepared by: -

Team Members-

S.NO	Name	Student ID	Email ID
1	Navni Gupta	20800951	n57gupta@uwaterloo.ca
2	Vaishali Singh	20800957	v47singh@uwaterloo.ca

December 7th, 2018

1. Introduction:

For this project, we are expected to analyze the efficiency of three different approaches or algorithms to solve the vertex cover problem for various sets of input, that is, the number of vertices for a graph. These three algorithms are: -

(i) Finding Vertex Cover using CNF-SAT
In this algorithm, we are creating a polynomial reduction of the decision version of vertex cover to CNF-SAT. We have to create our reduction and use minisat as a library to solve minimum VERTEX COVER problem for the graphs that are input to your program.

(ii) Finding Vertex Cover using APPROX-VC-1
In this algorithm, pick a vertex of highest degree (most incident edges). Add it to your vertex cover and throw away all edges incident on that vertex. Repeat till no edges remain. We will call this algorithm APPROX-VC-1.

(iii) Finding Vertex Cover using APPROX-VC-2
Pick an edge <u, v>, and add both u and v to your vertex cover. Throw away all edges attached to u and v. Repeat till no edges remain.

The analysis of all these three approaches to find the vertex cover are evaluated by their (a) running time and (b) approximation ratio, in which the computed vertex cover using CNF-SAT is used as optimal vertex cover.

2. Analysis of Algorithms:

This section analyses the efficiency of three different algorithms (CNF-SAT, APPROX-VC-1, APPROX-VC-2) that solves Vertex Cover Problem. “Efficiency” will be evaluated by two aspects: 1) Running Time and, 2) Approximation Ratio, which is characterized as the ratio of the size of the computed vertex cover to the size of an optimal (minimum-sized) vertex cover. In particular, the vertex cover computed by CNF-SAT algorithm was used as a benchmark to compute approximation ratios of other two algorithms, since CNF-SAT algorithm always guarantee an optimal (minimum-sized) vertex cover for any input graph.

Our objective is to measure, the running time and approximation ratio, for various values of |V| (number of vertices), for the graphs generated by graphGen. We perform this analysis for |V| = 5,10,15, and 20 for APPROX-VC-1 and APPROX-VC-2 but 5, 10, 15, and 16 for CNF-SAT. We generate 10 graphs for each value for |V|, compute the time and approximation ratio for each such graph. We measure the running time for 10 runs of each such graph. We, then compute the mean (average) and standard deviation across those 100 runs for each value of |V|. We then plot and analyse the mean for each value of |V| for which we made measurements, and the standard deviation as a yerrorbar. The plots will be discussed in the next section.

The analysis program is based on the multi-threads program assigned in final project. The main thread is responsible for I/O and the other three threads are responsible for the three algorithms for finding the vertex cover. In our program, initially I/O thread will be executed which will take graph input from GraphGen which randomly generates graph. After executing I/O thread,

it will execute three threads for three different algorithms concurrently. The pthread-getcpuclockid () function is used for getting the CPU time of each wrapper function instance. The analysis of the program is done on ECE Linux server.

2.1. Running Time Analysis

2.1.1 CNF-SAT Running Time Analysis

CNF-SAT is a well-known NP-Complete problem. According to the results, generally the CNF-SAT algorithm has the largest running time amongst the three algorithms to find the vertex cover for all the input graphs. Particularly, CNF-SAT will have exponential running time and as the vertices number increases, this algorithm will take hours to get a result.

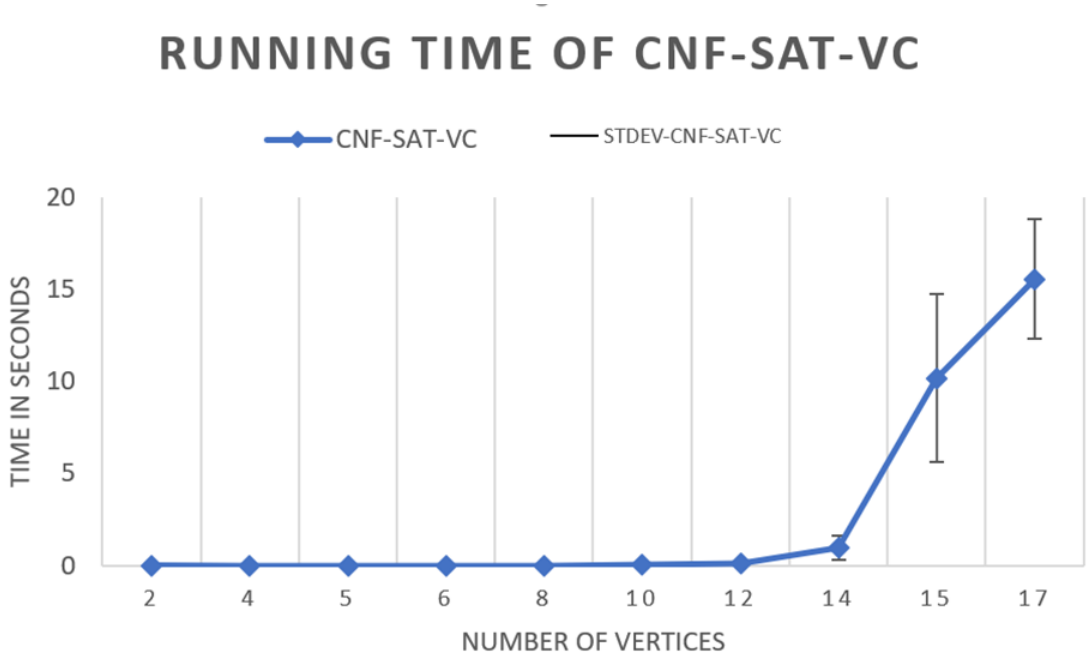

Figure 1: Mean Running time (in seconds) for CNF-SAT-VC

As we can see in Figure 1, the run time of CNF-SAT-VC approach increases polynomially with the number of vertices. Since it is an NP-hard problem, the run time increases a lot after |V|=14. The algorithm takes a lot of time to run for vertices ≥ 15 and was leading to time outs. This algorithm takes a lot of time as compared to both the approaches APPROX-VC-1 and APPROX-VC-2. The scale of the vertical axis in this graph is in seconds.

The huge difference in the time taken 2 by CNF-SAT-VC as compared to the other two algorithms is because of the time taken by Minisat to solve the CNF-SAT polynomial reduction of the vertex cover. For |V| > 14 we notice very high standard deviations. This may be because varying vertex cover size for the same vertex count will result in different run times. Since the run time for CNF-SAT-VC for higher vertex counts is high, it leads to higher standard deviations.

2.1.2 Running Time Comparison between APPROX-VC-1 & APPROX-VC-2

In order to compare the running time of both the approximate vertex cover algorithm, that is, APPROX-VC-1 & APPROX-VC-2, we will run them separately on the scale of vertex number 1 to 20 at an increment of 5.

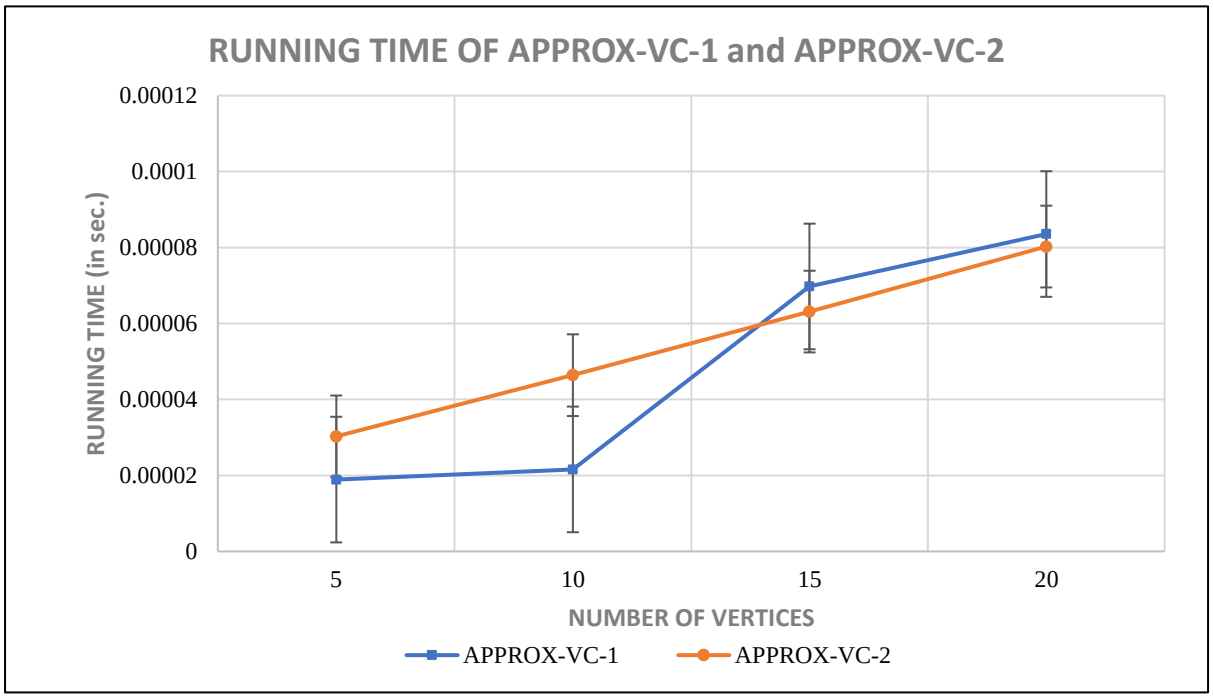

Figure 2: Mean Running time (in seconds) for APPROX-VC-1 & APPROX-VC-2

As we can see in Figure 2, for each value of vertex count |V|, the performance of APPROX-VC-2 is much better than that of APPROX-VC-1. This increased mean run time for APPROX-VC-1 can be attributed to the computation of the vertex with highest degree at each step for finding the vertex cover, which is an additional time intensive component of this approach. APPROX-VC-2, on the other hand, simply picks a random vertex at each step without much computation involved and hence is faster.

In general, we observe a polynomial increase in time for both APPROX-VC-1 and APPROXVC-2 approaches. We observe a higher standard deviation for |V| >10 for the mean run time in APPROX-VC-1 approach as compared to that in APPROX-VC-2 approach. The high standard deviation in the case of APPROX-VC-1 approach could be due to different run times (which includes time intensive computation of the highest degree vertices and which may vary with different graphs for the same vertex count) for each of the 10 graphs for a vertex count. On the other hand, in the case of APROX-VC-2 approach, for smaller vertex counts, almost all edges in the graph would be removed in no time (as the vertices are randomly selected) bringing the algorithm to end in almost similar times. For |V| >10, both the algorithms have a high standard deviation because we have used completely random 10 graphs to compute the run times. Each graph might be having a different run time and hence contributes to standard deviation.

2.2 Approximation Ratio Analysis

The approximation ratio analysis is computed as ratio of size of computed algorithm to the size of original algorithm. In our analysis, the approximation can be redefined as:

$$\text{Approximation Ratio} = \frac{Size\ of\ Computed\ Vertex\ Cover}{Size\ of\ Optimal\ Vertex\ Cover}$$

So, the approximation ratio for both the algorithm is calculated using the above formula.

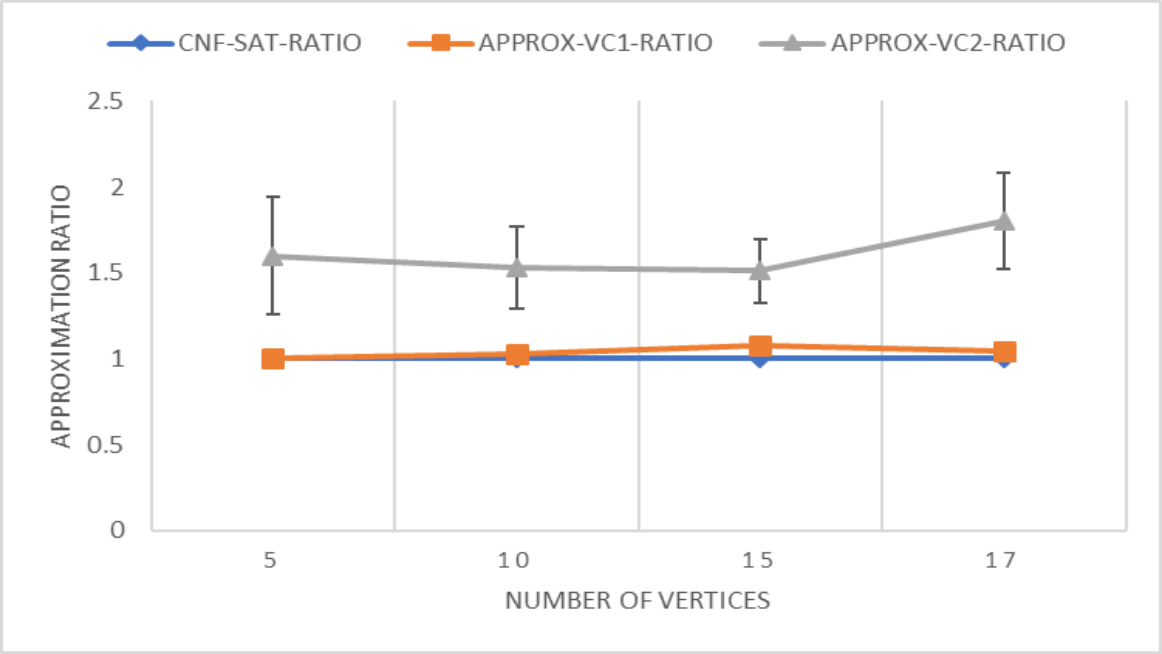

Figure 3: Mean Approximation Ratio plot for CNF-SAT-VC, APPROX-VC-1 & APPROX-VC-2

Figure 3 plots the approximations ratio of CNF-SAT-VC, APPROX-VC-1 and APPROX-VC-2 approaches. The CNF-SAT-VC computes optimal vertex cover, so, in this case my optimal and computed vertex cover will be same, thus giving approximation ratio equal to 1 always. We notice that APPROX-VC-2 approach is always more as compared to that of APPROX-VC-1 approach. This is because APPROX-VC-2 approach relies on randomness in terms of Vertex selection and hence deviates from the optimal vertex cover solution. APPROX-VC-1 approach, on the other hand, computes the Vertex with the highest degree at each step and hence achieves a solution which is much closer to the optimal solution. All the mean approximation values in APPROX-VC-1 approach lie in the vicinity of 1.000 and hence are very close to the optimal solution. APPROX-VC-2 approach, however, always results in more vertices in the vertex cover as compared to the optimal vertex cover. We notice that APPROX-VC-2 has higher standard deviations as compared to APPROX-VC-1. This is because APPROX-VC-2 algorithm relies on random vertex selection and provides vertex cover results based on the selection (which has no sense of optimality). This random selection in different graphs for the same vertex count may give different approximation ratios. Since, APPROX-VC-2 follows a probabilistic model, high standard deviation is expected.

3. Conclusion

According to efficiency in terms of running time, APPROX-VC-2 performs best followed by APPROX-VC-1 and CNF-SAT-VC respectively. CNF-SAT-VC, being a NP-COMPLETE problem, fails to finish in polynomial time for $|V| > 15$.

With respect to efficiency in terms of Approximation Ratio, APPROX-VC-1 is almost comparable to the optimal solution (CNF-SAT-VC) with the mean values almost approaching a ratio of 1. APPROX-VC-2, however, always results in high mean approximation values and higher number of vertices than the optimal solution in the vertex cover result. Thus, as far as Approximation Ratio is concerned, APPROX-VC-1 approach has better efficiency than APPROX-VC-2.

If we look at efficiency in terms of both Running Time and Approximation Ratio, APPROXVC-1 approach outperforms both the other approaches as it gives almost optimal vertex cover solution in very less time. APPROX-VC-2 approach performs good in terms of time but never results in an optimal vertex cover. CNF-SAT-VC approach guarantees an optimal vertex cover solution but fails to finish in polynomial time for higher values of vertex count.

In conclusion, the CNF-SAT algorithm will always find the minimum-sized or optimal vertex cover, but it is not time-efficient. On the other hand, the other two algorithms, that is, APPROX-VC-1 & APPROX-VC-2 are much more efficient both in time and space but it might be possible that they do not return optimal solution. Overall, APPROX-VC-1 has the best performance in terms of running time and returning minimum-sized vertex cover because it costs much lower time and space as compared to CNF-SAT but has lower approximation ratio compared to APPROX-VC-2.