PROJECT REPORT

ON

Quantitative Analysis of Vertex Cover Using Various algorithms

SUBMITTED BY:

Maninder Singh 21029657

Nishtha Kapoor 21050720

SUBMITTED TO:

Reza Babaee

APRIL 2023

Table of Contents

1 Introduction

In this Project our aim is to quantitatively analyse the vertex cover problem using various algorithms. We will analyse the efficiency of the the algorithms using the running time and approximation ratio. Vertex cover of a graph can be defined a the subset of vertices such that each edge of the graph is incident at one of the vertices in the vertex cover. Vertex cover is a NP hard problem which means there is no particular polynomial time algorithm to solve it for all cases.
In our project we are implementing six algorithms namely:

1. CNF-SAT-VC
2. CNF-3-SAT-VC
3. APPROX-VC-1
4. APPROX-VC-2
5. REFINED-APPROX-VC-1
6. REFINED-APPROX-VC-2

2 Design Of Algorithms

2.1 THREADING

We have used nine threads in our code one for user input, and one each for all 6 algorithms.
We added one thread named group-thread-1 which is supposed to create thread for Approx-VC-1 and then after it's completion it creates another thread for Refined-Approx-VC-1. We did it because We did not wanted the main thread to be waiting for Approx-VC-1 and then running refined-vc-1. Similarly we did it for group-thread-2 for running Approx-vc-2 and then Refined-approx-vc-2. Thus total nine thread for our project.

2.2 CNF-SAT-VC

CNF-SAT-VC:: In this algorithm we are using binary search algorithm to find the minimum vertex cover.This algorithm will return a boolean value depending if it found the vertex cover or not for a given size of vertex cover. We have set the minimum number of vertex cover as 1 and maximum vertex cover as the the number of vertices in the graph. This algorithm solves the vertex cover for a given boolean graph which is in CNF using a SAT solver. Since vertex cover can be between 1 to V where V is total number of vertices, we decide our search area between 1 to V. We check our Sat solver for vertex cover size to be mid of 1 and V. If is able to solve then we limit our search space from 1 to mid else from mid to V.

2.3 CNF-3-SAT-VC

CNF-3-SAT-VC:: In this algorithm we are solving vertex cover using 3-CNF approach. We are breaking down the first and fourth clauses of CNF-SAT-VC into clauses which has maximum three literals using the introduction of dummy variables. In this approach the function initially takes the number of vertices as the initial vertex cover and then divides the range into half and attempts to solve the 3 CNF formula. We vary the search space for vertex cover size in the same way we did for CNF-SAT-VC algorithm.

2.4 APPROX-VC-1

APPROX-VC-1:: In this we find vertex cover using the approximation algorithm. We choose the vertex with the highest degree from the graph and add it our set. We remove all the edges from this vertex and again find the vertex with highest degree from the remaining vertices. we do it until the vertices in our set form the vertex cover.

We optimized this algorithm by maintaining the ordered map to store the degree to vertex mapping, sorted by degree of the vertices. Thus we always pick the first item which has the highest degree and modify all the degrees of the vertices which are connected to the current vertex. We continue it till our vertices in our set form vertex cover.

2.5 REFINED-APPROX-VC-1

REFINED-APPROX-VC-1:: In this algorithm we take output of APPROX-VC-1 and remove certain vertices from it such that it still remains vertex cover. We actually implemented this algorithm using two ways and analysed both of them.

Algo RECURSIVE-VC-1 : In this algorithm it takes the vertex cover of APPROX-VC-1 and calls the recursive function to find the vertex cover which is the refined vertex cover and then sorts the list. The time complexity of the REFINED-APPROX-VC-1 algorithm is $O(2^n)$, where n is the size of the vertex cover returned by the APPROX-VC-1.

Algo GREEDY-VC-1: In this algorithm, we selected the vertex of the highest degree say "v" from the output of APPROX-VC-1, and add this vertex to our set. Now our goal is to remove a vertex from the output of APPROX-VC-1 which is not required in the minimum vertex cover. Now we will look if we have any vertex connected to vertex "v" and say the name of this vertex is "u". We can remove this vertex "u" from the output of APPROX-VC-1 if:
1. "u" is present in the output of APPROX-VC-1.
2. all the connected vertices to "u" are also part of APPROX-VC-1 so removing "u" is not a problem for vertex cover.
Repeat these steps till the complete output of APPROX-VC-1 is parsed.

since RECURSIVE-VC-1 is a naive approach to try out each combination of vertices and return the combination which has a vertex cover and whose size is minimum, it is taking significant longer time than GREEDY-VC-1. Thus for the rest of our analysis we took GREEDY-VC-1 as REFINED-APPROX-VC-1.

2.6 APPROX-VC-2

APPROX-VC-2:: In this algorithm we have implemented the approximation algorithm. In this algorithm we select two vertices which have an edge between them and are not in the vertex cover and then add it to our set. It then removes all the edges connected to these two vertices and does this till our set becomes the vertex cover. In each iteration we select the uncovered edge and its neighbouring vertex and add it to the vertex cover and remove the attached edges till all the edges gets removed. The final output of the vertex cover is then sorted in the ascending order.

2.7 REFINED-APPROX-VC-2

REFINED-APPROX-VC-2:: In this algorithm we take output of APPROX-VC-2 and remove certain vertices from it such that it still remains vertex cover. We actually implemented this algorithm using two ways and analysed both of them.
Algo RECURSIVE-VC-2 : In this algorithm it takes the vertex cover of APPROX-VC-2 and calls the recursive function to find the vertex cover which is the refined vertex cover and then sorts the list. The time complexity of the REFINED-APPROX-VC-2 algorithm is $O(2^n)$, where n is the size of the vertex cover returned by the APPROX-VC-2.
Algo GREEDY-VC-2 : In this algorithm, we selected the vertex of the highest degree say "v" from the output of APPROX-VC-2, and add this vertex to our set. Now our goal is to remove a vertex from the output of APPROX-VC-2 which is not required in the minimum vertex cover. Now we will look if we have any vertex connected to vertex "v" and say the name of this vertex is "u". We can remove this vertex "u" from the output of APPROX-VC-2 if:
1. "u" is present in the output of APPROX-VC-2.
2. all the connected vertices to "u" are also part of APPROX-VC-2 so removing "u" is not a problem for vertex cover.
Repeat these steps till the complete output of APPROX-VC-2 is parsed.

since RECURSIVE-VC-2 is a naive approach to try out each combination of vertices and return the combination which has a vertex cover and whose size is minimum, it is taking significant longer time than GREEDY-VC-2. Thus for the rest of our analysis we took GREEDY-VC-2 as REFINED-APPROX-VC-2.

3 Analysis of various Algorithms

We have done the analysis of various algorithms by comparing their efficiency. Efficiency will be compared based on two factors that are running time and approximation ratio. We are using pthread so that we can simultaneously find the running time of each algorithm and also the approximation ratio.
We have created four graphs
First for comparing CNF-SAT and 3CNF-SAT based on running time
Second for comparing APPROX-VC-1 APPROX-VC-2 REFINED-APPROX-VC-1 REFINED-APPROX-VC-2 based on running time
Third for comparing Approx-VC-1 and Approx-VC-2 algorithm based on running time
Sixth for comparing all six algorithm based on Approximation ratio.

For analysis we have also put a timeout of 5 minutes so as to avoid long time waiting for the result in case an algorithm takes more time than that time it would print timeout in the output.
We have plotted four graph based on running time and approximation ratio.

3.1 Comparision of Running time of CNF-SAT and CNF-3-SAT

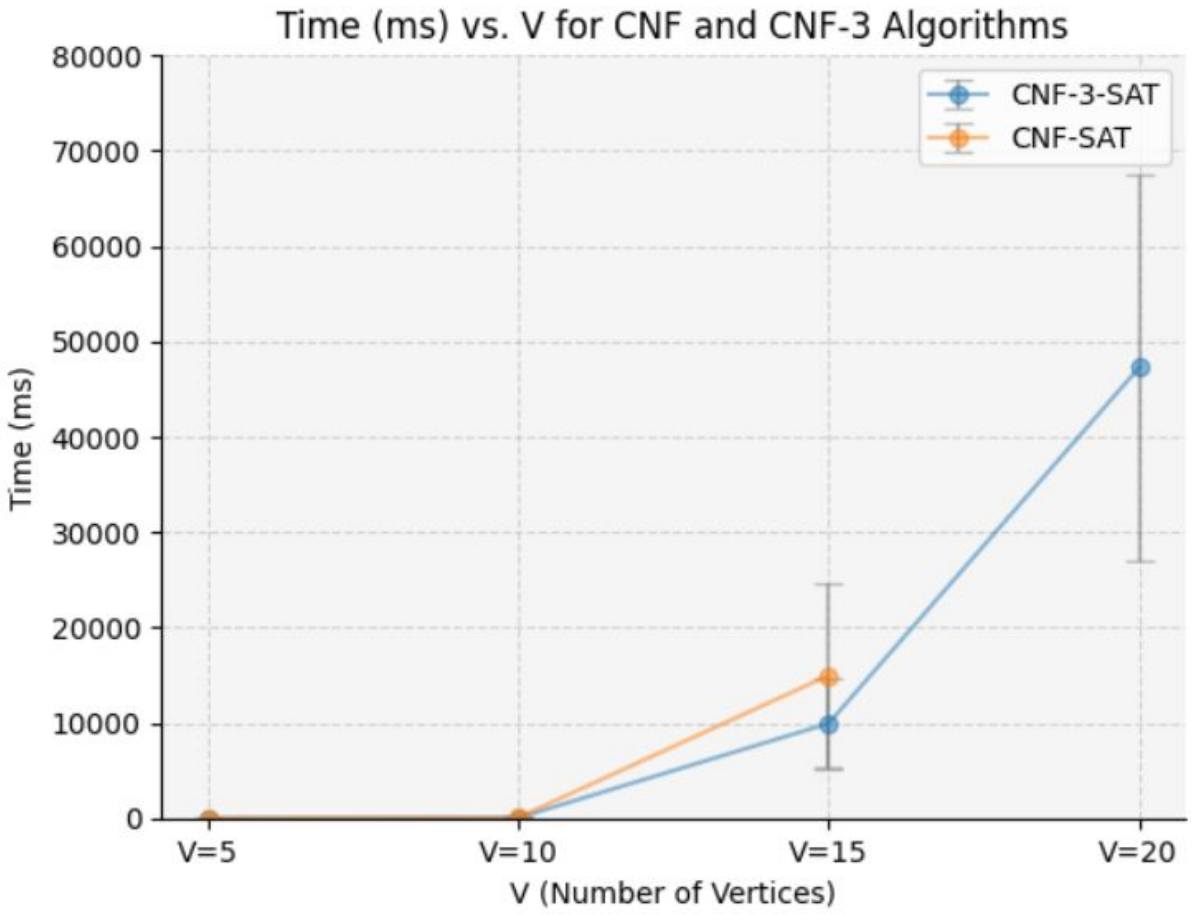

Figure 1: CNF-SAT vs CNF-3-SAT based on running time

While comparing the running time of CNF-SAT and CNF-3-SAT it was noticed that for V=5 and V=10 the execution time for both the algorithms is almost same. However, from Vertices=15 onwards time taken by the CNF-SAT algorithm is more than the 3CNF algorithm. Although, the number of clauses and literals is more in 3CNF but the size of each clause in 3CNF is at most 3 whereas size of each clause is large enough in the CNF algorithm. SAT solver computes smaller clauses relatively faster as compared to bigger ones and if vertex cover is unsatisfiable for any size of iteration the clause is computed to be false quickly because it only has 3 literals at max, on the other hand, a larger clause in CNF takes more time to compute. Additionally, whenever a clause is computed false vertex cover iterates to next iteration which saves time in 3CNF as it computes faster. So vertices are increased CNF-3-SAT algorithm finds the vertex cover faster than the CNF-SAT algorithm

We can also compute that as the number of vertices increases the standard deviation also increases and thus we can say that the standard deviation is dependent upon the number of vertices. Higher

the number of vertices more is the standard deviation.

3.2 Comparision of Running time of Approximation Algorithms

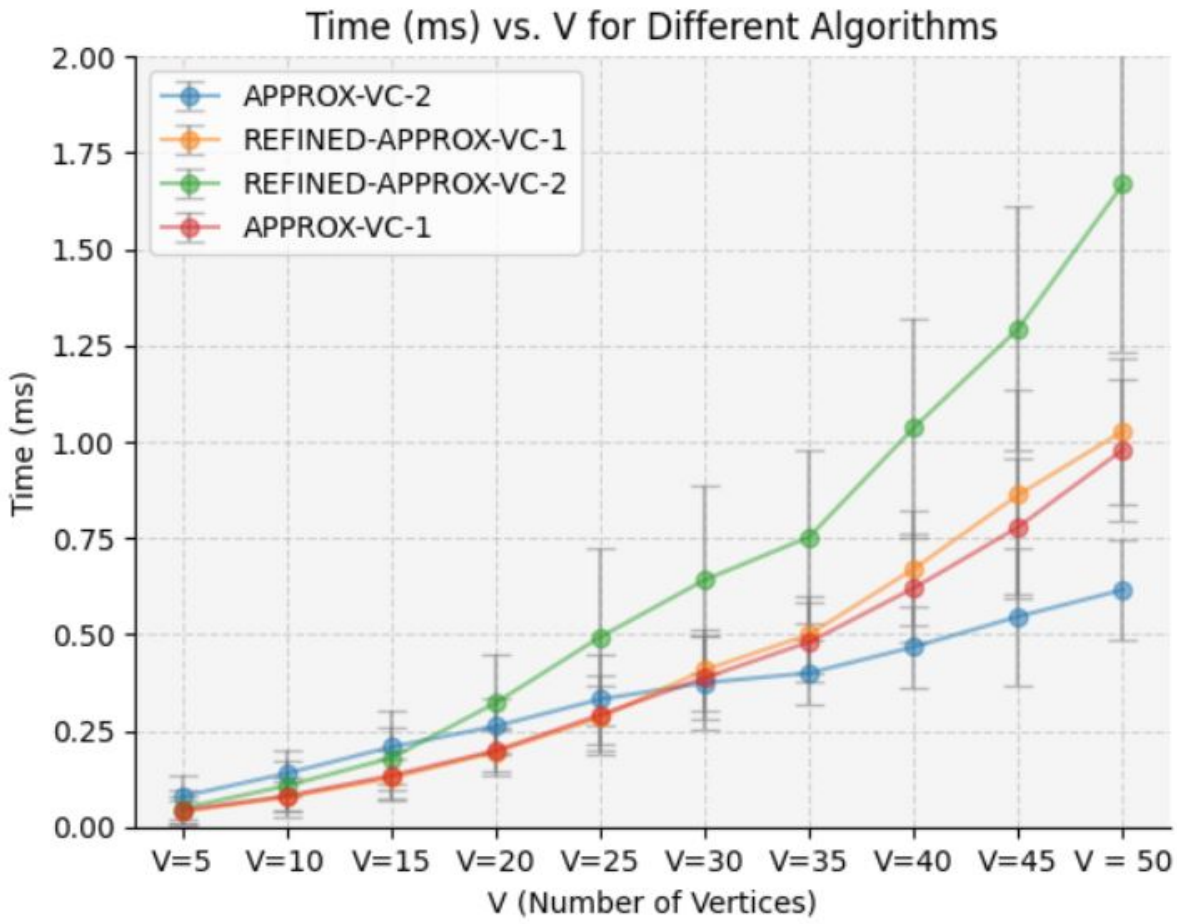

Figure 2: Comparison of various approximation algorithms

In Figure 2 we compare the running time of Approximation algorithms for vertex cover problem. In this graph we can see that the APPROX-VC-2 has the least running time whereas the REFINED-APPROX-VC-2 has the maximum running time for finding the vertex cover this is because REFINED-APPROX-VC-2 depend upon the output of APPROX-VC-2. Since APPROX-VC-2 have the highest approximation ratio with large standard deviation, which means its soluton is not close to optimal. It makes sense since we choose any arbitrary edge and add its vertices to our solution. and thus solution is not guaranteed to be optimal. So the size of vertex cover (output) of APPROX-VC-2 is large as compared to others, input of REFINED-APPROX-VC-2 is large and thus trying to remove vertices from larger set takes longer time, thus REFINED-APPROX-VC-2 takes more time than REFINED-APPROX-VC-1 whose input(APPROX-VC-1) is of lower size. APPROX-VC-1 has really less approximation ratio than APPROX-VC-2 with similar standard de-

viation of approximation ratios.

3.3 Comparision of Running time of Approx-VC-1 and Approx-VC-2

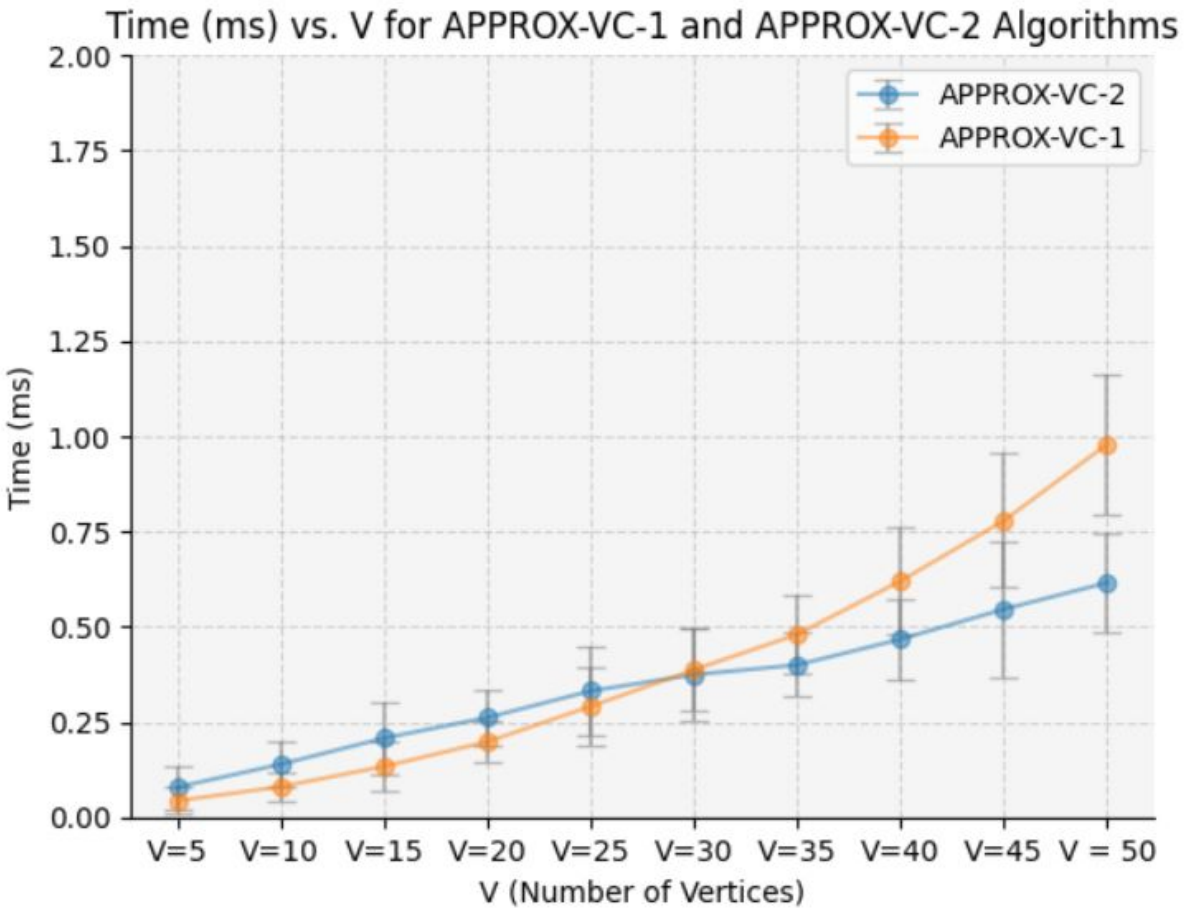

Figure 3: Approx-VC-1 and Approx-VC-2 based on their running time

In Figure 3 we have compared the running time of APPROX-VC-1 AND APPROX-VC-2 and it is seen that for V=5 to V=25 APPROX-VC-1 runs faster and for V=25 the running time is almost the same but after the vertices are increased the APPROX-VC-2 algorithm runs faster and has a steadily increasing graph. This is because as the no of vertices increase, APPROX-VC-1 takes more time to find the vertex with highest degree for which we are using map and it takes time to get the vertex with highest degree since it takes time for insertion and deletion in map and after that we need to remove every edge connected to this. Whereas for APPROX-VC-2 we simply select arbitrary edge and remove it from the vertex and thus it takes lesser time since no sorted need for selecting the vertex to be added in our set.

3.4 Comparision of Approximation Ratio of all six algorithms

For finding the approximation ratio we know that both CNF-SAT and CNF-3-SAT are giving the optimal solutions but from V=0 to V= 15 CNF-3-SAT was running faster than rest of the algorithms so we took this as the optimal and after that as these algorithm were timed out so we took the third best running algorithm based on time that is REFINED-APPROX-VC-1 as the optimal algorithm for V =20 to V=50.

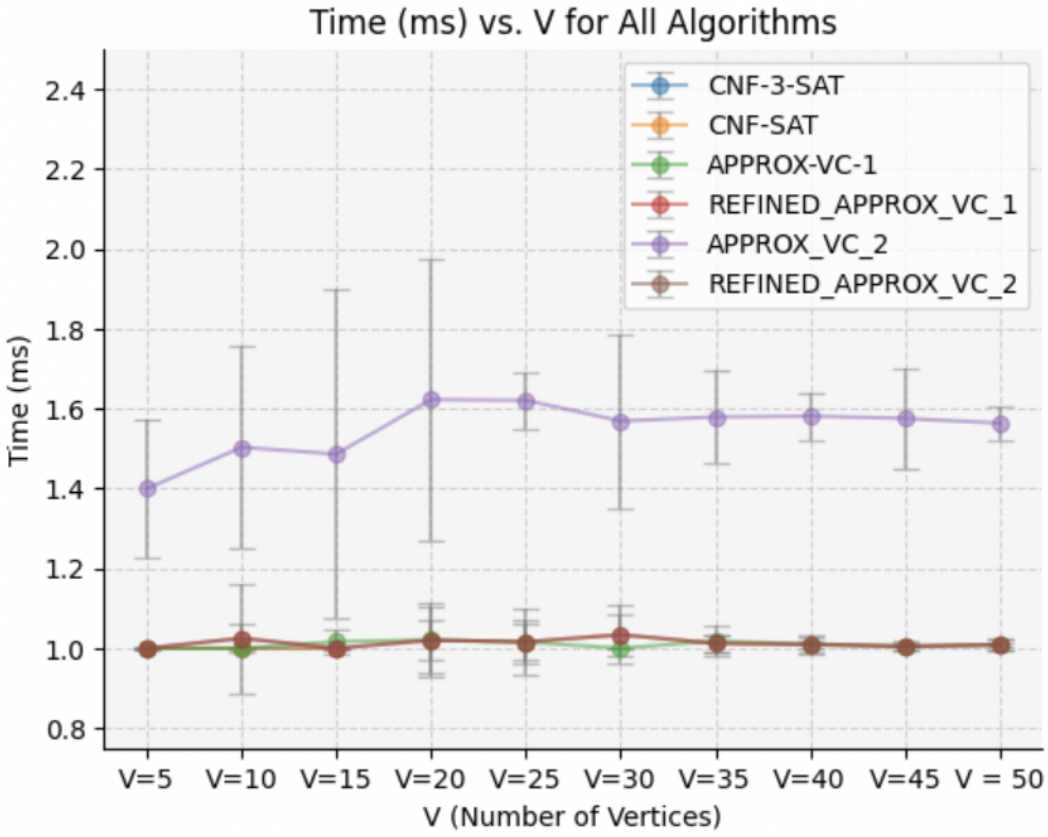

Figure 4: Graph of all 6 algorithms based on their approximation ratio

In Figure 4 we are comparing approximation ratio of all six algorithms and we can see that approximation ratio of :

1. CNF-SAT and CNF-3-SAT is always 1 with standard deviation 0 which shows that they have the best performance which is obvious since they are optimal and always have vertex cover of minimum size and thus even if we tried for 10 different graphs and 10 tries on each, everytime we get mean approximation ratio as 1 and thus it is a straight line at 1 and since none of its answers deviate from mean its sd is always zero and cannot be seen. CNF-SAT and CNF-3-SAT are plotted till V = 15 and V= 20 respectively.

2. APPROX-VC-2 has the highest approximation ratio with high standard deviation of approximation ratio which really implies that vertex cover returned by APPROX-VC-2 is not close to the minimum vertex cover returned by any other algorithm which is fine as we select any edge arbitrarily and include in our vertex cover which is not guaranteed to be in minimal vertex cover.
3. REFINED-APPROX-VC-1 show stable values across different graph sizes this means that the have relatively consistent performance for different vertex sizes.This is because APPROX-VC-1's solution is greedy and degree based and thus on an average its answer is close to minial vertex cover and thus it is lesser work for REFINED-APPROX-VC-1 to do as lesse vertex need to be removed.
4. REFINED-APPROX-VC-2 shows us some spike at V=10 , this implies there will be certain graphs for the input for which APPROX-VC-2 includes certain vertices in the answer and which are not removable from the set chosen by APPROX-VC-2. and thus REFINED-APPROX-VC-2 also deviates from the answer. It is because it greedily chooses the vertices from the APPROX-VC-2's answer but still not able to make it close to size of minimal vertex cover.

4 Conclusion

CNF-3-SAT is considered optimal till 20 number of vertices.Since CNF-SAT and CNF-3-SAT have always the approximation ratio as 1 with the standard deviation of 0. but beyond 20 vertices that CNF-3-SAT is timed out since we implemented timeout of 5 mins for our experiments.

For cases with V greater than 20, REFINED-APPROX-VC-1 have the approximation ratio close to 1, It can be considered as the best solution if we have to pick one regardless of the time taken.

APPROX-VC-2 is the fastest among all algorithms but the solution is not guaranteed to be optimal since we pick any edge and add their vertex to our solution set regardless if the vertex needed to be included in the minimum vertex cover.

REFINED-APPROX-VC-1 and REFINED-APPROX-VC-2 are bit slower than APPROX-VC-1 and APPROX-VC-2 algorithms but REFINED-APPROX-VC-1 is better solution and are more close to minimum vertex cover beyond 20 vertices.

For improving the encoding we have used binary search for both CNF-SAT and CNF-3-SAT for reducing time taken for higher vertices to give the optimal solution.

5 References

1. OpenDSA Data Structures and Algorithms Modules Collection CHAPTER 28 LIMITS TO COMPUTING
2. Encoding given for assignment 4