Analysis of Minimum Vertex Cover using MiniSAT and Approximation Algorithms

Ira Bhardwaj* and Aditi S Patade**

* Department of Electrical and Computer Engineering, Student ID : 20813150

** Department of Electrical and Computer Engineering, Student ID : 20868672

Abstract: This project report analyses the efficiency of different methods to find minimum vertex cover of a graph, which is an NP-hard problem. The three approaches to solve the minimum vertex cover problem used are MiniSAT solver (CNF-SAT-VC) and two other approximation algorithms, referred to as APPROX-VC-1 and APPROX-VC-2 in the report. The efficiency analysis is done based on running time and approximation ratio of each approach. CNF-SAT-VC is found to be the most effective way to solve the vertex cover problem.

1. INTRODUCTION

In graph theory, a vertex cover of a graph is a set of vertices that includes at least one endpoint of every edge of the graph. The problem of finding a minimum vertex cover is a classical optimization problem in computer science and is a typical example of an NP-hard optimization problem that has an approximation algorithm.

In our project, we have used three different approaches to solve the minimum vertex cover using a C++ program that uses four threads:

1. for I/O
2. for CNF-SAT-VC
3. for APPROX-VC-1
4. for APPROX-VC-2

Algorithm 1 : using MiniSAT:
This approach is based on a polynomial time reduction to CNF-SAT and use of a SAT solver. The SAT solver used to implement this, is MiniSat.

Algorithm 2 : APPROX-VC-1:
This approximation algorithm involves picking a vertex that has the greatest number of incident edges, adding it to the vertex cover and removing all the edges incident with the vertex. This is then repeated till all the edges are covered.

Algorithm 3 : APPROX-VC-2:
The third algorithm, which is also an approximation algorithm is a more random approach than APPROX-VC-1. It involves adding a random edge (u,v) to the vertex cover and removing all the

other edges attached to u and v. This is repeated till no more edges are left. This algorithm is implemented using recursion in our program.

2.ANALYSIS

The data was collected for 10 runs for each value of V (where V is the number of vertices in the input graph). The values of V, for which data is collected are 5,7,9…17,19 and after that in steps of five as 20,25,30…50. The reason for choosing these values, is the large different in execution times of CNF-SAT-VC for values V<19 and V≥19, as it is evident from the following analysis.

1.Run Time Analysis

For the run time analysis, the graphs have been plotted for the mean run time values for different values of V. The error bars show the standard deviation from the mean value of run time, as we collected data of ten graphs for each value of V displayed. The running time graphs for CNF-SAT-VC and the approximation algorithms have been made separately for better analysis as their running times differ by a huge amount.

Figure 1 shows the running time graph of CNF-SAT-VC. It can be seen from the graph that the run time values increase with the increase in no. of vertices. For values of number of vertices greater than or equal to 19, the run time for CNF-SAT-VC was more than two minutes (our code gave a timeout set for two minutes), so we limited our data sets to V=19 for CNF-SAT-VC. When executed without the timeout, program didn't return CNF-SAT-VC for more than three minutes, so we set the timeout value to 120 seconds (2 minutes).

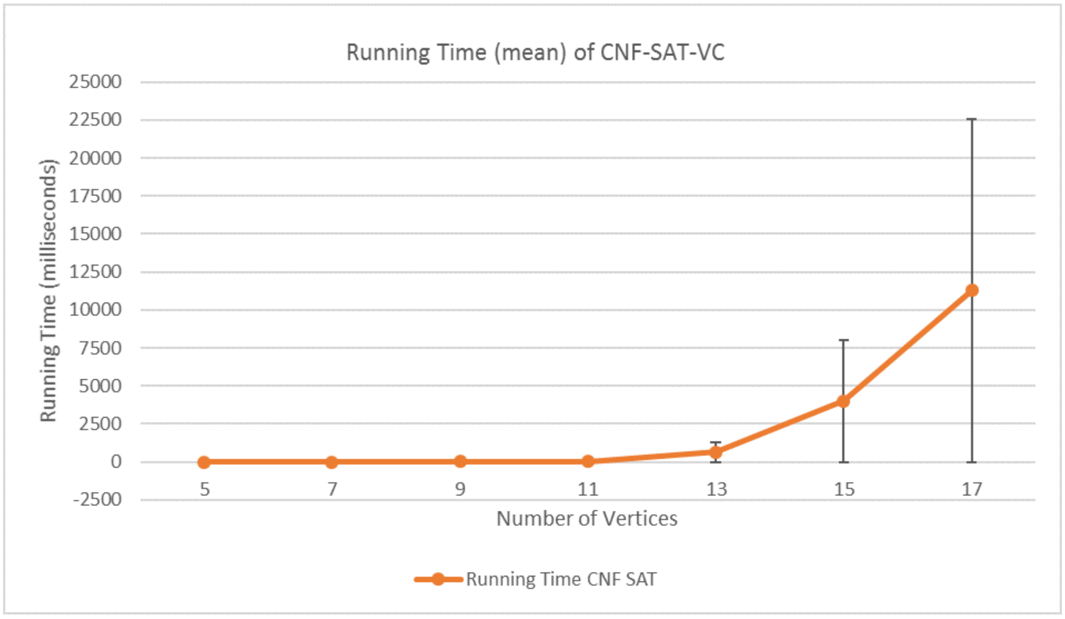

Figure 1: Mean Running Time plot for CNF-SAT-VC

For number of vertices upto 11, the graph is almost linear, the mean run time values for these range from 1.1035 ms to 51.597 ms. For number of vertices greater that 11, the mean run time keeps on increasing. V=11 onwards, the differences between the mean run time, as compared to the previous interval of V, also has a large difference. This rapidly increasing run time values with the increase in number of vertices may be because the SAT solver has greater number of clauses to satisfy for higher values of V. The highest run time value recorded for V=17 is 35 seconds. The standard deviation, for CNF-SAT-VC also shows an increase with the increase in number of vertices.

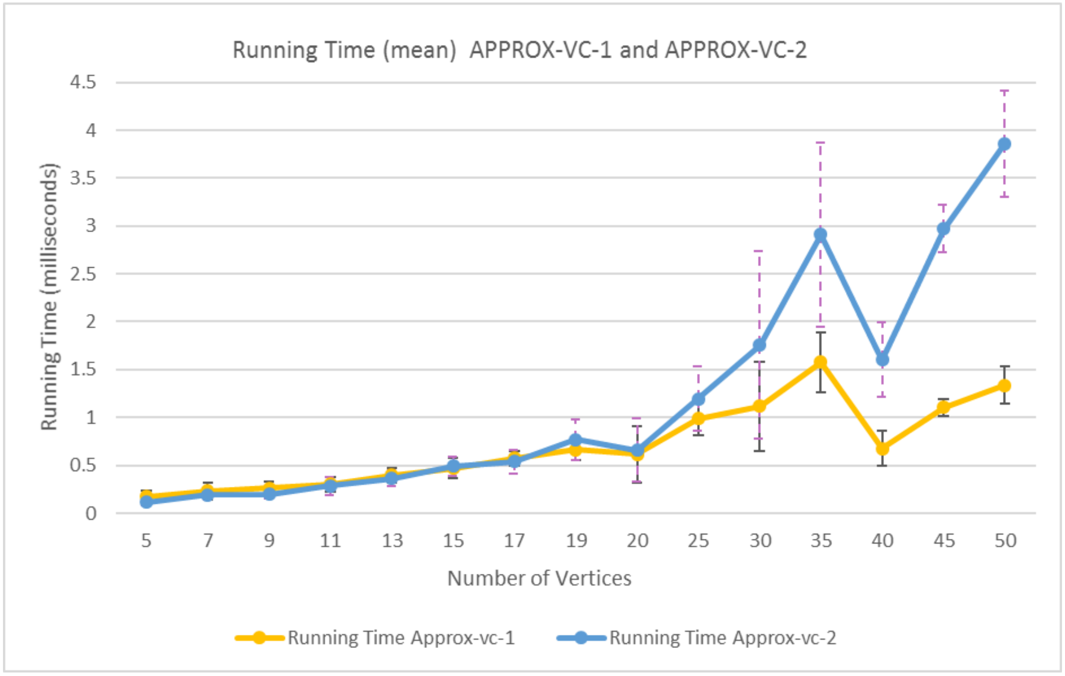

Figure 2: Mean Running Time graph for Approximation Algorithms (APPROX-VC-1 and APPROX-VC-2)

In the approximation algorithms, the running time increases with increase in number of vertices. For number of vertices until 13, the mean running time of APPROX-VC-2 is slightly less than that of APPROX-VC-1. But from number of vertices of 15, the mean running time for APPROX-VC-2 becomes higher than that for APPROX-VC-1. This may be explained by the fact that, until 13 the number of vertices is less, so the mean running times do not vary greatly. However, from V=17 the time for APPROX-VC-2 is significantly larger than APPROX-VC-1 which my be because APPROX-VC-2 algorithm is of random nature while APPROX-VC-1 limits the selection of vertex cover vertices by choosing the vertices with highest number of incident edges. This approach decreases the run time for APPROX-VC-1 as compared to APPROX-VC-2 for higher values of V.

If we see the overall mean running time for both figure 1 and figure 2, CNF-SAT-VC has greater running time than the approximation algorithms. As, vertex cover is a NP hard problem, and CNF-SAT-VC uses a SAT solver to find the vertex cover and the result cannot be achieved in polynomial time. The complexity in time for CNF-SAT-VC is more than the approximation algorithms. The implementation of CNF-SAT-VC includes a lot of loops(for and while loops) and conditions to form the clauses and then try to compute the vertex cover for each possible value of vertex cover size, starting from the minimum till a solution is reached that satisfies all the clauses.

2.Approximation Ratio

To analyze the approximation ratio, the approximation ratio was calculated for all the three algorithms for different values of number of vertices (V). Approximation ratio is the ratio of the size of the computed vertex cover to the size of an optimal (minimum-sized) vertex cover, which in this case is size of CNF-SAT-VC. Mathematically,

$$ApproximationRatio = \frac{Sizeofcomputedvertexcover}{Sizeofoptimalvertexcover}$$

For each value of V, the program was executed for ten different graph inputs. The mean value of approximation ratio was then calculated for each value of V.

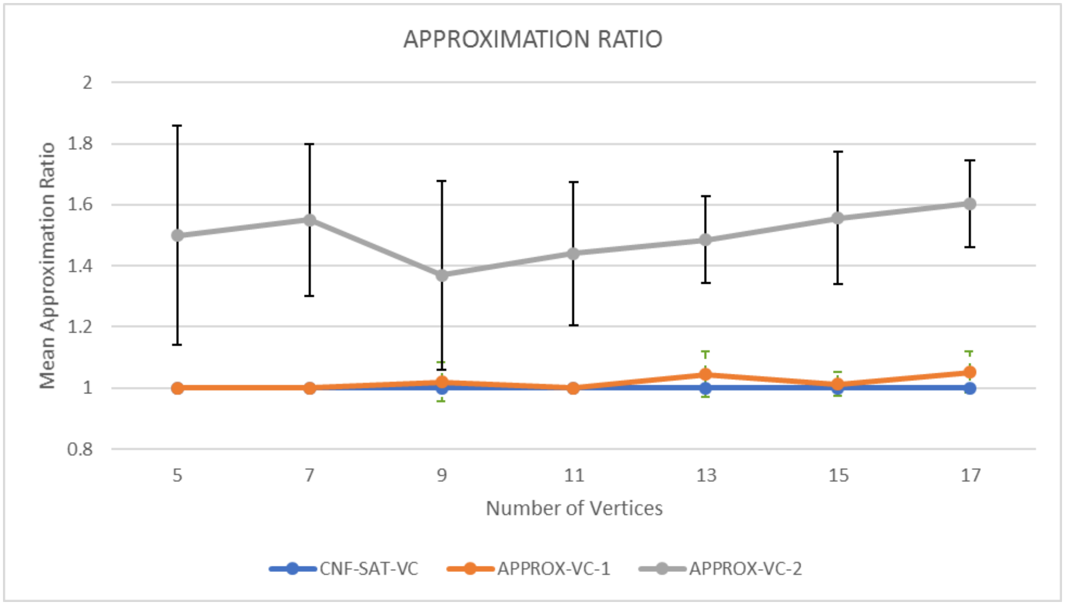

Figure 3: Mean Approximation graph for CNF-SAT-VC and Approximation Algorithms

For the analysis, we have taken the CNF-SAT-VC as the optimal solution for vertex cover. As seen from the graph, the approximation ratio for CNF-SAT-VC has a constant value of 1. The approximation ratio for APPROX-VC-1 is very close to the optimal vertex cover. This may be due to the nature of APPROX-VC-1 where it chooses the vertices having the most number of incident edges, and thus the vertex cover size is close to the optimal solution as the goal is to find the minimum cover. Some spikes can be seen where number of vertices are 9, 13 and 17.

APPROX-VC-2 shows a rather fluctuating trend which may be because of its random nature to choose vertices. The approximation ratio values for APPROX-VC-2 are higher than the other two algorithms at all instances. APPROX-VC-2 picks vertices randomly, which results in a vertex cover that has a higher approximation ratio as the vertex cover size is not minimal.
Thus, this analysis shows that APPROX-VC-1 solution is closer to giving a minimal vertex cover size than APPROX-VC-2.

3.CONCLUSION

If we consider the time complexity of the algorithms, for CNF-SAT-VC the execution time is larger than both the approximation algorithms and increases rapidly for as the number of vertices increases. So, it is not an effective way to get vertex cover solution for larger number of vertices. For, large number of vertices we can consider APPROX-VC-1 as it has lower running times than APPROX-VC-2. Also, looking at the approximation ratios, the CNF-SAT-VC gives an optimal minimum vertex cover solution for every value of number of vertices. The next algorithm which is close to the optimal solution is APPROX-VC-1.

Thus we can conclude that, as the research to find a more efficient way to solve vertex cover problem continues, we can consider using CNF-SAT for graphs with lower number of vertices and APPROX-VC-1 for larger number of vertices where CNF-SAT gives a large running time values.