Analysis of Minimal Vertex Cover Algorithms

Neethu Mariya
Department of Management Sciences
University of Waterloo
nmariya@uwaterloo.ca

Simrat Saini
Department of Electrical and Computer Engineering
University of Waterloo
s52saini@uwaterloo.ca

December 7, 2020

Abstract

In this project, we have implemented three algorithms to solve the minimum vertex cover problem. The first algorithm CNF-SAT-VC is a polynomial time reduction to a CNF-SAT problem and the other two algorithms, namely APPROX-VC-1 and APPROX-VC-2, are approximate approaches. To analyse the algorithms' efficiency , we collected the results from each algorithm and compared their mean running time as well as the approximate ratio. In conclusion, based on the data collected, APPROX-VC-1 turned out to be the best algorithm to solve the vertex cover problem.

1 Introduction

The minimum vertex cover is an optimization problem and it can be solved in polynomial time. Hence it is an NP complete problem. A vertex cover of an undirected graph is basically a subset of its vertices that covers all the edges of the graph. As an example, the red vertices shown in the figure below are part of the vertex cover.

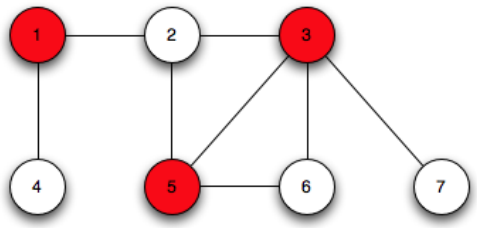

Figure 1: Minimum Vertex cover[1]

In the CNF-SAT-VC algorithm, we have reduced this to a SAT problem. We were provided with a SAT solver code that takes a propositional logic formula F in CNF form and produces an output True if the formula is satisfiable. Our task was to construct a formula F for the vertex cover using the following conditions for the clauses:

Assuming a graph $G=(V,E)$ with a vertex cover k where $k \in [1,V]$:

- At least one vertex is at position i where $i \in [1,k]$
- No vertex appears twice in the vertex cover
- Only one vertex appears at position i in the vertex cover
- For an edge (u,v) in G, at least u or v is in the vertex cover

The goal of the **CNF-SAT-VC** algorithm that we designed is to find the smallest k that makes the CNF formula F satisfiable i.e. to find the minimum vertex cover for the graph we entered. The algorithm always gives the correct output but the issue is it's efficiency which will be discussed in the later sections. The other two algorithms, **APPROX-VC-1** and **APPROX-VC-2** are approximate algorithms. As the name suggests, they give an approximate value of the minimum vertex cover that may not be always correct. APPROX-VC-1 picks the vertex with highest degree in each iteration until all edges have been visited. APPROX-VC-2 picks an edge (u,v) in random, adds both u and v to the vertex cover, discards edges connected to both of them and repeats the process till no more edges remain.

The paper is organized into four sections:Section 2 describes the implementation of code and test setup, Section 3 analyses the running time trends and Section 4 analyses the approximation ratio trends of the algorithms. In Section 4, we compare each algorithm based on the running time and approximate ratio and find out the optimum algorithm for the vertex cover problem.

2 Experimental Setup

For this project, we have taken two sets of data **S1** and **S2**. In the first set, running time data of each algorithm has been collected for 10 instances of graphs with vertices ranging from *[5,50]* with an interval of 5. In the second set, running time data and the approximation ratio of each algorithm has been collected for 10 instances of graphs with vertices *5,6,7,9,11,13,15,16*. The graphs have been generated using generator **graphGen** provided to us. The code **ece650-prj** that we designed implements multithreading: the main thread takes in I/O operations and the other three threads correspond to execution of the algorithms **CNF-SAT-VC**, **APPROX-VC-1** and **APPROX-VC-2**. We tested and collected our data by running our program remotely on **eceubuntu** linux machines.

3 Running Time Analysis

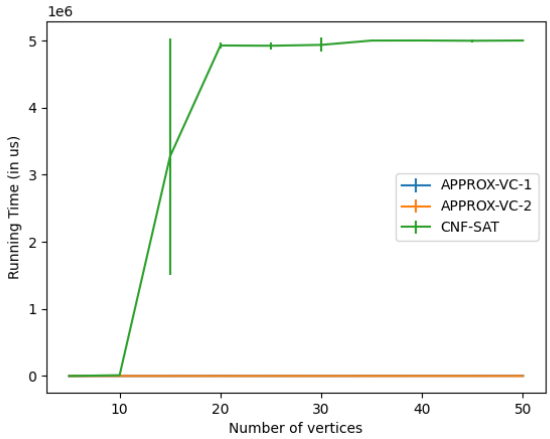

Figure 2: Running Time Analysis Set S1

The Figure 2 represents running time of the three algorithms for vertices *[5,10,....,45,50]*. There is a huge difference in the running time of CNF-SAT-VC algorithm and the approximation algorithms APPROX-VC-1 and APPROX-VC-2. The running time for CNF-SAT-VC shoots for *V>10*. In our code, while working on set **S1**, when the algorithm couldn't get the output in 5 seconds, it timeouts and saves the running time of 5 seconds, hence the straight line in the graph after *V>=20*. For graphs with small vertices, the CNF-SAT-VC algorithm is quick but the time increases exponentially as V gets higher. The main reason is the increase in number of clauses for the SAT solver to solve. Generally, for n vertices and a vertex cover of size k, the number of clauses is given by the formula:

$$\mathbf{k} + n\binom{k}{2} + k\binom{n}{2} + |\mathbf{E}| \tag{1}$$

Hence as the value of V increases, the number of clauses for vertex cover problem also increase and this results in increase in the running time. In order to properly see the trends in running time, set **S2** data is used.

In Figure 3, we can see an exponential rise in running time of CNF-SAT-VC plot as V increases, except at *V=15* where it dips and then again increases for *V=16*. One reason for this trend can be that the instances of graphs generated for *V 15* had well connected edges that allowed for the satisfiable condition for F to be found faster. Also, a huge deviation can be seen at *V=15* that can again be explained using the fact that some instances of graphs had well connected edges and some didn't. The former take less time than the latter to produce output. A significant standard deviation can also be seen at $V = 16$ and a minor one at $V = 13$.

Additionally, we can compare the running time trends of APPROX-VC-1 and APPROX-VC-2. Both of them compute the results in far less time than CNF-SAT-VC, especially as V increases. APPROX-VC-1 takes more time because the algorithm searches for vertex with highest degree whereas APPROX-VC-2 has no conditions and randomly picks up the edges to add to the vertex cover. Hence time complexity of former is more than the latter.

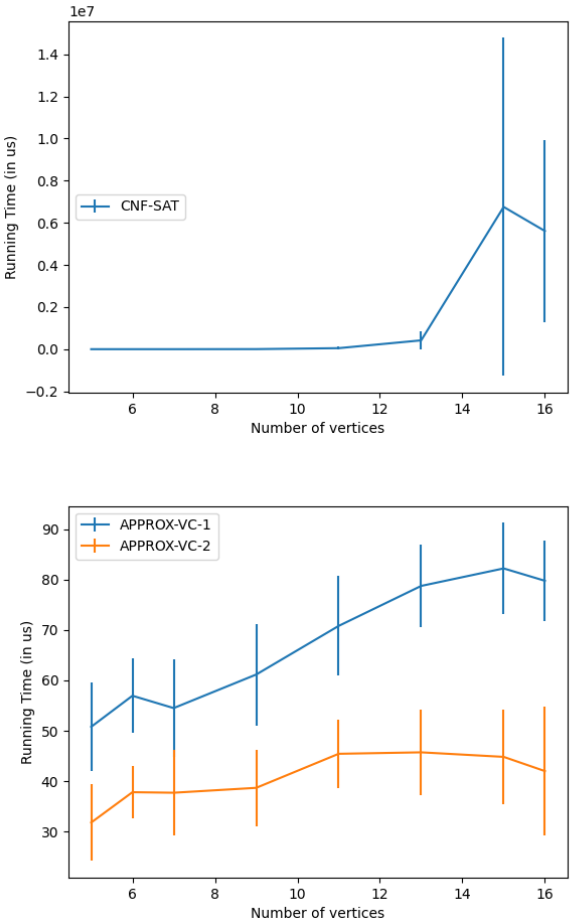

Figure 3: Running Time Analysis Set S2

4 Approximate Ratio Analysis

Approximate ratio is defined as the ratio of the size of computed vertex to the size of minimum sized vertex for the same graph. Hence:

$$\textbf{Approximate Ratio} = \frac{\textbf{size of the computed vertex cover}}{\textbf{size of optimal vertex cover}} \tag{2}$$

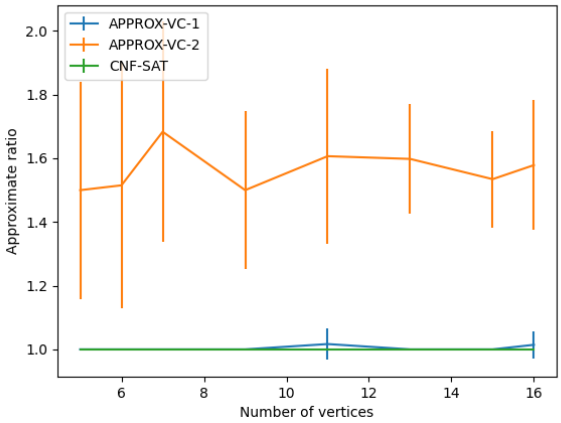

Figure 4: Approximate Ratio Analysis Set S2

We know that CNF-SAT-VC computes the minimum sized or the optimum sized vertex cover, though not necessarily unique. So we find the approximate ratio of each algorithm with respect to the CNF-SAT-VC output. The approximate ratio of CNF-SAT-VC is 1, hence closer the approximate ratio of the approximate algorithms to 1, more optimum the

result. The approximate ratio is inversely proportional to the correctness of the result i.e. greater the approximate ratio gives more inaccurate vertex cover.

We have used Set S2 to measure the trends in the approximate ratio for each algorithm. Set S1 is not used because for $V > 15$, we couldn't get the minimum vertex cover output from CNF-SAT-VC because it timed out. As we can see in Figure 4, the approximate ratio of APPROX-VC-1 and APPROX-VC-2 is always $\geq \mathbf{1}$. The approximate ratio of APPROX-VC-1 is close to the optimum value 1 because the algorithm has a condition for adding vertices to the vertex cover. The approximate ratio of APPROX-VC-2 is far from the optimum value and lies in the range $\mathbf{1.5 \leq x \leq 1.7}$. The reason for the inaccurate result is random picking up of edges with no condition to filter the results. Additionally from the graph we can analyse that the standard deviation for each vertex number is high in APPROX-VC-2. This is again owning to its random nature. For APPROX-VC-1, standard deviation is seen at $\mathbf{V = 13}$ and $\mathbf{V = 16}$. This can be because some outputs from the 10 different instances for each vertex number we recorded gave inaccurate results. However, the overall consensus is that the algorithm has greater chances of producing the optimum vertex cover.

5 Overall Comparison and Conclusion

Though **CNF-SAT-VC** produces correct results, it is not efficient because for higher vertices it takes too long to compute the result. In general, the time increases exponentially with the increasing number of vertices. We observed significant differences in the running time of CNF-VC-SAT and the approximate algorithms hence we have two bar graphs to give a comprehensive overview. As we can see from Figure 5, APPROX-VC-2 is the fastest with maximum time being **≈43 us** whereas CNF-SAT-VC takes about **≈ 7 s** at $\mathbf{V = 15}$ which is a huge difference. APPROX-VC-1 takes about **≈ 80 s** which is decent in comparison to CNF-SAT-VC run time.

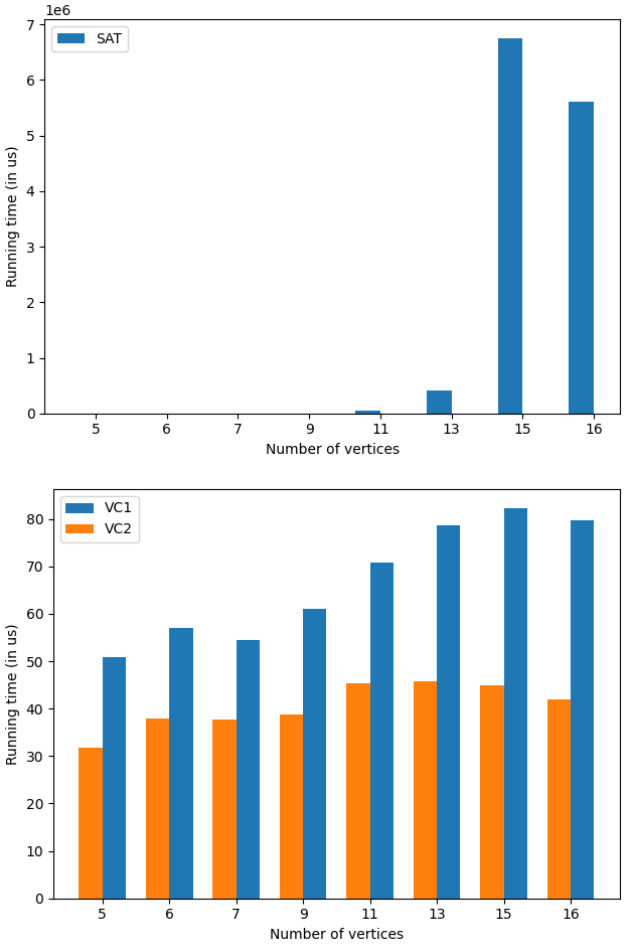

Figure 5: Running Time bar graphs

In terms of approximate ratio analysis, we have seen in the previous section that CNF-SAT-VC always gives the correct result. In addition, APPROX-VC-1 produces a vertex cover that is quite close or equal to the optimum vertex cover. Figure 6 gives an overview of our findings.

Hence we can come to the decision that keeping running time and approximate ratio readings in perspective, **APPROX-VC-1** is the most efficient algorithm of all the three that we designed. It takes an optimum running time to compute a vertex cover and the output is quite close or equal to the correct vertex cover.

To sum up our findings from set S2 :

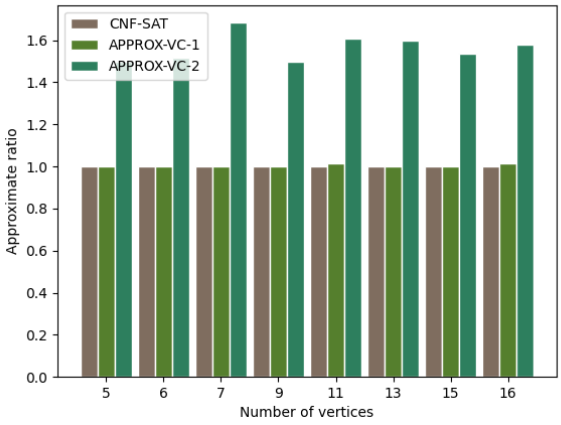

Figure 6: Approximate Ratio Bar Graph

- CNF-SAT-VC always produces the correct result i.e. a minimum sized vertex cover,but the issue with it is that as the number of vertices ($V > 15$) increases, the running time shoots up. Hence it is only effective as long as the number of vertices in a graph are small.
- APPROX-VC-1 produces result that has approximate ratio 1 for graphs with lesser number of vertices ($V < 11$ and shows a small deviation for higher number of vertices but the ratio stays close to 1. The running time increases as the number of vertices increase, but the increases is not as high as that in CNF-SAT-VC.
- APPROX-VC-2 performs well in terms of running time, also showing a decrease with increase in vertex number ($V > 13$). But,its performance is bad in terms of giving a correct output and shows a high deviation from the optimum result.

In conclusion, we have analysed all the three different algorithms we designed to compute a minimum vertex cover of a given graph and presented our findings based on their running time and approximate ratios.

References

[1] *CS 360: Lecture 29: Approximation Algorithms*. 2015. URL: `http://ycpcs.github.io/cs360-spring2015/lectures/lecture29.html`.

[2] P. A. Gurfinkel. *ECE650, Fall 2020, Final Course Project*. Fall,2020.

[3] P. A. Gurfinkel. *ECE650, Fall 2020, Section 01,A Polynomial-Time Reduction from Vertex-Cover to CNF-SAT*. Fall,2020.