UNIVERSITY OF WATERLOO
FACULTY OF ENGINEERING
Department of Electrical &
Computer Engineering

University of Waterloo
Department of Electrical and Computer Engineering
ECE 650

Final Course Project

Methods and Tools for Software Engineering

Nisarg Parikh & Jimisha Thakkar
20842540 20853543

Instructor:
Prof. Arie Gurfinkel

4th December 2019

Table of Contents

1 Introduction **1**

2 Methodology **2**

3 Analysis **3**

4 Conclusion **5**

1. Introduction

The project solves the Vertex Cover problem, a particular kind of optimization problem. It can help the local police department with their installation of security cameras at intersections. The idea is for the police to use/install minimal number of cameras and still effectively monitor the maximum streets possible. In this report, comparative analysis is performed on the three different algorithms to solve the vertex cover problem and then their efficiency is analysed.

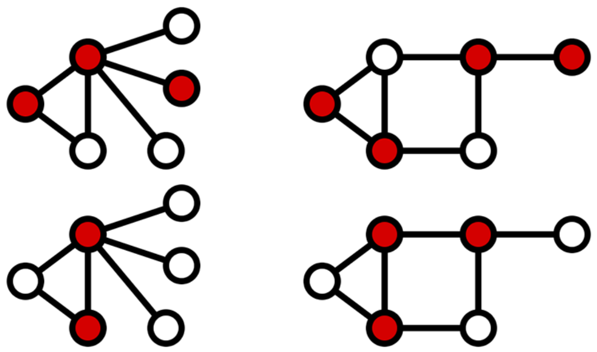

Figure 1.1: Simple Vertex Cover Problem
Image Source: http://isaacsteele.com/cv/edu/college/junior/vertexcover.shtml

The above figure demonstrates the graphs showing covered vertices which are the one's painted in red colour. The basic concept of a valid vertex cover is given every edge, at least one vertex is in the "covered" set. Minimum vertex cover requires the set of covered vertices to be as small as possible. Only the graphs along the bottom of the Figure 1.1 fulfill minimum vertex cover. Minimum vertex cover is considered to be an NP-Complete problem, and the best exact algorithm as it runs at $\Theta(2^n)$.

We quantitatively analyze our software for various kinds of inputs. The input graphs which have been used were generated by a graph generator. In order to measure the performance, the graphs were generated with an increasing number of vertices, as follows: 1) 10 Graphs with 5 vertices - Run each graph for 10 times 2) 10 Graphs with 10 vertices - Run each graph for 10 times 3) 10 Graphs with 15 vertices - Run each graph for 10 times

Detailed description of the various algorithms for solving Vertex Cover Problem can be summarised as follows: i)APPROX-VC-1: Pick a vertex of highest degree (most incident edges). Add it to vertex cover and discard all the edges incident on that vertex. Repeat till no edges remain. ii)APPROX-VC-2: Pick an edge $\langle u, v\rangle$ and add both u and v to your vertex cover. Throw away all edges attached to u and v. Repeat till no edges remain. iii)CNF-SAT-VC: This method is a reduction which consists of four main clauses upon which it mainly functions. 1)Atleast one vertex is the i^{th} vertex of the vertex cover 2)Same vertex cannot appear twice in the vertex cover 3)No more than one vertex appears in the m^{th} position of the vertex cover 4)Every edge is incident to at least one vertex in the vertex cover.

2. Methodology

The project and the report manifests the comparison between three different algorithms to solve the vertex cover problem on the basis of their respective run-time and their approximation ratio :

To satisfy the aforementioned purpose, we have built the project that solves the vertex cover problem using three algorithms by following the methodology below:

Firstly, the C++ programs namely APPROX-VC and CNF-SAT-VC and their respective header files had been created.

Secondly, the main.cpp file containing the main execution code of the program was created.

Thirdly, the python scripts were prepared for generating the inputs that uses GraphGen to produce the output '.txt' files for the various required inputs for number of vertices following an arithmetic progression of 5, 10,15...1500.

Finally, we analysed the data by using an analysis script using python and some input python files were used to get the required outputs of the Vertex Covers.

We successfully ran all the files and received the exact outputs for each set of input of the vertices provided in all the 3 algorithms.

On the later stage, the figures extracted after running the algorithms for number of times were entered in the excel sheets and graphs were generated which described interesting facts.

3. Analysis

The analysis framed out of the project clearly depicted results about the efficiency of the approaches based on the various inputs they were tested upon. Each algorithm's execution time was benchmarked taking the clock_gettime function as a reference. Each thread was timed based upon its cpuclockid() obtained by the function pthread getcpuclockid(). Having taken 5,10 and 15 vertices; 10 graphs each had been generated and were executed for 10 times.

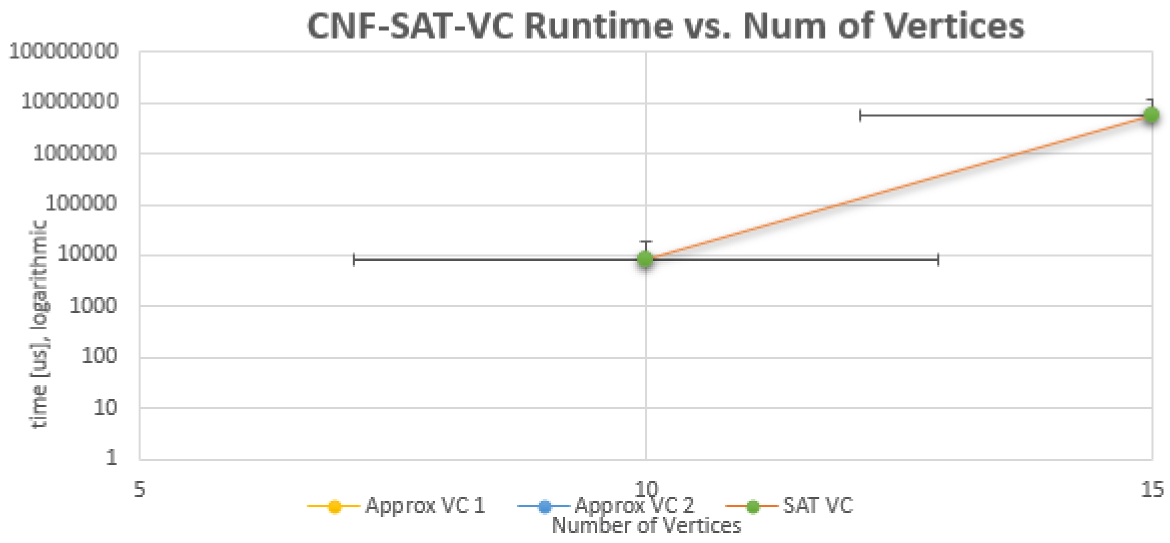

Figure 3.1: Graph on CNF-SAT-VC runtime VS Number of vertices

i) COMPARISON BASED ON RUN TIME: It can inferred from the figure 3.1, that the CNF-SAT-VC, takes more time to generate the graph with larger vertices like 10, 15 and greater than that.

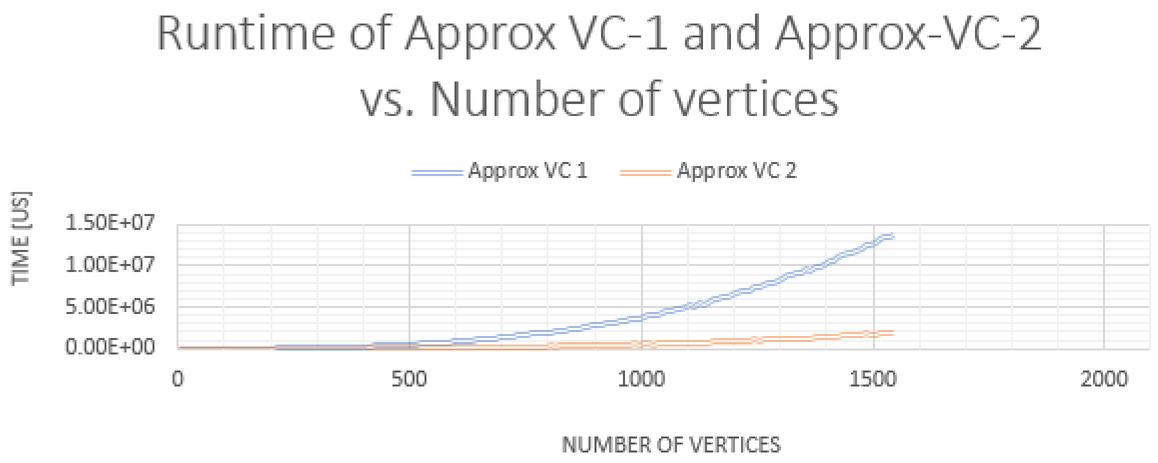

Figure 3.2: Graph on APPROX-VC-1,APPROX-VC-2 runtime VS Number of vertices

It can be observed from the figure 3.2 that for the graphs upto 15 Vertices; both the methods APPROX-VC-1 and APPROX-VC-2 can be worked out in less than one microsecond. In order to properly plot this trend, a Y-axis logarithmic data was considered which results in the error bars distorting. Therefore, a graph till 1500 vertices are taken as reference to view the proper growth of the curve which turns out to be exponential in nature. Both the algorithms showed similar run-time up to vertex number around 250, which can be considered as a breakdown point or split-away point. Surpassing this point, they dramatically differed in terms of performance due to the run-time complexities. Also, the APPROX-VC-1 algorithm computes its result in around 25 seconds while the APPROX-VC-2 takes less than 7 seconds to execute, which manifests that APPROX-VC-2 was more efficient for this case.

ii) COMPARISON BASED ON APPROXIMATION RATIOS: Approximation ratio is the ratio

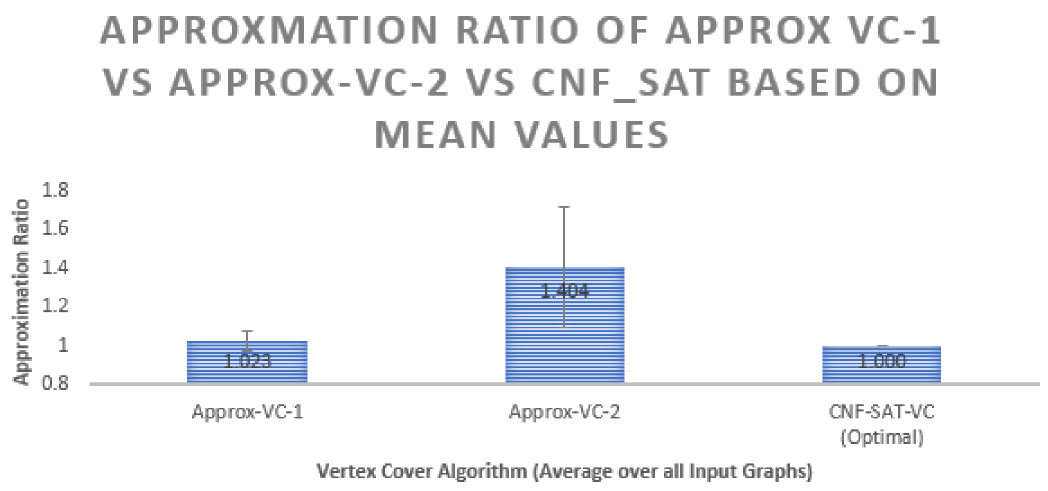

Figure 3.3: Graph on Approximation ratio VS APPROX-VC-1,APPROX-VC-2 and CNF-SAT-VC based on Mean

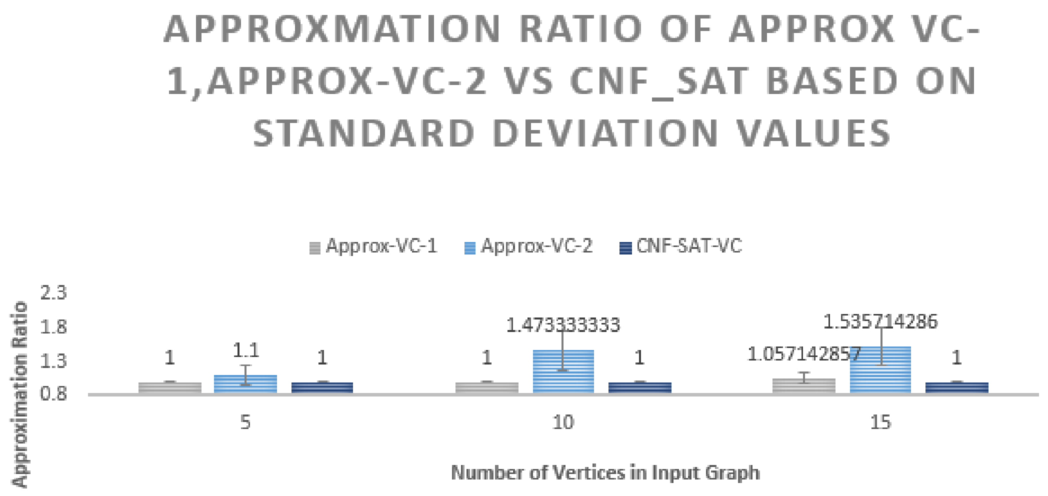

Figure 3.4: Graph on Approximation ratio VS APPROX-VC-1,APPROX-VC-2 and CNF-SAT-VC based on Standard Deviation

of the size of the computed vertex cover to the minimum sized vertex cover. The optimal vertex cover is achieved by the CNF-SAT-VC algorithm and the results of the performance are visualized in the bar charts above. The above figure describes the mean approximation ratios over the input graphs. APPROX-VC-1 is matching the optimal solution and is good performer whereas the approximation ratio of APPROX-VC-2 was 40% higher than the CNF-SAT vertex cover.

4. Conclusion

After completing a detailed study on the graphs of various methods it can be inferred that all the three algorithms are useful to solve the vertex cover problems but choices has to be made based on the necessity of either speed or accuracy.

As per the run-time analysis, APPROX-VC-2 provides speedy output and is better for higher number of vertices but is incapable to provide minimum vertex cover. So, it is can be used to obtain vertex cover in minimum time but not the minimum vertex cover.

On the other hand, the CNF-SAT-VC provides the minimum vertex cover but is restricted to only small number of vertices.

In terms of approximation ratio the APPROX-VC-1 provides a solution which matches more with the optimal solution i.e, CNF-SAT-VC, which is taken as benchmark as it definitely provides the minimum vertex cover.