ECE650 - METHODS AND TOOLS FOR SOFTWARE ENGINEERING

UNIVERSITY OF WATERLOO

FALL 2019

Efficiency Analysis of Vertex Cover Algorithms

Authors:
Sai Teja G (ID: 20825009)
Somesh kumar Gupta (ID: 20817245)

Date: December 1, 2019

1 Introduction

Vertex Cover or Node Cover of a graph is a set of vertices in such a way that each edge of the graph is incident to at least one vertex of the set. In computer science, finding minimum vertex cover of a graph is a classical optimization problem; it has an approximation algorithm and fits into the definition of NP-Complete problem in computational complexity theory.

In the first algorithm, we use Conjunctive Normal Form Satisfiability (CNF-SAT-VC) along with Minisat Solver. This algorithm presents a polynomial-time reduction from VERTEX-COVER to CNF-SAT. The minimum vertex cover is obtained once the SAT Solver return the result for the given clauses in the CNF form.

In second algorithm, Approximate Vertex Cover-1 (APPROX-VC-1) uses Greedy Algorithm. The Greedy Algorithm is an algorithmic paradigm which solves the Vertex Cover problem by making a locally optimal choice at each stage hoping to find a global optimum solution.

The last algorithm is Approximate Vertex Cover-2 (APPROX-VC-2). In approximate algorithm we pick an edge $u, v \in E$. We then pick all Edges that are incident to either u or v and remove those Edges from our list of Edges. This procedure is continued till the list of Edges becomes empty, we store picked u and v to our list and when the list of Edges becomes empty we return our list of u and v for minimum vertex cover for this algorithm.

Our main aim is to analyse the three algorithms for efficiency and then predict the optimum algorithm that is efficient in solving the Vertex Cover Problem for any given number of Vertices. The efficiency is measured as the minimum time taken by the algorithm to generate the Vertex Cover given the constraint that it generates Vertex Cover of minimum size (k is minimum).

2 Analysis

The analysis of efficiency for all the three algorithms were done on the basis of two factors the first is the Running Time (which is the CPU clock time for each algorithm) and the second is the Approximation Ratio (which is the ratio of the size of computed vertex cover to the size of optimal/minimum vertex cover). Here, CNF-SAT output is considered as an optimal approach than the other outputs.

The hypothesis of the program is based on the concept of multi-threading. In this we have implemented of four threads, namely I/O thread (for input), and three more threads (one from each of the vertex cover algorithm). The functionality of I/O thread is for distributing input to the other threads. The running time is computed at the end of each thread, pthread_getcpuclockid() function was used for getting the CPU time of each of the algorithm function.

2.1 Running Time Analysis of CNF-SAT-VC

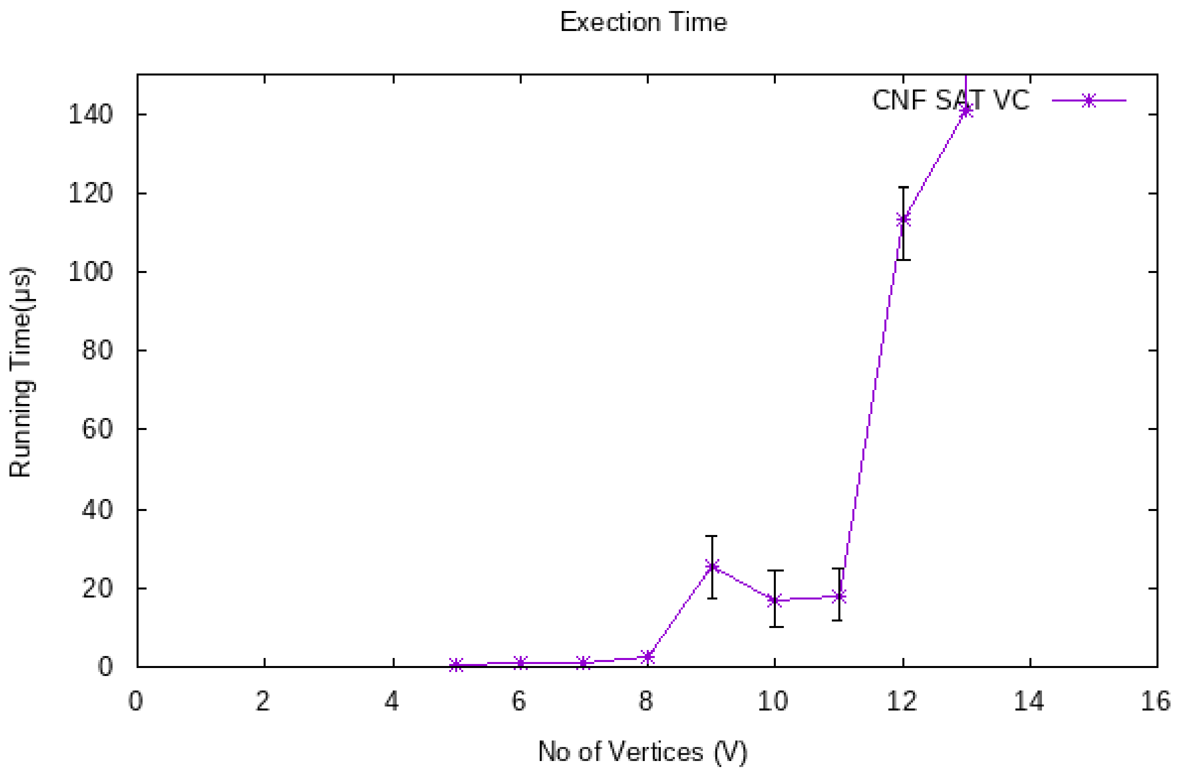

Figure 1: Running Time for CNF-SAT-VC

The graph shows the running time of CNF-SAT, X axis is the number of vertices, Y axis is the running time (μs), We have taken readings from 5 vertices to 15 vertices with the increment of 1. Since, CNF-SAT takes a lot of time to calculate vertex cover for higher vertices; therefore, we have bounded our analysis up to 15 vertices.

The running time for at least 10 runs of each vertex has been measured. Then, the mean and standard deviation across those 100 runs for each value of V were computed. There has been small standard deviation for lower number of vertices too. And this standard deviation gets amplified when the number of vertex increases for graph.

This approach is real and time consuming, and as the number of Vertices increase the running time is increasing exponentially. Due to the above functionality, this approach is a polynomial-time algorithm.

2.2 Running Time Analysis of APPROX-VC-1 and APPROX-VC-2

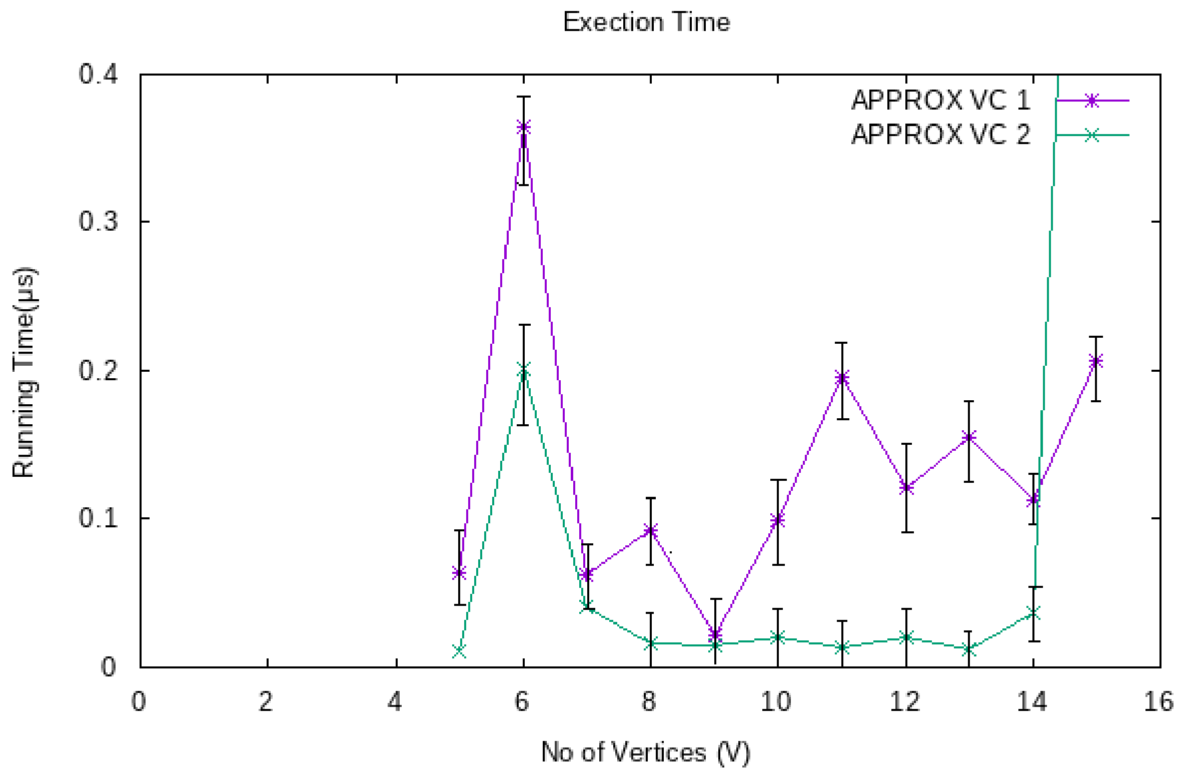

Figure 2: Running Time for APPROX-VC-1 and APPROX-VC-2

The graph shows the running time of APPROX-VC-1 and APPROX-VC-2 algorithms, X axis is the number of vertices, Y axis is the running time (us), The readings are taken in the intervals of 1 from 5 to 16 vertices.. Now from comparing the graph of CNF-SAT with the above graph, we can observe that CNF-SAT takes more time in comparison to APPROX-VC-1 and APPROX-VC-2.

However CNF-SAT follows an exponential increase in running time with respect to increase in the number of vertices, which is not same for Approximate Algorithms. As the above two Algorithms significantly take less running time, this is because this approach is a linear algorithm.

The number of clauses and hence time taken by SAT solver increases exponentially with the increase in vertices, it is given by the same aforementioned formula (i.e.

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

While in the case of other two algorithms, they have less time complexity as compared to CNF-SAT. Hence, they will never show exponential rise in their running time as oppose to CNF-SAT.

2.3 Analysis of Approximation Ratio

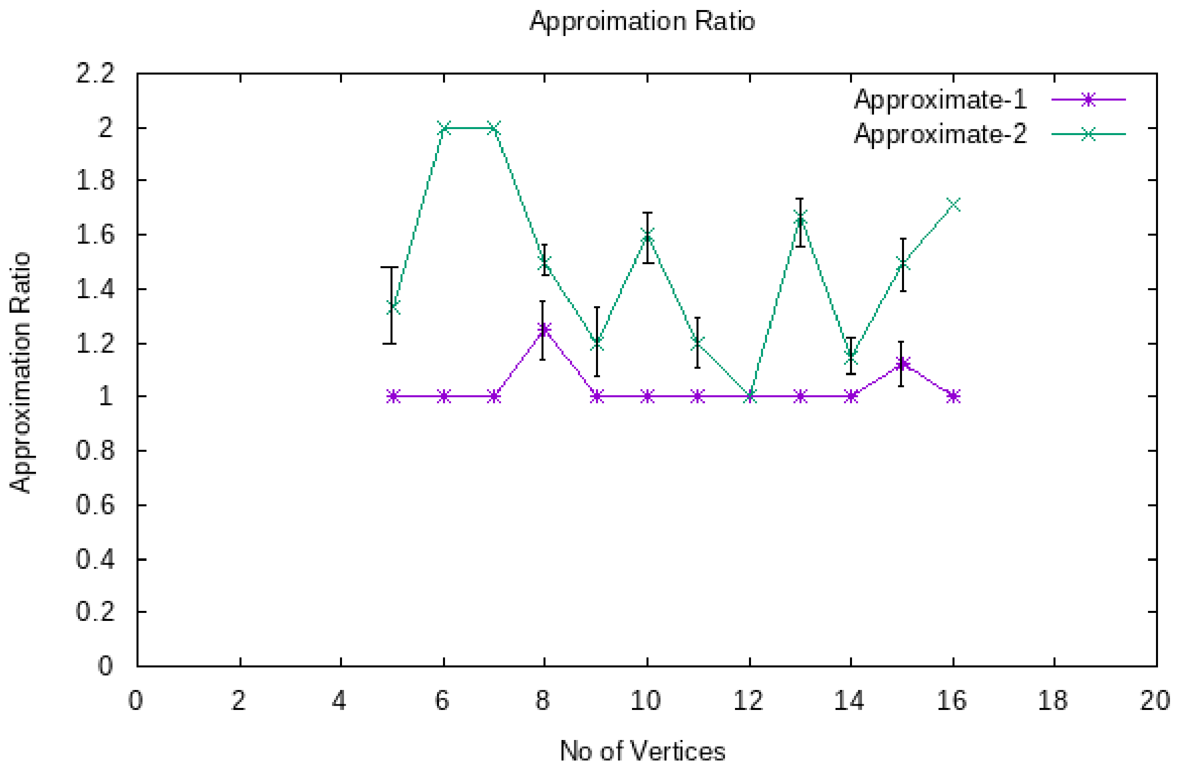

Figure 3: Running Time for APPROX-VC-1 and APPROX-VC-2

Figure 3. shows the approximation ratio of APPROX-1 and APPROX-2 over the output of CNF-SAT approach. Y axis is the approximation ratio, X axis is the number of vertices, each value for V each value was generate at least 10 graphs, compute the time and approximation ratio for each such graph.

As we infer from graph the approximation ratio for most vertices (lesser than 14) is 1 for APPROX-1 and this ratio is quite high for APPROX-2. So in Approx-1 algorithm the time it gives us the minimum vertex cover specifically for lesser number of vertices which was equivalent to the output of CNF-SAT. On the other hand, it a slight increase in approximation ratio for larger vertices but not as much as observed for Approx-2.

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

3 Conclusion

From the above data, we can conclude that the algorithm APPROX-VC-1 is a better approach to solve large scale vertex cover problem as compared to APPROX-VC-2 and CNF-SAT as APPROX-VC-1 takes less running time compared to APPROX-VC-2 and CNF-SAT-VC.

CNF-SAT-VC always gives optimal solution as compared to the APPROX-VC-1 and APPROX-VC-2 but the drawback of this algorithm is that the time for computation of Vertex Cover increases exponentially as the size of the input increases.In addition for larger inputs the algorithm APPROX-VC-1 is more efficient among all the three algorithms.