Analysis of Three Algorithms for Solving MIN-VERTEXCOVER Problem

Navpreet Kaur
n42kaur@uwaterloo.ca

Anmol Jawa
a2jawa@uwaterloo.ca

Abstract

A vertex cover of a graph is a set of vertices such that each edge of the graph is incident to at least 1 vertex in this set, and a minimum vertex cover is a vertex cover of smallest set size .The problem of finding a minimum vertex cover is a classical optimization problem in computer science and is a typical example of a NP-hard optimization problem that has an approximation algorithm. In this report, we calculate the Vertex Cover in the graph using the approximation approaches and the CNF-SAT algorithm. The result shows the most effective way to solve minimum vertex cover.

1 Introduction

The aim of this project was to compare the performance of three different algorithms which solve the minimum-vertex-cover problem. In this, A vertex cover (C) of a graph G=(V,E) is a subset of vertices V such that each edge in E is incident to at least one vertex in C. It takes an un-directed graph G = (V, E), and an integer k (k belongs to the range 0 to mod V) as input and gives True, if G has a vertex cover of size k, false otherwise as the output.
In this program, multi threading paradigm is used in such a way that all 3 algorithms run in parallel in different thread. This has been done to utilize CPU time.

2 Methodology

There are three methods, as following :

1. CNF-SAT-VC: The approach uses a polynomial time reduction to CNF-SAT, and then use a SAT solver to get the final results. In this, model values provided by the SAT solver are considered as solution to the vertex cover i.e. True, if F is satisfiable, false otherwise. It takes as input G, k and produces a formula F (a propositional logic formula) in Conjunctive Normal Form (CNF) with the property that G has a vertex cover of size k if and only if F is satisfiable. In this a polynomial-time reduction from VERTEX-COVER to CNF-SAT is presented.

2. APPROX-VC-1: The approach is to pick a vertex of highest degree (most incident edges) and add it to "vertex cover" container then throw away all edges incident on that vertex and repeat those steps till no edges remain.

3. APPROX-VC-2: The approach is to pick an edge (u, v) and add both u and v to your vertex cover and throw away all edges attached to u and v and repeat till no edges remain

These algorithms have been compared by analyzing both their respective runtime and their approximation ratio (size of min-vertices-set vs. size of optimal min-vertices-set) for a range of varied graphs with different sizes .

3 Analysis

3.1 Runtime Analysis

Each algorithm's execution time was bench-marked by using the `clock_getTime` function. Each thread in our C++ program for analysis was timed using its respective cpu clock id obtained by the function found in the pthread library. The performance results of the algorithms can be seen in figures below .

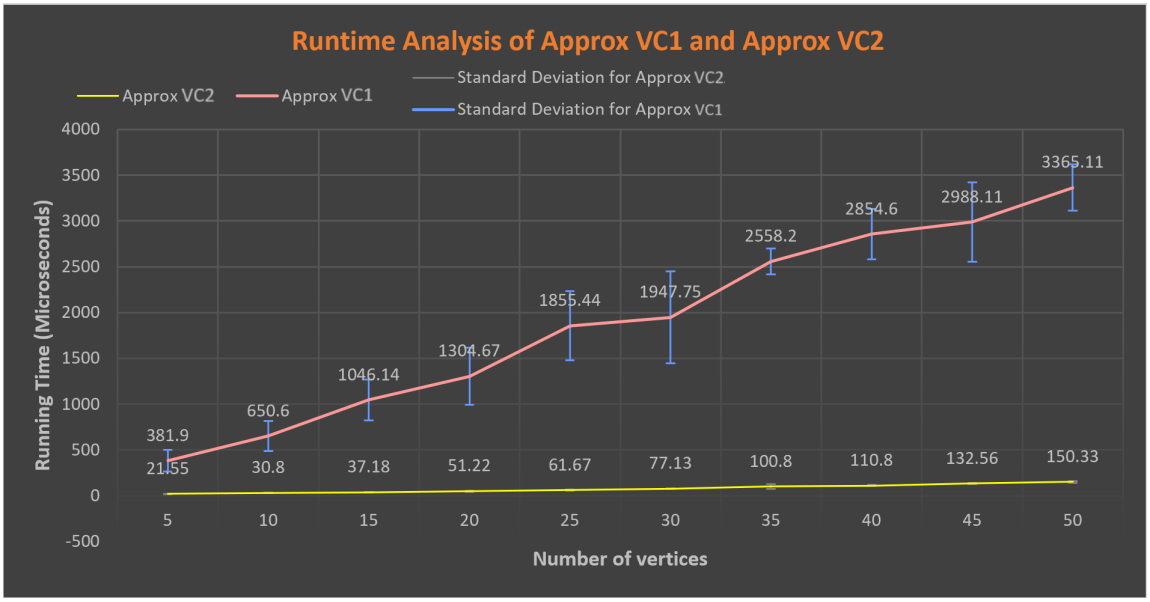

Figure 1: Runtime analysis for two approaches, Approx VC1 and Approx VC2 against graph size [5,50]

The analysis as portrayed in Figure 1, reveals that performances of both the approaches is similar with a negligible time difference, for smaller number of vertices. It can then be seen that the two algorithms diverge due to the run time complexities of the algorithms themselves, as the input graph size increases. At an input of graph with input size 50, the Approx-VC-2 took considerable lesser time than Approx-VC-1 by approximately 3000 micro seconds.

The CNF-SAT approach's analysis can be separately seen in Figure 2. This approach took substantially more time than the prior mentioned approaches for graph vertices more than 9. As the input graph size increased, the time taken increased non-linearly. With time at vertices 15 being approximately double than the time at vertices 13, it can be said that the graph showed exponential growth. The analysis were done till vertices = 16 as the approach took more than the set timeout time of 15 seconds, to come up with a solution. The analysis for CNF-SAT-VC were done separately as the algorithm timed-out after a

certain number of vertices and also had a different logarithmic scale than the Approx-VC-1 and Approx-VC-2 approaches.

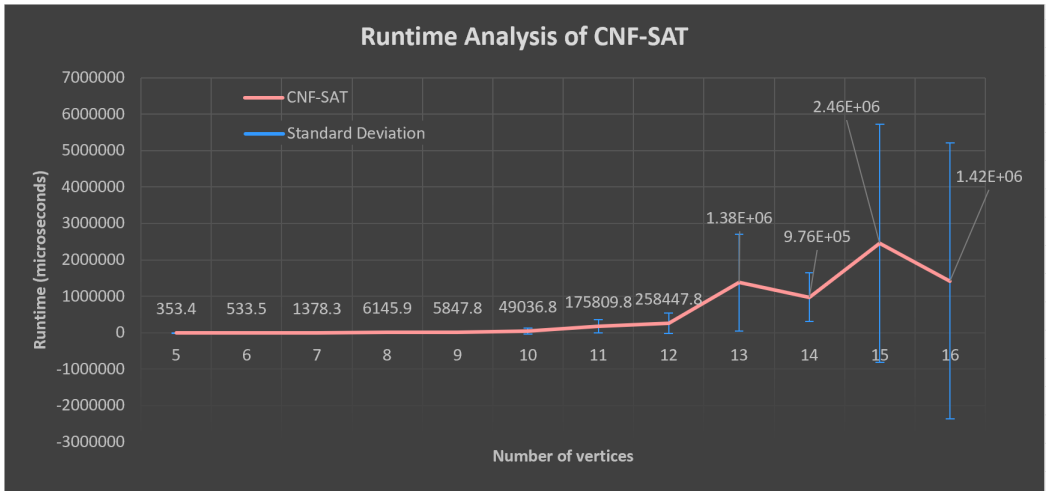

Figure 2: Runtime analysis for approach CNF-SAT against graph size [5,16]

In Figure 3, the analysis with Approx-VC-1 and Approx-VC-2 were re-done to portray a direct comparison with the CNF-SAT-VC approach, for the same range of vertices but with a different time-scale on y-axis. It can yet again be clearly seen that with run-time being the only parameter to judge, the Approx-VC-2 comes up to be the best approach and CNF-SAT-VC being the worst.

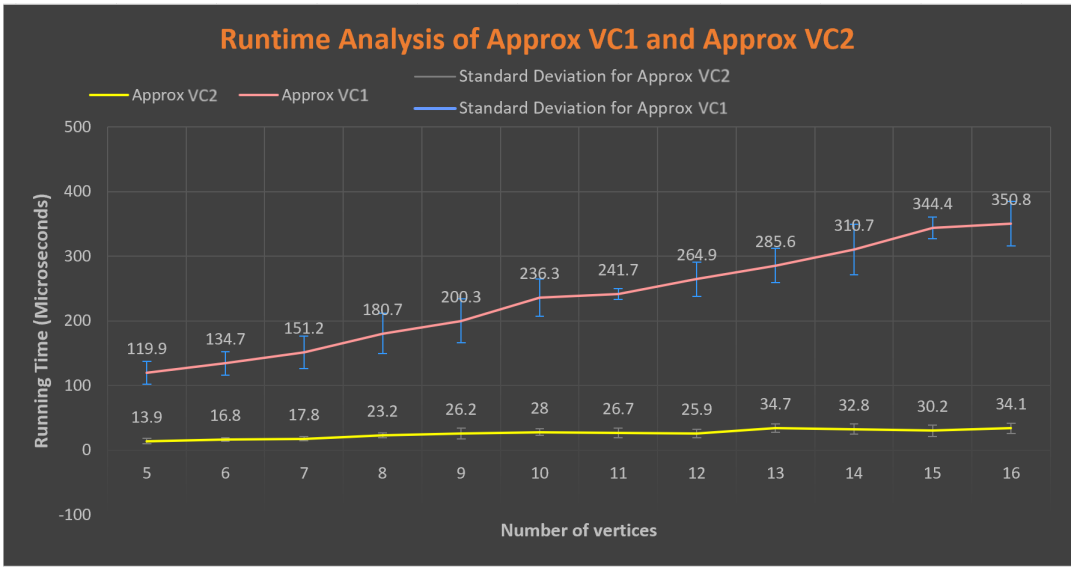

Figure 3: Runtime analysis for two approaches, Approx VC1 and Approx VC2 against graph size [5,16]

3.2 Approximation Ratio Analysis

In this part of our analysis we regarded the result of CNF-SAT-VC approach as the minimum sized vertex cover. The values seen in Figure 4 are the mean approximation rations

for input graphs in range [5,16], the range has been taken till graph size 16 as the CNF-SAT-VC algorithm timed-out for bigger graph sizes. The APPROX-VC-1 algorithm is highly accurate in returning a vertex cover near to the optimal value obtained by the sat solver as the approximation ratio for this approach remained close to 1 throughout the range of graph vertices. The APPROX-VC-2 algorithm did not perform well at reaching the optimal solution throughout the graph size range, in comparison with Approx-VC-1 approach. For APPROX-VC-2, the least approximation ratio across the range of graph vertices was `39%` larger than the optimal solution, which is a clear indicative of Approx-VC-2 being less reliable than Approx-VC-1, in terms of getting an optimal solution. Approx-VC-2 approach's standard deviation can be seen in the error bars which show the results fluctuate greatly between different inputs, reason being that this approach hugely relies on randomness of selection which leads to different approximation ratio for the same input graph.
From the above approximation analysis, it is evident that APPROX-VC-1 clearly produces more accurate and optimal results than APPROX-VC-2.

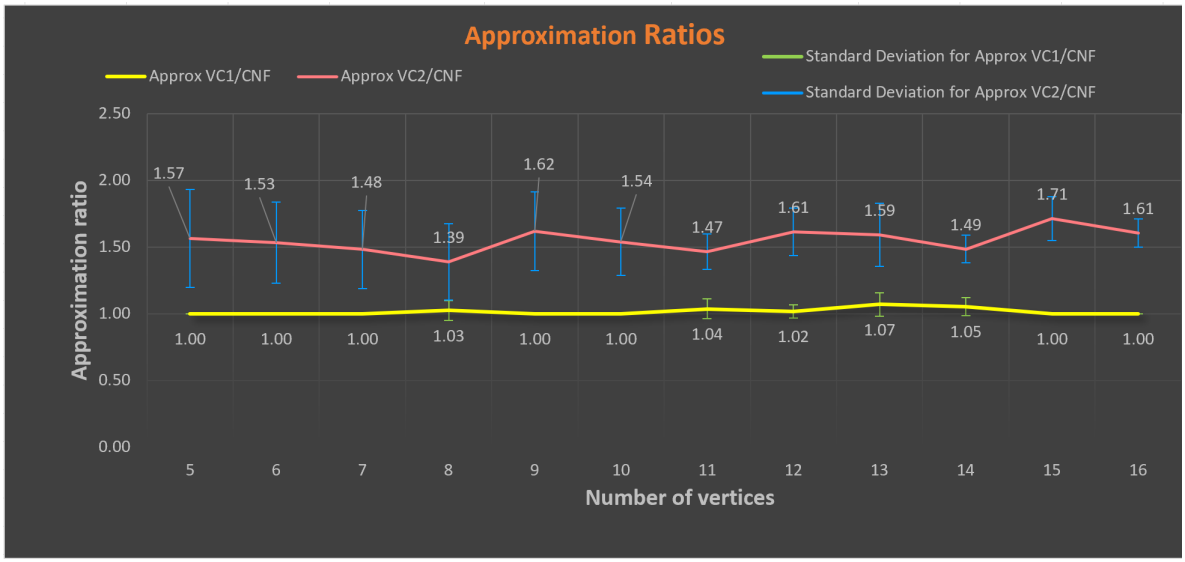

Figure 4: Approximation ratios of Approx VC 1/CNF and Approx VC 2/CNF

4 Conclusion

After analysing all the methods mentioned above, it is clear that CNF-SAT-VC approach gives the most optimal solution if the purpose is to find minimal vertex cover whilst APPROX-VC-2 is least preferred as it is least likely to give minimum vertex cover. However, CNF-SAT-VC is not suitable for solving vertex cover for large size graph since the time complexity increases dramatically with increase in number of vertices. Hence in such scenarios wherein the input graph size is huge, APPROX-VC-1 is a good option as it produces near optimal solution in substantially lesser time than CNF-SAT-VC. Our analysis also found that APPROX-VC-2 is the quickest approach amongst the three approaches and produces a solution with a minimum approximation ration of 1.39 .

References

1. Project problem statement
2. Assignment 4 problem statement
3. Github (https://github.com/agurfinkel/minisat)