Report

Analyzing Performance of SAT Solver for Vertex Cover Problem

ECE 650 Project

Group Members

Prateeti Deb Chaudhuri	pdebchau@uwaterloo.ca
Priya Patel	pb4patel@uwaterloo.ca

1. INTRODUCTION

In the mathematical discipline of graph theory, a vertex cover of a graph is a set of vertices such that each edge of the graph is incident to at least one vertex of the set. A minimum vertex cover is a vertex cover of smallest possible size. This problem of finding the minimum vertex cover is a classical optimization problem in computer science. It is a typical example of NP-hard optimization problem that has an approximation algorithm.

This report shows the analysis of finding Vertex Cover for a graph using different approaches.
In this report we are finding the minimum vertex cover by three techniques –

I. **CNF-SAT Solver**
Creating a polynomial time reduction of the decision vertex of Vertex Cover to CNF-SAT.
II. **APPROX-VC-1**
An algorithm called APPROX-VC-1, which is picking a vertex of highest degree(most incident edges), then adding this to the vertex cover, and then throwing away all the incident edges on that vertex till no edges remain
III. **APPROX-VC-2**
An algorithm called APPROX-VC-2, which is picking a random edge {u,v} and adding both u and v to the vertex cover. Then throwing away all the edges attached to u and v. This process will be repeated till no edge remains.

We are using multi-threaded approach to find these approximations of vertex cover in our program, where one thread is for I/O and one each for the different approaches to solve the minimum vertex cover. One thread is for SAT Solver to solve clauses generated by polynomial reduction. In other threads the approximation algorithms are run.

2. ANALYSIS

Analysis of the efficiency of these three algorithms is done according to the following two factors:-

I. Running Time
II. Approximation Ratio

2.1 Minisat

Minisat is a minimalistic, open source SAT Solver, developed to help both developers and researchers to get started on SAT. It is a SAT Solver, which can determine if an expression of Boolean variables can be true or not, provided the expression is written with AND, OR, NOT, parenthesis and Boolean variables.

2.2 Running Time

We are first comparing running time of the three algorithms. Since there is a huge difference between the running times of CNF-SAT and the other two algorithms, we have separated these into two graphs. Running time of all the three algorithms increases with the increase in number of vertices. This can be because of the reason that CPU has to process more data for more number of vertices and hence takes more time.

It can be found out that SAT Solver took the maximum time to run among all the three approaches. This is illustrated by generating the below graphs for SAT Solver and for the other two approaches.

We are calculating average running time of the graph with vertices in the range of 2 to 16. We have used matplotlib of Python to generate the graphs here. We are using graphGen to produce different graph inputs with different combinations of vertices and edges.

From Fig. 1 it can be seen that SAT Solver performs very fast for the lesser number of vertices (2 to 10 approximately). When the number of vertices is increasing, it is taking more time to find the vertex cover. It is expected that the exponential nature of the graph after vertex 13, will continue if we increase the number of vertices because if the number of vertices increases it will increase the number of clauses automatically. If there are v vertices then there will be
k*v*(v-1)/2 clauses where k is the size of the vertex cover. And since the edges will also increase it has to satisfy more clauses. In case of smaller number of vertices, it has to satisfy less number of clauses. Hence it takes less time to run as it has to satisfy a lesser combination of variables. So, the exponential growth of SAT Solver is justified here.

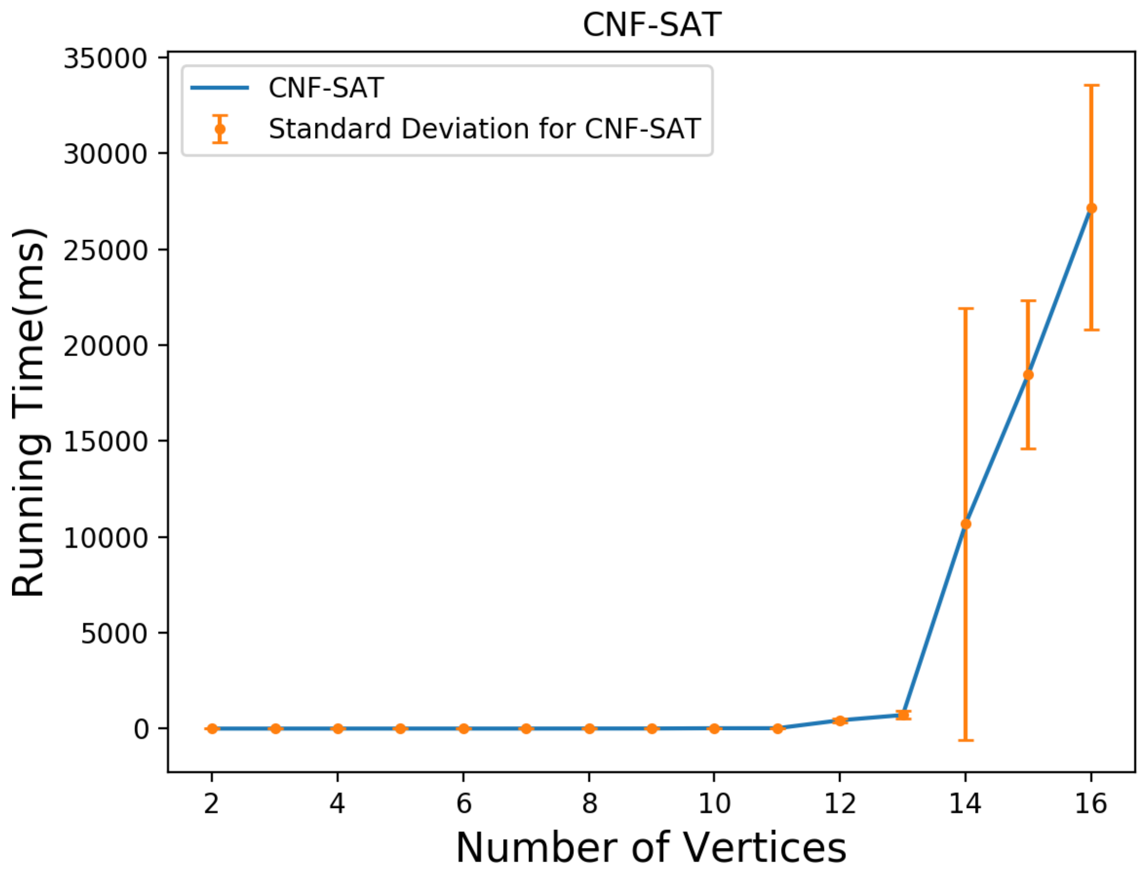

Figure 1 Running time Analysis for CNF-SAT Algorithm

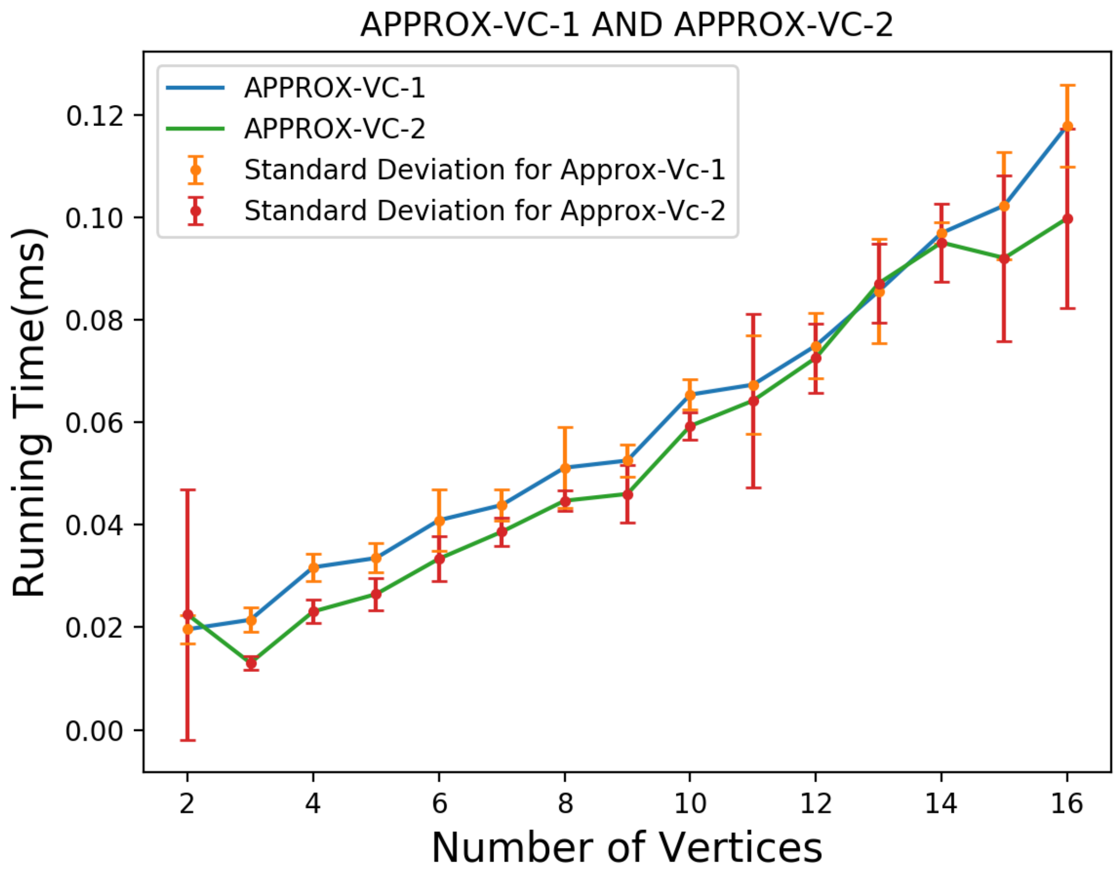

Figure 2 Running time Analysis for Approx-VC 1 and Approx-VC 2 Algorithms

However, time complexity of SAT Solver is Θ(n^4), which is much bigger than the other two algorithms. Also, since SAT is NP problem, it is not possible to get the result in polynomial time.

Running time of APPROX-VC-1 is a little higher than the running time of APPROX-VC-2. It seems that the cost of finding the highest degree vertex is higher than compared to picking only an edge without restriction. So, the complexity of APPROX-VC-1 is higher.

It can be seen from the graphs that the fluctuation in APPROX-VC-1 is much bigger than the fluctuations in APPROX-VC-2. This can be because APPROX-VC-1 is a greedy algorithm which is finding a highest degree vertex.

2.3 Approximation Ratio

Approximation Ratio is calculated according to the below formula:
Approximation ratio: (size of the computed vertex cover)/ (size of the minimum vertex cover)

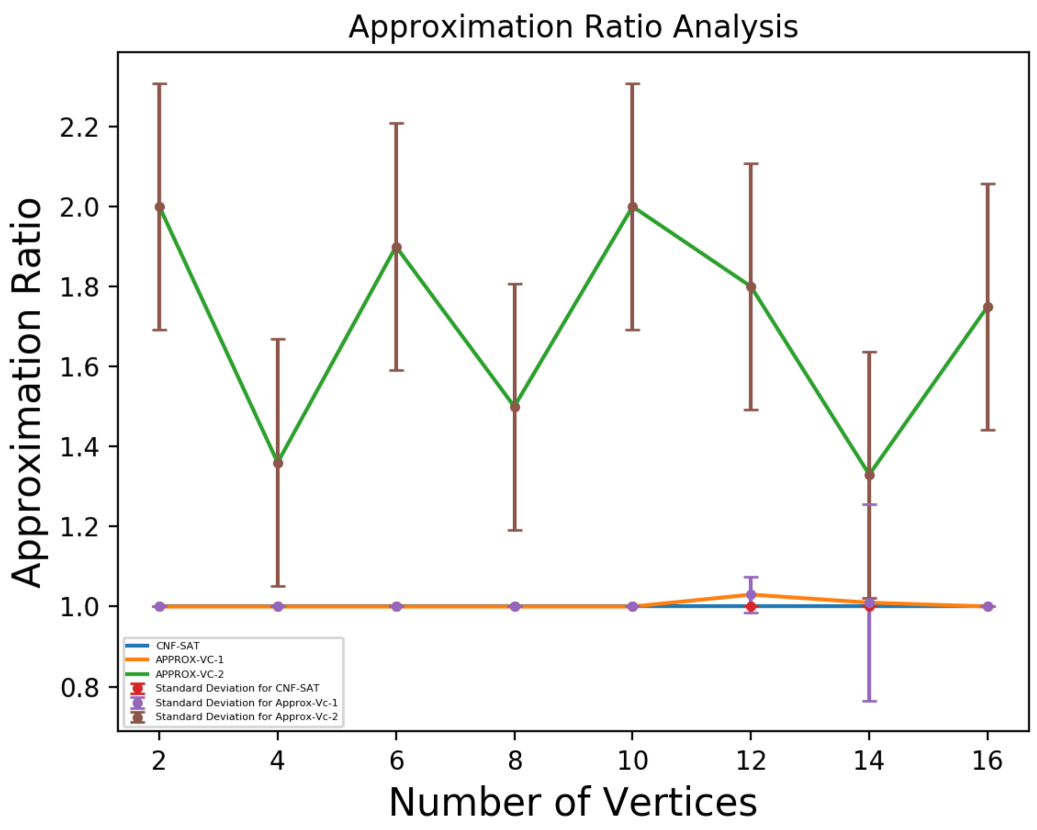

Figure 3 Approximation ratio Analysis for all three algorithms

In this part, analysis is done based on approximation ratio for the three algorithms. We can find from this graph that APPROX-VC-2 is taking less time but performing worst in case of Approximation Ratio. This can be because it is deleting edges without considering any other issues. So, it has more chances to calculate bigger vertex cover than APPROX-VC-1. The result of CNF-SAT, however, can be regarded as the minimum sized vertex cover. Apparently, the ratio of APPROX-VC-1 is also 1, according to the graph, at most times but the ratio of APPROX-VC-2 is much higher than APPROX-VC-1. So, the probability of getting a correct minimum vertex cover for APPROX-VC-2 is much higher than APPROX-VC-1.

CONCLUSION

In this report we studied about different approaches to get the minimum vertex cover of a graph. It can be hence concluded from this analysis that the running time of each approach increases with the number of vertices. If only running time is considered, then it can be concluded that total running time of CNF-SAT is much larger than that of the other two approaches, followed by APPROX-VC-1 and the least running time is for APPROX-VC-2 method. Thus APPROX-VC-2 is the most effective way to compute the minimum vertex cover problem according to the running time analysis.

However, if considering the analysis with respect to approximation ratio, it can be concluded that APPROX-VC-2 is more likely to fail to product precise answer for the minimum vertex cover. Hence, although the approximation algorithms are efficient in terms of time and space complexity, their results of approximations are not always guaranteed optimal, whereas CNF-SAT always gives the optimal result.