Analyzing Performance of SAT Solver for Solving MIN-VERTEX COVER Problem

Allada, Aishwarya Krishna
Student ID : 20816326
akallada@uwaterloo.ca

Nandakumar, Amith
Student ID : 20859891
a4nandak@uwaterloo.ca

December 2019

ABSTRACT

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. In this report, we will use three different approaches to solve the minimum vertex cover problem by using C++ program which is multithreaded having 4 threads, one for I/O, and one each for the three different algorithms. The result shows the most effective way to solve minimum vertex cover.

INTRODUCTION

The finding of the Vertex Cover for a graph is done using three different algorithms. They are –

1. CNF-SAT-VC – An algorithm that uses polynomial time reduction of the decision vertex of Vertex Cover to CNF-SAT, and then uses a SAT solver to get the final results.

2. APPROX-VC-1 - An algorithm which picks a vertex of highest degree (most incident edges), adds it to the vertex cover, and throws away all the incident edges on that vertex till no edges remain.

3. APPROX-VC-2 – An algorithm which picks a random edge u,v , adds both u and v to the vertex cover and throws away all the edges attached to u and v. And this process repeats until no edge remains.

Here, we are using multi-threaded approach, where one thread is assigned for the I/O, and one each for the three different algorithms to solve the minimum vertex cover problem.

ANALYSIS

Analysis of the efficiency of these three algorithms is done using two factors:

1. Running Time
2. Approximation Ratio

RELATING METHODS (METHODOLOGY)

1. MiniSat: MiniSat is a SAT solver which can determine if it is possible to find assignments to boolean variables that would make a given expression true, if the expression is written with only AND, OR, NOT, parentheses, and boolean variables. If it is satisfiable, most SAT solvers (including MiniSAT) can also show a set of assignments that make the expression true. Many problems can be broken down into a large SAT problem (perhaps with thousands of variables), so SAT solvers have a variety of uses.

2. Multithreading: Multithreading is the ability of a central processing unit (CPU) or a single core in a multi-core processor to execute multiple processes or threads concurrently, appropriately supported by the operating system. If a thread gets a lot of cache misses, the other threads can continue taking advantage of the unused computing resources, which may lead to faster overall execution, as these resources would have been idle if only a single thread were executed.

EXPERIMENTAL DESIGN

The whole experiment is done in three different steps -

First, architecture for a thread is designed for the whole project, where, in the main function, the program creates I/O thread to accept the input from the input device. Inside we are creating 3 more threads for the three algorithms. Meanwhile, the four threads run concurrently.

Second, the three algorithms will be implemented and the efficiency will be analyzed. Adjacency list is chosen to store the information of the input graph. After parsing the input, the program initializes the graph using the adjacency list. We then build the three algorithms, CNF-SAT-VC, APPROX-VC-1 and APPROX-VC-2 for analyzing, where each function outputs a vector which will be passed to I/O thread.

Third, to be more specific, for Approx-VC-2 algorithm, we are choosing the first edge and performing vertex cover associated with each vertex in that edge. And also in CNF-SAT-VC, as the running time increases with the number of vertices, we will wait for 2 seconds for the this algorithm to produce a vertex cover. Else, we will call the interrupt() and solveLimited() API from MINISAT. The following

part shows the efficiency of the three algorithms.

RESULTS AND ANALYSIS

Running Time Analysis

Comparing running time of the three algorithms, it's seen that there is a huge difference between the running times of CNF-SAT-VC and the other two algorithms. Hence separating them into two graphs, the results are as in the **Figure 1** and **Figure 2** for Running time Analysis for CNF-SAT-VC Algorithm and Running time Analysis for Approx-VC-1 and Approx-VC-2 Algorithms, respectively. It is seen that the running time of all the three algorithm increases with the increase in the number of vertices. This can be the reason why the running time is increasing as the CPU has to process more data.

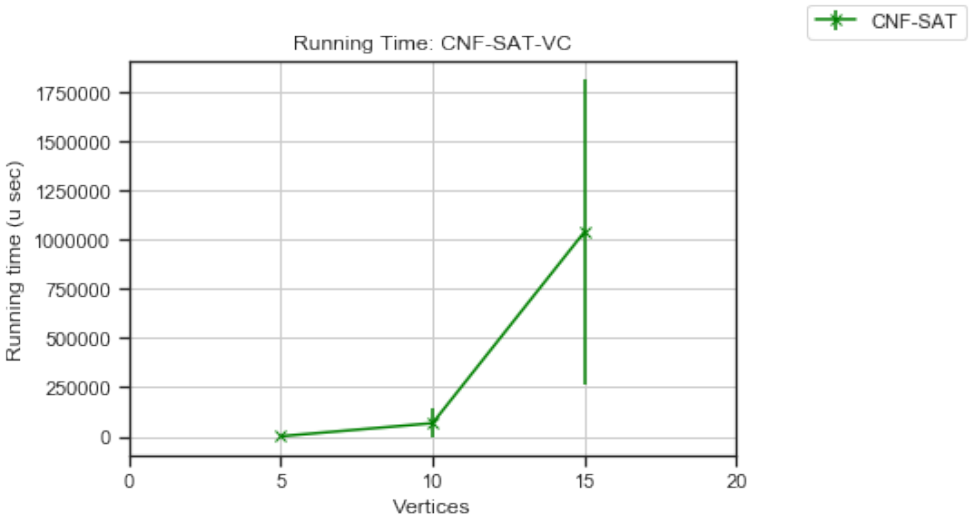

Figure 1: Running time Analysis for CNF-SAT-VC Algorithm

From the **Figure 1** and **Figure 2**, we can infer that the running time of CNF-SAT-VC is the highest when compared with APPROX-VC-1 and APPROX-VC-2 algorithms. The main reason behind this high running time in CNF-SAT-VC may be because of the usage of clauses to finish the reduction and get the minimum vertex cover.

However, time complexity of CNF-SAT-VC Solver is theta($n\hat{4}$), which is way higher than the other two algorithms. Meanwhile, CNF-SAT-VC is NP problem, which says that the output obtained by the SAT solver cannot be in polynomial time.

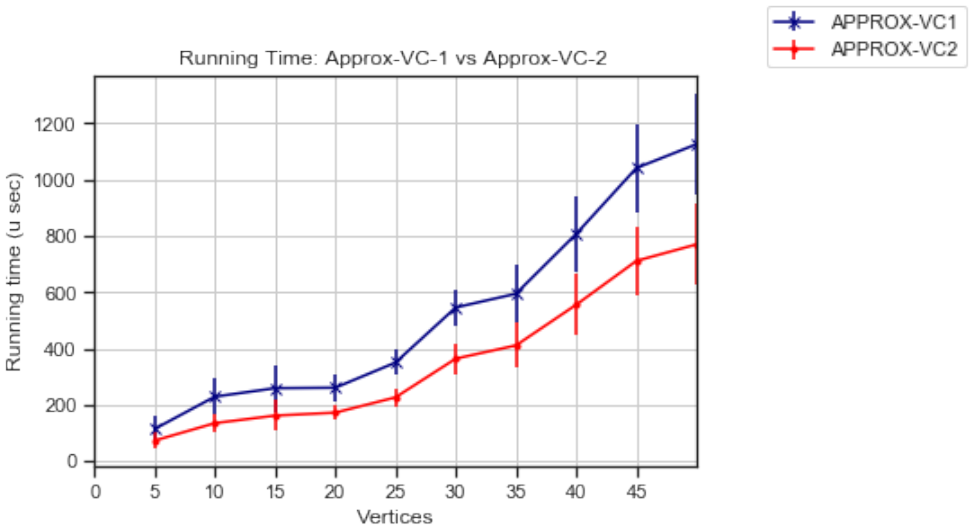

Figure 2: Running time Analysis for Approx-VC-1 and Approx-VC-2 Algorithms

When analyzing Figure 2 we can infer from the graph that the running time of APPROX-VC-1 is comparatively higher than APPROX-VC-2. The reason may be because of the time taken to choose the highest degree more when compared to choosing the first edge and performing vertex cover associated with each vertex in that edge. Also, Approx-VC-1 is fluctuating in way more, when compared to APPROX-VC-2, which is mainly because, APPROX-VC-1 is a greedy algorithm in finding the highest degree vertex.

Approximation Ratio Analysis

Approximation Ratio is calculated according to the below formula:

Approximation Ratio: (size of the computed vertex cover)/ (size of the minimum vertex cover)

We regard the result of CNF-SAT-VC as the optimal minimum-sized vertex cover. Now, as per the analysis done, though APPROX-VC2 is taking less time, it's performing worse in case of Approximation Ratio. Also, as APPROX-VC-1 is approximately 1, most of the time, the probability of this algorithm giving the correct minimum size of vertex cover at the most of time is high. Also, with respect to the graph, the ratio of APPROX-VC-2 is much higher than APPROX-VC-1, which implies that the probability of getting a correct minimum vertex cover for APPROX-VC-1 is much higher than APPROX-VC-2.

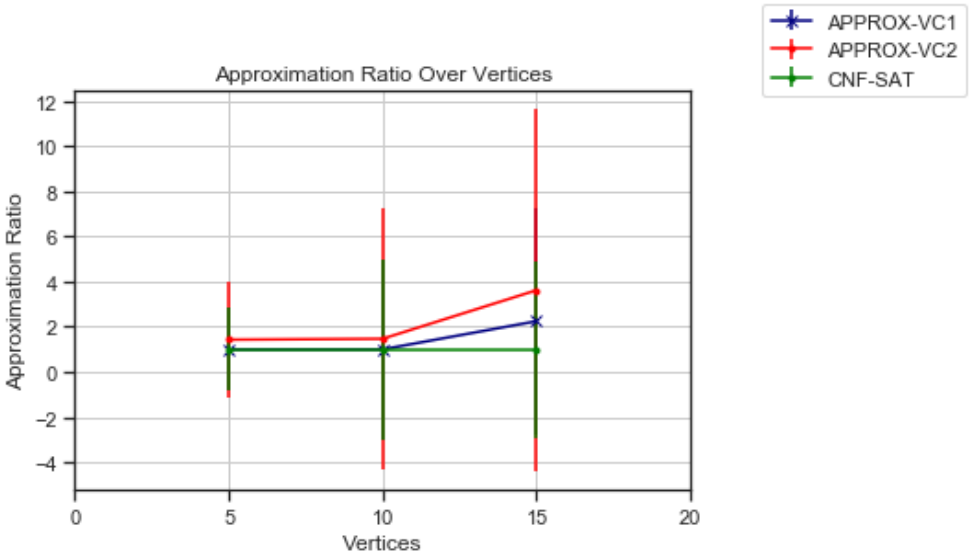

Figure 3: Approximation ratio Analysis for all three algorithms

CONCLUSION

In this report, we studied and compared three different approaches of solving the minimum vertex cover problem of a graph.

From the analysis, it is concluded that the running time of each approach increases as the number of vertices increase. If only the running time is considered, it is clear that the running time of CNF-SAT-VC algorithm is larger, followed by APPROX-VC-1 algorithm and the least running time being for APPROX-VC-2 algorithm. Hence from running time analysis, it is clear that the APPROX-VC-2 algorithm is the most effective way to compute the minimum-vertex cover problem.

However, with respect to the approximation ratio analysis, it can be said that the APPROX-VC-2 is more likely to fail to produce exact answer for the minimum vertex cover. So, though the approximation algorithms are efficient in both time and space complexity, the end results are not guaranteed to be optimal, whereas CNF-SAT-VC always gives the optimal result.

REFERENCES

1. https://dwheeler.com/essays/minisat-user-guide.html
2. https://en.wikipedia.org/wiki/Multithreading_(computer_architecture)
3. Assignment 4 handout and Assignment 5 Handout