University of Waterloo

Department of Electrical and Computer Engineering

Efficiency Analysis of Three Algorithms for Computing Vertex Cover

December 4, 2019

SHASHIKANTH SRINATH
Junhui Wang

1 Introduction

1.1 Implementation of Three Algorithms

In this report, three different approaches are analyzed to solve the vertex cover problem for the graph (V, E) where V represents the number of vertices and E represents the edge between two vertices. The first approach is to implement a polynomial reduction of the decision version of vertex cover to CNF-SAT and use minisat as a library to solve the minimum vertex cover problem. The second approach is to pick a vertex of highest degree (most incident edges), adding it to vertex cover, throwing away all edges incident on that vertex, and repeating this until no edges remain. This algorithm is called APPROX-VC-1. The third approach follows picking an edge <u,v>, adding both u and v to vertex cover, throwing away all edges attached to u and v and repeating this until no edges remain. This algorithm is called APPROX-VC-2. The efficiency of these approaches is analyzed for various inputs. It can be characterized through parameters such as running time and approximation ratio.

1.2 Improvement of CNF-SAT-VC Algorithm

Based on the polynomial reduction from Vertex Cover to CNF-SAT provided by our professor, we improved it by adding an extra clause. This clause follows a new rule: For atomic propositions $x_{i,j}$ and $x_{m,n}(i < m)$, they are both true if and only if $j < n$.

Adding this clause reduces the number of models for a certain k(number of vertices in the vertex cover). As a result, the time needed to find a model is much shorter compared to the former algorithm, which makes it possible for this algorithm to be scaled to 50 vertices.

2 Analysis

Multithread programming is implemented with 4 threads. One thread handles input and output to the console, and three other threads handle different approaches to solve the minimum vertex cover problem. The I/O thread will supply input to other threads and running time is calculated at the end of each thread using pthread-getcpuclockid() function which constitutes running time analysis. The Approximation ratio is calculated as the ratio of the size of the computed vertex cover to the size of the minimum-sized vertex cover.

The algorithms are tested with values of V ranging from 5 to 50 in the increments of 5. 10 graphs are generated for each V and are used to compute running time and approximation ratio. To obtain an accurate analysis of running time, each graph is run 10 times and the mean (average) and standard deviation across 100 runs for each value of vertices are computed

2.1 Running Time Analysis

The amount of time taken by each algorithm to return vertex cover is calculated and plotted against the number of vertices. Standard deviation is computed from numerous runs at each value of V and it is plotted as error bar in the graphs.

2.1.1 Running Time of CNF-SAT-VC

The time to find a vertex cover which has k vertices increases as the number of vertices grows. This is because the number of atomic propositions $x_{i,j}$ grows. More importantly, the number of different combinations of atomic propositions increases significantly, which means that minisat will take a much longer time to find a model. We use binary search to find the minimum vertex cover. The number of times minisat is invoked is proportional to the number of vertices.

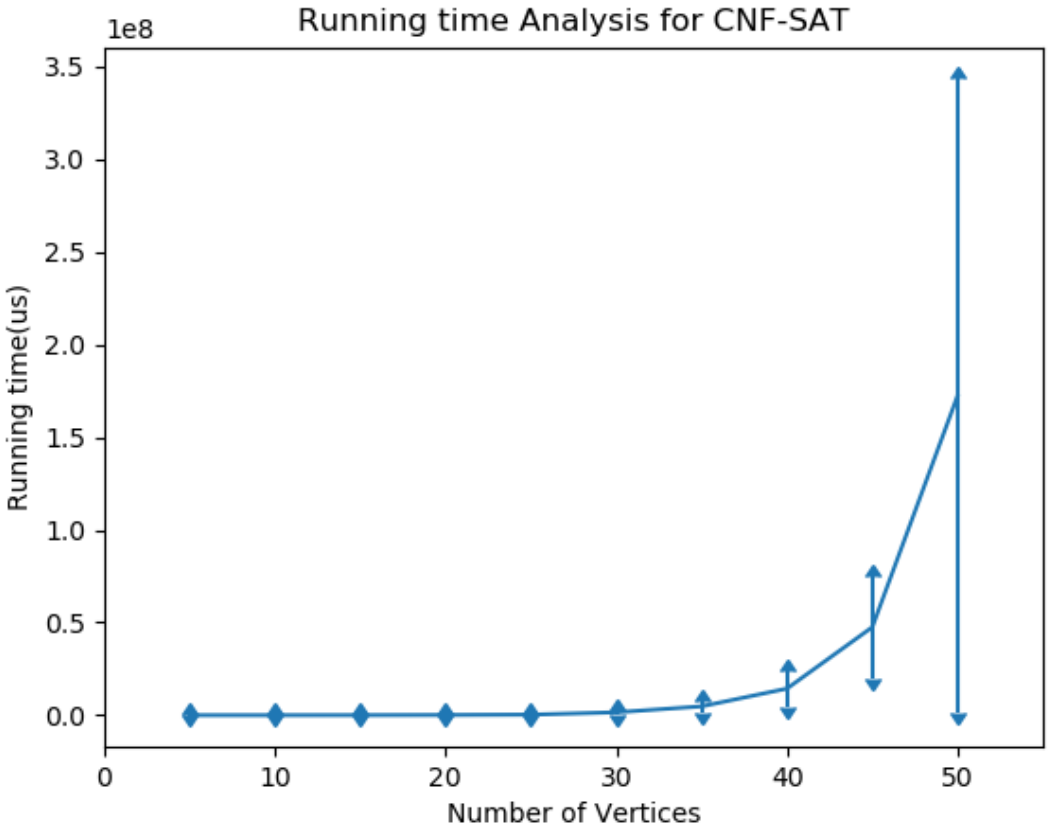

Figure 2.1: Running time Analysis of CNF-SAT with standard deviation as error bar

2.1.2 Running Time of APPROX-VC-1 and APPROX-VC-2

For a higher number of vertices, it can be seen that Approx-VC-2 performs faster than Approx-VC-1. This is because as the number of vertices increase, Approx-VC-1 takes a longer amount of time searching for the vertex of highest degree.

For a lesser number of vertices, it can be observed that Approx-VC-2 takes an almost equivalent

or greater amount of time than Approx-VC-1. This is due to the randomness involved in Approx-VC-2 in selecting the edge to constitute its vertex cover.

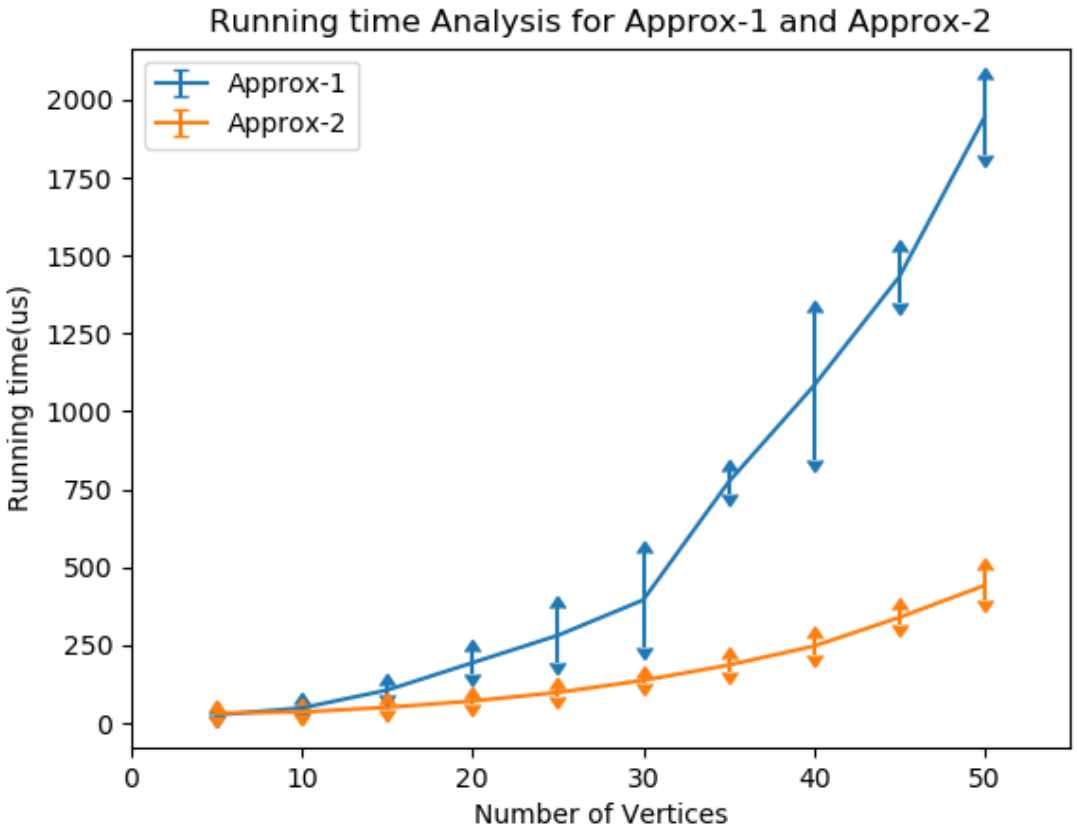

Figure 2.2: Running time Analysis of Approx-VC-1 and Approx-VC-2 with standard deviation as error bar

2.1.3 Comparison of Run Time

Since running time for CNF-SAT-VC is much greater than the other two algorithms, logarithmic scale is used to plot their comparison in 2.3. It can be observed that running time of CNF-SAT-VC increases exponentially with number of vertices, and is much larger than running time of Approx-VC-1 and Approx-VC-2. This is due to increase in number of clauses and as CNF-SAT has to traverse through all the clauses to find a suitable assignment. The other two algorithms have lesser time complexity than CNF-SAT-VC

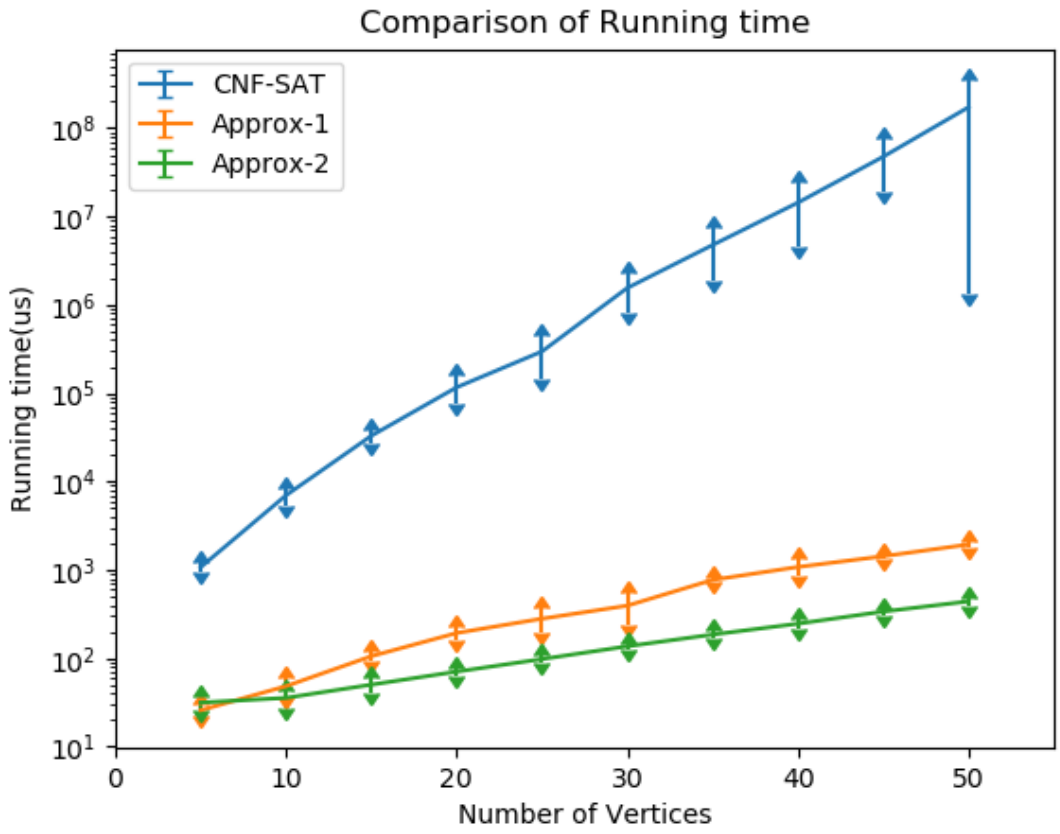

Figure 2.3: Comparison of the running time of three algorithms

2.2 Approximation Ratio Analysis

Approximation Ratio is defined as the ratio of the size of calculated vertex cover to the size of an optimal(minimum-sized) vertex cover. Since CNF-SAT-VC algorithm can generate one of the optimal vertex covers, we compute the approximation ratios of the other two algorithms by comparing their output to the output of CNF-SAT-VC algorithm. The figure 2.4 below is plotted according to the data of the approximation ratio we collect.

If we want to make the number of vertices in the vertex cover as smallest as possible, an intuitive way is to find the vertices with most incident edges. That is exactly what we do in APPROX-VC-1 algorithm. It's easy to find out that picking vertices and putting them into vertex cover randomly is not an efficient way. Therefore, in the figure of approximation ratio, the approximation ratio of APPROX-VC-1 is much smaller than APPROX-VC-2 in general.

When diving deeper, it is easy to find out that the approximation ratio of APPROX-VC-1 keeps consistent, but the approximation ratio of APPROX-VC-2 keeps increasing linearly until it reaches 1.6. This is because APPROX-VC-1 just does similar things no matter how many vertices are involved. In other words, each time it just searches all the edges left with highest degree(most incident edges). This method guarantees that it has almost the same approximation ratio. However, in the APPROX-VC-2 algorithm, it picks an edge randomly every time, therefore it becomes harder and harder to pick the vertices with many incident edges when the number of vertices increases. As a result, the approximation ratio becomes larger. After the value of V exceeds 25, the APPROX-VC-2 algorithm produces non-optimal solution, with consistent higher

approximation ratio.

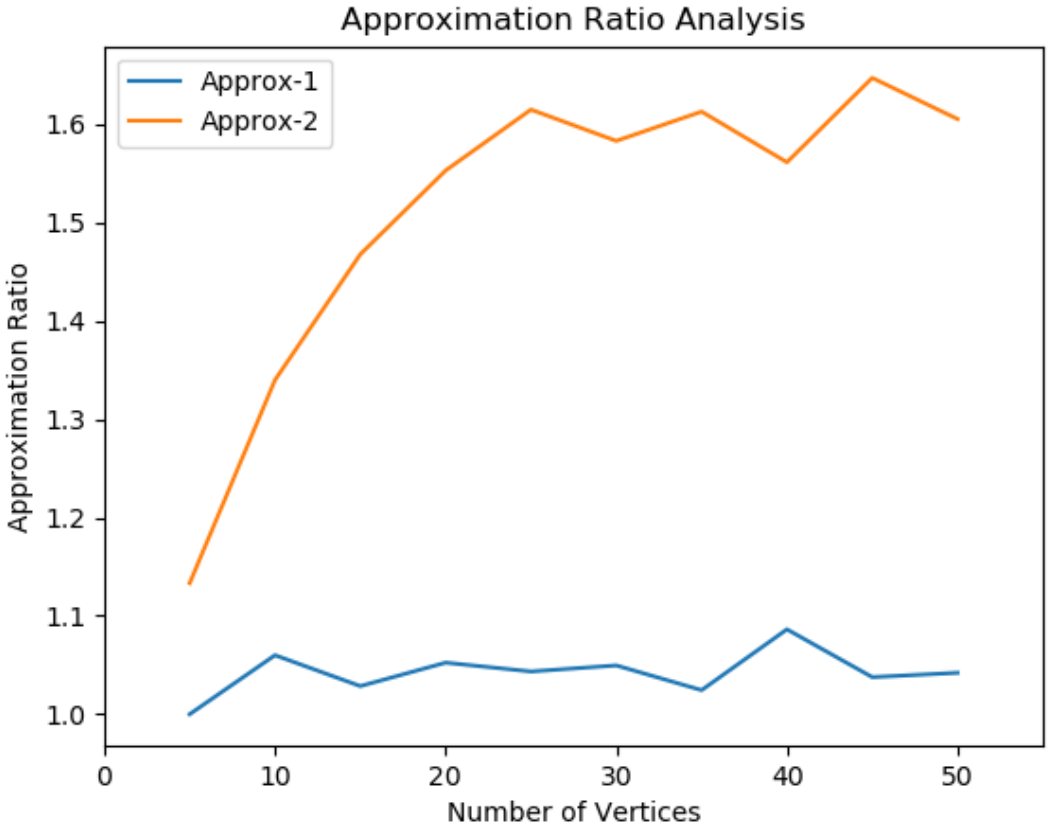

Figure 2.4: Approximation ratios of APPROX-VC-1 and APPROX-VC-2

3 Conclusion

CNF-SAT-VC takes the highest amount of running time when compared with Approx-VC-1 and Approx-VC-2. CNF-SAT-VC's running time increases exponentially with the number of vertices, hence it is not scalable for a larger number of vertices. The standard deviation of the running time of CNF-SAT-VC also increases with the number of vertices. From the analysis, it can be observed that Approx-VC-2 has the least running time followed by Approx-VC-1. Both of these algorithms are scalable for a larger number of vertices.

Approx-VC-1 is observed to have a consistent Approximation ratio with value which is almost equal to 1. Approx-VC-2 produces non-optimal solution for most of the vertices. This proves that Approx-VC-1 can be an efficient algorithm to find vertex cover.