ECE 650 - METHODS AND TOOLS FOR SOFTWARE ENGINEERING

UNIVERSITY OF WATERLOO

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Project : Optimal Analysis of Vertex Cover using various Algorithms

Authors:
Rahul Sharma (ID: 20980992)
Arshad Momin (ID: 20986176)

Date: April 10, 2022
Submitted to: Prof. Arie Gurfinkel

Contents

1 Introduction

The main objective of the project is to cover Vertices in an optimum possible way. This goal can be achieved by using Minimum vertex cover optimization problem which is a typical example of an NP-complete problem that has an approximation algorithm.
These algorithms are:
- Approx-VC-1 (as described in the instructions)
- Approx-VC-2 (as described in the instructions)
- CNF-SAT-VC (as described in the instructions)

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).
A vertex cover of a graph G(V, E) is a subset of vertices V such that for every edge (u, v) is a subset of E, at least one of the vertices u or v is in the vertex cover. In this project we discuss different techniques and algorithms used to solve the parameterized Vertex Cover problem. The first approach is to encode vertex cover to CNF-SAT clauses in a graph (using MiniSat). We also have two approximation algorithms Approx-VC-1 and Approx-VC-2 that are used to calculate the vertex cover.

2 Algorithms

CNF-SAT:

The reduction of vertex cover to CNF-SAT is a polynomial time reduction process. The graph uses literals and clauses in CNF form. The clauses are generated using A4 encoding process.
CNF is a dis-junction of literals which means logical OR is performed on literals and Logical AND is performed on clauses. These clauses are then provided to MiniSat SAT solver inorder to determine satisfiable conditions (if vertex cover is present for current graph).

Multithreading:

We have also used Multithreading which will provide a concurrent execution of the program for optimal utilization of CPU. To start a thread we simply need to create a new thread object and pass the executing code to be called . Once the object is created new thread is launched and this executes the code in specified call.

Approx-VC-1 :

Vertex with highest degree is considered which means the vertex which occurs most frequently. This vertex is then added to the vertex cover list and edges being removed that are attached to these vertex. The process is repeated until no edges remains.

Approx-VC-2 :

In this approach the Vertex Cover uses an arbitary edge (u,v) from set of edges E. The Edges connected to these both vertices are deleted and this process is continued until no edges remain.

3 Analysis Methodology

1. Running Time
2. Approximation ratio

These above factors provide efficiency of the Algorithms. The graphs are generated for each vertices. With help of this, we compute the Running time and the Approximation ratio for each graph.

4 Analysis

Run time of CNF-SAT-VC

Three running time vs number of vertices graph plots are plotted representing running time on Y-axis and number of vertices on X-axis.
Figure 1 represents the running time of CNF-SAT Algorithm,where we plot the mean and standard deviation of running times with increments of 5. In CNF-SAT-VC algorithm, the running time increases exponentially with the increase in the number of vertices. For less literals and less clauses it takes less time to find satisfiability and large value of vertices, literals and clauses leads to increase in time consumption. This graph will steadily increase exponentially with increase in vertices. We can also clearly see that the standard deviation has increased a lot when the vertex is 15. The duration for the **CNF-SAT is divided by 100000** so that we compare the values of the three algorithms. As from Figure 1 it was difficult to comprehend, so after decreasing the value of the CNF-SAT we could compare the 3 algorithms in Figure 2. Furthermore, we have plotted the graph in the Figure 3 from 5 to 15, so we could differentiate better. It is to be noted that the value of duration CNF-SAT is still reduced by 100000.
Also, it can be seen that the error bars in Figures reveal fluctuation in CPU time for each vertex, indicating that the running time is not consistent through the vertices also. In addition, it can be seen that the standard deviation is not that significant for the vertices

count less but the deviation increases with increase in number of vertices. Also, if we see the figures closely, we can notice that the running time of CNF-SAT is significantly higher as compared to both approx-VC-1 and approx-VC-2. The trend follows an exponential rise as we increase the number of vertices which inturns increase the number of the edges, however, the results of CNF-SAT are the most optimal.
Figure

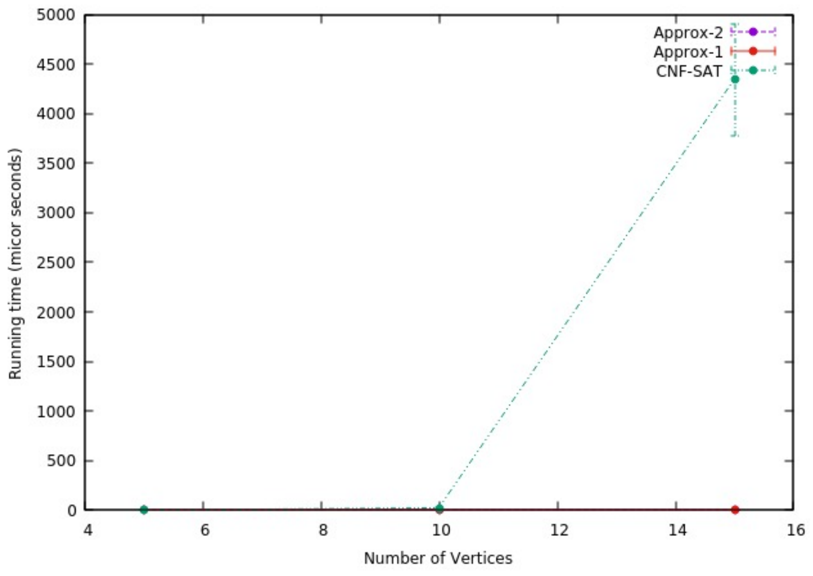

Figure 1: Run Time analysis for Vertice range [5,15]

The interval duration for CNF-SAT algorithm is set to 10 pico seconds and hence in below plots we are able to get runtime values for individual value of vertices.

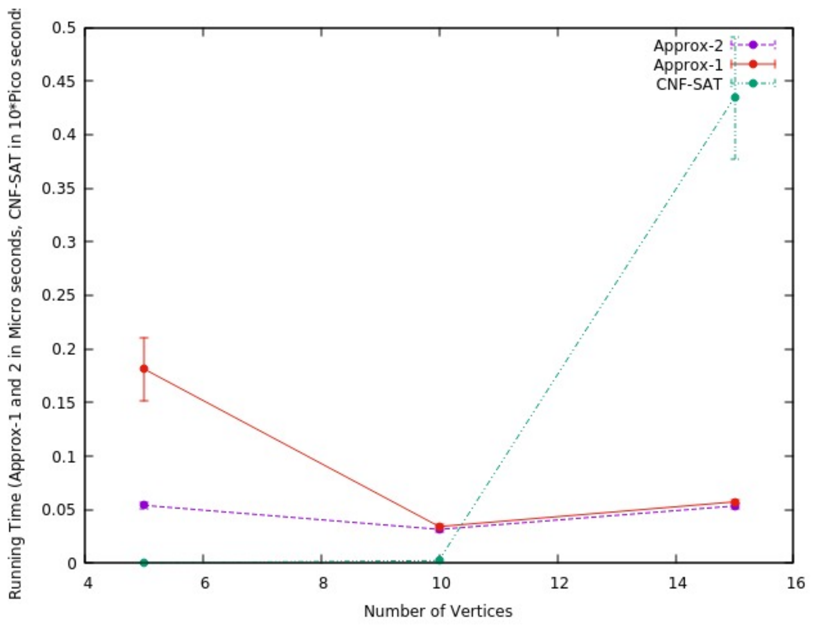

Figure 2: Run Time analysis for Vertice range [5,15]

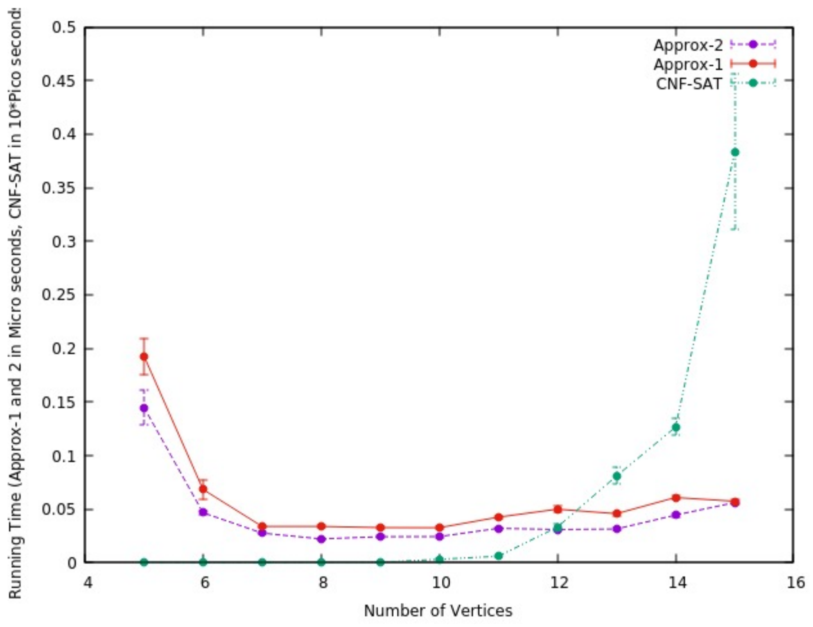

Figure 3: Run Time analysis for Vertice range [5,15]

Run time of Approx-VC-1 and Approx-VC-2

The running time of Approx-VC-1 and Approx-VC-2 is analyzed by plotting mean standard deviation of running times for each value of V (Vertices) in [5,50]. However, the running time of Approx-VC-1 is observed higher compared to Approx-VC-2. This is because APPROX-VC-1 is used in optimization problems which calculate the maximum degree of the vertex and then finds the vertex cover list by visiting each node in every iteration. In case of Approx-VC-2 it doesn't find the highest degree Vertex in every iteration and it also does not return back to the node that was visited before. This results in the less number of nodes to be visited in next iteration. Therefore, the time complexity of Approx-VC-1 is greater than that of Approx-VC-2 for most of the points. However, for the vertices uptill 10 it was seen that the Approx-VC-1 is taking less time as compared to the Approx-VC-2.
Figures:

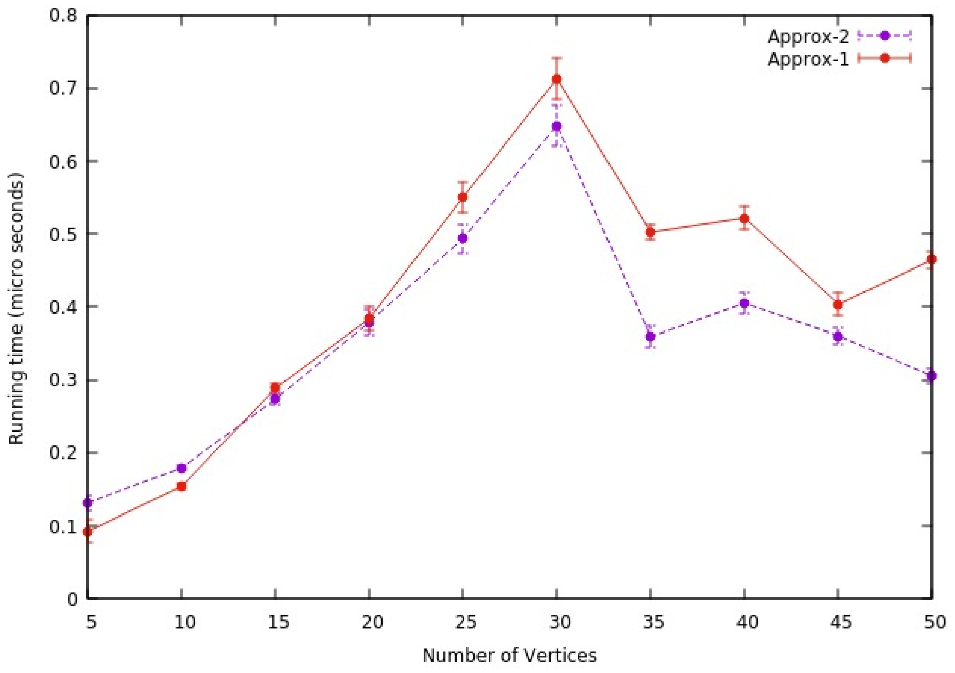

Figure 4: Approximate Ratio for Vertice range [5,50]

Approximation Ratio:

In this figure we plot Approximation ratio vs Number of vertices V, representing approximation ratio on Y-axis and number of vertices on X-axis. With respect to the project problem statement it is decided that CNF-SAT-VC is an Optimal solution. Therefore, we define the approximation ratio by comparing the Vertex cover size of CNF-SAT-VC with vertex cover obtained from Approx-VC-1 and Approx-VC-2.
This results in terms of minimal Vertex cover and from the graph we understand that the vertex cover generated by Approx-VC-1 is almost equal to that of CNF-SAT. As CNF-SAT provides minimum number of vertices cover and hence the approximation ratio is equal to 1 in all vertex cases here which is different in the case of Approx-VC-2. It is observed that this algorithm has degraded the approximation ratio amongst the other two because it doesn't find vertex with the highest degree instead it takes the arbitrary vertex and vertex cover usually not close to the minimum vertex cover. The figure 5 was for the vertices 5,10,15 only and it was difficult to get any deduction, therefore we plotted the graph for the vertices from 5,6,7,...to 15 continuously. The ratio of vertices for Approx-VC-2/CNF-SAT is always greater than 1.5. However, for the Approx-VC-1/CNF-SAT is always lesser than 1.1.
Figures:

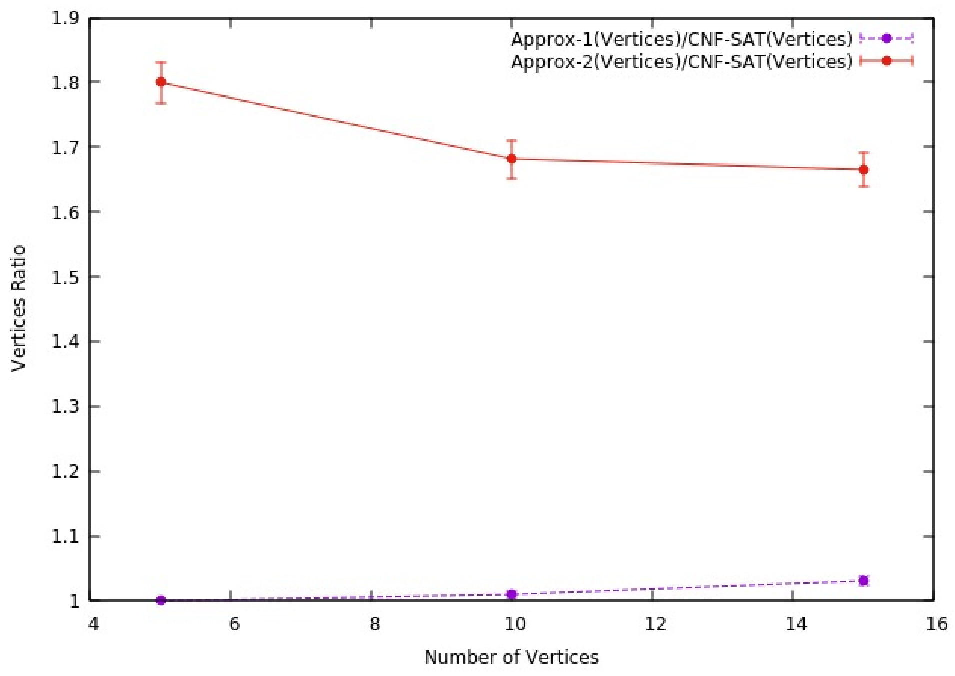

Figure 5: Approximate Ratio for Vertice range [5,15]

The interval duration for CNF-SAT algorithm is set to 10 pico seconds and hence in below plots we are able to get runtime values for individual value of vertices.

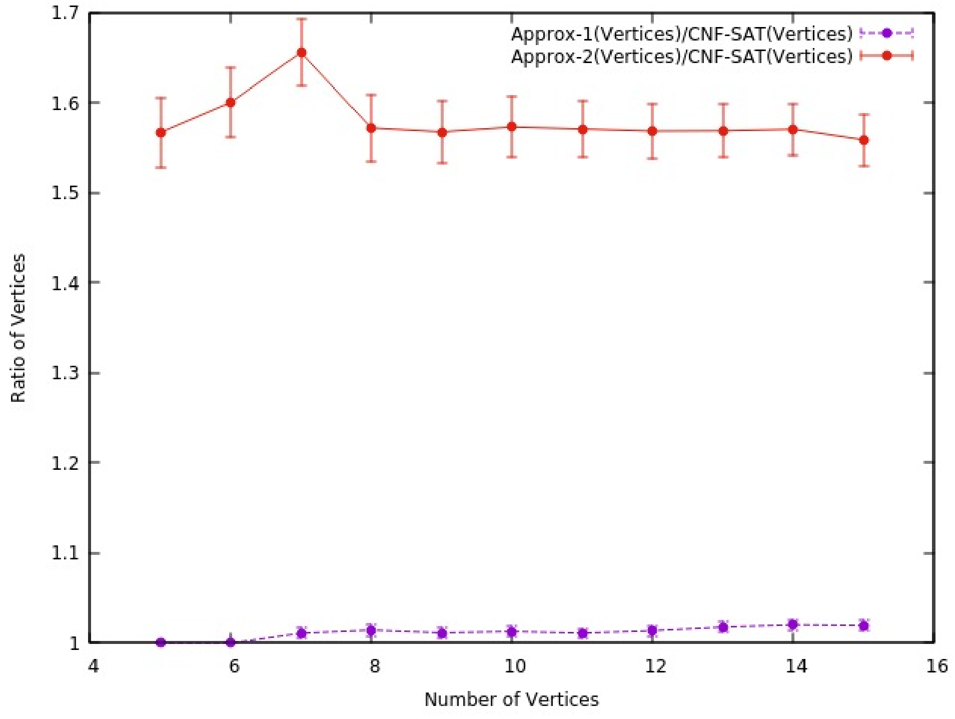

Figure 6: Approximate Ratio for Vertice range [5,15]

5 Conclusion

Obviously the sat solver returned optimal results and was used as the control and benchmark for the approximation ratio results. It is important to note that for larger graphs the CNF-SAT-VC algorithm quickly becomes intractable due to its exponential nature. Depending on the application, it would be good to consider an approximation method when graph inputs become larger than 15 vertices. The results are evident from the figure 1, Figure 2 and Figure 3.
In conclusion, when we consider Approximation ratio APPROX-VC-1 algorithm becomes more efficient than the APPROX-VC-2. When running time is the constriction of design APPROX-VC-2 should be the preferred algorithm to compute vertex covers because it takes the least time amongst all the three. However, if the main focus is to find minimum vertex then, APPROX-VC-2 is more likely to fail.

Reference :

1. CNF-SAT (https://www.geeksforgeeks.org/2-satisfiability-2-sat-problem/)

2. MultiThreading (https://www.geeksforgeeks.org/multithreading-in-cpp/)

3. Vertex Cover (https://www.geeksforgeeks.org/vertex-cover-problem-set-1-introduction-approximate-algorithm-2/)

4. Concurrency (https://www.toptal.com/software/introduction-to-concurrent-programming)

5. Run time measure (https://man7.org/linux/man-pages/man3/pthread$_getcpuclockid.3.html$)