ECE 650 - Methods and Tools for Software Engineering

University of Waterloo

Department of Electrical and Computer Engineering

Vertex Cover Problem

Authors:
Muhammad Areeb Ansari (ID: 20812646)
Muhammad Mohsin Tahir (ID: 20812155)

Date: December 3, 2019

Contents

1 Problem Statement

In this project we had to solve a classical optimization problem known as Vertex Cover problem.The input is of graph which is undirected and is defined as G = (V,E). We were required to find minimum sized Vertex Cover set which includes vertices such that all the edges are adjacent to those vertices in the Vertex Cover set.This basically means that if every edge is defined as (i,j) then either i or j is in the vertex cover set. Vertex Cover is NP-Complete problem i.e., there is no polynomial time solution for this unless P = NP. The following figure have highlighted the minimum number of vertices which should be included in the Vertex Cover Set.

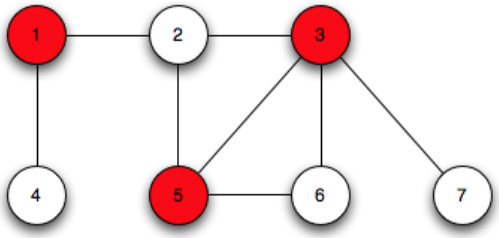

Figure 1: Minimum Vertex Cover Set (www.ycpcs.github.io)

2 Algorithms

We posed following three approaches as solutions for finding the Minimum Vertex Cover Set.

2.1 APPROX-VC-1

In this algorithm we selected that Vertex which had the maximum number of edges incident to it. The selected Vertex was then moved to a list called as minimum Vertex Cover Set. Then we removed all those edges from the edges data-set which were incident to the selected Vertex. The process was repeated until the edges list gets empty. This approach can be referred to as greedy approach as we are finding the local optimum solutions in order to global optimum solution.

2.2 APPROX-VC-2

In this algorithm we selected randomly an edge (x,y) from the edge data-set. Then we put the x and y in a data-set called VC. Then we removed all the edges which were incident to x and y from the edges data-set. This process was repeated until the data-set

of edges became empty. VC in this case contains the minimum Vertex Cover Set. We notices in this case that VC always contains even number of vertices as in every cycle two vertices are added in VC.

2.3 CNF-SAT-VC

This algorithm takes polynomial time in order to reduce VERTEX-COVER to CNF-SAT. CNF contains literals i.e the literals are OR with each other and the clauses are AND with each other. CNF is basically a conjunction of disjunction. In this approach we made clauses and used MINISAT to provide a satisfiable solution for all of these clauses. Clauses were designed such that it met all the following conditions. 1) One vertex cannot come more than once in the Vertex Cover Set. 2) There should be minimum of one Vertex in Vertex Cover Set. 3) Each and every edge is incident to minimum of one vertex in the Vertex Cover Set. 4) Single position in the Vertex Cover Set can be occupied by one Vertex only.

3 Analysis

For Analysis part, we have considered the following two factors in order to determine efficiencies of the above mentioned Algorithms.

Running Time

It refers to the total time consumed by the algorithm to find the vertex cover. We have used 4 threads in our project, one for Inputs and three for the each algorithm. We have made use of “pthread-getcpuclockid” to calculate the value of thread clock Id and “clock-gettime” to determine the clock time. Running time for each thread calculated tells us about the efficiency of each algorithm. We tested each of our algorithm on inputs varying from V = 5,10,15. These inputs were generated through graphGen. For each input we tested our code 10 times and took average of the Running Time for each algorithm.The graph for running time is provided on the next page.

Approximation Ratio

Approximation Ratio in this case can be defined as the ratio of the size of calculated vertex cover to the size of an optimum minimum vertex cover. We have observed in our project that CNF-SAT produces the most optimum Vertex Cover Set. Thus, We can use it as reference and will take ratios of the APPROX-VC-1 and APPROX-VC-2 with it. The graph for Approximation Ratio is provided on the next page.

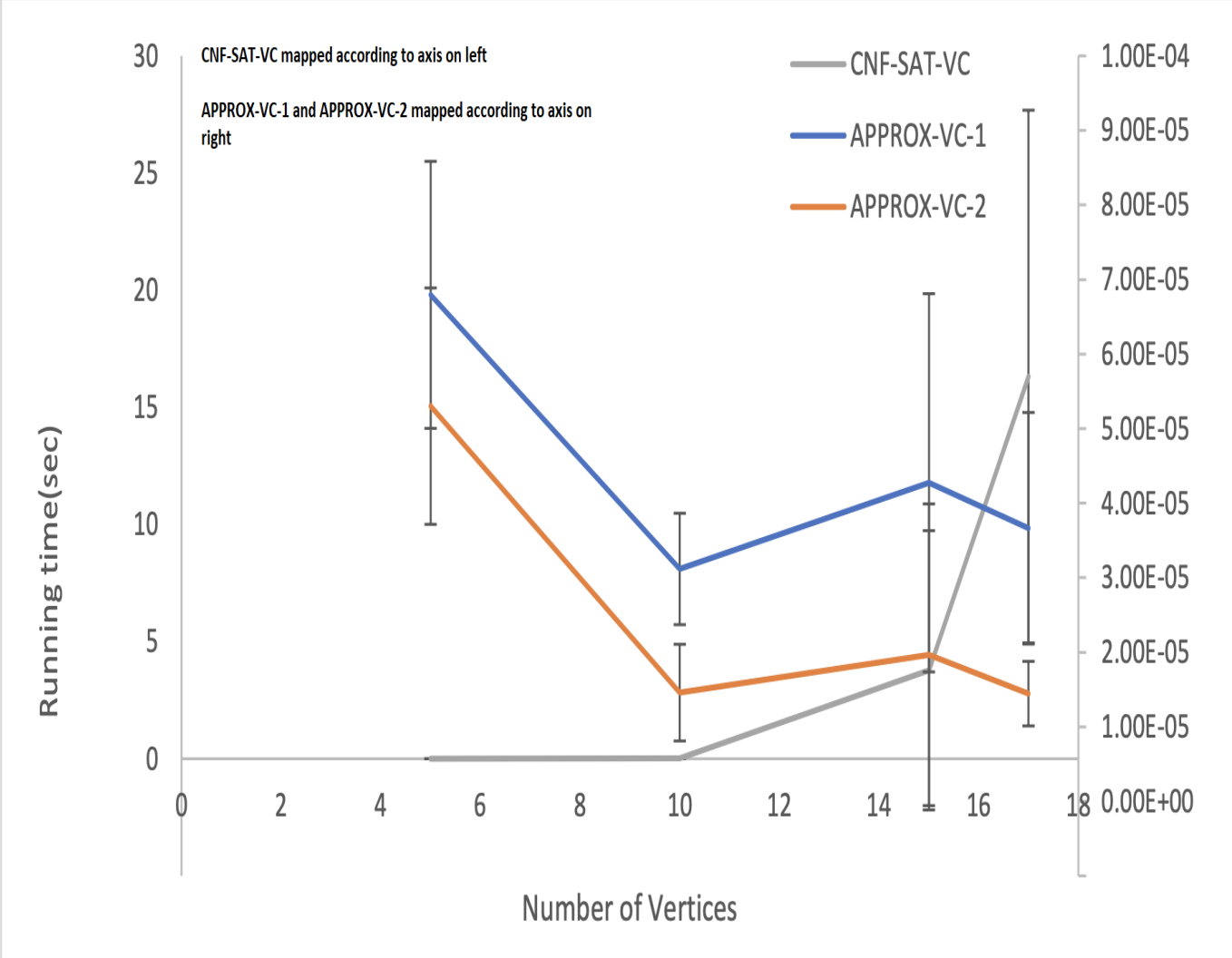

Figure 2: Running Times for all 3 Algorithms

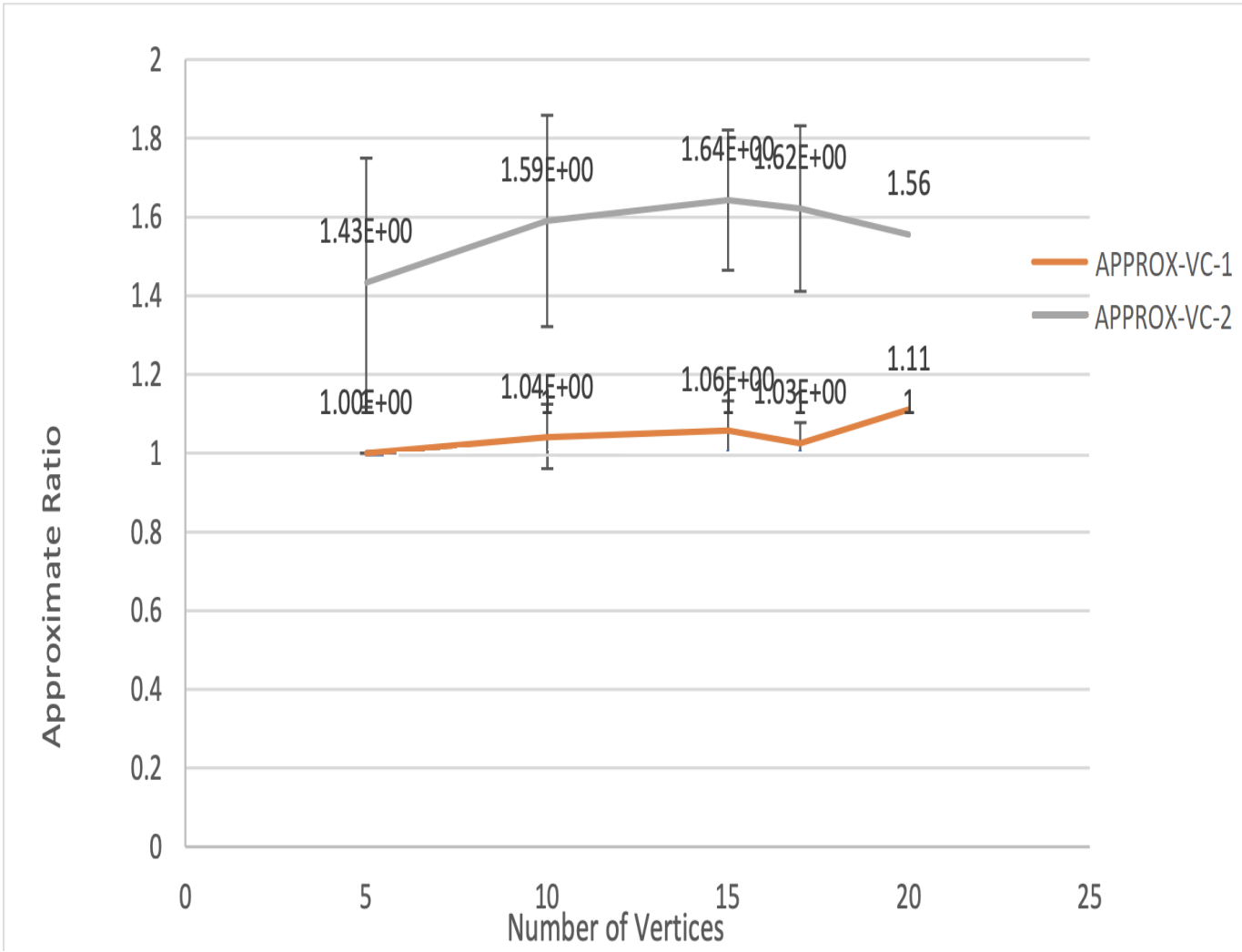

Figure 3: Approximation Ratio with respect to CNF-SAT-VC

3.1 Analysis of CNF-SAT-VC

Running Time:

In the Graph 1 it can be observed that for lower values of vertices e.g. 5 the running time for CNF-SAT-VC is quite low. According to the value calculated it is around 0.003 seconds. Whereas, as we increase the number of vertices from 10 to around 15 then it gets to 1 seconds. However, if we increase the number of vertices to 16 the time consumed rises exponentially to around 1350 seconds. One of the probable reason for these observations can be the fact that number of literals in CNF-SAT-VC depends on the input of number of vertices. As the number of vertices increase time taken by this algorithm increases due to exponential time taken by MINISAT to solve and find the model which solves the problem. At start the number of literals are low where as if number of vertices increase then number of literals increase as well. Our algorithm runs for k number of times for finding the satisfiablity and k is defined such that it is less than equal to number of vertices defined at start. Thus, if number of vertices are less then time taken to get to satisfiablity is less as compared to the case when number of vertices increase. As we know that, this algorithm gives output with good time complexity only up till vertices number of 16 to 17.We are trying to come up with 2 new approaches in order to cater this problem.We are working on the 1)Implementation of Binary execution for selection of the value of K and 2) Trying to pass the vector of unique vertices present in the edges as an input to SAT Solver.

Approximation Ratio:

As CNF-SAT-VC provided optimal minimum cover set so the ratio of CNF-SAT-VC to optimal vertex cover set will always remain 1.

3.2 Analysis of APPROX-VC-1

Running Time:

This approach first looks for the vertices with the most number of edges being incident to it. Thus it has a higher time complexity as compared to APPROX-VC-2. As far as spikes are concerned then may be due to the kind of output of the graphGen ,there would have been greater number of edges thus would have resulted in increase in time complexity in some cases. It has time complexity of O(n2). If we compare this approach with CNF-SAT then time complexity for higher number of vertices is better for this algorithm. CNF-SAT running time increases exponentially.

Approximation Ratio:

We can observe from the graph in figure 3 that approximation ratio for APPROX-VC-1 is lower as compared to APPROX-VC-2. The main reason for this is the fact that in this

algorithm we first check for the highest degree vertex.

3.3 Analysis of APPROX-VC-2

Running Time:

From the figure 2 we can see that the running time for APPROX-VC-2 is slightly less than APPROX-VC-1. The main reason for this is that this algorithm doesn't find the highest degree vertex at the start so its time complexity is less as compared to APPROX-VC-1. Time complexity for this algorithm is better than CNF-SAT as in CNF-SAT the running time increases exponentially for the higher number of vertices.

Approximation Ratio:

We can observe from the graph in figure 3 that approximation ratio for APPROX-VC-2 is higher. The main reason for this is the fact that in this algorithm we select the edges set randomly at start. Secondly we put two vertices in the set at once which means that it will always give even number of vertices in the vertex cover set. Thus it has a higher approximation ratio value as compared to all other algorithms.

4 Conclusion

To conclude our discussion, we can say that if we need highly optimal solution as far as size of the vertex cover set is concerend then CNF-SAT-VC can be used. However, there is a constraint in it. We can use it only up till 16 number of vertices. After that it takes a lot of time to compute the vertex cover so it is not recommended. As far as APPROX-VC-1 and APPROX-VC-2 are concerned then time complexity of APPROX-VC-2 is better than APPROX-VC-1 however, it has the worst optimal output as is evident from its very high Approximation Ratio. So if we have high number of vertices e.g from 15-50 we can make use of Approx-VC-1 because its time complexity is far better than CNF-SAT-VC and it provides optimum sized vertex cover as compared to APPROX-VC-2.