Analyze the Efficiency of Vertex Cover Algorithms

Haozhen Zhao and Tianao Meng

1 Introduction

For the vertex cover problem in the final project, our group use three algorithms to solve the minimum vertex cover problem. (Reduction-to-CNF-SAT, APPROX-VC-1 and APPROX-VC-2). For the Reduction-to-CNF-SAT, it create a reduction of the decision version of Vertex Cover to CNF-SAT and solve the CNF-SAT using the MiniSAT SAT solver. This algorithm is called CNF-SAT in the following part of our report. For the APPROX-VC-1, we use the greedy algorithm to solve the Vertex Cover problem. It picks a vertex of highest degree, add it to our Vertex Cover list, then remove all the edges incident on that vertex. We repeat this process until no edges remain in our edge list. For the APPORX-VC-2, pick the first edge of the input edge list firstly. Because the graph generated by graphGen is randomly, every edge in the edge list has the same possibility to appear in the first slot of the edge list. After picking the first edge ¡u , v¿, we remove all the edges incident on u and v. Repeat this process until no edges remain in the graph.

2 Analysis of running time

2.1 Background

Our group use graphGen to generate 10 graphs for each vertex from 5 to 50, with increments of 5. Because the MiniSAT solver is not a very time-efficient method to solve our problem, we also generate 6 more graphs for each vertex from 5 to 17 with increments of 2 to analyze the CNF-SAT algorithm. For each graph we run ten times and record those data in excel, then generate the plot using gnuplot with the data we get.

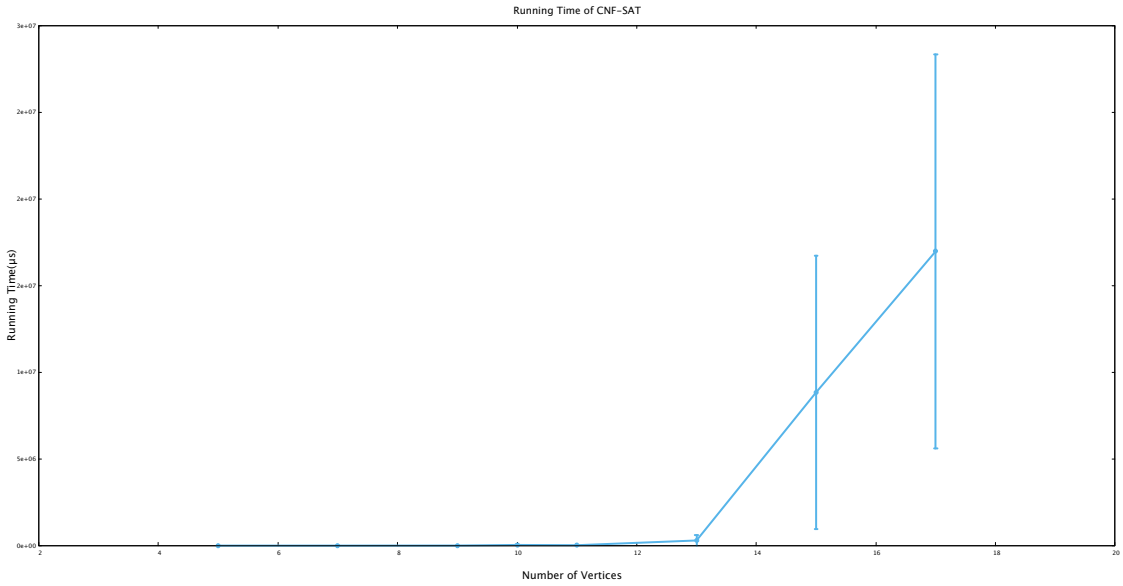

Figure 1: Running Time of SAT with vertices [5, 17] in increments of 2

2.2 Running Time Analysis of CNF-SAT

From Figure 1, we can know that the running time of CNF-SAT and the error bar is too small to see before 17; however, the running time experiences a dramatic increase with the vertex number increase from 13 to 17, and the error bar is also remarkable. From the a4_encoding.pdf, we can know that the number of clauses in the reduction to CNF-SAT is k+$\binom{n}{k}$ +$\binom{n}{2}$+$|E|$ where n is the total number of vertex and k represents the size of minimum vertex cover list got from this algorithm. With the increase of the vertex number, k and n both increase, which means that the total clauses in the reduction increase a lot while the size of some clauses inclines. Thus, from vertex number 13 to 17 has that remarkable increase trend. From Figure 1, we can also get that the standard deviation can be ignored below 13 compared with that over 15

Because the trend and the error bar are not obvious for us to explore, we will plot the logscale CNF-SAT figure, which will provide us clearer figure telling us how the running time of CNF-SAT changes.

This figure is much clearer than Figure 1. From Figure 2, the running time of CNF-SAT almost experiences an increasing trend with some fluctuation. We can also get that the CNF-SAT is not stable algorithm to solve

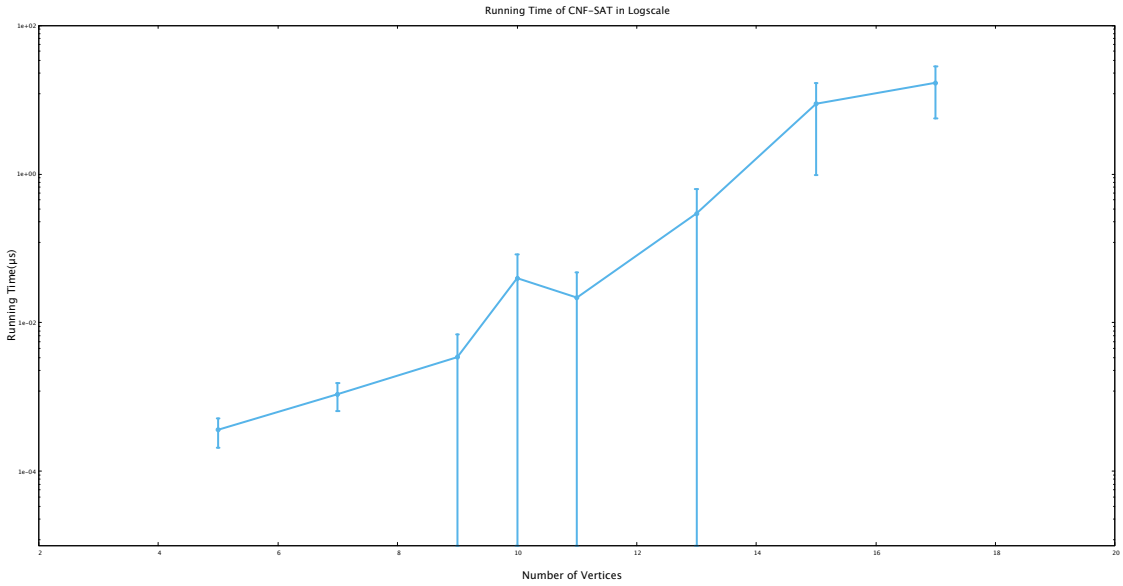

Figure 2: Logscale Running Time of CNF-SAT with vertices [5, 17]

minimum vertex cover problem.

As for the unusual running time for vertex 10, when I check the data at the vertex number 10, I get that there are two unusual data whose running time is 0.11s and 0.09s. These two abnormal data increase the mean running time at vertex number 10, causing the fluctuation in Figure 2; furthermore, if we remove these two abnormal data, we can get a steady increase trend.

2.3 Running Time Analysis of APPROX-VC-1 and APPROX-VC-2

Both APPROX-VC-1 and APPROX-VC-2 have an inclining trend, while the standard deviation of APPROX-VC-1 is much higher than that of APPROX-VC-2, especially after 10; therefore, the running time of APPROX-VC-1 algorithm is much more stable than the APPROX-VC-2 algorithm, and the APPROX-VC-1 algorithm is a time-consuming algorithm compared with APPROX-VC-2.

We have known graphGen generate more edges when the vertex number increases. APPROX-VC-1 algorithm greedily picks the highest degree vertex and removes the edges incident on it until no edges remain. As for the reason why APPROX-VC-1 almost have increasing trend, when the total

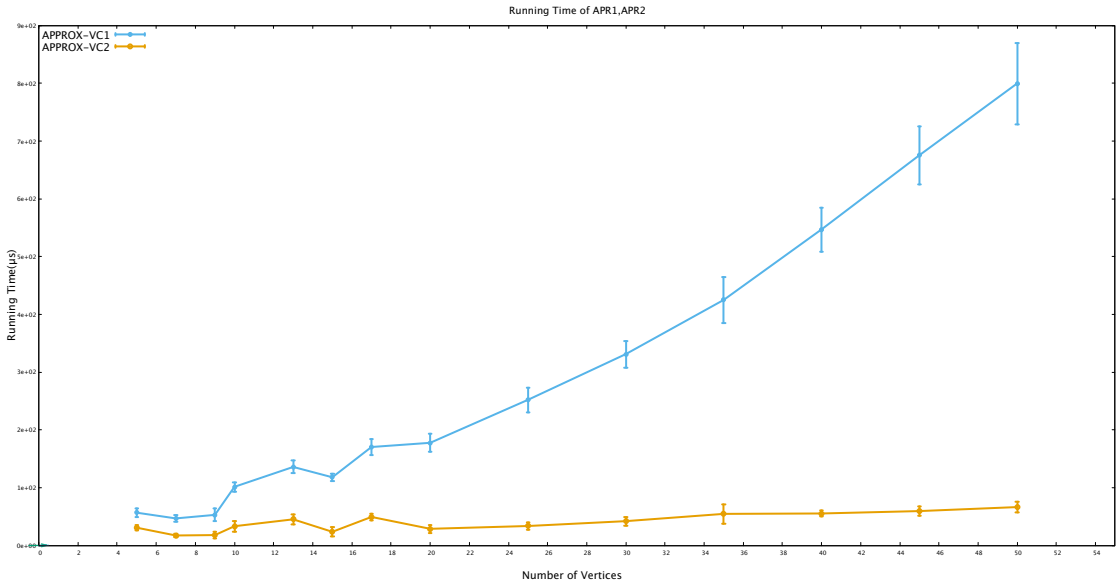

Figure 3: running time of APPROX-VC-1, APPROX-VC-2 with vertices [5, 50]

edges number increase, it will take more time to remove all the edges in the graph. In other words, this algorithm will loop more times to remove the edges in edge list.

APPROX-VC-2 algorithm picks the first edge of the input edge list firstly. As the vertex number increases, it also experiences a slightly increasing trend compared with APPROX-VC-1. The reason for this increase is similar with that for APPROX-VC-1; however, this less dramatic increasing trend for APPROX-VC-2 algorithm is because each loop of APPROX-VC-2 do not have to calculate which vertex has the highest degree like APPROX-VC-1. And this calculate process changes a lot when the number of vertexes increases. Thus, APPROX-VC-2 increases slower compared with APPROX-VC-1. And our APPROX-VC-2 algorithm does not have those random components and does not have to calculate the highest degree vertex; therefore, APPROX-VC-2 cost less time.

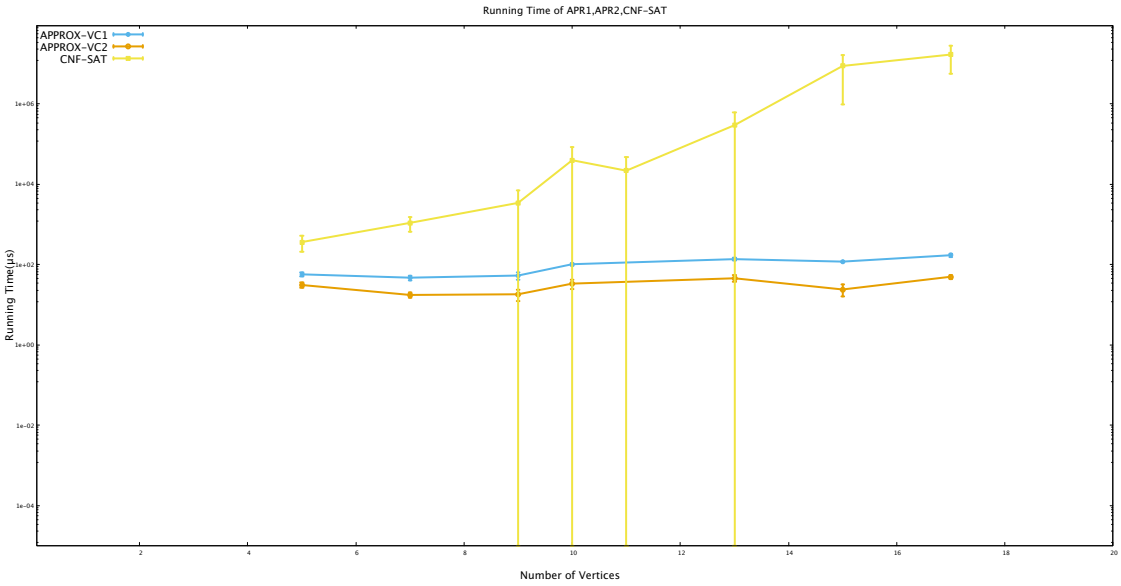

Figure 4: running time of CNF-SAT, APPROX-VC-1, APPROX-VC-2 with vertices [5, 17] in increments of 2 in logscale

2.4 Running Time Analysis of CNF-SAT, APPROX-VC-1 and APPROX-VC-2

From Figure 4, we can get that APPROX-VC-1 and APPROX-VC-2 are much more time-saving. The CNF-SAT is reduced from SAT problem and examines the instance for CNF-SAT takes polynomial time; therefore CNF-SAT is a NP-complete problem. Although CNF-SAT algorithm takes polynomial time $O(n^a)$ the index of CNF-SAT has a higher index "a" compared with APPROX-VC-1 and APPROX-VC-2 because the increase of total clauses with the increase of —V— contributes much more running time.

Seen from Figure 4, the standard deviation of CNF-SAT is also much higher than that of APPROX-VC-1, APPROX-VC-2, thus CNF-SAT algorithm is the most unstable of these three algorithms.

3 Analysis of Approximation Ratio

3.1 Background

According to ece650project.pdf, the approximation ratio is the ratio of the size of computed vertex cover to the size of the optimal vertex cover. We can easily get that the approximation is larger or equal to 1 and while the approximation ratio of the algorithm is higher, the algorithm is less accurate. In this project, the result of the CNF-SAT is considered as the optimal vertex cover. When our group run all the graph generated, we also record the vertex cover output for each graph and for each run.

3.2 Approximation Ratio of Analysis of APPROX-VC-1 and APPROX-VC-2

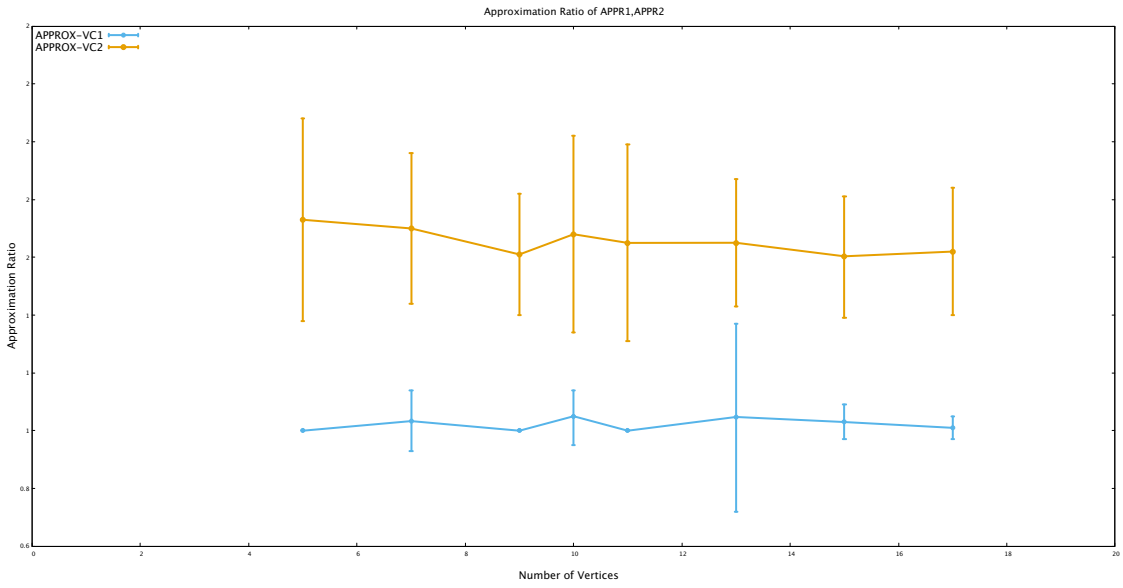

Figure 5: Approximation Ratio of APPROX-VC-1 and APPROX-VC-2

Comparing with APPROX-VC-2 algorithm, the APPROX-VC-1 has a lower approximation ratio in Figure 5, which means that APPROX-VC-1 can get a better vertex cover result. The APPROX-VC-2 has a higher standard deviation almost at each —V—. Although APPROX-VC-2 has a

higher standard deviation, the lowest approximation ratio of APPROX-VC-2 is still higher than the highest approximation ratio of APPROX-VC-1.

Because the vertex with the highest degree is most likely in the minimum vertex cover list, the APPROX-VC-1 always get the approximation rate at 1. At some special case, the approximation rate is a little higher than 1, which means that selecting vertex with the highest degree cannot get the optimal vertex cover list.

Because we pick the first edge in edge list which can mean that we randomly pick an edge in edge list, we are more likely to choose the vertexes that are not in the optimal vertex cover list. This is the main reason that contributes the bad result of approximation ratio.

4 Conclusion

From the analysis of running time and approximation ratio, the CNF-SAT can get the best result of vertex cover list, but the running time and standard deviation are much higher than those of APPROX-VC-1 and APPROX-VC-1. When comes to the result of APPROX-VC-1, APPROX-VC-1 can get a better result at the expense of higher running time and higher standard deviation. As for APPROX-VC-2, it has the lowest running time and lowest standard deviation with the worst vertex cover result.