ECE650, Fall 2020, Final Course Project

Xun Lu, 20820321 & Zhijie Wang, 20856733

Dec 1, 2020

1 Introduction

The project is meant to conduct a comparison and evaluation of performance of 3 different algorithms based on the minimum vertex cover problem and C++ implementation based on assignment 4 is utilized to evaluate it. The efficiency can be characterized in running time and approximation ratio with respect to different number of vertices range from 5 to 50 by running different graphs generated by /home/agurfink/ece650/graphGen/graphGen on ecelinux. This report consists of two parts. Firstly, a brief introduction to three minimum vertex cover algorithms will be given.Secondly, the analysis on performance will be demonstrated with figures.

2 Related Algorithms

In this section, we propose a brief review of three algorithms used in this project.

2.1 CNF-SAT-VC

This is an optimal (minimized sized) vertex cover algorithm that reduce the vertex cover problem to CNF-SAT problem in polynomial time. The vertex cover problem[1] is:

Given a graph $G =< V, E >$ and an integer k, does G have a vertex cover of size k?

And the CNF-SAT problem[1] is :

Given input a boolean formula in propositional logic that is in CNF, is it satisfiable?

The input graph is obtained from the vertex cover problem and a formula F is produced according to the encoding rules in [2] with the property that if and only if F is satisfiable, the graph has a vertex cover of size k. After that, an SAT solver is used to make verification of the satisfiability of the formula F.

2.2 APPROX-VC-1

This is an approximate vertex cover algorithm which widely picks a vertex of highest degree and removes all the edges incident on that vertex. Such process is repeated till no edge remains there. The time complexity is characterized as $O(|V| + |E|)$. This greedy algorithm will result in a vertex cover with size at most $log(|V|)$ times as the optimal solution[1].

2.3 APPROX-VC-2

This is an approximate vertex cover algorithm as well which randomly picks an edge and adds both vertices of the edge to vertex cover, and then removes all the edges incident on those vertices. Similarly, the process like this is repeated if there is any edges remaining. The time complexity is characterized as $O(|V| + |E|)$. This algorithm leads to a vertex cover with size at most twice as the optimal solution for the reason that at least one vertex of an edge should be in vertex cover[1].

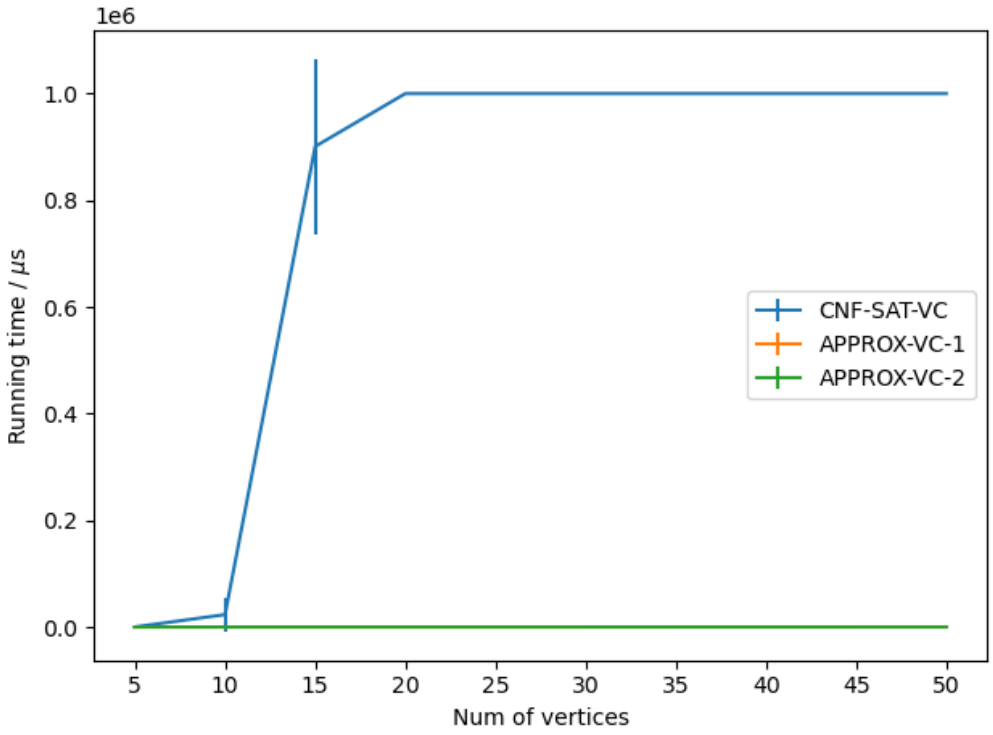

Figure 1: Running time of 3 algorithms v.s number of vertices

3 Experiments Setup

In order to evaluate the average running time and approximation ratio, we use **graphGen** to generate 11 graphs for each $|V| = 5, 10, \ldots 50$, and for each graph, we run 10 times. The graphs generated by **graphGen** and used in our project is saved as `output_graph` in our git repository. In each run, the running time of each algorithm is calculated by its thread time, and the approximation ratio of

two approximate algorithms are calculate by their output vertex size divided by the output vertex size of **CNF-SAT-VC**, which is guaranteed to be optimal. Note that we also generate 11 graphs for each $|V| = 5, 7, 9 \ldots 17$ in output_graph2, to calculate the approximation ratio of two approximate algorithms because **CNF-SAT-VC** would timeout when $|V|$ is greater than 17.

4 Results Analysis

In this section, we analysis three algorithms through: 1) running time, and 2) approximation ratio.

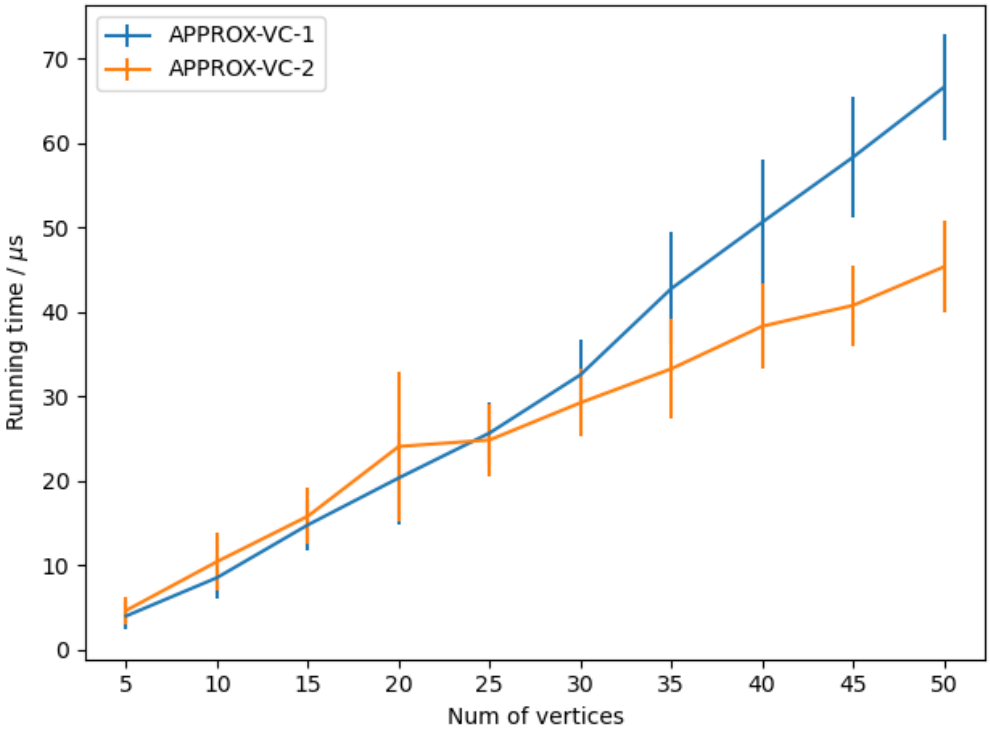

Figure 2: Running time of 2 approximate algorithms v.s number of vertices

4.1 Running Time

Figure 1 shows the running time of three algorithms, as we can see, the running time of **CNF-SAT-VC** is much higher than other two approximate algorithms, and go to timeout (1 second) when the number of vertices is bigger or equal to 20. (Note that the running time of two approximate algorithms are much lower, thus theirs plots are nearly overlapped in this Figure.) In **CNF-SAT-VC**, the number of clauses used in reduction is $k+n\binom{k}{2}+k\binom{n}{2}+|E|$, thus, in the worst case suppose the minimum vertex cover has a size of $|V|$, then the total running time would be $O(|V|^2)+O(|V|^3)+O(|V|^3)+O(|V|\times|E|)$. Besides, the running time of **Minisat** is also not polynomial since **CNF-SAT** problem is **NP-Complete**, thus the running time would grow exponentially with the increasing size of input.

Figure 2 shows the running time of two approximate algorithms, as we can see, the running time

of both two algorithms increase with the number of vertices, whereas when the number of vertices is greater than 25, the running time of **APPROX-VC-1** is longer than **APPROX-VC-2**, this might because **APPROX-VC-1** need to sort the vertices by their degrees, hence, more vertices would lead to a large running time comparing to **APPROX-VC-2**.

Figure 2 also shows that the running time of **APPROX-VC-1** and **APPROX-VC-2** both grow nearly linearly, which is $O(|V| + |E|)$ in the worst case.

4.2 Approximation Ratio

As we have discussed in Section 2, the upper-bound of the approximation ratio of **APPROX-VC-1** is $\log(|V|)$, which is $2|V|$ in **APPROX-VC-2**. Figure 3 shows the approximation ratio of two approximate algorithms in our experiments, since **CNF-SAT-VC** timeouts after 20, so we can only calculate the approximation ratio when $|V| = 5, 10, 15$. As we can see, the greedy one **APPROX-VC-1** is nearly optimal, while the random algorithm **APPROX-VC-2** is much higher, which proves that the analysis in Section 2 is correct.

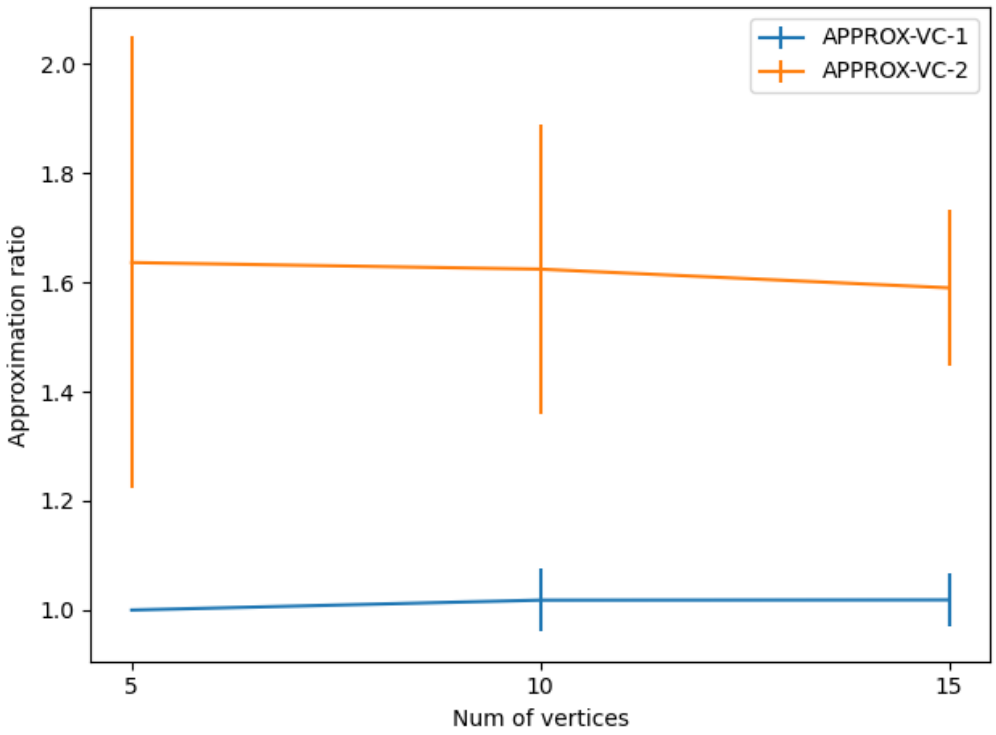

Figure 3: Approximation ratio of 2 approximate algorithms v.s number of vertices

To better illustrate the relationship between approximation ratio and number of vertices while considering about the running time of **CNF-SAT-VC**, Figure 4 shows a more detailed plot of approximation ratio, in which the interval of number of vertices was reduce to 2. As we can see, in most cases **APPROX-VC-1** have an optimal results which reflects by a zero standard deviation (therefore they don't have error-bar plots).

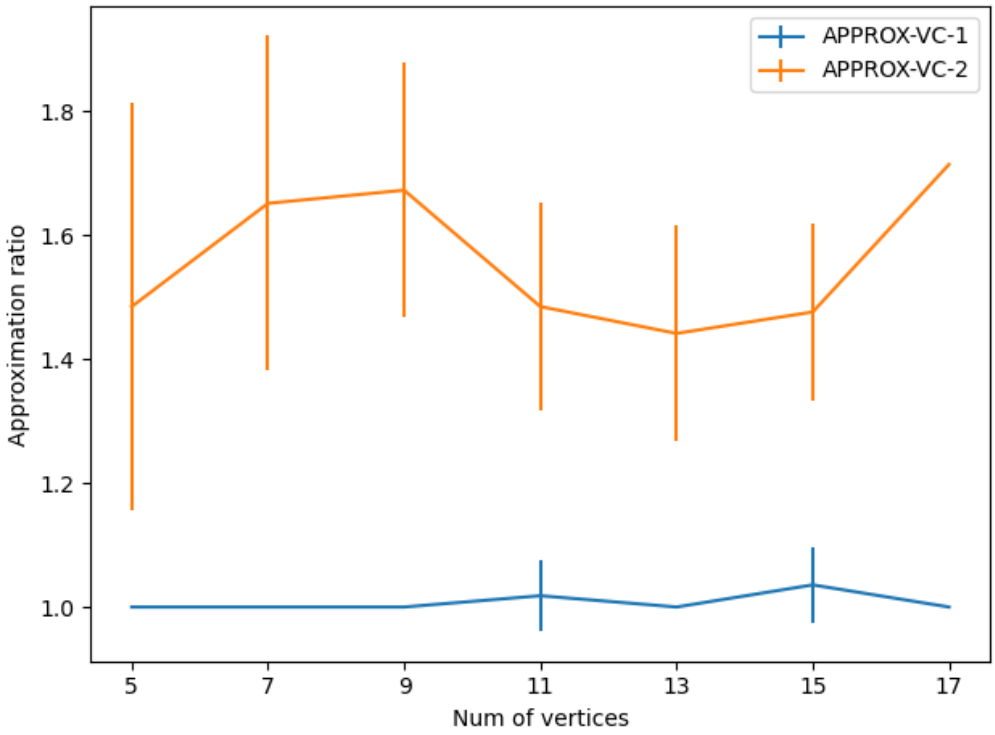

Figure 4: Approximation ratio of 2 approximate algorithms v.s number of vertices

5 Conclusion

In this project, we explored the comparison of different algorithms to solve vertex cover problem. In conclusion, the reduction based method **CNF-SAT-VC** could give an optimal solution, however, it's easy to timeout when input graph has a large number of vertices. Two approximate algorithms, **APPROX-VC-1** and **APPROX-VC-2** have much lower running time, **APPROX-VC-2** is slower than another when input graph has a large number of vertices but can provide a nearly optimal solution.

References

[1] Thomas H Cormen et al. *Introduction to algorithms*. MIT press, 2009.

[2] Arie Gurfinkel. *A Polynomial-Time Reduction from VERTEX-COVER to CNF-SAT*. URL: `https://git.uwaterloo.ca/ece650-1209/pdfs/-/blob/master/ece650.a4.encoding.pdf`.