Report of Analysis of MIN-VERTEX-COVER Problem

Shi Cheng, Yijun Pan
November 29, 2017

1 INTRODUCTION

Vertex cover of a graph is a set of vertexes such that each edges of the graph is incident to at least one vertex of the set. In this project, we use three different algorithms (CNF-SAT-VC, APPROX-VC-1, and APPROX-VC-2) that solving the vertex cover problem. The algorithm CNF-SAT-VC create a polynomial reduction of the decision version of VERTEX COVER to CNF-SAT. Then we add other two additional ways to solve MIN-VERTEX-COVER, APPROX-VC-1 and APPROX-VC-2 respectively. APPR1 just picks a vertex of highest degree and add it to the vertex cover, meanwhile, throw away all edges incident on that vertex and repeat till no edges remain. APPR2 is an algorithm which pick an edge randomly and delete all edges which attach to the previous picked one until no edges left.

2 ANALYSIS

2.1 ANALYSIS FOR RPROGRM

Different algorithms have different efficiency to solve the Vertex Cover problem. In this part, we will analyze the difference between three algorithms. As the report claimed before, efficiency will be evaluated by two aspects: 1) running time, various values of |V| were taken into consideration. We do this by generating graphs for |V|$\in$ [3,18] using graphGen, in increments of 3. That is, graphs with 3,6,9,12,15 and 18 vertices. 2) approximation ratio, we characterize the approximation ratio as the ratio of the size of the computed vertex cover to the size of an optimal (minimum-sized) vertex cover. That is $\frac{size\ of\ computed\ vertex\ cover}{size\ of\ optimal\ vertex\ cover}$. To guarantee the accuracy of running time, 10 graphs for each value of |V| were generated then 10 runs of each graph were tested to get the running time for each run. For measuring approximation, the output of CNF-SAT-VC, which is guaranteed to be optimal, was used as the standard.

The code was made multithreaded since the outputs of three different approaches were generated at the same time. There are four threads, one for I/O, and one each of the three others for the different approaches respectively. The main thread I/O takes responsibility for getting input from graphGen and outputting the result for each approach.

2.2 ANALYSIS FOR RUNNING TIME

2.2.1 ALL THREE ALGORITHMS

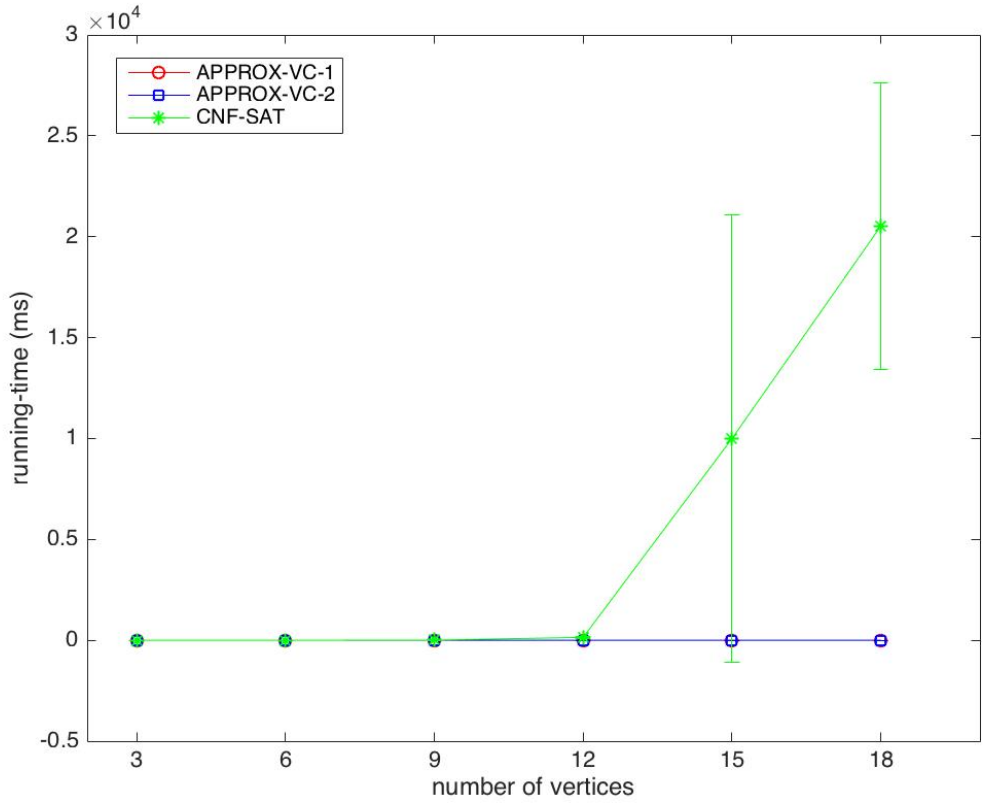

Figure2.1: Running Time of SAT, APPROX-VC-1, APPROX-VC-2

As is clearly shown in the Figure2.1, SAT algorithm has the most running time among the three algorithms. and there is a small gap between three algorithms before the |V| of 12. In addition, a dramatic exponential increase happened after the |V| of 12, meanwhile, the difference become quite bigger between CNF-SAT and other two. In contrast, running time of algorithm APPROX-VC-1 and algorithm APPROX-VC-2 maintain on a steady level all the time.

For CNF-SAT algorithm, it has to traverse the whole solution space and tried all possible assignments of the given CNF in order to satisfied every clause to find the minimum-sized vertex cover. For other two algorithms, their time complexity is $O(V^2)$ and O(V) respectively, and this is the reason why the running time of them is steady.

2.2.2 CNF-SAT

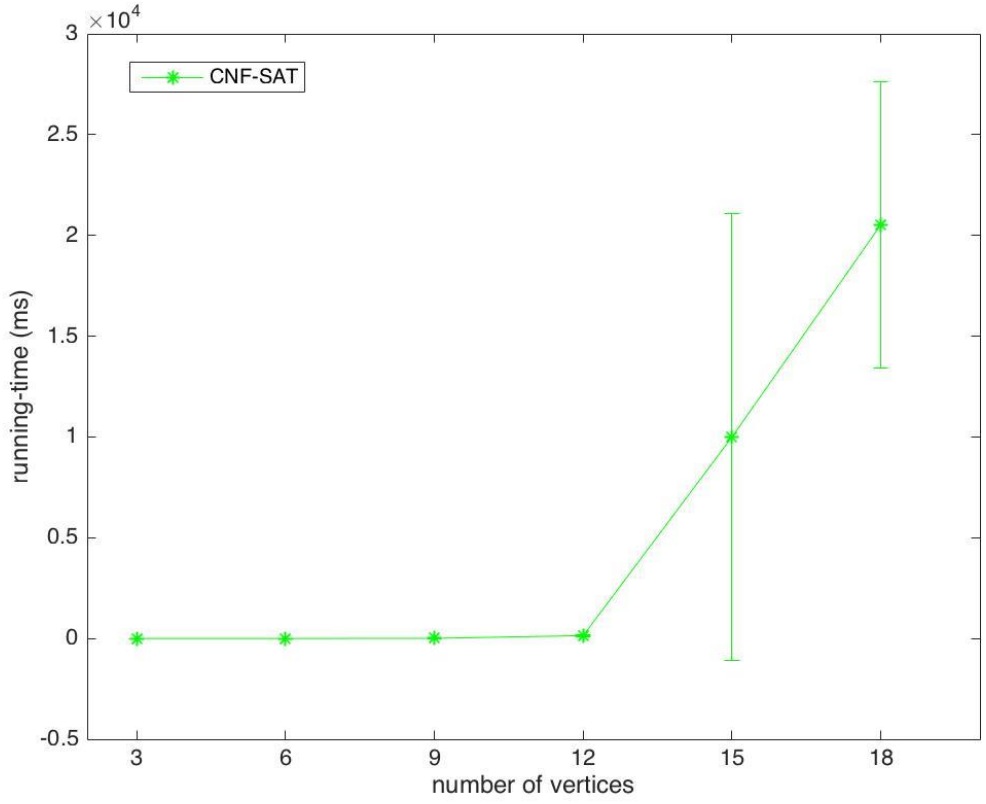

Figure2.2: Running Time of SAT

As we can see in Figure2.2, there is a dramatic exponential increase of running time between |V| of 12 and 18 and 12 is the turning point. Before it, the machine calculates the algorithm at a very fast way. After that, the running time grew at a considerable speed and it peaked at |V| of 18 which is 2.1×10^4 ms. There is no doubt that the running time will continue to rise after 18, but it will take several hours to compute when the graphs get complex. The reason is the polynomial-time reduction which is conducted in the approach has a strong relationship with the number of vertices of a graph. As is shown in the assignment 4, the clause number of SAT algorithm is $k + n\binom{k}{2} + k\binom{n}{2} + |E|$, in which k is the size of vertex cover that was tried in the minisat loop, n is the actual vertex number that was given. As we could see in the formula, the number of clauses rises in exponential form with the loop of k. And k rise with n, thus when |V| is higher, number of clauses is higher, time of solving the minimum vertex cover is higher.

To rise the veracity of the result, we not only compute the average running time of each |V|, but also calculate the standard deviation. It can be seen that after the |V| of 12, the standard deviation of SAT running time become much bigger. There is a phenomenon that the size of minimum vertex

cover would be quite different for different edges and a settled |V|. This is why the standard deviation increase with the rise of |V|.

2.2.3 APPROX-VC-1 AND APPROX-VC-2

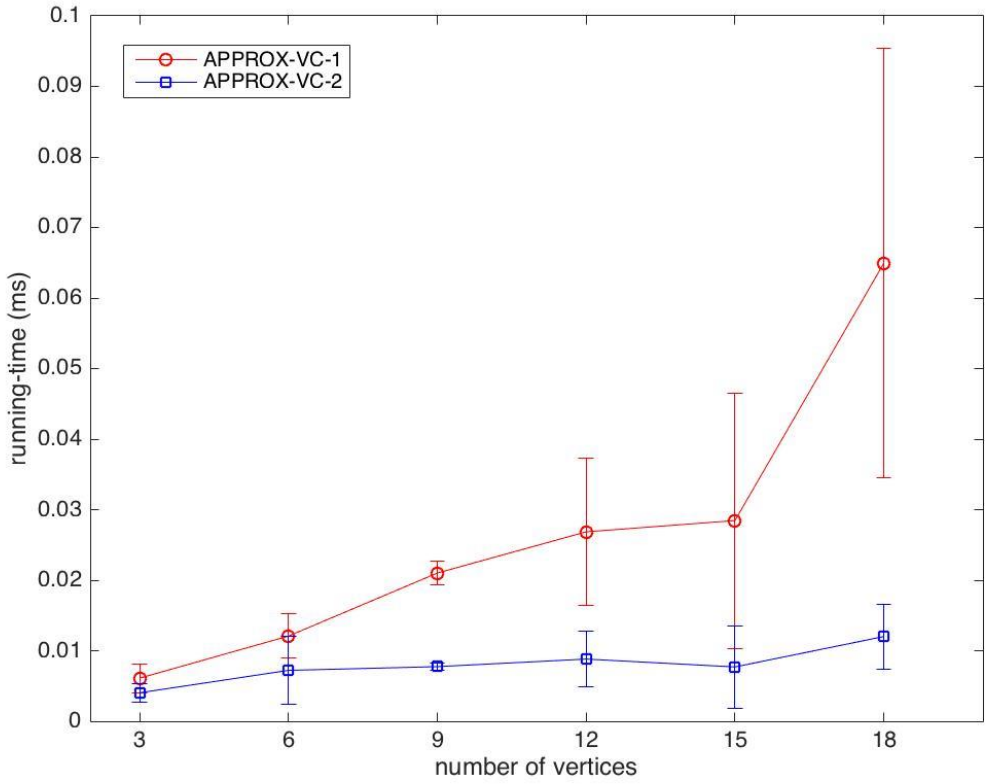

Figure2.3: Running Time of APPROX-VC-1 and APPROX-VC-2

After the experiment was conducted, we found that the running time of APPROX-VC-2 is much lower than APPROX-VC-1, and although running time of these two algorithms increase with the increasing number of vertices, increasing rate of APPROX-VC-2 is much slower than APPROX-VC-1. The main reason is that the algorithm in APPROX-VC-1 uses two for loop to pick edges and therefore its time complexity is $O(V^2)$, while the algorithm in APPROX-VC-2 is linear when picking vertex and therefore its complexity is O(V).

The standard deviation is almost like the comparison of the running time of them. APPROX-VC-2 has much bigger standard deviation after 12 vertices than APPROX-VC-1, and the reason is that with rising number of vertex, the graph is getting more and more complex and therefore, the running time fluctuates more greatly.

In all, from the perspective of running time, APPROX-VC-2 is much more stable and faster than others.

3. APPROXIMATION RATIO ANALYSIS

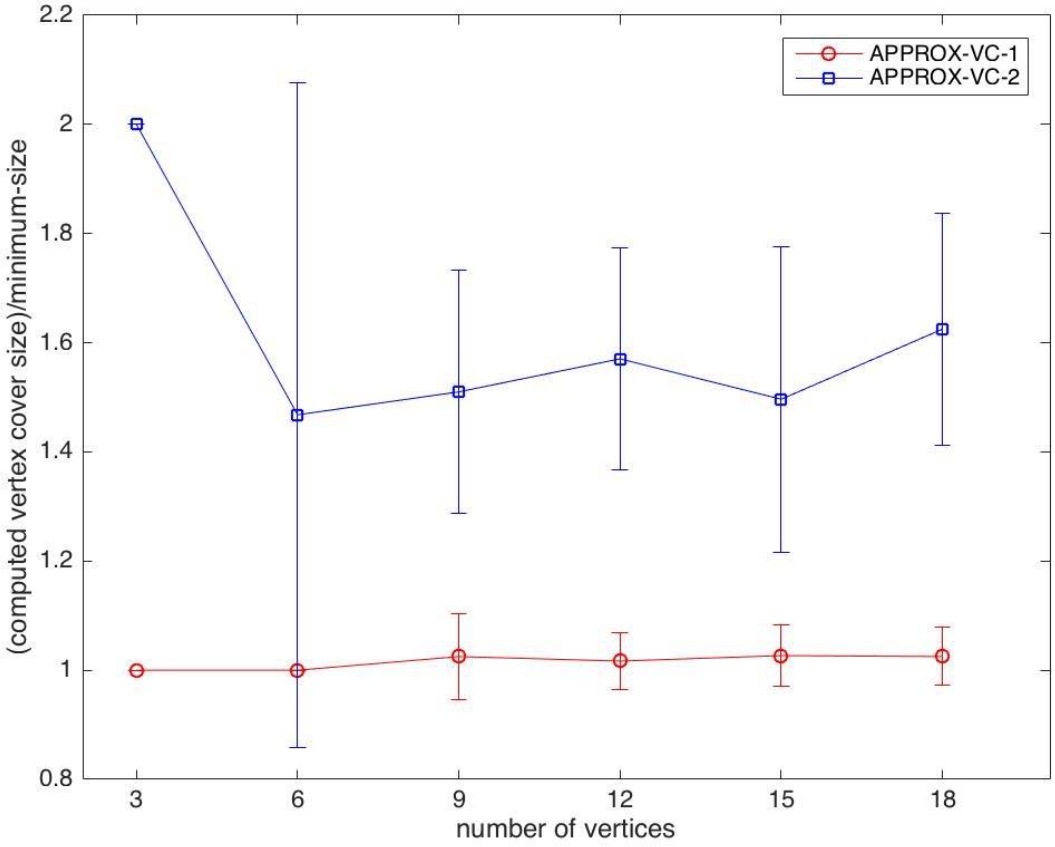

Figure 3.1: approximation ratio plot, generated using MATLAB.

The approximation ratio formula is $\frac{computed\ vertex\ cover\ size}{minimum\ vertex\ cover\ size}$.

From the above formula, the result of CNF-SAT is the minimum-sized vertex cover, so if the ratio is lower, the algorithm will be better. And if we only pay attention to the approximation ratio, algorithm VC-1 seems to be better.

In these two algorithms, we didn't pick edge randomly, so for each graph, we can always get the same result when repeating running the same graph.

3.1 Algorithm VC-1

For algorithm VC-1, we always pick a vertex of highest degree first, until no edges remain. As shown in Figure 3.1, red line shows the trend of the approximation ratio in VC-1. It is almost a straight line with slight fluctuation. And in the most cases, this algorithm can always return the right result, which is the minimum-sized vertex cover, and the standard deviation is 0. But sometimes it would be wrong, and the standard deviation will be non-zero. It is because that algorithm VC-1 is a greedy algorithm, but greedy algorithm is not always optimal.

When the vertex size is small, the standard deviation is not visible, with size increasing, the standard deviation is apparent.

3.2 Algorithm VC-2

For algorithm VC-2, it is easier. We just repeatedly pick an edge and throw both endpoints into the cover. Throw the vertices and its adjacent edges out of graph, and continue.

For this algorithm, it is to show that VC-2 uses at most twice as many vertices as the optimal vertex cover. This is because each edge that gets chosen must have one of its endpoints in the cover; hence in the worst case, we may throw double points into the cover.

According to the figure, the approximation ratio fluctuated moderately. Especially when V = 3, it is the worst case, because we always have to pick 2 points according to VC-2, when the optimal size is 1. And when the size is increasing, the ratio is always changed and the standard deviation is much more apparent. In short, it always returns more vertices than the optimal solution.

In conclusion, from all cases we have tested, CNF-SAT can always return the minimum-sized vertex cover. In most cases, algorithm VC-1 could return optimal result. But only several VC-2 results are optimal.

4 SUMMARY

All in all, for running time, algorithm VC-2 would be better, but for approximation ratio, VC-1 be better. Last but not least, SAT can always output the optimal vertex cover with much longer running time.