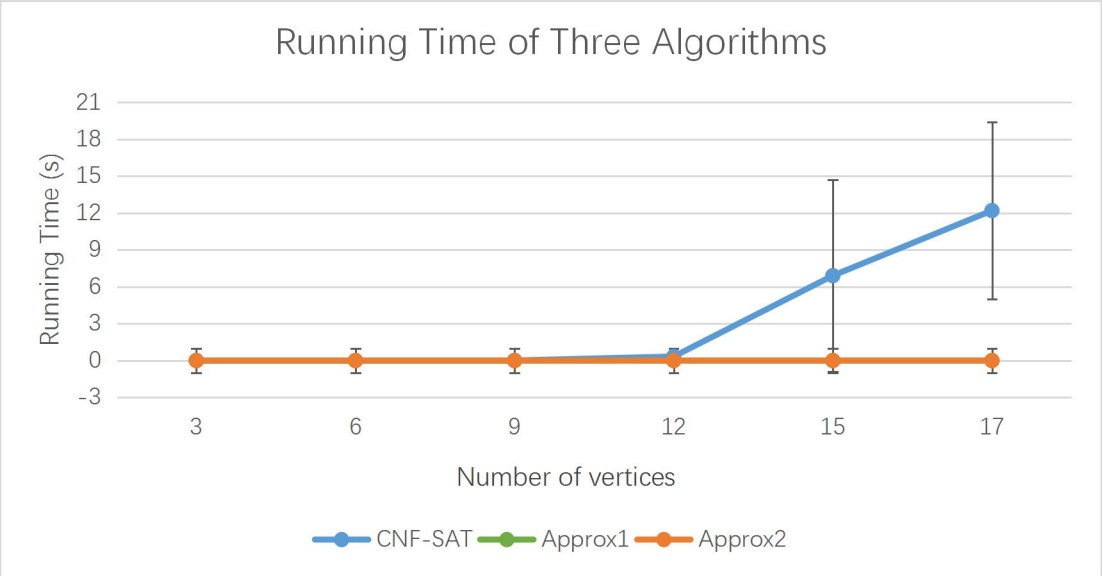
Analysis of Vertex Cover Algorithms

1 Introduction

In order to solve the vertex cover problem, we apply three distinct algorithms and compare the efficiency of them. The first one is CNF-SAT-VC, which uses conjunctive normal forms to find mini-vertex cover of a graph. The second one is called APPROX-VC1 calculating the vertex cover by picking highest degree vertices. The last one called APPROX2-VC2 find a vertex cover through randomly picking pairs of vertices. The line charts in this report show the algorithms' performances based on running time and approximation ratio. By analyzing the data and the algorithms, we give the possible reasons to interpret the reasons of the trends in charts and find out the advantages and disadvantages of each approach.

2 Analysis

2.1 Analysis for running time

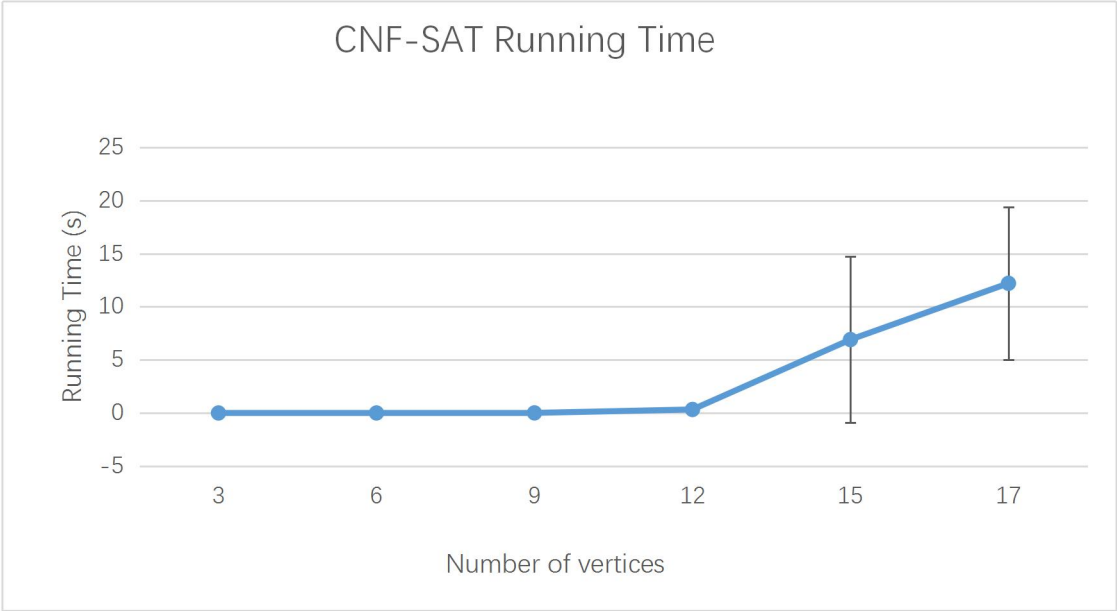


Graph-1 running time of three algorithm

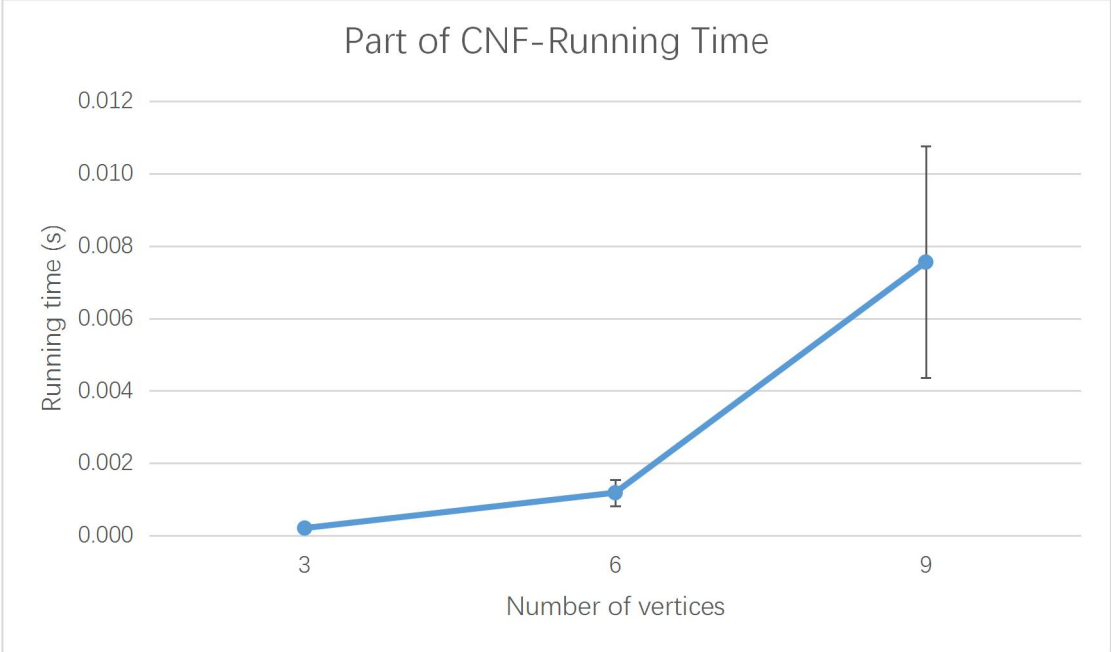
It can be seen from **Graph-1** that running time of CNF-SAT-VC, Approx1 and Approx2 are different in magnitude. It is obvious that the running time of the approach of CNF-SAT-VC is rather longer than those of Approx1 and Approx2, so we will analyze them separately by generating two graphs for CNF and one graph for the latter two algorithm relatively. In **Graph-2**, we use "second" unit to describe the trend here, and we

use “microsecond” unit for **Graph-4**.

In **Graph-1,** the reason why the running time of CNF-SAT-VC is longer is that the former algorithm uses the library of minisat, and it takes long time to calculate the clauses in the CNF formula.



Graph-2 CNF-SAT-VC running time



Graph-3 CNF-SAT running time when V=3,6,9

The error bars of V=3, V=6, and V=9 are so small that they are invisible in the graph, so we generate part of the graph, i.e. **Graph-3** alone so that we can see the error bars clearly and it is easier for us to analyze the standard deviation.

We can see as the increase of the number of vertices, the running time increases and the standard deviation also increases.

When the number of vertices is fewer than 12, the running time is small and the standard deviation is also small. When the number of vertices is more then 12, the running time surges dramatically and there are fluctuations between running time of each graph in the same the number of vertices and cause that the standard deviation can be quite large.

For the approach of CNF-SAT-VC, we use reduction to calculate the vertex cover and the number which need to be added in the solver is related to the size of k, that is, the size of the vertex cover. The number of clauses in the reduction is as follows:

$$k + n \times \binom{k}{2} + k \times \binom{n}{2} + |E|$$

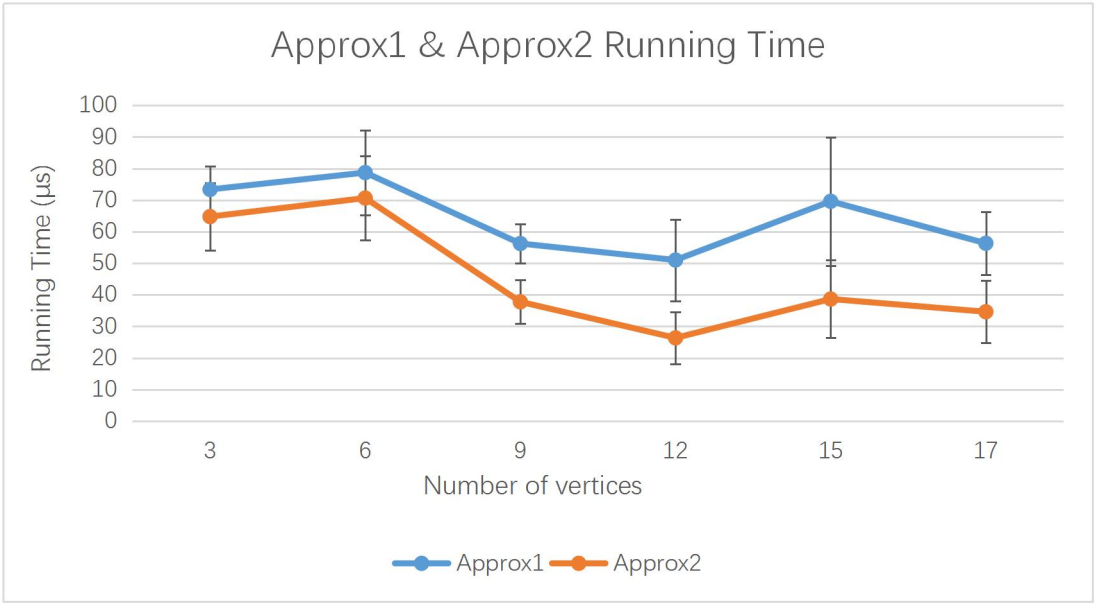
The reason why the running time increases is that when the number of vertices increases, the number of edges in the set E also increases, which causes the number of clauses that add into the solver of Minisat increases as well. In other words, the solver need to calculate more clauses, so the running time increases.

The reason why the standard deviation increases is analyzed as follow.

The more a graph is connected, the smaller the size of minimum vertex cover is. So when the number of vertexes and the number of edges are same in different graphs, the size of minimum vertex cover for weakly connected graph is bigger than strongly connected graph.

Amplitudes in the running time are bigger, because although graphs have the same number of vertices, graphs generated by the graphGen in the Linux are different and they are connected in different degree, so that the sizes of minimum vertex cover may also be different. The searching method for k in the algorithm we use in the project is to increase the size of k from 1 to |V|-1. When the algorithm finds a k-sized vertex cover, the algorithm terminates and output the minimum vertex cover. When k increases, after execution of the searching for (k-1)-sized vertex cover, the procedure need to add clauses into the solver of the minisat library for searching for k-sized vertex cover, so the total running time is longer. So when the number of vertices is same in different graphs, the running time may be very different because of the size of k so that the standard deviation is bigger.

Also, the standard deviation seems larger when running time becomes longer, because the absolute value of running time increases and the standard deviations are too small to be observed when V=3,6,9.



Graph-4 Approx1 and Approx2 running time

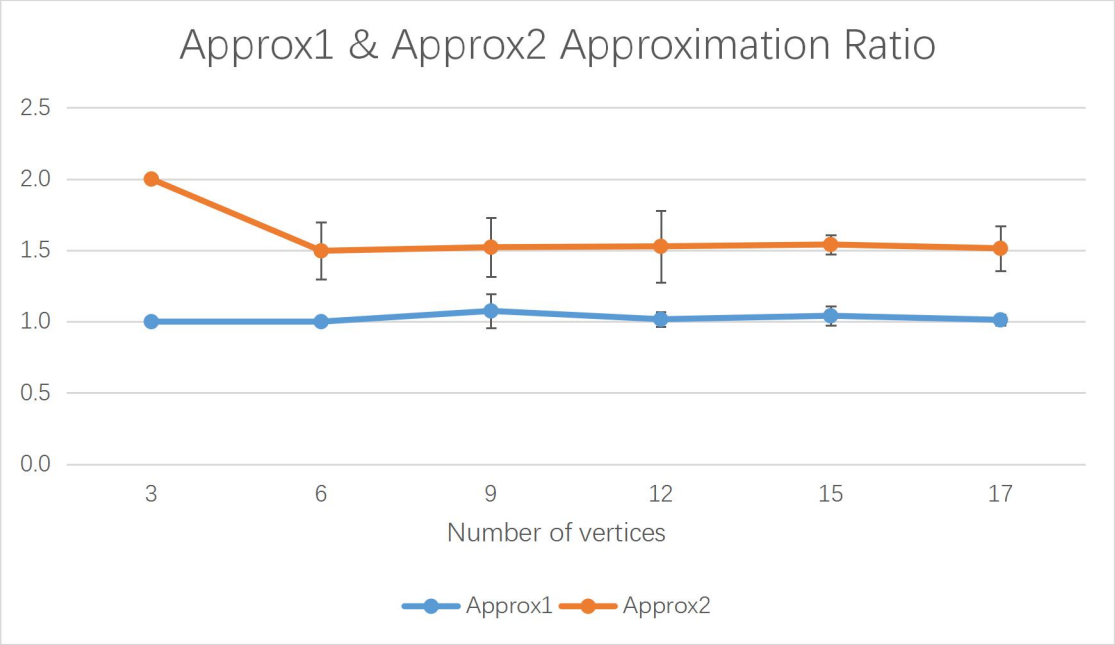
The algorithm for Approx1 is to traverse every entry in the adjacent matrix and find the vertice which has the maximum degree from these V vertices in the graph. Then check if the maximum degree is 0. If the maximum degree is not 0, then pick it and change the entry incident on that vertice to 0 in the matrix. Then we continue traversing the entries in the matrix. The running time complexity of the algorithm is $O(V^3)$.

The method of the Approx2 is to traverse every entry in the matrix. If the entry is 1, it shows that there is an edge between the index of row and column. Then pick the both indexes in the set of the vertex cover and change this entry to 0. when traversing the entry in the matrix, every entry is 0 then the algorithm terminates. The running time complexity of the algorithm is $O(V^3)$ as well.

Both algorithms' running time increase when the numbers of vertices increase because both algorithm are in polynomial time.

The running time is in microseconds, so when the thread runs, it is easily to be disturbed by the CPU interference factors in the Linux. This is the reason why the running time of every execution is different and standard deviation exits.

2.2 Analysis for approximation ratio



Graph-5 Approx1 and Approx2 approximation ratio

The output of the approach of CNF-SAT-VC is exactly an optimal (minimum-sized) vertex cover.

The **Graph-5** shows that the output of Approx1 is almost the minimum vertex cover.

The algorithm of Approx2 picks random edges, so every execution of Approx2 may output different vertex cover and the sizes of the vertex cover are also different. Sometimes the output is minimum vertex cover, but most time its output is not minimum vertex cover.

From the **Graph-5**, we can see that there is a spike in the line of Approx2 when V=3. The reason may be that when given a graph with 3 vertices, the minimum vertex cover is either 1 and 2. We random pick an edge in the algorithm of Approx2 and add both vertices into the set the vertex cover. So when V is equal to 3, the size of the output of Approx2 is likely to 2("likely" means when the E is empty, the output is nothing.) So when E is not empty, the approximation ratio of the vertex cover of approx2 is more likely to be 2.

3 Conclusion

In conclusion, these three algorithms differ from the running time and the approximation ratio. The running time of Approx1 and Approx2 are almost equal and both of them are much smaller than the CNF-SAT-VC. Approximation ratio of Approx1 shows better performance than Approx2 when calculating the mini-vertex cover, and the outputs of both algorithm are not guaranteed to be the minimum vertex cover.