METHODS AND TOOLS FOR SOFTWARE ENGINEERING

UNIVERSITY OF WATERLOO

DEPARTMENT OF ELECTRICAL AND ELECTRONICS ENGINEERING

Quantitative Analysis of Mini-Vertex Cover Using Various Algorithms

Authors:
Student 1 Yunyi Qiu(ID: 20739473)
Student 2 Keqi Shu (ID: 20803058)

Date: April 20, 2021

Contents

1 Introduction

In this project, different techniques and algorithms have been used to solve the parameterized Vertex Cover problem, and the corresponding results and performance of different algorithms are presented in the following report. The first approach is to encode vertex cover to CNF-SAT clauses in a graph. Then MiniSat solver have been used to solve the problem in a more effective way. Also two alternative approximation algorithms which is Approx-VC-1 and Approx-VC-2 have been used to calculate the vertex cover for comparison. More detailed description of the algorithms are explained in further details in the following sections.

2 Algorithms

2.1 CNF-SAT

The CNF-SAT problem is a boolean satisfiability problem, which the reduction of the vertex cover of this sort is a polynomial-time reduction issue. The encoding method of this issue is A4 which form the clauses. Different literals that are conjunction of dis- junction forms the CNF. These clauses are then send to the MiniSat SAT solver to compute the final result.

2.2 Approx-VC-1

The Approx-VC-1 is a extension or other option of CNF-SAT. Here, only the vertices that are happening more frequently given the edge sets are selected. Then this vertex is deleted from the list and all the relevant edges are also removed. This goes on until there are no more edges remaining in the edge list.

2.3 Approx-VC-2

Here, we select a random edge from the edge list, then we add this edge to the VC list, where we assum that the VC is the vertex cover. then all the edges corresponding with all the vertices on both ends of the edge is removed from the graph. This goes on until the remaining edge list is clear.
To solve the problem and enable the comparison, we used 4 threads in this program. The 4 threads are: I/O which is used in the main function for cooperation of the input and output. It reads the (E,V) from the user and provides it to the algorithms which provides proper outputs. The other three algorithms' thread (CNF-SAT, Approx-VC-1 and Approx-VC-2) are called by the main functions too and cooperates with the I/O thread.

3 Experiment and Analysis

3.1 Input data preparation

For better fairness of the evaluation of the algorithms. The input are generated through a graph generator called GraphGen which is in /home/agurfink/ece650/graphGen/graphGen on eceubuntu. This tool could generate different edges with same numbers of vertices. Thus, graphs corresponding with different numbers of vertices are generated from 5 vertex to 100 vertex with an increment of 5. For each number of vertex, 5 graphs are generated. Thus in this case, 100 graphs are generated.
However, through the experiment, we found that when the vertex exceed 15, the SAT algorithm could not find a proper path in the limited amount of time. Therefore, more graphs are generated for v ¡= 15. That is, we generated graphs from 5 to 15 with an increment of 2 for more thorough evaluation of different algorithms.

3.2 Running time of CNF-SAT-VC

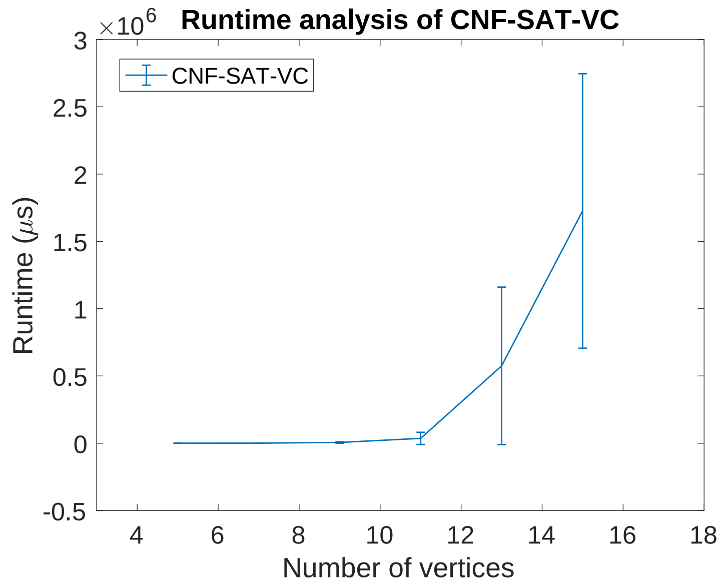

Figure 1: Running time CNF-SAT for the range of vertices [5, 15].

The result of running time of CNF-SAT-VC with graph size from 5 to 15 is presented in Figure 1. The standard deviation of the result is also plotted as error bars. From the graph, we can observe that the computation time increases tremendously after the graph size reached to 15. The overall running time increases exponentially when larger graphs are inputted. This is sounded since larger graphs means that the clauses also increase exponentially, which makes the MiniSat solver very hard to find the proper result, which as a result, makes the time consumption increase exponentially too. The standard deviation also increased tremendously as larger graphs are being calculated.

3.3 Running time of APPROX-VC-1 and APPROX-VC-2

APPROX-VC-1 and APPROX-VC-2 support high speed calculation of VC problems, that is, it is able to find VC in limited amount of time for very large graphs up to V = 100. As shown in Figure 2, the analysed running time of APPROX-VC-1 and APPROX-VC-2 is presented. From the figure we can see that overall, the running time of both algorithms increase with the increment of the size of the graph, as well as the standard deviations. Also, the computation time of APPROX-VC-1 is always shorter that APPROX-VC-2. This is due to the heuristics used in APPROX-VC-1. In APPROX-VC-1, the vertex with the highest degree is selected and all the corresponding edge deleted, which the remaining edge is decreased in a quicker speed. Thus, the vertices are searched in a quicker manner. This lead to a quicker search speed. In short, the time complexity of Approx-VC-2 is higher than Approx-VC-1.

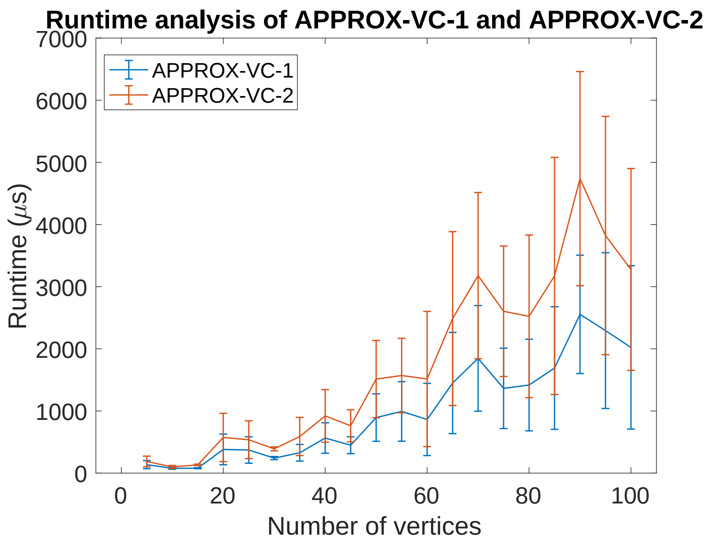

Figure 2: Running times of APPROX-VC-1 and APPROX-VC-2 for the vertices in range [5, 100].

Figure 3 showed the result of the runtime of APPROX-VC-1 and APPROX- VC-2 with a more careful evaluation. The result still shows that APPROX-VC-1 could provide shorter computation time comparing to APPROX-VC-2, and the overall trend of the computation time increases with larger graphs being calculated.

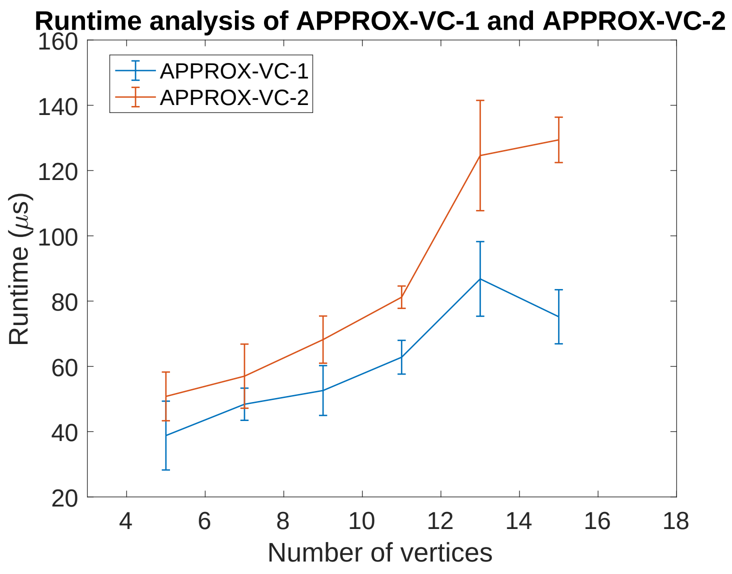

Figure 3: Running times of APPROX-VC-1 and APPROX-VC-2 for the vertices in range [5, 15].

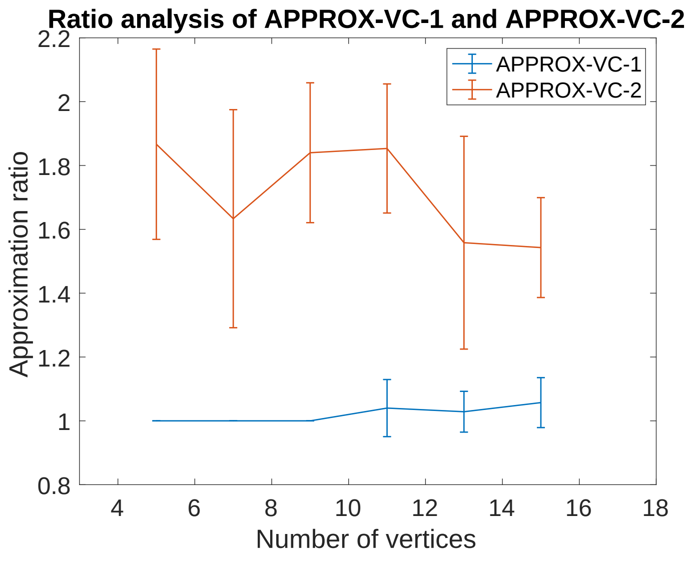

Figure 4: Approximation Ratios of APPROX-VC-1 /CNF-VC and APPROX-VC-2/CNF-VC

3.4 Approximation Ratio

Approximation ratios are also used to evaluate the efficiencies of the algorithms. Approximation ratios is defined as the vertex cover produced by APRROX-VC-1 and APPROX-VC-2 over the vertex cover size of CNF-SAT-VC. Since we assume that the CNF-SAT-VC is the optimal solution. Therefore, a smaller ratio means that the solution is closer to the

optimal result.
Figure 4 records the approximation ratios of the two algorithms. The results shown that the cover vertex generated by the APPROX-VC-1 is very close to the optimal solution CNF-SAT-VC, since all the values are close to 1. While APPROX-VC-2 is less optimal, since it generates almost 50% more of the vertices than the optimal solution. This is due to that this algorithm does not use heuristics, it only use randomness to find the cover vertex, which would lead to un-optimal results.

4 Conclusion

CNF-SAT-VC generates the best solution among all the algorithms, however, when the graph size increases, the computational time increases tremendously, in our case, when the number of vertices is greater than 15, the computational time encounters a steep increase. Therefore, in cases where larger graphs are used, more computational time is better to be allowed. However, we observed that in all the cases, when dealing with larger graphs, the time needed to find a solution increases.
In conclusion, if time is not a consideration, CNF-SAT-VC would be the best choice. If the time needed to find a solution is the highest concern, then APPROX-VC-2 would be your first choice. However, if time and optimal solution all needs to be considered, then APPROX-VC-1 would be the best choice. In most cases where these two goal is needed to be satisfied, APPROX-VC-1 seems to be the optimal solution to choose.