ECE650 - Methods and Tools for Software Engineering

University of Waterloo

Department of Electrical and Computer Engineering

Project : Analysis of Three Approaches to Vertex Cover Problem

Name:
Xin Yi (ID: 20834941)

Email:
x22yi@uwaterloo.ca

Date: December 3, 2019

1 Introduction

Vertex cover of a graph is a set of vertices such that each edge of the graph is incident to at least one vertex of the set. The problem of finding a minimum vertex cover is a classical optimization problem in computer science and is a typical example of an NP-hard optimization problem that has an approximation algorithm. In this report, it will be implemented by three different approaches, related methods and experimental details will be introduced in section 2 and section 3. And to analyze the efficiency of each approach, the running time and approximation ratio will be showed and discussed in section 4. Finally, in section 5, there will be a summary.

2 Related Methods

1.Algorithms

Here we introduce three solutions to solve minimum vertex cover problem:

CNF-SAT: We present a polynomial time reduction from VERTEX-COVER to CNF-SAT. A polynomial time reduction is an algorithm that runs in time polynomial in its input.And then we use a Minisat SAT solver to find the solution.

APPROX-1: (Step 1) First we find a vertex which has the highest degree. (Step 2) add it to the vertex cover and throw away all edges incident on that vertex. (Step 3) Repeat these steps until no edges remain.

APPROX-2: (Step 1) First we pick an edge ¡u,v¿. (Step 2) Add both u and v to the vertex cover, throw away all edges attached to u and v. (Step 3) Repeat the step 1 and 2 till no edges remain.

2.Mutithread

In computer architecture, multithreading is the ability of a central processing unit (CPU) (or a single core in a multi-core processor) to provide multiple threads of execution concurrently, supported by the operating system. In this project, except for the main function, there are 4 threads: one for I/O, and one each for the dierent approaches to solve the minimum vertex cover problem. Because the shared data are ”read-only”, there would be no problem when multiple threads are accessing data, but if the threads are trying to change the data at the same time, it will cause problems and locks will be required.

3 Experimental Details

We use the output of graphGen on eceubuntu as input, which generates graphs with the same number of edges for a particular number of vertices. Then process the inputs in the I/O thread, and store the vertices and edges in two vectors.

Each algorithm threads take the information of the graph as input, and the parameters taken are an integer number of vertices V and a vector stored edges of each graph. For the vertex cover program, after computing by each approach in the three threads separately, the output from the program is the minimum vertex cover, which is a vector sorted from value low to high.Here we defined three different vectors to store the output vertex cover, so it will not cause the conflict if changes were made.

4.Analysis of Experimental Results

To analyze how efficient each approach is, we generated the running time and approximation ratio of each approach. For measuring them, we generated 10 graphs for each value V (V= 5,10,15,20...,45,50), and run the program for 10 times for each graph, and calculate the mean and standard deviation across those 100 runs for each value of V. Next, we will analyze the experimental results in these two aspects:

1.Running Time

We recorded the running time of each algorithm threads and computed their average and standard variance.We draw the graphs separately because compared to the first two approaches, CNF-SAT takes a much longer running time. Figure 1 and figure 2 separately shows the running time of APPROX-1 and APPROX-2, and CNF-SAT, where x-axis represents for the number of vertices, and the y-axis stands for the running time(μs).

For the APPROX-1 and APPROX-2, it can bee seen in figure 1 that both of them grow with increasing number of vertices, because the time complexity is . When comparing these two algorithms, APPROX-1 costs more running time than APPROX-2 because in my program it takes more sequential loops and computations to pick the vertex which has the highest degree while APPROX-2 just picks the edge randomly.

When it comes to CNF-SAT algorithm, it takes a much longer running time when there are many vertices and edges in the graph. During the experiment, we found that when trying to run the program which takes the graph of 20 vertices or more, it will cost an extremely long time, that is also the reason why they are not shown in figure 2. From figure 2 , the running time rises exponentially as number of vertices increase, this is because it is a NP-complete problem, which cannot be done within polynomial time.

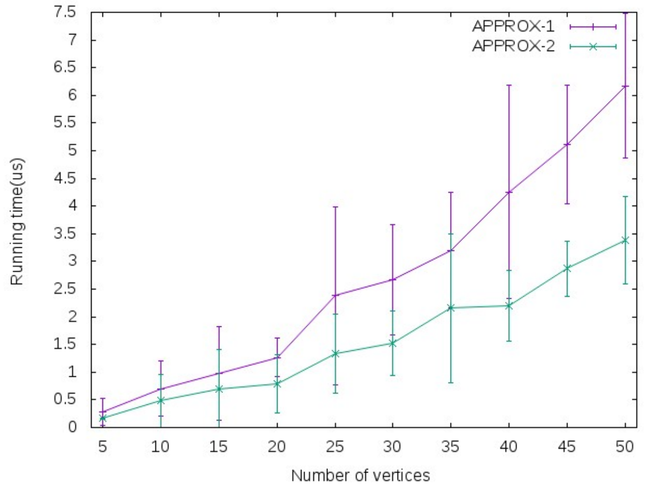

Figure 1: Running time of APPROX-1 and APPROX-2.

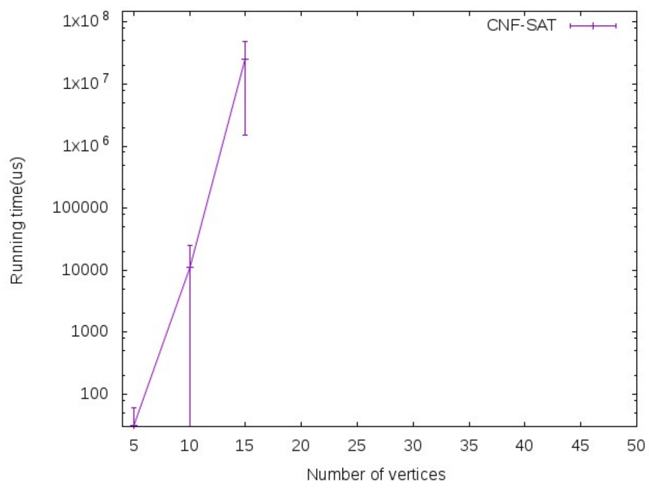

Figure 2: Running time of CNF-SAT.

2.Approximation Ratio:

We computed the approximation ratio of APPROX-1 and APPROX-2, which are the the ratio of the size of the computed vertex cover to the size of an optimal (minimum-sizes) vertex cover. Figure 3 shows the approximation ratio of APPROX-1 and APPROX-2, where x-axis represents for the number of vertices, and the y-axis stands for the approximation ratio. We draw the graphs with values V=5,10,15.

As is shown in figure 3, it is obvious that the result of APPROX-VC-1 is more close (basically 1:1) to the the optimal ((minimum-sizes) solution, while APPROX-2 produces larger size of vertex cover which means it cannot guarantee the correctness of minimum-sizes results.

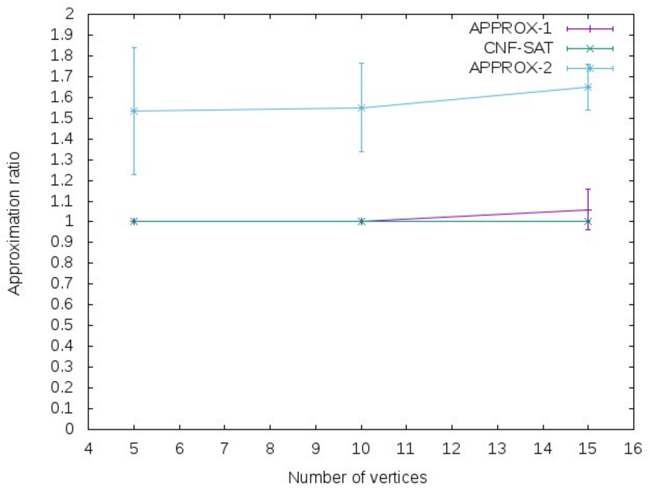

Figure 3: Approximation ratio.

Overall, the efficiency of CNF-SAT algorithm is the best of all to find minimum vertex cover when the data scale is small, however, when there are more vertices and edges, the running time will rise up significantly compared to the other two algorithms, so it is difficult to realize. APPROX-2 takes the least running time, however, the approximation ratio shows it cannot always generates the minimum vertex cover accurately. APPROX-1 also takes much smaller running time and it works well when it comes to approximation ratio, which is very close to the optimal results, so we can also regard it as a more efficiency solution when the data becomes larger.

5 Summary

In this report, the implement of three different approaches to vertex cover problem are introduced first, and then there is an analysis of the efficiency of each methods by running time and approximation ratio, and finally we made an overall comparison for each of them under different situations.

References

[1] https://ece.uwaterloo.ca/ rbabaeec/ECE-650/

[2] Advanced Linux Programming, Mark Mitchell, Jeffrey Oldham, and Alex Samuel

[3] Introduction to Algorithms, Second Edition ,Thomas H.Cormen, Charles E.Leiserson, Ronald L.Rivest and Clifford Stein