ECE650 - Final Project

Ibtehaj Pervez (20915891)

8th April 2022

Contents

1 Introduction **3**

2 Algorithms **3**

3 Multithreading **3**

4 Experiment Setup **3**

5 Experiment Results **4**

5.1 Running Time 4

5.2 Approximate Ratio 6

6 Summary **7**

7 References **7**

1 Introduction

According to graph theory, a vertex cover of a graph is a set of vertices that include at least one endpoint of every edge in the graph. The vertex cover problem is a NP-hard problem when

$$P \neq NP$$

. In the above expression, P represents polynomial time and NP represents nondeterministic polynomial time. This means that in the size of the input, it problem cannot be solved in polynomial time. This report will discuss three algorithms of solving the vertex cover problem along with their performance with respect to time taken to solve the problem and approximation ratio.

2 Algorithms

The three algorithms used for determining the vertex cover are:

1. **CNF-SAT**: We use a polynomial time reduction which converts a graph edge list into a CNF-SAT problem (details of the algorithm are proivded in the project question). Afterwards, we use the MINISAT SAT solver to find the solution of the CNF-SAT which results in finding the solution of the minimum vertex cover problem for a perticular graph.
2. **APPROX-1**: We pick a vertex of the highest degree (most incident edges) and add it to our vertex cover. Then we remove all edges incident on that perticular vertex and keep repeating it till no edges remain.
3. **APPROX-2**: We pick an edge and add both the vertices of that perticular edge to our vertex cover. Then we throw away all edges attached to those two vertices and repeat till no edges remain.

3 Multithreading

In computer architecture, multithearding provides the ability execute multiple threads of a process concurrently. The threads share the same resources of the process they are part of. For this project, each of the three aforementioned algorithm had its own thread that executed the algorithm. This will result in all the three algorithms to run concurrently. Furthermore, the program waits for all the three programs to finish before presenting the result. There is a timeout function on the CNF-SAT algorithm incase the algorithm is taking considerable time to generate the vertex cover.

4 Experiment Setup

To analyze the efficency of the algorithm, we generated the running time and the approximation ratio of each algorithm. For standardization of the experiment, we first generated 10 graphs of perticular number of vertices. The number of vertices for this experiment were [5, 10, 15, 20, 25, 30, 35, 40, 45, 50]. Then we ran the three algorithms 10 times for each perticular graph resulting in 100 runs for a perticular number of vertices. This means that for graphs which contain 5 vertices, the program will be run 100 times to record the running time and the approximation ratio.

Then we plotted the graph of the running time versus the number of vertices and the approximation ratio vs number of vertices.

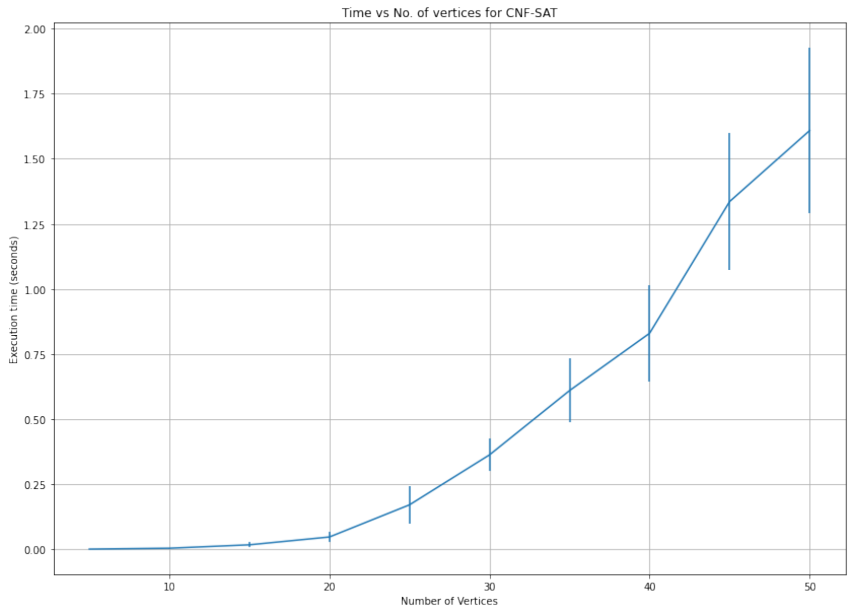
5 Experiment Results

5.1 Running Time

The running time of three algorithms were compared and analysed. The CNF-SAT has considerably higher times than that of APPROX-1 and APPROX-2 algorithm. For this reason two graphs for the running time have been plotted. One graph will only contain the running time plot of the CNF-SAT algorithm against the number of vertices and the second graph will contain the running time plot of both APPROX-1 and APPROX-2 against the number of vertices.

The running time graph for the CNF-SAT algorithm is shown in Figure 1.

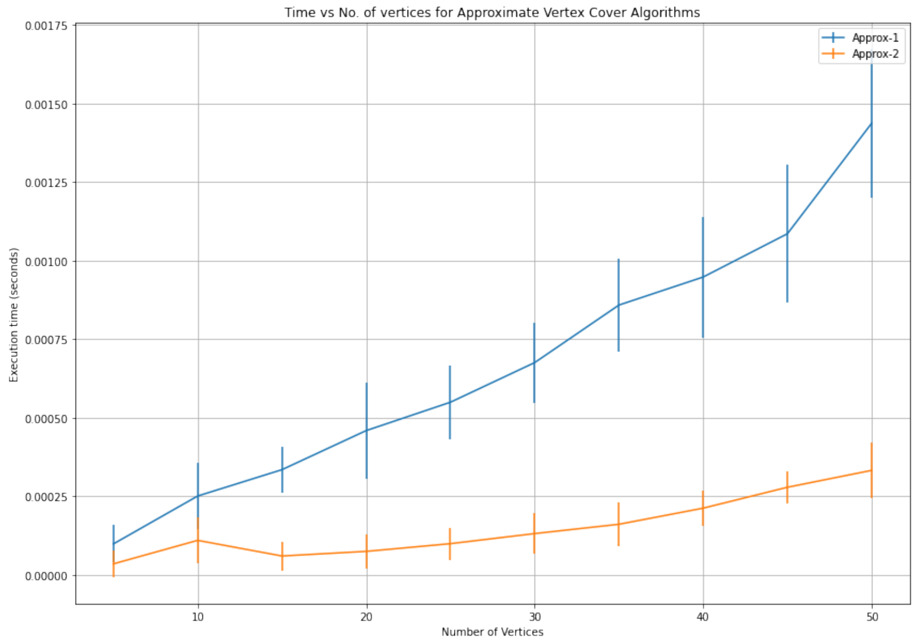
Figure 1: Time plot of the CNF-SAT algorithm



In reference to the graph of the running time of CNF-SAT, as the vertex number of the graph increases, the running time for the graph also increases. The increase is significant and approaches exponential as the number of vertices for a graph increase.

The running time graph for both the APPROX-1 and APPROX-2 is shown in Figure 2.

Figure 2: Time plot of the Approximation algorithm



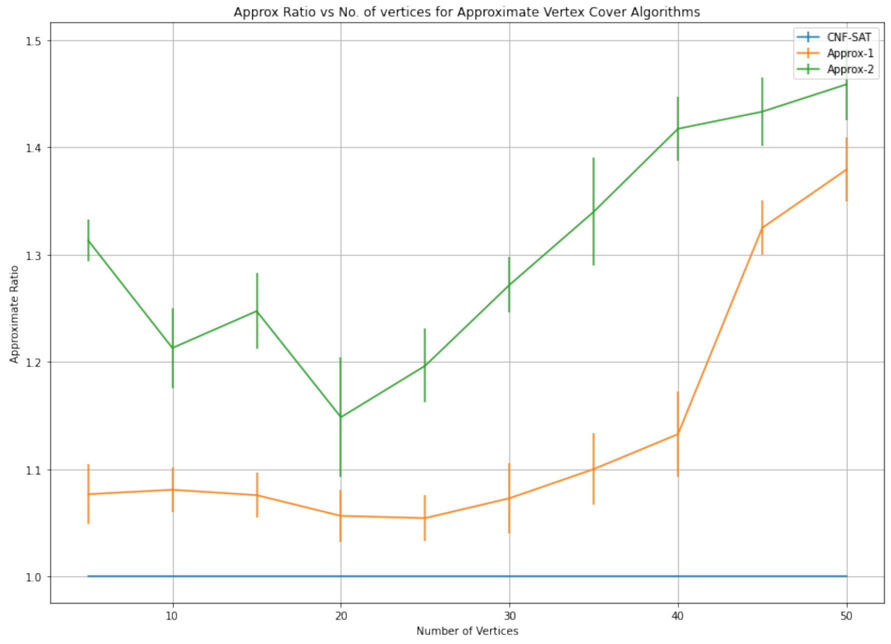
CNF-SAT takes considerable longer time than both the APPROX-1 and APPROX-2 because to convert the vertex cover problem to the CNF-SAT problem, nested for loops are use which increase time. The main for loop tries all the number of vertices to find the possible CNF. Then clause 1 use 2 nested for loops, clause 2 and 3 use 3 nested for loops, and clause 4 there are 2 nested for loops in the CNF-SAT algorithm. All of the clauses run inside the main for loop which results in the maximum nested for loop size to be 4. This increases the running time considerable. Furthermore, the minisat solver also takes considerable time to run through all the vertices values to confirm whether the graph is solvable or not.

For both APPROX-1 and APPROX-2, the running time increases as the number of vertices increases, but the overall time required by both algorithms is less than that of CNF-SAT. APPROX-1 and APPROX-2 do not have to do any of this as they dont have nested for loops and most of the time is used for reading the arrays which is an efficient process. Therefore, the time taken by these algorithms is less than that of CNF-SAT algorithm. As for the deviation from the mean running time, in general, as the number of vertices increases, the error tends to increase as well. This highlights the fact that as the number of vertices increase, the difference in edge list can increase and hence cause deviation in the running time.

5.2 Approximate Ratio

The approximate ratio of all the three algorithms were calculated. The CNF-SAT was used for normalizing the results as it is the best algorithm among the three. Therefore, CNF-SAT generates a straight line at 1. Both APPROX-1 and APPROX-2 provide considerably higher number approximate ratios for all the graph with the different number of vertices. This result is shown in Figure 3.

Figure 3: Approximate Ratio of CNF-SAT and Approximation algorithms



APPROX-2's approximation ratio starts to decrease initially but then it follows the overall trend of increasing with the number of vertices in the graph. As for APPROX-1, the overall trend is that the approximate ratio increases with the number of vertices, specially as the number of vertices begin to go beyond 40. As for the deviation from the mean, in general, as the number of vertices increases, the error tends to increase as well. This highlights the fact that as the number of vertices increase, the difference in edge list can increase and hence cause deviation in the approximation ratio.

6 Summary

This report presented the performance of three algorithm which solve the vertex cover problem. The performance was measured in terms of running time and approximate ratio. CNF-SAT takes considerably longer than the other two algorithms to generate a vertex cover. Although, CNF-SAT takes more time than the others, it generates the smallest vertex covers everytime.

Therefore, the use of the algorithm depends upon the use case. If accuracy of the minimum vertex cover is more important, than CNF-SAT should be used. However, if running time of the algorithm is more important for the application, than either of APPROX-1 or APPROX-2 should be used.

7 References

1. **ECE-650 Website**: ece.uwaterloo.ca/ agurfink/ece650/
2. **ECE-650 Project Handout**
3. **Overleaf Tutorials**: https://www.overleaf.com/
4. **Logic for Computer Scientists**, by Uwe Schoning
5. **Advanced Linux Programming**, by Mark Mitchell