ECE 650 PROJECT REPORT

Hoyos Velasquez, Natalia; Gomez Gonzalez, Juan Manuel

University of Waterloo, Electrical and Computer Engineering Department

ABSTRACT

The vertex cover algorithm is of great utility for a range of applications, yet in can become computationally expensive to determine when V in a given graph is large. Vertex cover approximations, although may not be optimal in being the minimum possible cover, can be computed in a fraction of the time compared to the most ideal, minimum-sized solution. In this paper, three vertex cover algorithms are compared. The first one is based on a SAT solver and the other two are based on approximation algorithms. The approximation algorithms are evaluated and compared to an optimal vertex cover using their running time and approximation ratio. The approximation ratio is the proportion of the size of the vertex cover computed to the size of an optimal vertex cover. Graphs from a range of 5 to 50 in increments of 5 are used, 10 for each interval and each run 10 times.

Index Terms— Vertex cover, MiniSat, Conjunctive Normal Form, Approximation algorithm.

1. INTRODUCTION

The vertex cover problem is a problem with extreme significance due to the range of applications that it can have, from finding phylogenetic trees based on protein domain information in the field of computational biology [1] to the prevention of denial of service attacks in networking [2]. Vertex cover can be defined as the subset of vertices of a graph that contain at least one of the endpoints of each edge of the graph. Given an undirected graph G, it finds a subset of vertices S so that all the edges in the graph are incident to at least one of the vertices in the subset S [3]. An example and trivial solution for the vertex-cover problem would be the subset of all the vertices in the graph. Nonetheless, the vertex cover problem is to find an optimal solution which is the minimum-sized subset of vertices so that the fewest vertices are required to cover the entire graph. An optimal subset might not be unique given that more than one subset with the same number but different vertices might exist for a given graph.

However, this problem cannot be solved by brute force in polynomial time due to the fact that as the graph grows, the possible combinations for the solutions grow exponentially [2]. The vertex cover problem has actually been proved to be NP-complete and thus it is not expected to find a polynomial time solution [3]. Nonetheless, instead of trying to find a minimum-sized optimal solution, approximations can be made to reduce the computation time it might take to find a solution at the cost of risking not having the minimum size.

In this project, three algorithms are analyzed and both the time taken to obtain a result and the proportion to the most optimal solution are tested. The first algorithm studied is labeled CNF-SAT and uses an encoding of the vertex cover to Conjunctive Normal Form (CNF) and is subsequently solved using a SAT solver [4], more specifically, the MiniSat solver, developed by Eén and Sörensson in 2003 [5].

The other two algorithms are based on approximation algorithms, whose basic operation will be briefly explained in the methodology of the paper but can be further consulted in [3].

The paper is then divided in four sections. In the first section, the methodology for the experiment is defined. Subsequently, the results obtained with the given methodology are displayed. A discussion section is presented afterward and it further deals with the findings obtained in the result section. Closing the paper are the conclusions.

2. METHODOLOGY

As mentioned before, three algorithms are compared in this paper. The first algorithm is labeled CNF-SAT-VC and is an encoding of the decision version of the vertex cover to a CNF problem. A CNF is comprised of a number of clauses each one including a series of literals in disjunctive form [4]. The CNF is then used as the input to a MiniSat solver. The MiniSat solver looks for combinations of truth values that can satisfy the specific problem. CNF-SAT-VC is guaranteed to return an optimal solution.

The second algorithm is labeled APPROX-VC-1 and is polynomial-time. This approximation of the vertex cover is implemented as follows:

1. Pick the vertex with the most incident edges (highest degree) and add it to the vertex cover.
2. Discard all edges incident to that vertex.
3. Repeat until there are no more vertices to add.

The third algorithm is labeled APPROX-VC-2. It guarantees its size to be no more than twice the size of an optimal vertex

cover and is solved in polynomial-time [3]. APPROX-VC-2 is implemented as follows:

1. Select an edge.
2. Add both vertices composing the edge (u and v) to the vertex cover.
3. Discard all other edges incident to both u and v.
4. Repeat until there are no more edges to add.

The efficiency of the different algorithms is then evaluated. This efficiency is defined as the running time of each algorithm and the approximation ratio of its solutions. The approximation ratio is considered as the proportion of the size of the computed vertex cover to the size of the minimum-sized optimal vertex cover, given by CNF-SAT-VC.

To test how efficient each approach is, graphs with different number of vertices (V) are used as input into the system. The graphs' number of vertices range from 5 to 50 in increments of 5. 10 graphs are generated for each value of V, and each of these graphs is run 10 times for a total number of runs of 100 per value of V and 1,000 total. The approximation ratio and the computation time are analyzed by calculating the average or mean value and the standard deviation (SD) of these measurements, comparing them between algorithms. The results were obtained using the Windows Subsystem for Linux running Ubuntu 18.04 on a desktop computer with an Intel Core i7 3770k processor @ 4.1 GHz.

Due to the high values of the vertices that are introduced into the program, the difficulty to scale the CNF-SAT-VC algorithm to large number of vertices and in order to avoid lengthy waits, a timeout of 2 minutes is set. This means that if the program takes more than 2 minutes to compute a result, it will abort and not show a solution for that algorithm.

3. RESULTS

The results of the computing time can be seen on Figs. 1, 2, and 3. Fig. 1 displays the comparison of the computing time vs number of vertices for the three algorithms. As expected, CNF-SAT-VC took the longest and it was only able to compute a solution for up to V = 15 vertices due to the timeout of 2 minutes for its convergence. Fig. 2 displays a close-up of the same results for the CNF-SAT-VC algorithm alone, as the time difference between the approximation algorithms and CNF-SAT-VC was large enough to make it difficult to read the values on the graph. These two graphs display the time in μs and in logarithmic scale vs the number of vertices. Fig. 3 shows the complementary close-up, displaying the comparison between the running time (in μs) of the two approximation algorithms, APPROX-VC-1 and APPROX-VC-2, for each value of V. Table 1 shows the quantitative results.

Finally, Fig. 4 displays the approximation ratio vs V. This graph displays the results up to V = 15 vertices since, as was

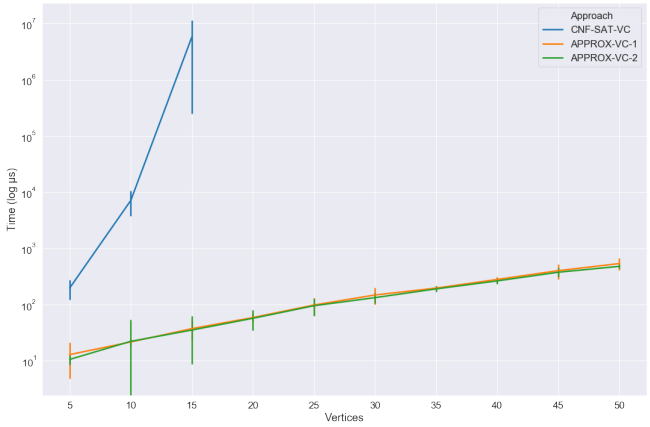

Fig. 1. Running time (in log μs) comparison between the different vertex values of the three different approaches.

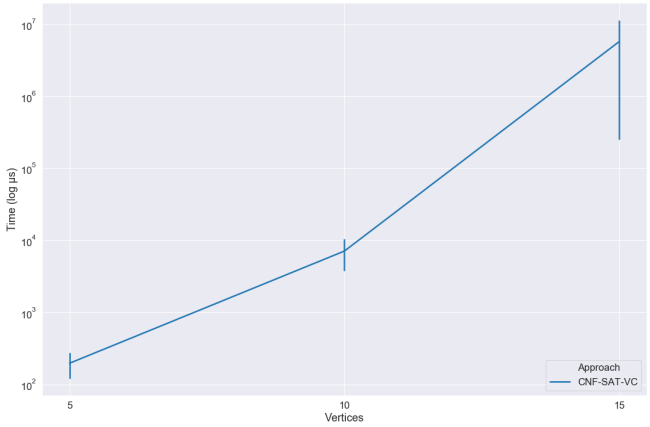

Fig. 2. Running time (in log μs) comparison between the different vertex values of the CNF-SAT-VC.

mentioned above, CNF-SAT-VC reached the time limit set in the timeout when trying to compute further results. Table 2 shows this information quantitatively.

4. DISCUSSION

As it can be seen in Fig. 1 and Table 1, when V is small, the difference in the running time between the approximation algorithms and CNF-SAT-VC is somewhat contrasting. With small Vs, the approximations' running times are in the tens of microseconds while the CNF-SAT-VC's is in the hundreds, giving a ten-fold difference. The data also displays that the time difference increases with larger values of V, being 100,000 times longer to run CNF-SAT-VC than any approximation when using V = 15.

As expected, CNF-SAT-VC had the difficulty to scale to a large V. While V increased, the CNF-SAT-VC had trouble to obtain a result within the time limit as the number of pos-

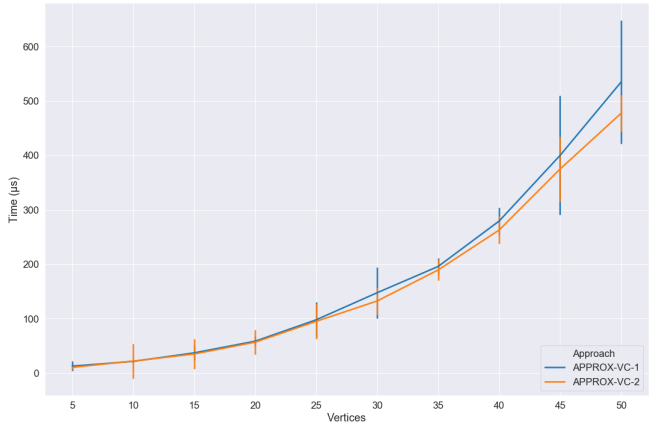

Fig. 3. Running time (in μs) comparison between the different vertex values of the approximation methods.

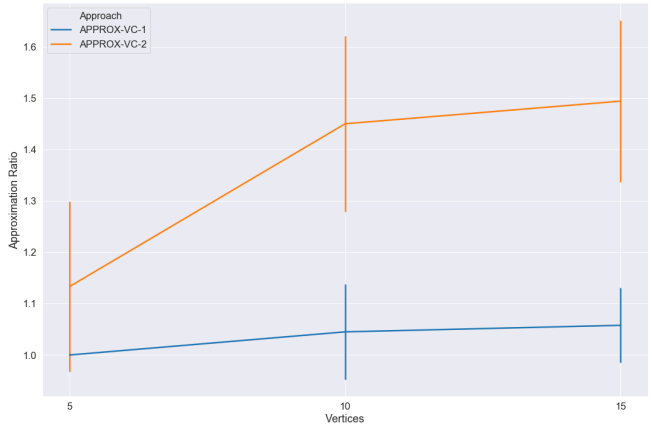

Fig. 4. Approximation ratio vs number of vertices of the two approximation methods.

sible solutions that it had to try increased exponentially. This can also be perceived in Fig. 2 where the exponential behavior starts to be noticeable with the considerable change in the graph's slope before and after V = 10. Additionally, the results show an increase on the standard deviation with a larger V. This high variance might be explained by the randomness in the graph generation, since a larger V permits a larger variation in the graph configuration and the degrees of the vertices, influencing the complexity to find a solution and thus the running time.

Fig. 3 indicates that APPROX-VC-1 and APPROX-VC-2 have a tendency to take more time after each increase in the value of V. Notice that unlike Figs. 1 and 2, Fig. 3 does not display the time in a logarithmic scale. The graph shows the polynomial behavior of the approximation algorithms and reinforce that their growth speed is far slower than that of CNF-SAT-VC. These results are coherent with the way the algorithms are implemented. The running time of APPROX-VC-1 depends on the number of vertices since it affects the number of iterations for which the highest degree vertex has to be selected. The running time of APPROX-VC-2 is based on the number of edges and thus the number of vertices that have to be selected and added to the vertex cover. The number of edges can be influenced by the change in V.

Table 1. Mean and standard deviation of the running time (in μs) it took to compute each of the different approaches.

Vertices	Approach	Mean	SD
5	APPROX-VC-1	12.85	7.89
	APPROX-VC-2	10.55	1.83
	CNF-SAT-VC	197.09	73.71
10	APPROX-VC-1	21.57	2.23
	APPROX-VC-2	22.13	30.75
	CNF-SAT-VC	7,098.51	3,224.4
15	APPROX-VC-1	37.22	13.29
	APPROX-VC-2	35.1	26.28
	CNF-SAT-VC	5,738,641.54	5,480,986.27

Table 2. Mean and standard deviation values of the approximation ratios for both approximation algorithms.

Vertices	Approach	Mean	SD
5	APPROX-VC-1	1.0	0.0
	APPROX-VC-2	1.13	0.16
10	APPROX-VC-1	1.05	0.09
	APPROX-VC-2	1.45	0.17
15	APPROX-VC-1	1.06	0.07
	APPROX-VC-2	1.49	0.16

It is also noticeable that the more vertices a graph has, the larger the difference in speed between APPROX-VC-1 and APPROX-VC-2. The time it takes to compute both approximations starts to diverge from V = 25, yet it is more evident from V = 35 onward. Table 3 shows these results and allows for a better comparison of the different computation times. With a small V, both approximations have a similar running time because both algorithms need to complete similar steps: first find or select a vertex, then add it to the vertex cover and finally eliminate all its adjacent vertices. The biggest differences between the approximation algorithms are that the former has a stricter rule to select the vertex to add to the vertex cover, as it has to find the one with the highest degree, and the later finds and adds the two vertices corresponding to one edge at the same time. With larger values of V the process to apply the rule to select vertices in APPROX-VC-1 gets prolonged as it has to compute the degree of a larger number of vertices and compare all of them to select the maximum.

Fig.3 also indicates through the error bars that there is high variability in the APPROX-VC-2 running time. The high

variance of APPROX-VC-2 might be caused by the different degree of the vertices found on the graphs. It might be that a solution to a graph with a higher degree could be found faster than the solution to a graph of lesser degree.

Table 3. Mean and SD of the time of the approaches according to the number of vertices of the graph, with the difference between the respective statistical measurements in each set of vertices.

Total Vertices	Approach	Mean	Mean Diff	SD
25	APPROX-VC-1	97.9	2.88	31.01
	APPROX-VC-2	95.02		30.52
30	APPROX-VC-1	147.62	15.16	46.28
	APPROX-VC-2	132.46		23.75
35	APPROX-VC-1	195.83	6.43	14.66
	APPROX-VC-2	189.4		18.24
40	APPROX-VC-1	279.23	16.5	23.57
	APPROX-VC-2	262.73		24.42
45	APPROX-VC-1	399.87	25.1	108.49
	APPROX-VC-2	374.77		58.8
50	APPROX-VC-1	534.49	57.36	112.69
	APPROX-VC-2	477.13		32.65

Analyzing Fig. 4, it can be seen that both approximations have a similar response and the sizes of their solution for the vertex cover are within 1.5 times the size of the optimal solution. The approximation ratio of the two algorithms starts relatively close from one another when they are dealing with small graphs (i.e. V = 5), staying between 1 and 1.2 in average. However, as V increase the approximation ratio of APPROX-VC-2 increases by a higher amount compared to that of APPROX-VC-1, finally situating at 1.06 $\pm$0.07 and 1.49 $\pm$0.16 for APPROX-VC-1 and APPROX-VC-2 respectively. What this seems to indicate is that APPROX-VC-2 is able to maintain a closer similarity with CNF-SAT-VC with respect to the vertex cover size, making it more accurate for this range of V values. What can be appreciated in Fig. 3 coupled with what has just been found in Fig. 4 seem to suggest that there exists a compromise between these two algorithms, where if time performance is prominently required APPROX-VC-2 should be selected, while if a smaller vertex cover is desired APPROX-VC-1 should be the one to be used. These considerations appear to be more relevant when the number of vertices V increases.

5. CONCLUSION

Due to the exhaustive search nature of the CNF-SAT-VC algorithm, the running time it takes to compute a vertex cover grows more rapidly than that of both of the approximation algorithms studied. Nonetheless, its solution guarantees an optimal vertex cover with the minimum sized vertex subset possible for that specific graph. The same cannot be said for the approximation algorithms.

Apart from this, the findings of this experiment appear to indicate that both approximations have a similar approximation ratio response. The sizes of their solutions for the vertex cover problem are within 1.5 times the size of the optimal solution for this range of values of V. APPROX-VC-1 had a better performance when compared to APPROX-VC-2 with a response of just 1.06 times the size of the optimal solution. Speed-wise, APPROX-VC-2 had a faster computation time compared to APPROX-VC-1. This, combined with the previous approximation ratio assertion, means that if a vertex cover calculation is required and depending on the size of the graph it might be necessary to choose between speed and accuracy of the vertex cover.

6. REFERENCES

[1] F. Abu-Khzam, R. Collins, M. Fellows, M. Langston, W. Suters, and C. Symons, "Kernelization algorithms for the vertex cover problem: Theory and experiments," in *Proceedings of the 6th Workshop on Algorithm Engineering and Experiments (ALENEX)*, 2004.

[2] F. Hüffner, R. Niedermeier, and S. Wernicke, "Techniques for practical fixed-parameter algorithms," *The Computer Journal*, vol. 51, no. 1, pp. 7–25, 2008.

[3] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, MIT Press, Cambridge, Massachusetts and London, England, 3rd edition, 2009.

[4] P. Jackson and D. Sheridan, "Clause form conversions for boolean circuits," in *Proceedings of the 7th International Conference on Theory and Applications of Satisfiability Testing (SAT 2004 - LNCS 3542)*, H. H. Hoos and D. G. Mitchell, Eds., Berlin, Germany and Heidelberg, Germany, May 2005, pp. 183–198, Springer-Verlag.

[5] N. Eén and N. Sörensson, "The MiniSat page," `http://minisat.se/Main.html`, 2007, [Online; accessed 28-Nov-2019].