ECE 650 - Methods and Tools for Software Engineering

University of Waterloo

Department of Electrical and Computer Engineering

Final Project

Authors:
Pranav Kumar Ayee Goundar Venkatesan (ID: 21079726)
Giridharan Sankaran (ID: 21076860)

Date: December 05, 2023

Abstract

The report implements and analyses the performance of three distinct Vertex Cover algorithms, i.e., CNF-SAT-VC, Approx-VC-1 and Approx-VC-2. Two different metrics are computed for each of these algorithms: run-time and approximation ratio. Upon analysis, it is identified that the run-time of CNF-SAT-VC increases exponentially with the number of vertices. Hence, an IMPROVED-CNF-SAT-VC is implemented, which is twice as fast as the original CNF-SAT-VC.

1 Introduction

A Vertex Cover [1] in graph theory refers to a subset of vertices within a graph such that every edge in the graph has at least one of its endpoints in this subset. The goal is often to find the smallest possible vertex cover, the minimum vertex cover. The Minimum Vertex Cover problem involves finding the most miniature set of vertices in a graph such that each edge is incident to at least one vertex in the set.
Vertex cover problems have applications in diverse fields, including network design, resource allocation, and optimization. The quest to find optimal vertex covers has led to the developing of various algorithms and approaches to address this NP-hard problem efficiently.
The report is organized as follows. Section 2 discusses the distinct algorithms implemented in this report. Section 3 describes the experimental setup used to run the algorithms. Section 4 analyses the performance of the different algorithms. Section 5 proposes an optimal approach to reduce the running time of CNF-SAT-VC and analyses the results. Lastly, Section 6 concludes the paper.

2 Vertex Cover Algorithms

2.1 CNF-SAT-VC

A SAT solver, short for Boolean satisfiability solver, is a software tool designed to determine the satisfiability of a given propositional logic formula in conjunctive normal form (CNF) [2]. CNF-SAT-VC is a specific type of problem that combines aspects of Boolean satisfiability and the minimum vertex cover problem. In this project, MiniSAT [3] is used as the SAT solver.
In this algorithm, a polynomial time reduction is performed and transformed into a propositional logic formula in Conjunctive Normal Form (CNF). If the formula is satisfiable, the minimum vertex cover is identified from the satisfiability assignment.

2.2 APPROX VC 1 and APPROX VC 2

Apart from implementing CNF-SAT-VC to identify the minimum vertex cover, two unique algorithms are also implemented to identify the minimum vertex cover, namely, APPROX-VC-1 and APPROX-VC-2.
APPROX-VC-1: Choose a vertex of the highest degree (with most incident edges). Include the chosen vertex to the vertex cover and remove all the edges attached to the chosen vertex. Follow this procedure until no edges remain.

APPROX-VC-2: Choose an edge e (an edge connecting vertices u and v), and insert both u and v to the vertex cover. Remove all edges incident on u and v. Follow this procedure until no edges remain.

3 Experimental Setup

The program is implemented in C++ language. The program is multi-threaded [4]. It has four threads, one for I/O and one each for the different approaches to solve the minimum vertex cover. Adoption of multi-threading allows us to take advantage of the unused computing resources and leads to faster execution of the program. The implementation of the program and the analysis of different approaches were run on the eceubuntu server.

4 Performance Analysis

Two different metrics are adopted to evaluate the performance of the algorithms: run-time and approximation ratio.

4.1 Run-time

In this project, *pthread_getcpuclockid()* [5] is used to measure the running time of each algorithm. Input graph to CNF-SAT-VC has a vertex count between 2 and 20 with an interval of 1. At each size, ten graphs are chosen from the output of "/home/agurfink/ece650/graphGen/graph Gen" on *eceubuntu*. Furthermore, each graph is iterated ten times. Finally, the mean and standard deviation is identified. The running time of CNF-SAT-VC against the varying number of vertices is depicted in Figure 1.

Figure 1 illustrates the running time of CNF-SAT-VC with the increase in the number of vertices. The figure shows that the running time of CNF-SAT-VC increases exponentially with the increase in the number of vertices.

Input graph to APPROX-VC-1 and APPROX-VC-2 has a vertex count between 2 and 50. The same graph which is used for CNF-SAT-VC is used for the vertex count between 2 and 20. New graphs are generated for the higher vertex value, starting from a vertex size of 20 until a vertex size of 50 with an interval of 5. The running time for APPROX-VC-1 and APPROX-VC-2 are identified similarly to that computed for CNF-SAT-VC. The running time of APPROX-VC-1 and APPROX-VC-2 against the varying number of vertices is described in Figure 2.

Figure 2 depicts the running time of APPROX-VC-1 and APPROX-VC-2 with the increase in vertices. It is evident from the figure that the running time of both algorithms increases with the size of the vertices. Comparing Figure 1 and Figure 2, CNF-SAT-VC exhibits the maximum running time among the three algorithms.

Additionally, the running time of CNF-SAT-VC increases exponentially as V increases. It is because, as the number of vertices increases, the propositional logic formula grows, which results in increased processing time. Specifically, the running time increases significantly after a vertex size of 15. Hence, a timeout feature of 2 minutes is implemented. Hence, if the running time takes longer than 2 minutes, the calculation is terminated for larger graphs.

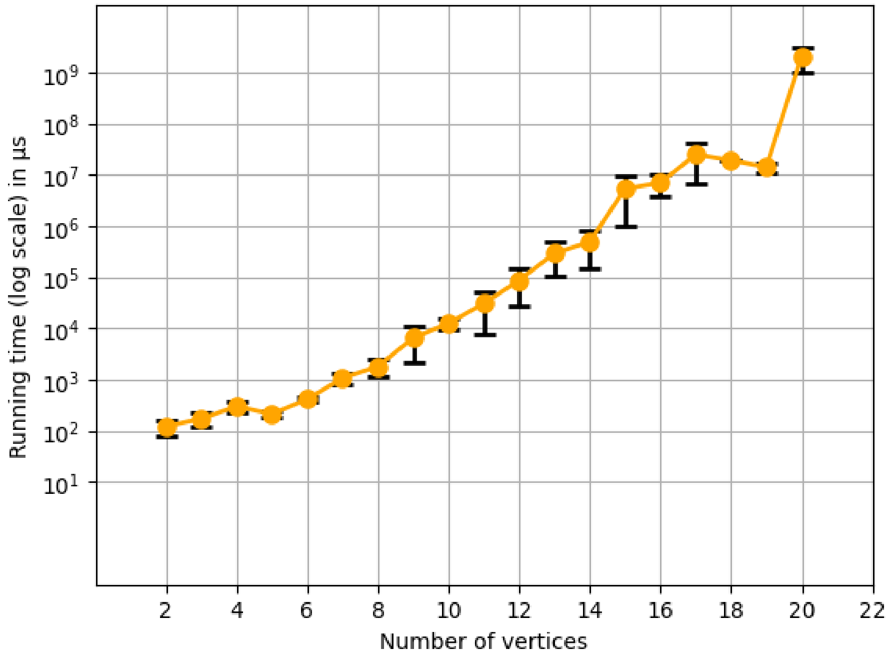

Figure 1: Running-time of CNF-SAT-VC vs Number of Vertices in Log Scale.

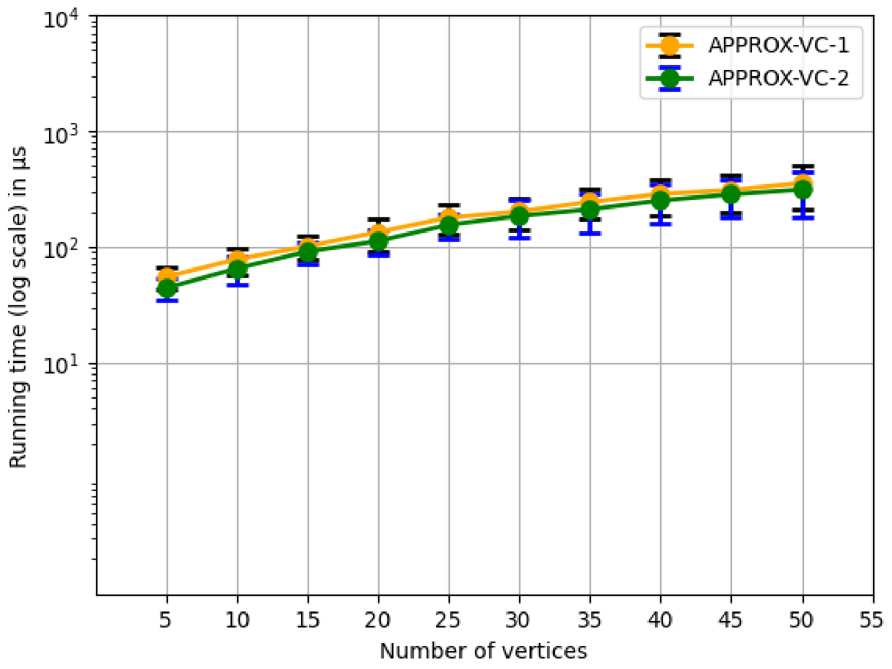

Figure 2: Running-time of APPROX-VC-1 and APPROX-VC-2 vs Number of Vertices.

4.2 Approximation Ratio

To compare the correctness of APPROX-VC-1 and APPROX-VC-2, a metric called Approximation ratio is calculated. Amongst the three algorithms, CNF-SAT-VC computes the most optimal solution. Hence, the vertex cover computed from CNF-SAT-VC is used as the base, and the approximation ratio is computed for APPROX-VC-1 and APPROX-VC-2. The formula to compute the approximation ratio is denoted in Equation 1.

$$Approximation\ ratio\ = \frac{Size\ of\ \ VC\ (Computed\ by\ approximation\ algorithm)}{Size\ of\ \ VC\ (Computed\ by\ CNF-SAT-VC)} \tag{1}$$

Using the formula in Equation 1, the approximation ratio for APPROX-VC-1 and APPROX-VC-2 is computed and presented in Figure 3.

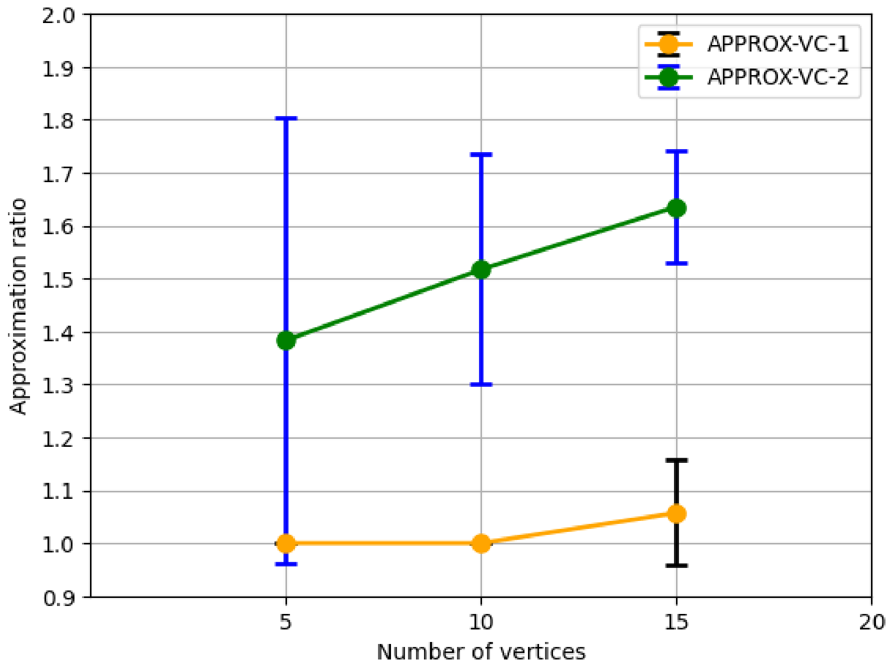

Figure 3: Approximation ratio for APPROX-VC-1 and APPROX-VC-2 vs Number of Vertices.

Figure 3 illustrates the approximation ratio for both APPROX-VC-1 and APPROX-VC-2. From the figure, it is noticeable that APPROX-VC-1 is the most optimal solution between APPROX-VC-1 and APPROX-VC-2. Since the approximation ratio of APPROX-VC-1 is closer to 1, this implies that the vertex cover identified from APPROX-VC-1 is almost the same size as the vertex cover identified by CNF-SAT-VC. Hence, this concludes that APPROX-VC-1 has a better approximation ratio than APPROX-VC-2.

5 Optimal Approach: Implementation and Analysis

From Section 4.1, it is clear that there is an exponential increase in the run-time of the CNF-SAT-VC with the increase in the number of vertices. Hence, a few improvements have been made to the CNF-SAT-VC, and an improved version has been presented, called the IMPROVED-CNF-SAT-VC.

5.1 Improvements

Two distinct improvements have been made. The improvements are described below.

- **Avoiding Isolated vertices in the clauses:** CNF-SAT-VC is used to identify the minimum vertex cover, and isolated vertices are not included in the minimum vertex cover. As isolated vertices do not contribute to covering any edges, they are not necessary for a vertex cover and are typically excluded from the minimum vertex cover.

  Hence, before adding clauses to the MiniSAT, whether the adjacency list of the current vertex is non-empty is checked. If it is empty, it indicates that the chosen vertex is isolated and no clauses are created. If it is non-empty, then clauses are added to the MiniSAT. The number of clauses in the reduction is described in Equation 2.

  $$Number\ of\ Clauses = k + n\binom{k}{2} + k\binom{n}{2} + |E| \tag{2}$$

  In equation 2, n represents the total number of vertices, $|E|$ represents the total number of edges, and k represents the vertex cover size. Therefore, if the number of vertices used to create the clauses decreases, it reduces the total number of clauses. This helps reduce the algorithm's running time.

- **Individual implementation and Ordering of Clauses:** The input to the CNF-SAT is in Conjunctive Normal Form (CNF). Hence, if any clause fails, the whole formula fails as each clause is joined using an AND operation. Therefore, if any of the clauses fails, it is unnecessary to validate the remaining clauses.

  There are four types of clauses (as mentioned in a4-encoding). So, the program is modified such that if any type of clause is identified to fail, the remaining clauses are not validated.

  Furthermore, the running time of type 1 and type 4 clauses takes less time. Hence, initially, type 1 and type 4 are validated. If it succeeds, then further types are validated. By doing so, if the CNF-SAT-VC is identified to fail for type 1 and type 4, then the rest of the clauses are not validated, which reduces the running time.

5.2 Results and Analysis

Using the improvements described in Section 5.1, the original CNF-SAT-VC is modified to improve CNF-SAT-VC. Furthermore, the same graphs used for the original CNF-SAT-VC are used as inputs to identify the running time of the IMPROVED-CNF-SAT-VC. The running time for IMPROVED-CNF-SAT-VC is identified similarly to that computed for CNF-SAT-VC. The running time of IMPROVED-CNF-SAT-VC and CNF-SAT-VC against the varying number of vertices is described in Figure 4.
Figure 4 and Table 1 compare the mean running time of IMPROVED-CNF-SAT-VC and CNF-SAT-VC against the varying number of vertices. Although the graph does not show the evident reduction in running time, it can be witnessed through Table 1. By analysing the table and figure, it is evident that IMPROVED-CNF-SAT-VC is almost twice as fast as the original CNF-SAT-VC for the higher number of vertices. Since the running time of IMPROVED-CNF-SAT-VC is half the running time of the original CNF-SAT-VC.

Table 1: Comparison of mean running time of IMPROVED-CNF-SAT-VC and CNF-SAT-VC

S.No	Number of Vertices	Mean Running time before Optimization(in sec)	Mean Running time after Optimization(in sec)
1	10	0.0118944	0.0109404
2	12	0.0870766	0.0770728
3	14	1.021029	0.5354362
4	16	12.560106	8.879476
5	18	27.543063	17.694616
6	20	2022.123541	1163.110183

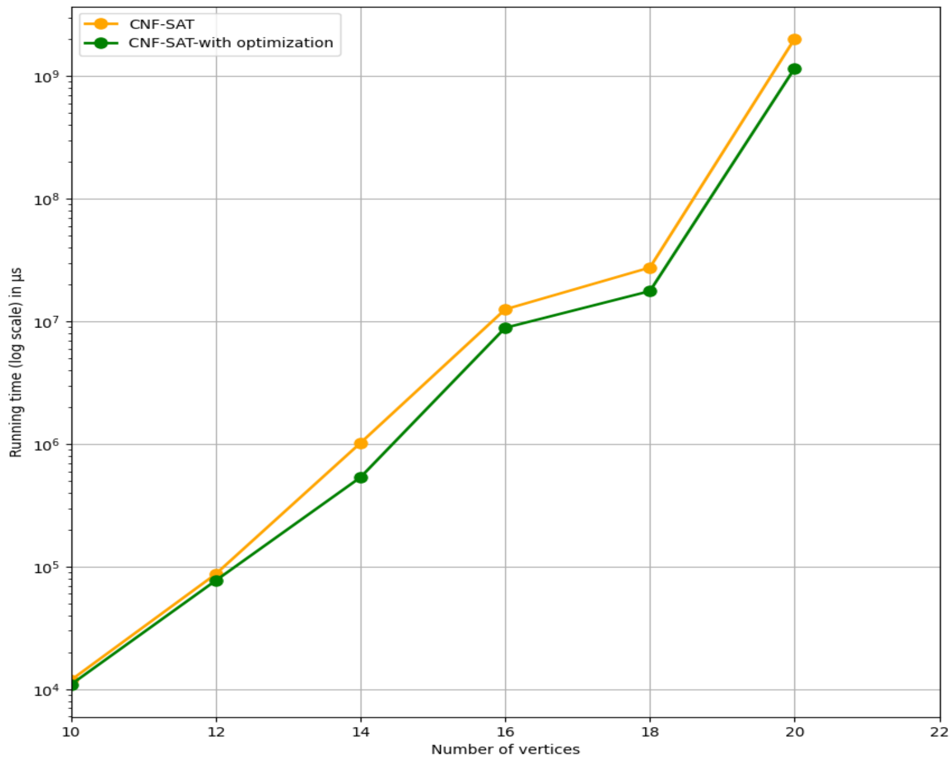

Figure 4: Comparison of mean running time of IMPROVED-CNF-SAT-VC and CNF-SAT-VC against the varying number of vertices.

Even for lower vertices, there is a reduction in the running time, but it is much more significant in higher vertices. Since the running time of the original CNF-SAT-VC, until 13 vertices, is significantly less. So, the improvement only reduces the running time to a few microseconds. However, from 14 vertices, the running time is significant. Hence, avoiding a few isolated vertices significantly impacts the running time. Therefore, the figure and table mentioned prove the effectiveness of the improvements described in Section 5.1.

6 Conclusion

Three distinct algorithms are implemented in this report. From the analysis of these algorithms, it is observed that CNF-SAT-VC always prints the accurate answer. Apart from CNF-SAT-VC, APPROX-VC-1 produces almost the same accurate solution. Although the CNF-SAT-VC provides the desired solution, its running time increases exponentially with the number of vertices. However, APPROX-VC-1 is much faster than CNF-SAT-VC. Additionally, APPROX-VC-2 is faster than both CNF-SAT-VC and APPROX-VC-1. Although APPROX-VC-2 is the fastest algorithm, it does not produce the desired output.
In the later part of the project (Section 5), an improved version of the CNF-SAT-VC is displayed. Although the IMPROVED-CNF-SAT-VC is twice as fast when compared to the original algorithm, it is still much slower than APPROX-VC-1.
Hence, picking the suitable algorithm for the application involves a trade-off between running time and accurate answers. If an accurate answer is demanded without any slack in the running time, then CNF-SAT-VC is chosen as the algorithm. However, if there is a slack in the running time, then APPROX-VC-1 is chosen as the algorithm.

References

[1] J. Chen, I. A. Kanj, and W. Jia, ”Vertex cover: further observations and further improvements,” Journal of Algorithms, vol. 41, no. 2, pp. 280-301, 2001.

[2] R. Schuler, ”An algorithm for the satisfiability problem of formulas in conjunctive normal form,” Journal of Algorithms, vol. 54, no. 1, pp. 40-44, 2005.

[3] N. Sorensson and N. Een, ”Minisat v1.13 - a sat solver with conflict-clause minimization,” in SAT 2005, 2005, pp. 1-2.

[4] D. M. Tullsen, S. J. Eggers, and H. M. Levy, ”Simultaneous multithreading: Maximizing on-chip parallelism,” in Proceedings of the 22nd Annual International Symposium on Computer Architecture, 1995.

[5] Linux manual page. *URL:https://man7.org/linux/man-pages/man3/pthread_getcpu clockid.3.html.*