# PROJECT REPORT


**SUBMITTED BY:**
Jobanpreet Singh Thind (21057311)
Sunvir Singh (21100544)

**SUBMITTED TO:**
Arie Gurfinkel

APRIL 2024

# Contents

# 1 Objective

The project involves solving vertex cover problem using multithreading. The concept of multithreading is used to run 3 different algorithms in parallel. The 3 algorithms are as follows:

1. CNF-SAT-VC
2. APPROX-VC-1
3. APPROX-VC-2

Further we will be calculating and analyzing the running time and approximation ratio of all the algorithms.

## 1.1 What is vertex cover?

- Given a graph G (V, E) where V is set of vertices and E is set of edges, vertex cover is used to find the set of vertices such that for every edge either u or v is present in that set of vertices.
- Minimum vertex cover is used to find set of minimum vertices. This problem is not solved in polynomial time and thus is it considered as NP-hard problem. Therefore, this problem is converted into propositional logic which can be solved by SAT solver. SAT solver helps in finding whether the formula is satisfiable or not.
- The propositional logic used by SAT solver is CNF.

## 1.2 What is CNF?

Conjunctive Normal Form (CNF) is a standard form used in propositional logic, defined as a conjunction of one or more clauses, where each clause is a disjunction of one or more literals. A clause is a disjunction of literals, which are either propositional variables or their negations. Formally, a CNF formula $F$ is represented as:

$$F = C_1 \wedge C_2 \wedge \ldots \wedge C_n$$

where each $C_i$ is a clause represented as:

$$C_i = L_{i1} \vee L_{i2} \vee \ldots \vee L_{im}$$

and each $L_{ij}$ is a literal, defined as:

$$L_{ij} = P \quad \text{or} \quad \neg P$$

where $P$ is a propositional variable.

# 2 Algorithms

## 2.1 CNF-SAT-VC

In this algorithm, the vertex cover problem was reduced to CNF-SAT which was solved with the help of Minisat SAT solver in C++. The formula was created using following 4 conditions.

1. At least one vertex is present in vertex cover.
2. The vertex will appear only once in vertex cover.
3. Only vertex can be present at the same position in vertex cover.
4. For every edge (u, v), either of both will present in vertex cover.

## 2.2 APPROX-VC-1

In this algorithm,

1. The vertex with highest degree is picked.
2. Add that vertex to the vertex cover.
3. Remove the edges incident to that vertex.
4. Pick the next highest degree.
5. Repeat the above steps till no edge is remaining.

## 2.3 APPROX-VC-2

In this algorithm,

1. Pick the edges from set of edges.
2. Add u, v to the vertex cover.
3. Remove every edge incident on either u or v.
4. Repeat the above steps till no edge is remaining.

## 3 Graphs

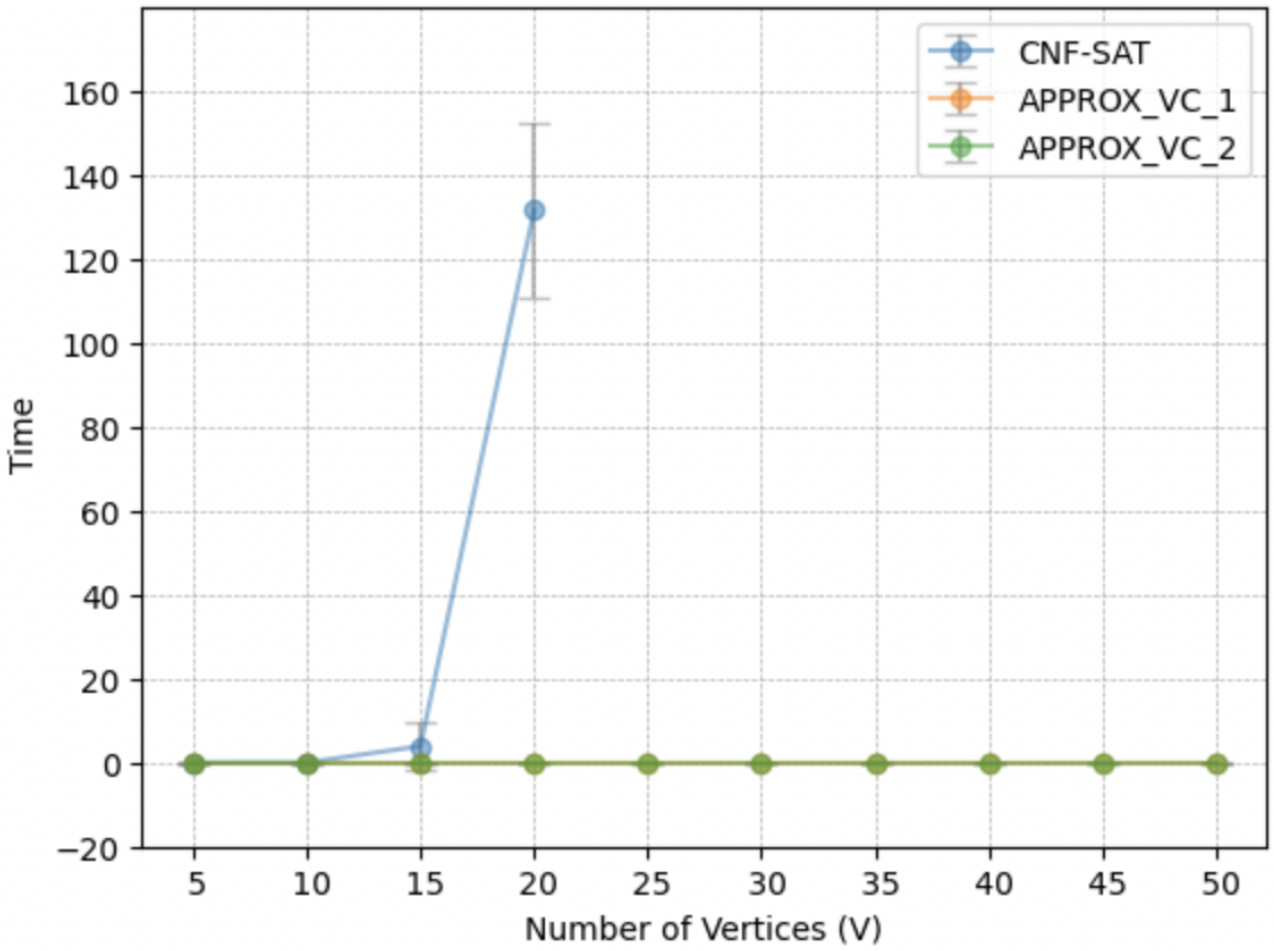


**Figure 1:** Run time(secs) comparison of CNF-SAT, APPROX-VC-1, APPROX-VC-2

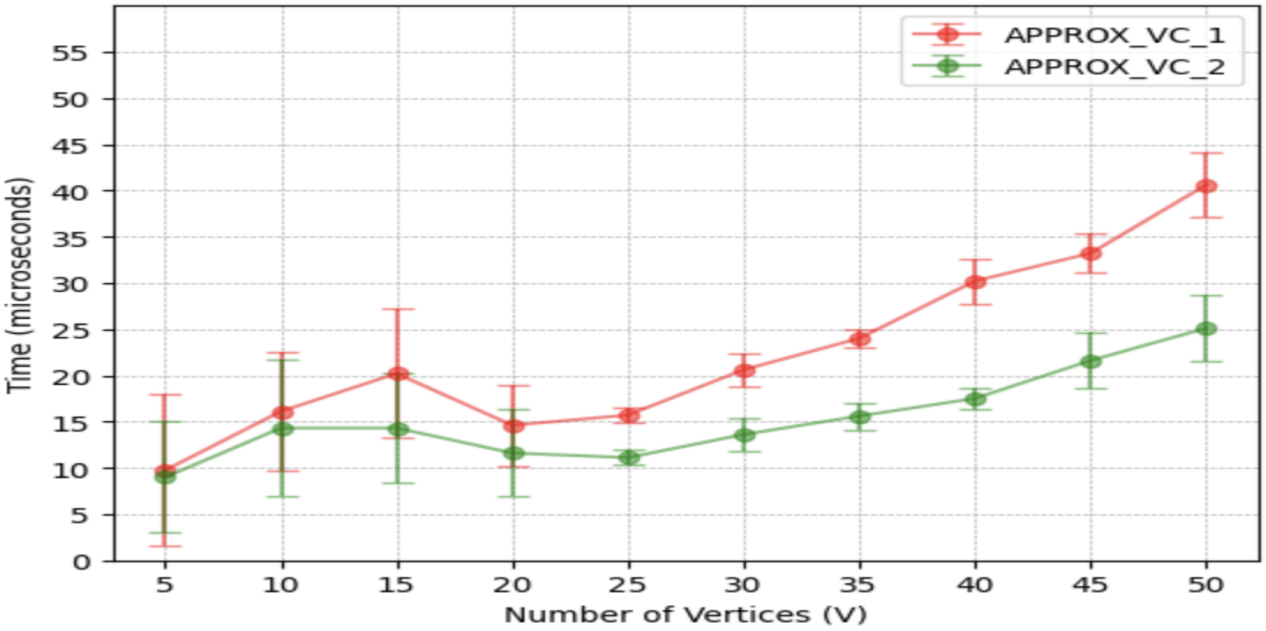


**Figure 2:** Run time(micro secs) comparison of APPROX-VC-1, APPROX-VC-2

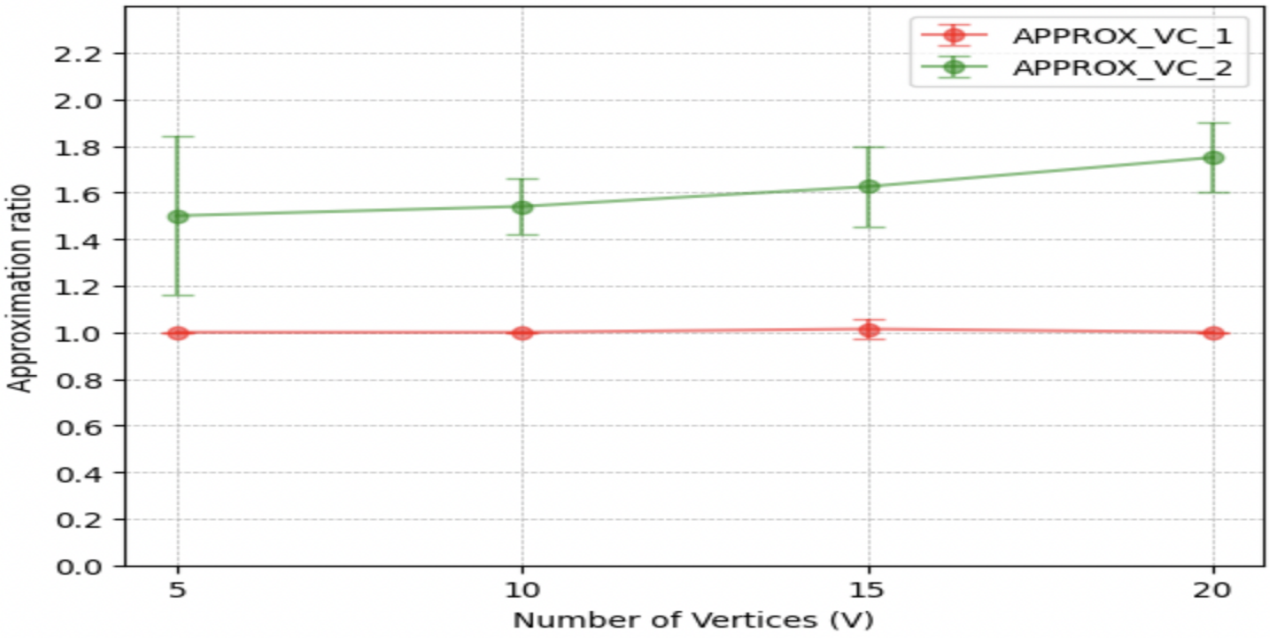


**Figure 3:** Approximation Ratio

# 4 Statistical analysis

| V | CNF-SAT | SD CNF-SAT | Approx-VC-1 | SD Approx-VC-1 | Approx-VC-2 | SD Approx-VC-2 |
|---|---|---|---|---|---|---|
| 5 | 2.37E-4 | 7.83E-05 | 9.74E-06 | 8.16E-06 | 8.99E-06 | 6.00E-06 |
| 10 | 1.31E-02 | 0.013 | 1.61E-05 | 6.35E-06 | 1.43E-05 | 7.40E-06 |
| 15 | 3.92 | 5.79 | 2.02E-05 | 6.99E-06 | 1.43E-05 | 5.91E-06 |
| 20 | 131.68 | 20.85 | 1.46E-05 | 4.35E-06 | 1.16E-05 | 4.69E-06 |
| 25 | Timeout | Timeout | 1.57E-05 | 7.72E-07 | 1.11E-05 | 7.89E-07 |
| 30 | Timeout | Timeout | 2.06E-05 | 1.81E-06 | 1.36E-05 | 1.77E-06 |
| 35 | Timeout | Timeout | 2.40E-05 | 9.91E-07 | 1.56E-05 | 1.47E-06 |
| 40 | Timeout | Timeout | 3.02E-05 | 2.46E-06 | 1.75E-05 | 1.14E-06 |
| 45 | Timeout | Timeout | 3.32E-05 | 2.11E-06 | 2.16E-05 | 2.97E-06 |
| 50 | Timeout | Timeout | 4.06E-05 | 3.46E-06 | 2.51E-05 | 3.52E-06 |

**Table 1:** Mean values of time and standard deviation(SD) in seconds for the different algorithms

| V | Approx-VC-1 | SD Approx-VC-1 | Approx-VC-2 | SD Approx-VC-2 |
|---|---|---|---|---|
| 5 | 1 | 0 | 1.5 | 0.34 |
| 10 | 1 | 0 | 1.54 | 0.12 |
| 15 | 1.014 | 0.04 | 1.625 | 0.17 |
| 20 | 1 | 0 | 1.75 | 0.15 |

**Table 2:** Approximation ratio and standard deviation(SD) for the different algorithms

# 5 Analysis

## 5.1 Analysis of Running time

All the 3 algorithms were run on different number of vertices from 5 to 50. Different observations were made. We are sure that CNF-SAT-VC will give us the optimal solution, but it takes longer duration of time than APPROX-VC-1 and APPROX-VC-2.

We took timeout as 140 seconds. CNF-SAT-VC was taking longer time than 140 seconds for vertices greater than 20. For a greater number of vertices, there will be larger number of literals for each clause in CNF thus SAT solver will have to compute every combination of literal to check if it is satisfiable or not. Thus, it is taking more time for large number of vertices. The standard deviation will also increase as vertices increases in CNF-SAT-VC.

Also, from the above tables and graphs, it can be seen that APPROX-VC-1 is taking more time than APPROX-VC-2. This is because some of the computation time might have gone in computing the highest degree vertex. Whereas, in the latter algorithm, we are randomly picking the vertex and sometimes add unnecessary vertices to the answer.

It can be also seen that for majority of cases, the standard deviation of APPROX-VC-2 is greater than that of APPROX-VC-1. It is because the former algorithm depends on the input data of the graph and size of the vertex cover will varies.

### 5.2 Analysis of Approximation Ratio

The approximation formula is defined as:

$$\frac{Sizeof MinimumVertexCoversetbyAPPROX - VC}{Sizeof MinimumVertexCoversetbyCNF - SAT}. \tag{1}$$

Thus, we can say that if the result is closer to 1, we are getting the accurate results. We can observe from the above table that the approximation ratio of APPROX-VC-1 is equal to 1 for V = 5, 15, 20. It means that it is giving us the accurate results. Because of the timeout of CNF-SAT-VC for V greater than 20, we were not able to calculate the approximation ratio of vertices greater than 20.
For the standard deviation of approximation ratio, the vertices with size 5, 15, 20 gives us the standard deviation as 0 and for V = 15 it is also very small as the results of vertex cover are consistent with that of CNF-SAT-VC.

## 6 Optimization

We used binary search for calculating minimum size of vertex cover. Earlier we were using linear search from 1 to V, which was calling the SAT solver for each of these values until it found a satisfying result, where the algorithm stopped. The idea here is to replace linear search with binary search. We start with the middle value of the low and high values, which initially are 1 and V. If we get a satisfying result, we try to find an even smaller k value that is satisfying, by changing the high to mid-1. If not, since we got an unsatisfying result for the current value of k(mid), we move low to mid+1 to find a satisfying result. This is repeated until the low value goes above high, indicating that the search interval has been exhausted. Using this approach, we achieved a time complexity of O(logn) as compared to the earlier O(n), thus making a significant time improvement.

## 7 Conclusion

To conclude, in this project we solved minimum vertex cover problem using 3 different algorithms including CNF-SAT-VC, APPROX-VC-1, APPROX-VC-2. Further, an analysis was performed to compare running time and approximation ratio. It was observed that CNF-SAT-VC gave us the optimal result, but it was taking a lot of time as compared to other algorithms which are not guaranteed to produce optimal vertex cover.