Analysis of Minimal Vertex Cover Algorithms

Linsen Gao
watiam: l78gao
student: 21134283
`l78gao@uwaterloo.ca`

Jiejia Shi
watiam: j279shi
student: 21100145
`j279shi@uwaterloo.ca`

November 24, 2024

Abstract

This study examines three algorithms for solving the minimum vertex cover problem: CNF-SAT-VC, APPROX-VC-1, and APPROX-VC-2. We compare their performance based on runtime and approximation ratio. The CNF-SAT-VC algorithm is improved with an enhanced SAT encoding, which reduces the number of variables and simplifies constraints, improving performance, especially for larger graphs. Among the approximation algorithms, APPROX-VC-1 provides more accurate solutions with a smaller approximation ratio, while APPROX-VC-2 performs faster. The results suggest that the improved SAT encoding offers a significant advantage in handling larger graphs.

1 Introduction

The minimum vertex cover is an optimization problem and it can be solved in polynomial time. Hence it is an NP-complete problem. A vertex cover of an undirected graph is a subset of its vertices that covers all the edges of the graph.

In the CNF-SAT-VC algorithm, we reduced this to a SAT problem using a SAT solver. Our task was to construct a formula F for the vertex cover using several conditions. APPROX-VC-1 and APPROX-VC-2 are approximate algorithms. As the name suggests, they give an approximate value of the minimum vertex cover that may not be always correct. APPROX-VC-1 picks a vertex with highest degree in each iteration until all edges have been visited. APPROX-VC-2 picks an edge (u,v) in random, and throws away all edges attached to u and v until no edges remain.

2 Improved SAT Encoding for Vertex Cover[Bonus]

2.1 Traditional SAT Encoding

The goal of the conventional CNF-SAT-VC algorithm that we designed is to find the smallest k that makes the CNF formula F satisfiable:

Variables: $x_{i,j}$ where i is the vertex index (from 1 to V) and j is the position in the vertex cover of size k. **Total Variables:** $\mathcal{O}(V \times k)$.

Constraints:

- *At Least One Vertex in Each Position:*

$$\bigvee_{i=1}^{V} x_{i,j} \quad \text{for each } j = 1 \text{ to } k.$$

- *No Vertex Appears Twice:*

$$\neg(x_{i,p} \land x_{i,q}) \quad \text{for all } i \text{ and } p \neq q.$$

- *No Position Contains Multiple Vertices:*

$$\neg(x_{i,p} \land x_{j,p}) \quad \text{for all } p \text{ and } i \neq j.$$

- *Edge Coverage:* For each edge (u, v):

$$\bigvee_{j=1}^{k} (x_{u,j} \lor x_{v,j}).$$

Limitations:

- Scalability issues due to $\mathcal{O}(V \times k)$ variables.
- The number of clause in traditional encoding is $N_{tratidional} = k + V\binom{k}{2} + k\binom{V}{2} + E$

2.2 Improved SAT Encoding

The improved method simplifies the encoding as follows:

Variables: A single Boolean variable v_i for each vertex i, indicating whether it is included in the vertex cover. **Total Variables:** $\mathcal{O}(V)$.

Constraints:

- *Cardinality Constraint:* At most k vertices in the cover, encoded using cardinality constraints:

$$\text{Sum}(v_i) \leq k.$$

- *Edge Coverage:* For each edge (u, v):

$$v_u \lor v_v.$$

- *Sorting Network:* If sorting networks are used for enforcing the cardinality constraint, each comparator adds multiple clauses. The total number of clauses in Sorting Network is:

$$O(V \log^2 V) \quad \text{(for sorting } V \text{ literals).}$$

Advantages:

- Reduced variables: $\mathcal{O}(V)$ instead of $\mathcal{O}(V \times k)$.
- The number of clause in improved encoding is $N_{improved} = \binom{V}{k+1} + E + V \log^2 V$

2.3 Analysis of Efficiency

The efficiency of the improved encoding was analyzed based on three primary metrics: variable reduction, constraint simplification, and Experiment.

2.3.1 Variable Reduction

The traditional encoding for the vertex cover problem introduces a large number of variables, proportional to both the number of vertices (V) and the size of the vertex cover (k), leading to a complexity of $\mathcal{O}(V \times k)$ variables.

In contrast, the improved encoding reduces the dependency on k by restructuring the problem formulation. As a result, the number of variables is proportional solely to V, achieving a complexity of $\mathcal{O}(V)$. This reduction is particularly beneficial for large graphs where k can grow significantly.

2.3.2 Constraint Simplification

In traditional encoding, positional constraints are required to ensure that the selected vertices form a valid vertex cover. These constraints contribute to increased computational overhead. The total number of clauses in traditional encoding is:

$$N_{tratidional} = k + V\binom{k}{2} + k\binom{V}{2} + E$$

The improved SAT encoding eliminates positional constraints by utilizing a streamlined logical representation, significantly reducing computational overhead. The algorithm begins with a cardinality constraint, which generates $\binom{n}{k+1}$. clauses to ensure that at most k vertices are included in the cover. Next, the edge coverage condition adds E clauses to guarantee that every edge is covered by at least one vertex. Eventually, wo techniques are combined: Sorting Networks and Binary Encodings. Sorting Networks efficiently arrange Boolean variables representing vertex inclusion in order, requiring $\mathcal{O}(V \log V)$operations. Binary Encodings then apply the cardinality constraint to the sorted sequence, adding $\mathcal{O}(logV)$operations per vertex. Together, the two methods result in $\mathcal{O}(V \log^2 V)$ clauses. The total number of clauses in improved encoding is:

$$N_{improved} = \binom{V}{k+1} + E + V \log^2 V$$

2.3.3 Traditional vs Improved SAT Encoding

From Figure 1, we can observe two important points:

1. Improved Efficiency: Improved SAT Encoding demonstrates greater efficiency compared to Traditional SAT Encoding when $10 < V < 20$. The runtime remains low for larger vertex counts, while the Traditional SAT Encoding shows a significant increase and eventually times out.

2. Larger Graph Handling: Improved SAT Encoding can handle larger graphs, as it completes execution even when $V = 25$, , staying well within the timeout threshold of 200 seconds. In contrast, the Traditional SAT Encoding times out for $V > 15$

2.3.4 Comparative Summary

- **Variables:** Reduced from $\mathcal{O}(V \times k)$ to $\mathcal{O}(V)$.
- **Clauses:** Simplified with sorting networks and ordering Boolean variables from $N_{tratidional} = k + V\binom{k}{2} + k\binom{V}{2} + E$ to $N_{improved} = \binom{V}{k+1} + E + V \log^2 V$.
- **Data:** Our experiment should Imporved SAT Encoding has higher efficient.

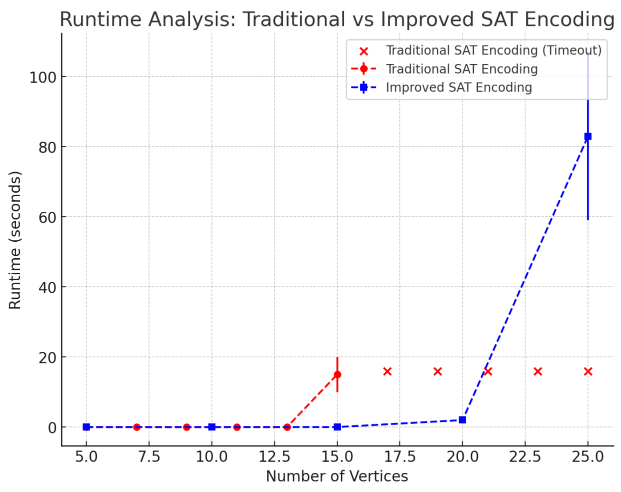

Figure 1: Traditional vs Improved SAT Encoding

3 Analysis of Experimental Results

3.1 Running Time Analysis

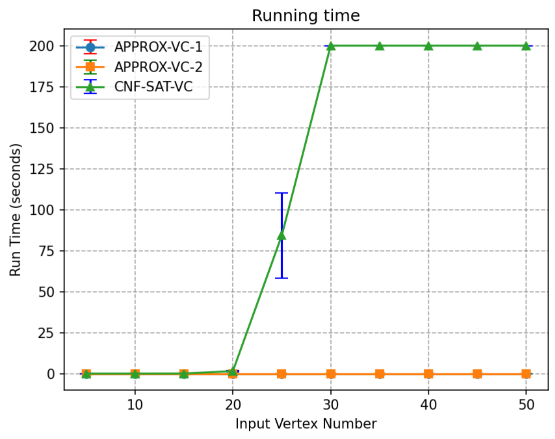

Figure 2: Running time for all three methods from V 5 to V 50

Figure 2 presents the running times of the three algorithms across vertex counts [5,10,15,...,45,50]. A significant disparity is observed between the running time of the CNF-SAT-VC algorithm and those of the approximation algorithms, APPROX-VC-1 and APPROX-VC-2. Due to optimizations applied to the CNF-SAT-VC algorithm, it demonstrates comparable performance to the approximation algorithms when the vertex count is less than or equal to 20. However, as the vertex count reaches 25, the average running time of CNF-SAT-VC escalates to nearly 80 seconds. Beyond this point, the running time continues to grow, eventually reaching a timeout limit of 200 seconds.

The primary cause of this increase is the use of a SAT solver to address the problem. As the number of input vertices rises, the corresponding number of clauses also grows, leading to longer computation times. For a graph with V vertices, the number of improved CNF-SAT-VC clauses can be estimated by the formula:

$$N_{improved} = \binom{V}{k+1} + E + V \log^2 V$$

As the number of vertices increases, the number of edges also grows, leading to a higher number of clauses in the vertex cover problem. This, in turn, results in increased running time.

Figure 3 shows that the running times for both APPROX-VC-1 and APPROX-VC-2 algorithms exhibit linear growth. This behavior occurs because both algorithms select vertices based on either the highest degree (in the case of APPROX-VC-1) or directly from the edges (for APPROX-VC-2), resulting in a time complexity of $\theta(V)$. Generally, the running time of APPROX-VC-1 is greater than that of APPROX-VC-2. This is due to the additional step in APPROX-VC-1, where we calculate the incident edges for each vertex and then sort them to identify the highest-degree vertex, a process that requires more time compared to APPROX-VC-2, which selects edges randomly without additional sorting.

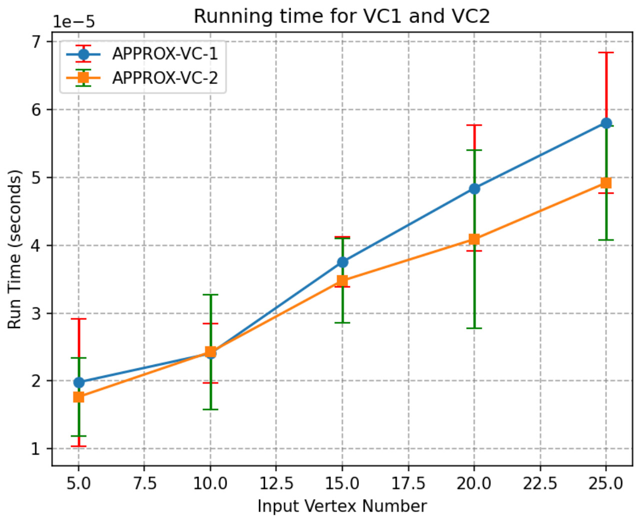

Figure 3: A zoomed-in analysis of running time for APPROX-VC-1 and APPROX-VC-2

3.2 Approximate Ratio Analysis

Figure 4 shows that the approximation ratios of APPROX-VC-1 remain consistently just above 1, while those of APPROX-VC-2 range from 1.3 to 1.6, consistently exceeding those of APPROX-VC-1. This difference occurs because APPROX-VC-1 employs a greedy strategy by selecting vertices based on their degree, which is more efficient, whereas APPROX-VC-2 selects vertices based on random edge choices, making it more likely to include redundant vertices.

4 Conclusion

In this study, we analyzed and compared three algorithms for solving the minimum vertex cover problem: CNF-SAT-VC, APPROX-VC-1, and APPROX-VC-2. Our experimental results show that the improved SAT encoding significantly enhances performance, particularly for larger graphs, by reducing the number of variables and simplifying constraints. Among the approximation algorithms,

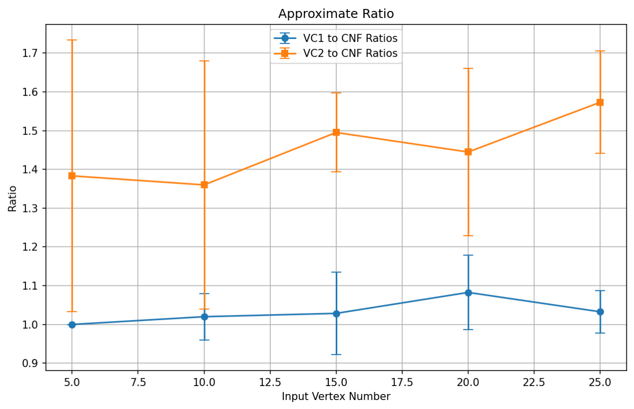

Figure 4: Approximate Ratio Analysis for APPROX-VC-1 and APPROX-VC-2

APPROX-VC-1 provides more accurate results with a smaller approximation ratio, while APPROX-VC-2 offers faster execution times.

References

[1] E. Goldberg and Y. Novikov, *Bounding variable occurrences in CNF formulas*, Design, Automation and Test in Europe Conference and Exhibition, 2003.

[2] O. Bailleux and Y. Boufkhad, *Efficient CNF Encoding of Boolean Cardinality Constraints*, International Conference on Principles and Practice of Constraint Programming, 2003.