Analysis of Vertex Cover Algorithms And Their Efficacy

ECE 650

PREPARED FOR: DR. ARIE GARFINKEL

Samudra Perera (20621037)
sperera@uwaterloo.ca

Kevin Guo (21108061)
z45guo@uwaterloo.ca

April 12th, 2024

Contents

1 Introduction

This project aims to addresses the challenge of optimizing surveillance at traffic intersections. We can frame traffic surveillance as a vertex cover (VC). A VC is a set of vertices that contains an endpoint of every edge in a graph. The beginning, ending and intersection of streets can be considered as vertices and the streets themselves can be considered the edges. Framing the problem like this, it makes sense how finding the VC of a street map would lead to an optimized solution. The VC is NP-complete and requires considerable computational difficulty. We implemented three methods to solve VC: CNF-SAT-VC, APPROX-VC-1, and APPROX-VC-2. This report aims to evaluate their effectiveness and efficiency. The outcome is a nuanced understanding of the trade-off between computational efficiency and solution accuracy.

2 Algorithms

2.1 CNF-SAT-VC

This approach uses a polynomial time reduction from vertex cover to CNF-SAT by encoding the graph with a matrix of CNF clauses. These clauses are then used by an SAT-solving library to identify the minimum vertex cover. The encoding we implemented was prescribed in Assignment 4 and followed 4 distinct clauses.

2.2 APPROX-VC-1

This approach is an estimate to solve the vertex cover. The Approx-VC-1 algorithm works by iterating over the edges finding the vertex with the most incident edges (highest degree of incidents), and throwing away all edges contained within that vertex. This vertex is added to the result and this process is repeated until no edges in the graph remain. The runtime for this algorithm is O ($m * n$), where m is the number of edges, and n is the number of vertices.

2.3 APPROX-VC-2

This approach to estimating the vertex cover is done by selecting an arbitrary edge (u, v) and adding vertices u and v to the cover. Then all edges connected to the vertices u and v are removed and the process is repeated until no edges remain. The runtime for this algorithm can vary from O(m^2) if very few edges are removed from all arbitrary selections to O ($m + logn$), where m is the number of edges and n is the number of vertices.

3 Results

3.1 Approximation Ratio

We use an approximation ratio to calculate the accuracy of the two approximation algorithms. This is the vertex cover size from the approximation algorithm over the minimum vertex cover size given by CNF-SAT. The minimum approximation ratio value of 1 would correspond to the approximation algorithm having a vertex cover of the same size as the minimum computer by CNF-SAT. The closer the approximation ratio is to 1, the more accurate the prediction of

the vertex cover size is by the approximation algorithm. It is important to note, that the approximation ratio is not a measure of the accuracy of the cover generated by the approximation algorithms. We have calculated the approximation ratios for APPROX-VC-1 and APPROX-VC-2 compared to CNF-SAT-VC for vertices ranging from 2 to 14. From the graph, we can summarize the findings as follows:

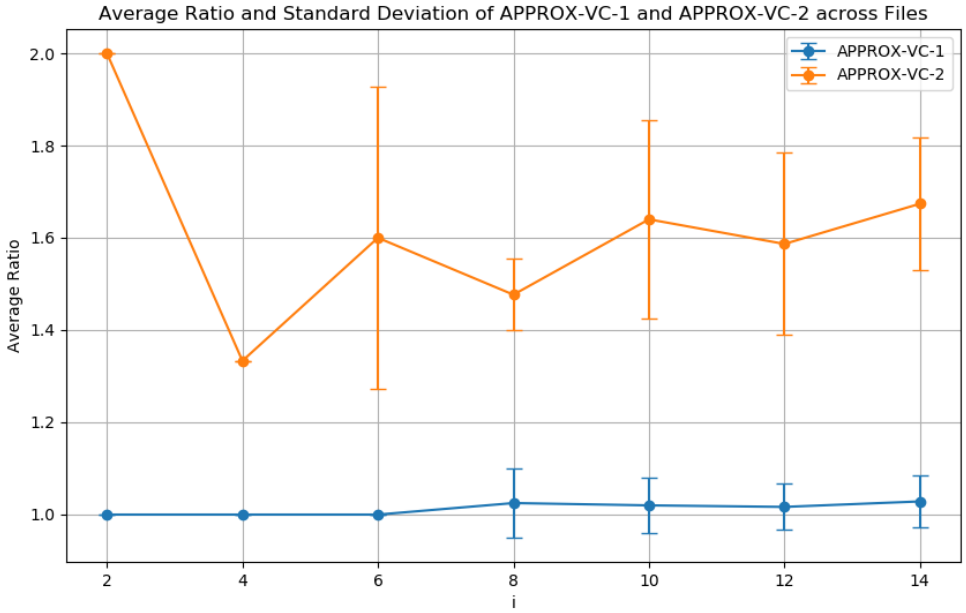

Figure 1: Approximation Ratio of APPROX-VC-1 and APPROX-VC-2

- APPROX-VC-1 generally maintains a lower approximation ratio than APPROX-VC-2 across the tested range, suggesting a closer approximation to the minimum vertex cover size obtained by CNF-SAT, indicating a relatively accurate estimation during these cases.
- APPROX-VC-2, displays higher variability and shows a trend of increasing approximation ratio as the vertex count grows. This is especially noticeable after vertices greater than 10. Despite the higher ratios compared to APPROX-VC-1, the ratios remain below 2, across the tested range. It is important to know that the ratio at V = 2 is 2.0 due to the fact the algorithm selects both vertices from the single existing edge, leading to a cover size of 2.
- The standard deviation for APPROX-VC-2 is notably larger than that of APPROX-VC-1, indicating less consistency in the performance. Interestingly, as the Vertices increase the standard deviation appears to be decreasing, however, our sample size is too small to assert this as fact.

Overall, while both algorithms provide covers larger than the optimal, APPROX-VC-1 consistently performs closer to the CNF-SAT and with less deviation in its approximation ratio, suggesting a more reliable and accurate performance in approximating the minimum vertex cover size.

3.2 Runtime

We have completed an experimental evaluation of three algorithms (CNF-SAT-VC, APPROX-VC-1, APPROX-VC-2) for solving the minimum vertex cover problem. Regarding runtime, we observed that the runtime of the CNF-SAT-VC algorithm exponentially increases with the number of vertices. It was found for vertices exceeding 20, the runtime of the CNF-SAT-VC algorithm exceeded 10 minutes. As a result, a two-minute runtime limit was set for CNF-SAT-VC; if this threshold was exceeded, the output would read ”Timeout”. This consideration was done to avoid excessive wait times for data compilation.

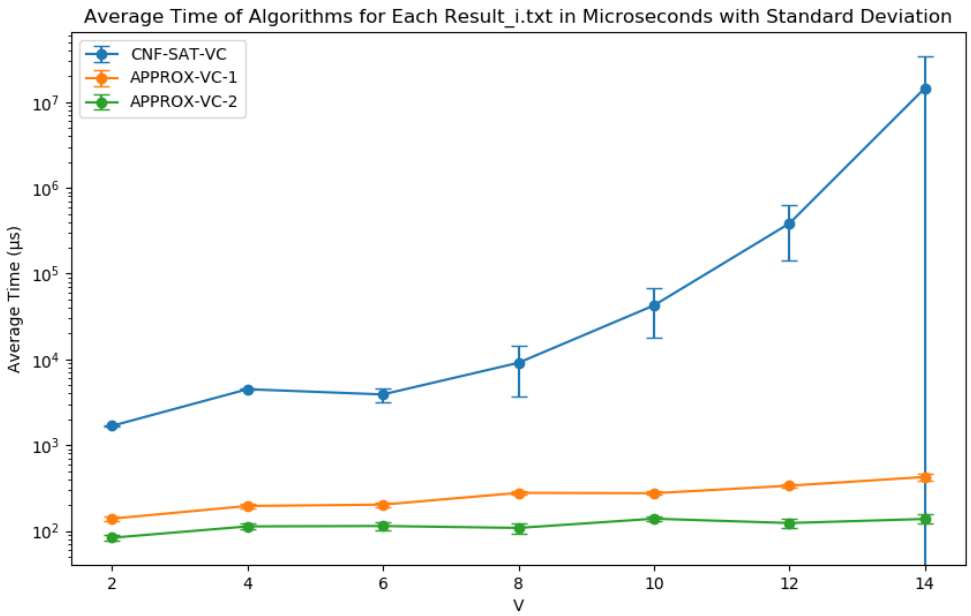

Figure 2: Run Time of 3 Algorithms With Log-scale on Y-Axis

As depicted in Figure 2, we utilized a logarithmic scale on the y-axis using microseconds as our unit of measurement. The runtime of CNF-SAT-VC consistently exceeded that of the other two algorithms. The significant increase in runtime for CNF-SAT-VC compared to the APPROX-VC algorithms can be attributed to its computational approach. CNF-SAT-VC involves a polynomial time reduction from vertex cover to CNF-SAT, which while effective, results in an exponential complexity due to the combinatorial nature of the SAT-solving process. There is a four-step process that builds a list of clauses using literals to encode the solution. This encoding requires traversing the entire matrix of clauses multiple times. This is in contrast to the heuristic approaches used by the APPROX-VC algorithms, which are designed to provide approximate solutions.

For APPROX-VC-1 and APPROX-VC-2, given their lower runtimes, we extended our analysis to vertex counts from 2 to 50 with a step size of 2, as illustrated in Figure 3. APPROX-VC-1 exhibited longer runtimes than APPROX-VC-2, with the growth rate of its runtime also significantly outpacing that of APPROX-VC-2. This difference can be explained by their approaches to finding the vertex cover. APPROX-VC-1 prioritizes vertices finding the vertex with the highest degree, which can be computationally intensive as it requires frequent recalculations

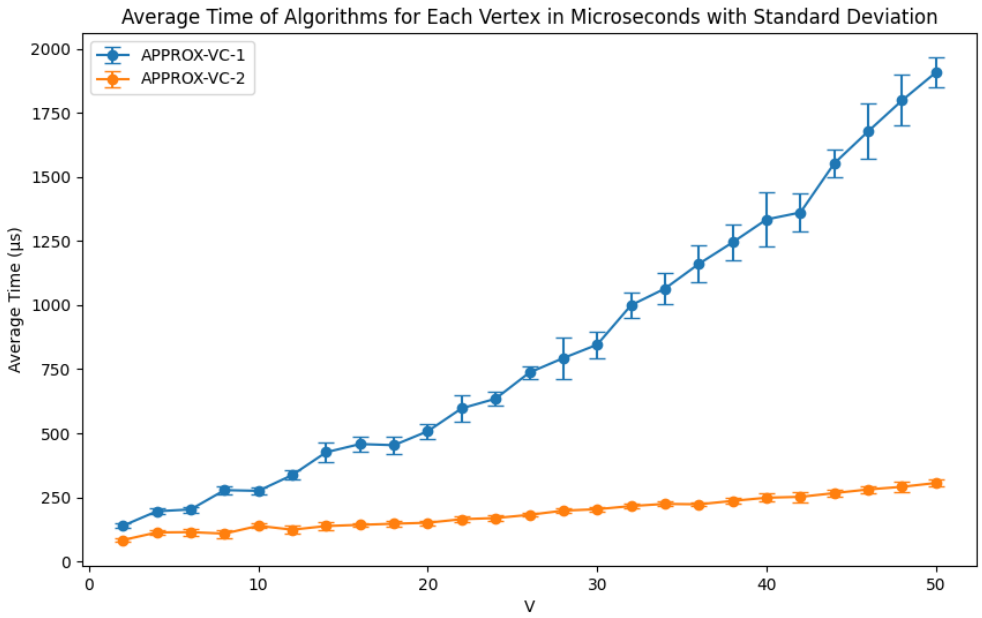

Figure 3: Runtime Performance Comparison of APPROX-VC-1 and APPROX-VC-2 Algorithms

of vertex degrees as edges are removed. On the other hand, APPROX-VC-2 randomly selects an edge and adds both of its vertices to the cover, generally requiring fewer steps to reduce the remaining graph and thus, typically running faster. The downside of the random selection is that the vertices may not be part of any optimal solutions. As described in the algorithms section, the time complexity of APPROX-VC-1 is O($m * n$) and the runtime complexity of the APPROX-VC-2 is O($m + logn$).

4 Conclusion

Our report findings outline some interesting relationships between the different algorithms. Firstly, the CNF-SAT-VC algorithm was the most accurate and served as the basis for our approximation ratio. However, in exchange for this accuracy, it was computationally and memory intensive. Between the two approximation algorithms, APPROX-VC-2 was consistently faster, but on average the computed cover was around 50% larger than both APPROX-VC-1 and CNF-SAT-VC for Vertices 2 to 14. The APPROX-VC-1 algorithm hit a perfect middle ground between accuracy and computational efficiency. On the same sets of vertices, the computed cover for APPROX-VC-1 was only around 10% larger. When taking into consideration which solution to implement in a city network, the requirements must be well defined. Do they require a perfect answer, or would a great one suffice? In a real-world scenario where costs and time limitations need to be taken into consideration, the APPROX-VC-1 implementation offers a balance of speed and accuracy.