Final Course Project

Demirci, Yekta and Sosa, Eric

December 2019

1 Introduction

The purpose of this project is to help a local police department with choosing the correct locations of their security cameras. The hypothetical police department needs to put enough cameras at traffic intersections in order to cover all edges adjacent to said intersections. The problem solved in here is choosing the minimum number of cameras to be installed in order to cover all edges. This is an optimization problem called Vertex Cover problem.

"Given a hypergraph H (V,E), a vertex cover of H is a set $C \subset V$ such that, for every $e \in E$, there exists $v \in e \cap C$. The vertex cover problem is the problem of finding a vertex cover of minimum size." [1]

We used 3 different approaches to solve the minimum vertex cover problem, in this project all three approaches run at the same time by using multi-threading.

"THREADS, LIKE PROCESSES, ARE A MECHANISM TO ALLOW A PROGRAM to do more than one thing at a time. As with processes, threads appear to run concurrently; the Linux kernel schedules them asynchronously, interrupting each thread from time to time to give others a chance to execute." [2]

1.1 Approaches

- The first approach was implementing a polynomial time reduction to CNF-SAT using minisat library in order to solve the minimum vertex cover problem.
- The second approach was implemented by picking the vertex of highest degree (most incident edges) and removing all edges incident to that vertex. This was repeated until there were no edges left. We will call this algorithm APPROX-VC-1.
- The third approach was to pick an edge randomly and adding both of its vertices to the vertex cover. Then, we removed all edges attached to those vertices and repeated until there were no edges left. We will call this algorithm APPROX-VC-2.

2 Analysis

The output from /home/agurfink/ece650/graphGen/graphGen on eceubuntu was used to generate our graphs. GraphGen always generates the same number of edges for a number of vertices, but the edges that it produces are different. For this project, 10 graphs for each number of vertices within [5:50] in increments of 5 were tested. That is, graphs with 5; 10; 15; : : : ; 50 vertices.

We used pthread_getcpuclockid() to measure the time that it took each of the approaches to compute the vertex cover. We also calculated the approximation ratio for both of the approximations. The approximation ratio is the ratio of the size of the computed vertex cover to the size of a minimum vertex cover which is given by the CNF-SAT.

Each of the graphs was tested 10 times. We then calculated the mean and the standard deviation across those 100 runs for each different number of vertices using the 3 approaches.

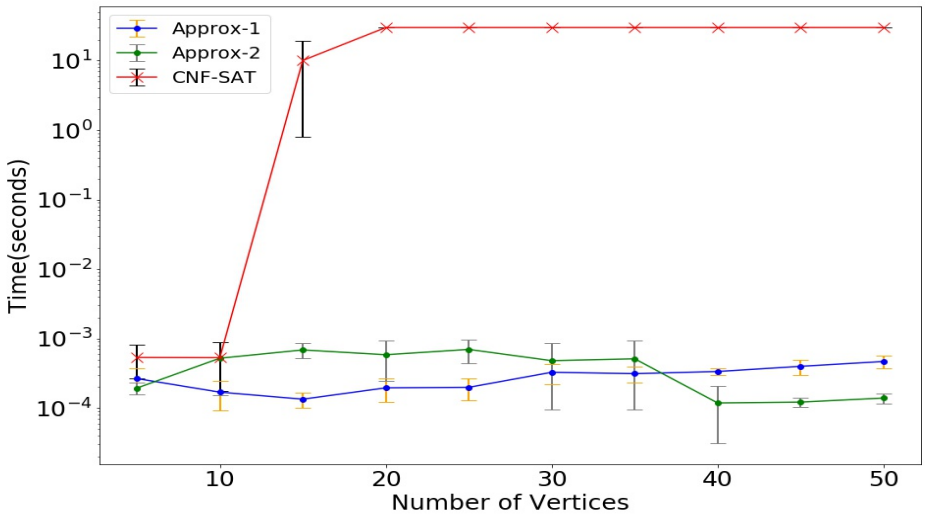

Figure 1: Logarithmic Time Comparison.

Fig. 1 represents the time measurement in logarithmic scale across all of the tests afore mentioned. Logarithmic scale was used in this case in order to be able to compare the behavior of all the approaches since CNF-SAT time grew exponentially, which can be seen here.

A 30 seconds timeout was implemented into CNF-SAT since the approach takes too long for large number of vertices. This can be seen in Fig. 1 as the horizontal portion of CNF-SAT which occurs after 20 vertices, this means we were only able to solve graphs with 5, 10 and 15 vertices using this approach.

CNF-SAT took a similar amount of time to both approximations in small graphs but the amount of clauses it needs to analyze grows exponentially for each vertex added.

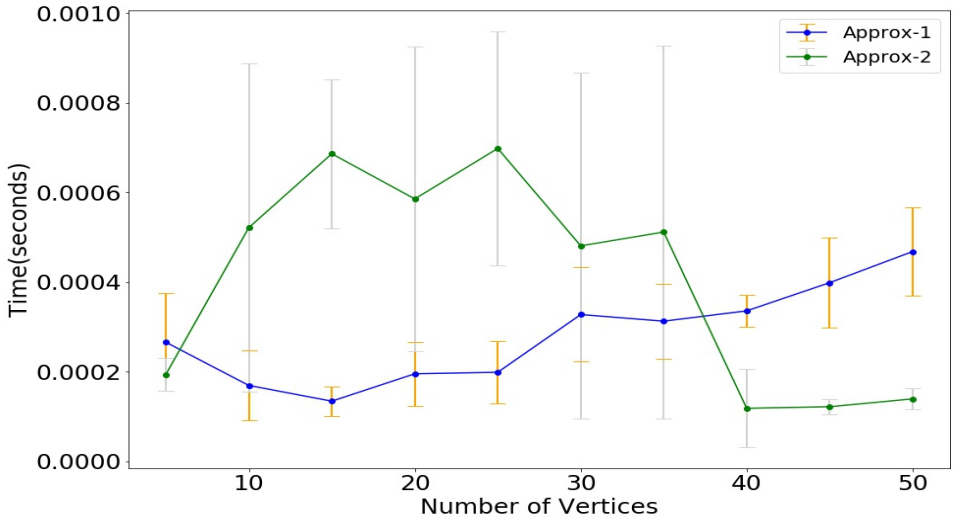

Figure 2: Time Comparison of Approximation 1 and 2.

As seen in in Fig. 2 approximation 2 takes less time when having 5 vertices, this is probably due to the

fact that Approx-1 needs to find the highest degree vertex while Approx 2 selects an edge randomly, the benefits of Approx 1 begin at graphs of size 10.

The randomness of Approx-2 regain its time benefits somewhere after 35 vertices as can be seen with the intersection of the 2 plots.

It is important to mention that there is a high variance in the time used for Approx-2 which decreases significantly after 35 vertices, this is probably due to the fact that selecting the best edge at the beginning is not relevant anymore as there are several that can be good first choices. The standard deviation in Approx-1 stayed within the same range across all graph sizes.

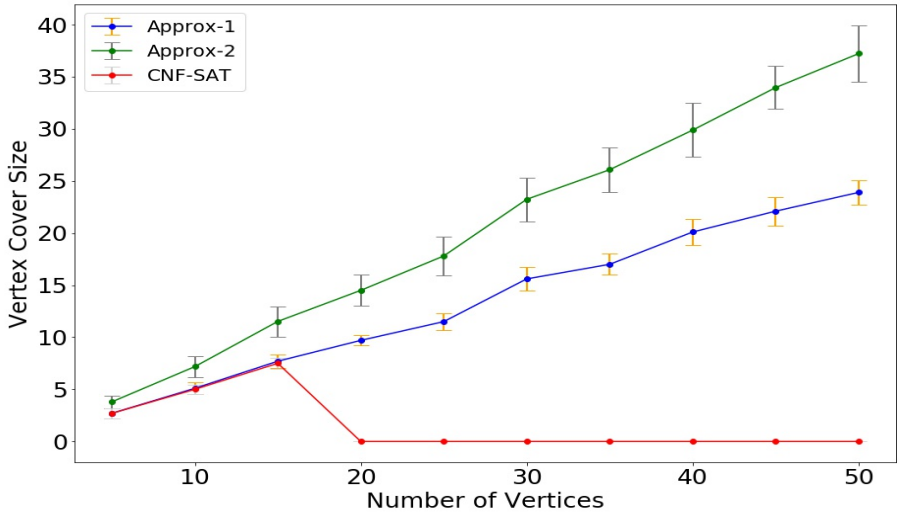

Figure 3: Vertex Cover Size Comparison.

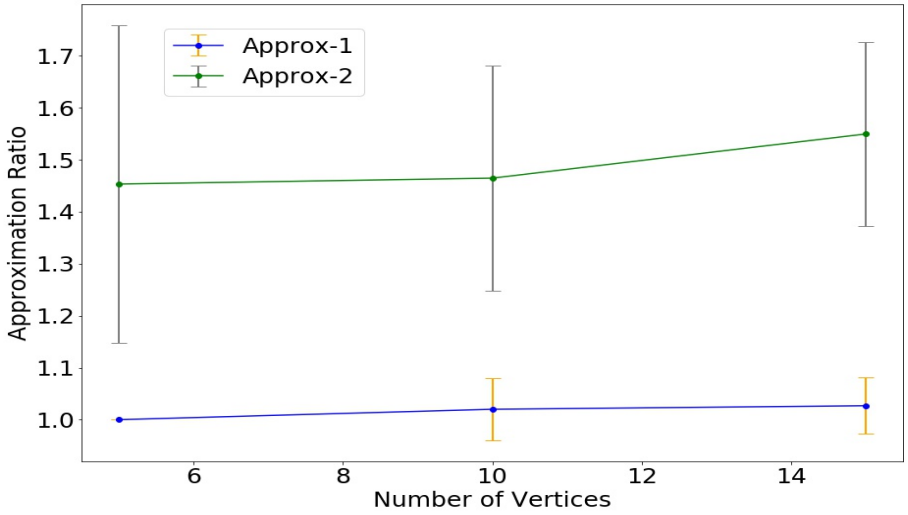

Figure 4: Approximation Ratio Comparison.

From Fig. 3 we can observe that Approx-1 has a lower Vertex Cover Size than Approx-2. Unfortunately CNF-SAT needed to be aborted after 30 seconds without being able to get any minimum Vertex covers for graphs above 15 vertices. Although, it could be inferred from the first 3 measurements that Approx-1 has a similar vertex cover size to the one from CNF-SAT.

According to the time needed to compute the vertex cover Approx-2 is the most efficient approach, but this is not the case for its approximation ratio which can be seen in Fig. 4. In this case Approx-1 is obviously a better choice by having a similar vertex cover size to the minimum of CNF-SAT.

Approximation 2 used a random approach to finding the vertex cover, for this reason, the following graphs were created in order to analyze its standard deviation in each of the graphs.

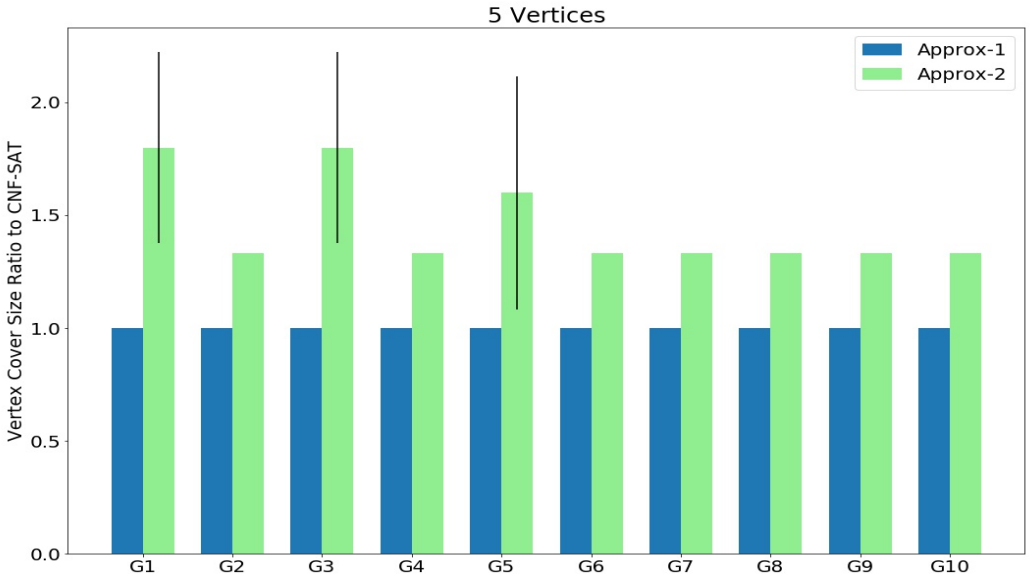

Figure 5: Approximation Ratio for Each 5 Vertex Graph.

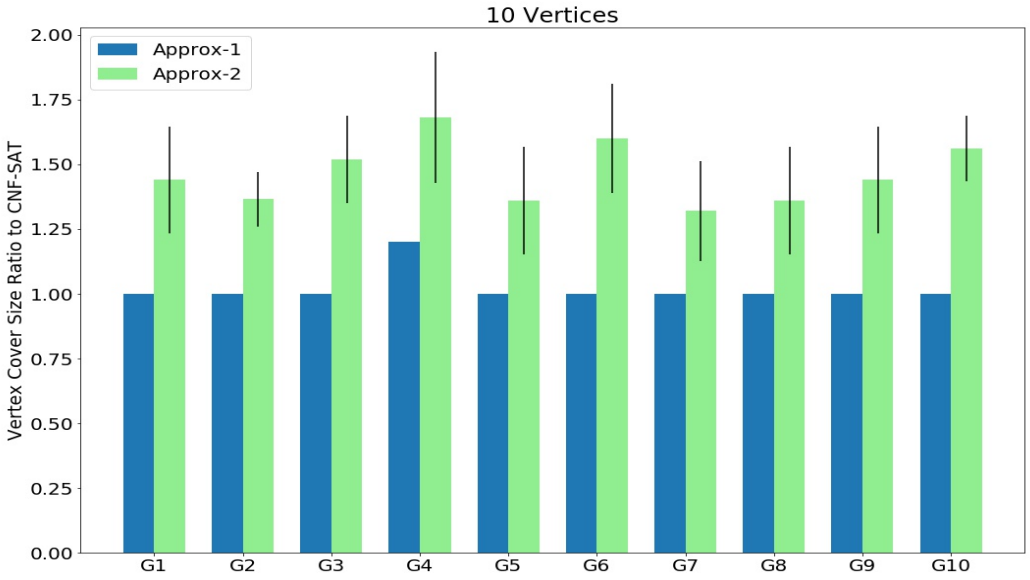

Figure 6: Approximation Ratio for Each 10 Vertex Graph.

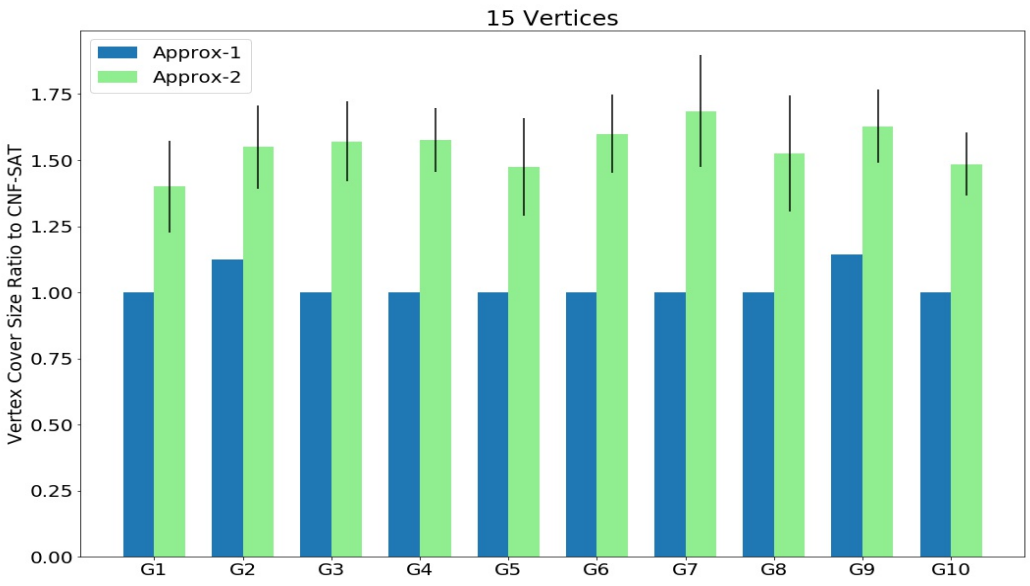

Figure 7: Approximation Ratio for Each 15 Vertex Graph.

From Fig. 5 we can see that Approx-1 always gave the minimum vertex cover while approx-2 never did since it always uses at least 2 vertices when probably only 1 was necessary. There was some standard deviation on only 3 graphs, this is due to the low number of possibilities when having only 5 vertices.

From Fig. 6 and Fig. 7 we can confirm that Approx-1 almost always gave the minimum vertex cover, while the approximation ration for Approx-2 increases when the number of vertices increase. There was always some variance in Approx-2 due to its random approach.

3 Conclusion

In order to be able to use CNF-SAT we would need to implement a significant reduction to its encoding, otherwise it is almost impossible to use for any graph with more than 20 vertices. Using Approx-1 is a good compromise in every sense, if a police department was already buying 20 video cameras, buying 1 more than the minimum would be completely reasonable.

References

[1] Halperin, E.: Improved approximation algorithms for the vertex cover problem in graphs and hypergraphs. SIAM J. Comput. 31(5), 1608–1623 (2002)

[2] M. Mitchell, J. Oldham and A. Samuel, Advanced Linux programming. Indianapolis, Ind.: New Riders, 2003.