ECE 650 Final Project Report

Youmei Zhang, 20706451, ECE

1. Background

In this project, we implemented three different algorithms for computing the Vertex Cover for a given graph, namely CNF-SAT, Approx-1 and Approx-2. In this report, we explain and analyze the theory behind these three methods. Then, we run these methods for graphs with different numbers of vertices. For each vertex size |V|, we generate 10 graphs and run 10 times for each. Combining with these data we achieved from experiments, we did the analysis and draw some conclusions.

2. Analysis of Running Time

For Approx-1, even though it is not guarantee to output the minimum vertex cover, it is thought to be highly efficient. All we need to do is just scan the input graph once. The worst case for Approx-1 is that we scan every vertices in the graph, so we can calculate the upper bound for this algorithm is $O(|V|)$ which is linear with the size of input. Therefore, even though the number of vertices keeps increasing, the actual running time will not increase dramatically as CNF-SAT which we will see later in the graph. Especially in our case, the numbers of vertices are relatively small which are all under 20. The change of running time for Approx-1 is relatively small.

For Approx-2, it is similar to Approx-1, we pick an edge randomly and throw away other edges that are connected to these two vertices and then repeat the process until no edge left in the graph . The worst case is that we scan every edge in a given graph. Therefore the upper bound for the running time is $O(|E|)$ which is linear with the size of input as well.

For CNF-SAT, since SAT Solving is a NP-complete problem. The worst case of running time is $2^{|V|}$ which is exponential. We can see from Assignment 4 that the number of clauses in the reduction is $k + n\binom{k}{2} + k\binom{n}{2} + |E|$ where k is the vertex cover size and n is the number of vertices. The running time depends of n, k and $|E|$. Therefore, when the number of vertices get increasing, the running time will get dramatically increasing as well.

Fig.1 is the running time of these three algorithms. From this figure, we can easily conclude that when the number of vertices are below 11, the running time for all three methods does not show a significant difference. However, with number of vertices increasing to 15, the running time increases dramatically for CNF-SAT. The average running time is around 2000000 us which is around 2 seconds for each run. However, unlike CNF-SAT, both Approx-1 and Approx-2 are still very stable and quite similar. They don't change a lot and it is nearly constant time compared with CNF-SAT when the number of vertices are around 15 which is consistent as the theory that we analyze above.

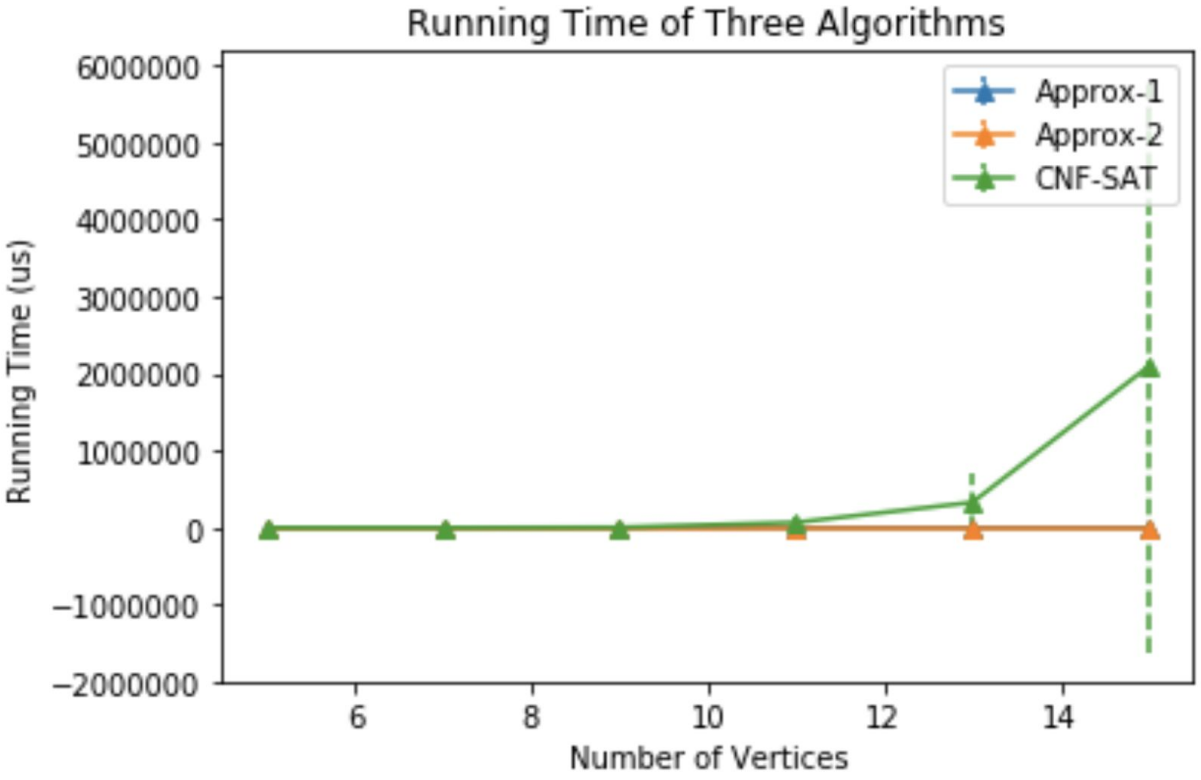

Fig.1 Comparison of running time for three methods

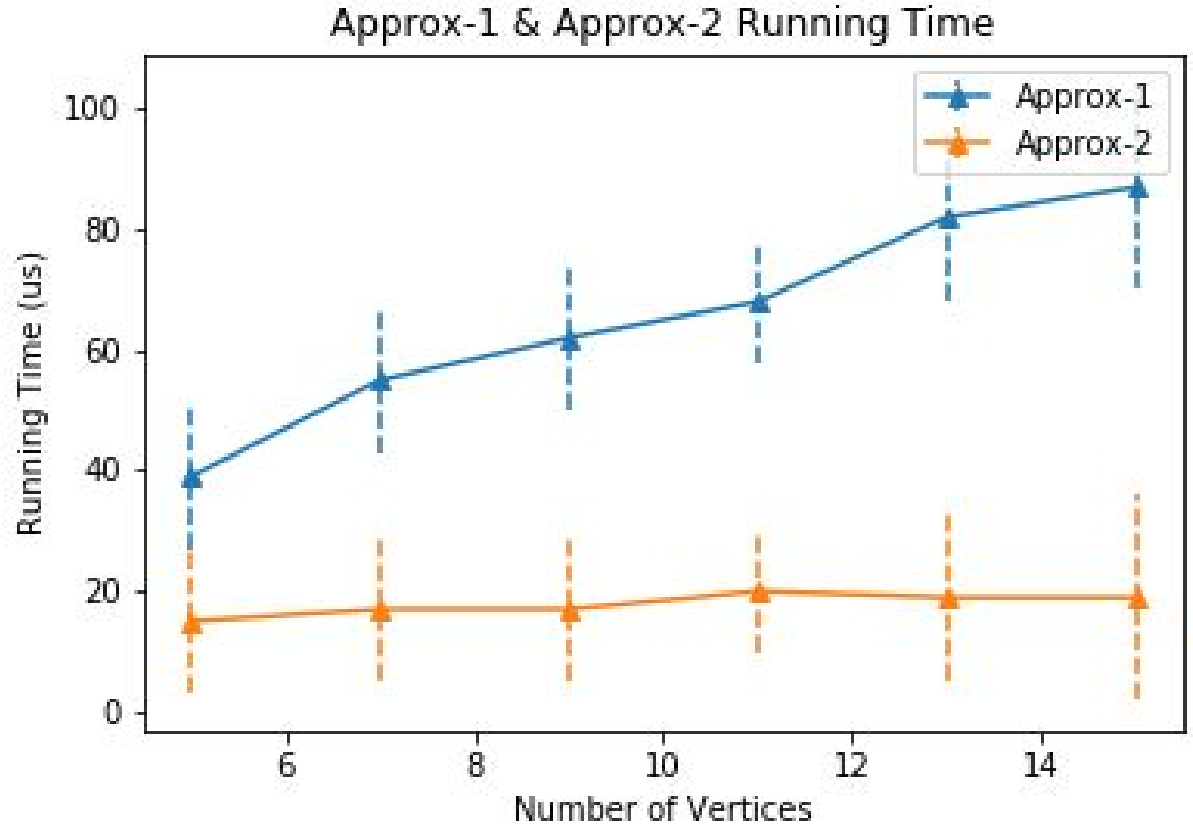

Fig.2 Comparison of running time for Approx-1 and Approx-2

The standard deviation is very different as well. For CNF-SAT computing vertex cover for a graph with 15 vertices, the standard deviation is 3715849us which is around 3.7 seconds while the standard deviations of Approx-1 and Approx-2 are still very small.

For better comparing the difference between Approx-1 and Approx-2, we plot the result in Fig.2 as shown above. On average, Approx-2 takes less time than Approx-1. With the number of vertices increasing, the running time of Approx-1 gets increasing steadily while for Approx-2, it still does not show any significant changes. The running time for Approx-1 is around 40us - 90us while for Approx-2 is around 18us. The difference between these two are very small and one of the reason for this might be the way of implementing the algorithms. We can see that the standard deviation does not change significantly no matter the number of vertices is large or small.

3. Analysis of Running Time

We calculate the approximation ratio based on the ratio of the size of the computed vertex cover to the size of the vertex cover produced by CNF-SAT which is guarantee to be optimal. For Approx-1 and Approx-2, suppose $c*$ is the minimum size for a vertex cover of a graph G, then the output C produced by Approx-1 or Approx-2 on input graph G is guaranteed to be $|C| \leq 2c*$. Because at least one of u or v must be in every vertex cover. Thus, the size of the output from these two methods can be no more that twice a minimum sized vertex cover. Therefore, we can check on Fig.3 that the approximation ratio will never exceed 2.

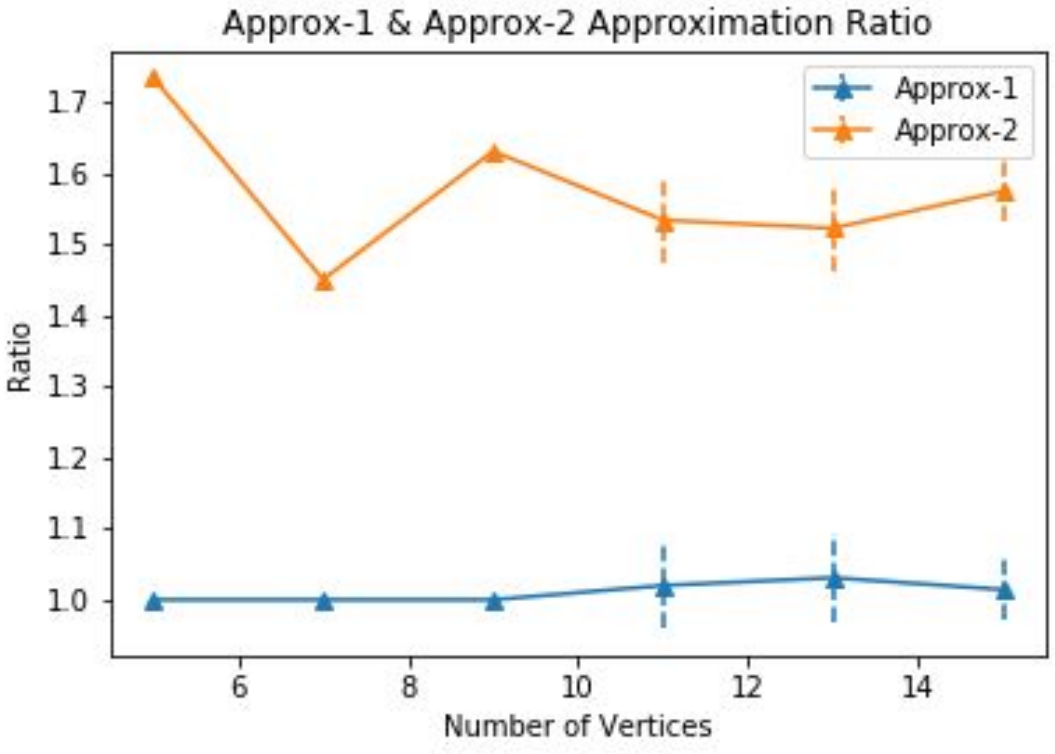

Fig.3 Comparison of approximation ratio for Approx-1 and Approx-2

Now, we explain why Approx-2 is slightly inefficient in terms of approximation ratio compared with Approx-1. Because in Approx-1, we are guarantee that we pick the vertex with most vertices connecting to it, in this way we are trying to minimize the number of vertices needed to cover all the edges. However, in Approx-2, at each round, the edge $< u, v >$ is picked randomly, it is possible that a vertex with only another one vertex connecting to it is picked and both u and v are added to the result. But actually, only u or v is needed. This might result in more vertices so as to cover all edges in the graph. As a consequence, it might pick more vertices than Approx-1. Therefore, the approximation ratio for Approx-2 is on average larger than Approx-1.

From Fig.3 above, the experiment result also shows that the approximation ratio for Approx-2 is larger than Approx-1 which shows its less efficiency. For all the number of vertices tested, the approximation ratio for Approx-2 is above 1.4 which means on average if the optimized vertex cover is only 10, then Approx-2 will output a vertex cover with size 14. However, even though Approx-1 takes more time to run than Approx-2, the vertex cover it outputs is definitely better than Approx-2 and it stays around 1. This shows that Approx-1's output is almost as optimal as the output that CNF-SAT gives. That's also consistent with what we mention before that Approx-1 is highly efficient.

4. Conclusion

After comparing these three methods, we can conclude that Approx-1 algorithm is a highly efficient algorithm to solve the vertex cover problem. Even though the output produced by Approx-1 is not optimal, we can still accept it. However, for achieving the optimal results, we have to sacrifice so much time especially when the number of vertices is above 13.