Analysis on different approaches to solve the minimum vertex cover problem

Shiyun Qin, (20819760) and Weng Yu, (20826718)

Abstract—This is a final course project of ECE 650 section 1 in group of 2. For the project, this paper analysis on different approaches to solve the minimum vertex cover problem. The minimum vertex cover problem are solved by three approaches: CNF-SAT, APPROX VC-1, and APPROX-VC-2, and the efficient of output for each approach are analyzed. An input is characterized by the number of vertices. “Efficient” is characterized in one of two ways: (1) running time, and (2) approximation ratio. In this paper, the efficient of running time and approximation ratio are expected in the method through calculation, and the result of data can prove the expectation through plot and analysis.

Index Terms—Vertex Cover, CNF-SAT, Running time, Approximation ration, LaTeX.

1 Introduction

FROM the definition of vertex cover problem, a vertex cover of a graph is a set of vertices such that each edge of the graph is incident to at least one vertex of the set. For an undirected graph $G =< V, E >$, the vertex cover is a subset of the vertices $VC \subseteq V$, where for every edge (u, v) of the graph E either $u \in V$ or $v \in V$ is in the set. In this paper, there are three difference approaches to solve the vertex cover problem: CNF-SAT, APPROX-VC-1, and APPROX-VC-2, and each approach have unique efficient of running time and approximation ratio.

We characterize the approximation ratio as the ratio of the size of the computed vertex cover to the size of an optimal (minimum-sized) vertex cover. For measuring the approximation ratio, compare the output of APPROX-VC-1 and APPROX-VC-2 to the output of CNF-SAT-VC, which is guaranteed to be optimal, which means it can output exact minimum number of vertex cover. In the Method section, we discussed the approximation ratio for APPROX-VC-1 and APPROX-VC-2 through prove, and the data certificate our prove. According to the ratio for APPROX-VC-1 and APPROX-VC-2, the size of vertex cover of those approach should not go over the optimal result multiply the ratio for same graph. Vertex Cover Problem is a known NP Complete problem, there is no polynomial time solution for this unless P = NP. There are approximate polynomial time algorithms to solve the problem though. For measuring the running time, the optimal approach (CNF-SAT-VC) difficult to scale to large number of vertices. In order to record the data for large graph, we also talk about the improvement of the optimal approach (CNF-SAT-VC) in the paper. Because the vertex cover problem is NP-complete finding an exact answer is very difficult and time consuming. Many times, approximation algorithms are useful. These run much faster than exact algorithms, but may produce a sub-optimal solution.To measure the running time, we run each graph for at least 10 runs and compute the mean (average) and standard deviation across those 100 runs for each value of $|V|$.

2 Background

As one of the fundamental NP-hard problems, the minimum vertex cover problem attracted notable attention during the recent few decades [1]. Explained by Gary and Johnson, all NP-complete problems share two common properties:

- No NP-Complete problem is known as solvable in polynomial time.
- If a polynomial-time solution is found for any NP-Complete problem, all NP-Complete problems can be solved in polynomial time.

Although, try to explicitly locate the minimum vertex cover is very difficult, a curve ball can be thrown here for finding an approximation solution is relatively easy. So far, various approximation algorithms have been introduced. A qualified approximation algorithm must generate a reasonable solution within the complexity of polynomial time. Such a method is then evaluated in terms of running time and approximation ratio. The approximation ratio of the chosen algorithm is the ratio between the result of such an algorithm and an optimal approach. In other words, the smaller the approximation ratio is, the more proximate the result of the algorithm is from the optimal approach.

In 2001, Chen, Kanj, and Jia indicated several new properties of the vertex cover problem and reduced the running time of the solving algorithm to O(1.2852k + kn) [2]. Superior to that, in 2010, Chen, etc. presented a momentously improved algorithm and further reduced the running time to O(1.2738k +kn) [2].

3 Method

In this final project, we argument our code in the following ways. This project used multithreaded, there are 4 threads in the code: one for I/O, and one each for the different approaches to solve the minimum vertex cover problem. For the efficient of running time and approximation ratio, the following subsection introduce the pseudo code and running time for each approach.

3.1 CNF-SAT-VC

Notation: Suppose, graph $G = (V, E)$ throughout this paper are considered undirected and unweighted. We denote n as the number of vertices $(|V|)$, and m as the number of edges $(|E|)$. For any vertex v $\exists$ V, we denote by N(v) the set of adjacent vertices of v that are the set of vertices sharing an edge with v, and d(v)) the degree of v. A cover K of G, where k $\leq$ n, is a subset of V that for every edge(i, j) there exists a vertex x in K such that x is equal to either i or j. The following pseudo code is the version before optimization. For the vision we create 4 clauses for the minimum-set, whihc present a polynomial-time reduction from VERTEX-COVER to CNF-SAT. A polynomial-time reduction is an algorithm that runs in time polynomial in its input.

Run time For $m = [n(n-1)]/2$, we calculate the worst run time as below:

$$\begin{aligned} C &= O(k*n + n*k*k + k*(n-1)*(n-1) + 2mk) \\ &= O(nk2 + kn2 + mk) \\ &= O(nk^2 + kn^2) \end{aligned} \quad (1)$$

Algorithm 1 CNF-SAT-VC

```
1:  procedure GETCNF()
2:      $k \leftarrow 1int$
3:      $solver \leftarrow$ new Solver()
4:      while k <n do
5:          for ci=0; ci$\leq$ k; ci++ do
6:              vector clause1
7:              for cn = 0; cn < n; cn++ do
8:                  clause1 $\leftarrow$ ci,cn
9:              solver $\leftarrow$ clause1
10:         for cn=0; cn <n; cn++ do
11:             for cp=0; cn <k; cp++ do
12:                 itrp $\leftarrow$ cm,cp
13:                 for cq=0; cn$\leq$ n; cq++ do
14:                     itrq $\leftarrow$ cm,cq
15:                     solver $\leftarrow$ clause2( $itrp$, $itrq$)
16:         for ck=1; ck$\leq$ k; ck++ do
17:             for cp=0; cn <n-1; cp++ do
18:                 itrp $\leftarrow$ cp,ck
19:                 for cq=cp+1; cn <n; cq++ do
20:                     itrq $\leftarrow$ cq,ck
21:                     solver $\leftarrow$ clause3( $itrp$, $itrq$)
22:         for ce=1; ce$\leq$ m; ce++ do
23:             vector clause4
24:             Edge(i,j) $\leftarrow$ m[ce]
25:             for ck=1; ck$\leq$ k; ck++ do
26:                 itrp $\leftarrow$ i,ck
27:             for ck=1; cn$\leq$ k; ck++ do
28:                 itrq $\leftarrow$ j,ck
29:                 solver $\leftarrow$ clause4( $itrp$, $itrq$)
30:             solver $\leftarrow$ clause4
```

3.2 APPROX-VC-1

Notation: The following is the first greedy algorithm to compute a minimum-sized vertex cover of an input undirected graph $G =< V, E >$. Repeatedly pick a vertex of highest degree, and remove all edges that have been covered by it before picking the next vertex. This algorithm has an polynomial running time, but the number of vertex cover may not by the minimum.

The algorithm works as follows. The set U contains, at each stage, the set of remaining uncovered vertex. The set VC contains the cover being constructed. Line 5 is the greedy decision-making step, choosing a subset S that covers as many uncovered vertex as possible. After S is selected, line 6 removes its vertex from U, and line 7 places S into VC. When the algorithm terminates, the set VC contains a subset of E that covers V.

Algorithm 2 APPROX-VC-1

```
1: procedure GEEDY-VERTEX-COVER($V, E$):
2:     $U \leftarrow V$
3:     $VC \leftarrow \emptyset$
4:     while $U$ not $\emptyset$ do
5:         select $S \in E$ that maximizes $S \cap U$
6:         $U \leftarrow U - S$
7:         $VC \leftarrow VC \cup \{S\}$
8:     return $VC$
```

It can easily implement GREEDY-VERTEX-COVER to run in time polynomial in V and E. Since the number of iterations of the loop on lines 4–7 is bounded from above by $min(|V|, |E|)$, and it can implement the loop body to run in time $O(|V||E|)$, a simple implementation runs in time $O(|V||E|min(|V|, |E|))$.

This greedy vertex cover has size $\leq \ln(n)\cdot$ minimum size of CNF-SAT (optimal solution) where n is the size of input.

$$\begin{aligned} |VC| &\leq \sum_{S \in \text{VC}^*} H(|S|) \\ &\leq |\text{VC}^*| \cdot H(|S| : S \in V) \end{aligned} \tag{2}$$

Where VC is the Greedy Vertex Cover and VC* is the optimal approach. The ratio of greedy and optimal is the harmonic function H. The result of function H is $\ln|n| + 1$-approximation algorithm, so the upper bound is $O(\log n)$. Therefor, Vertex Cover is efficiently approximately with a ratio of $O(\log n)$, where n is the size of input.[4]

3.3 APPROX-VC-2

Notation: The following is the first greedy algorithm to compute a minimum-sized vertex cover of an input undirected graph $G =< V, E >$. Random pick an edge $< u, v >$ from E, and add both u and v to your vertex cover VC. Throw away all edges attached to u and v. Repeat till no edges remain. The set of edges picked by this algorithm is a matching, no 2 edges touch each other (edges disjoint). In fact, it is a maximal matching. We can then have the following alternative description of the algorithm as follows.

- Find a maximal matching M
- Return the set of end-points of all $edges \in M$.

This algorithm gives a greedy approach of vertex cover. Every $edge \in M$ is clearly covered. If an edge, $e \in M$ is not covered, then $M \bigcup e$ is a matching, which contradict to maximality of M.

Algorithm 3 APPROX-VC-2

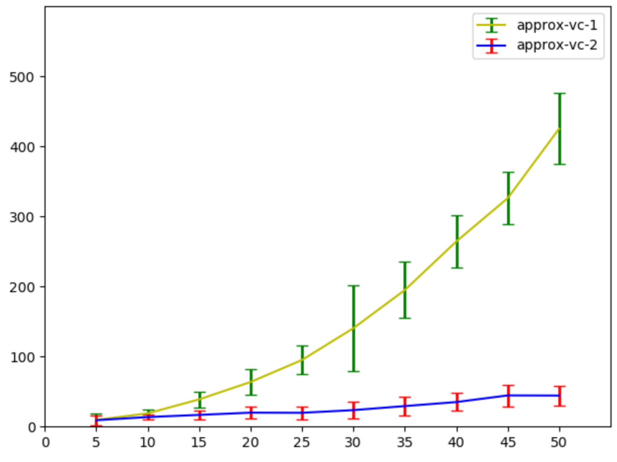
```
1: procedure GEEDY-VERTEX-COVER($V, E$):
2:     $VC \leftarrow \emptyset$
3:     while $E$ not $\emptyset$ do
4:         pick any $\{u, v\} \in E$
5:         $VC \leftarrow VC \cup \{u, v\}$
6:         delete all edges incident to either $u$ or $v$
7:     return $VC$
```

This GREEDY-VERTEX-COVER also run in time polynomial in V and E. Since the number of iterations of the loop on lines 3–7 traverse E and V, so the time efficiency of this algorithm is $O(|V||E|)$.

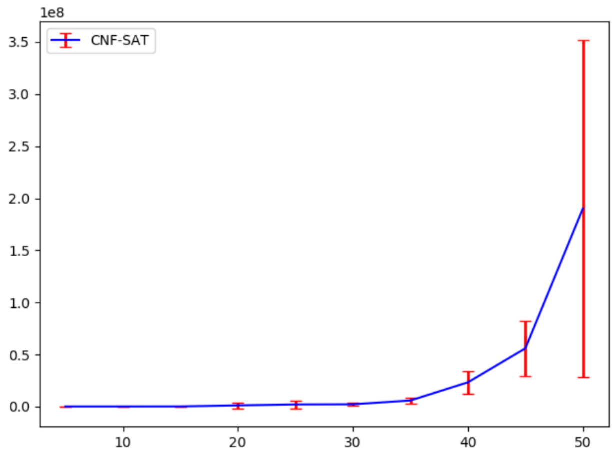
This greedy vertex cover has size $\leq 2 \cdot$ minimum size of CNF-SAT (optimal solution). The optimum vertex cover must cover every edge in M. So, it must include at least one of the endpoints of each edge $e \in M$ where no 2 edges in M share an endpoint. Hence, optimum vertex cover must have size. $CNF-SAT(i) \geq |M|$ But the greedy algorithm $APPROX - VC - 2$ return a vertex cover of size $2|M$—, so $\forall i$ we have APPROX-VC-2(i) $= 2|M| \leq 2 \cdot$ CNF-SAT(i) implying that APPROX-VC-2 is a 2-approximation algorithm.

4 Optimization

Method 1: in the notation we assumed $|V| = n$. However, there may exist the case when $|V|$ is greater than n, denote n' which n' stand for the actual number of vertex in $G(V, E)$, and n denotes the maximum number of vertices allowed in graph G. In this case, we rename the vertex in n′ from $1...n$ to $1..n'$ where $1 < n' < n$. Thus, we can reduce the matrix size from $n \cdot k$ to $n' \cdot k$ which is guaranteed to be smaller. Method 2: after optimization through method 1, we notice the run time significantly improved when $n' < 10$. However, it barely changed in larger test case. Assume at iteration k', $1 < k \leq k$, we have conducted a matrix size $n \leq k'$. Assume the minimum vertex cover is first to be found in this iteration, and let set $S[1...j]$ denotes all other potential solutions. Then $\forall i$, $1 \leq i \leq j$, $k' \leq S[i]$. In iteration k', there are at most k'! different ways to rearrange such solution which means we have k'! Solutions to graph G. By limiting the order of k', we can limit the number of solutions from k'! to 1. Denote M as the matrix $n \cdot k'$. Assume Edge $E(i, j)$ in M is picked at iteration s. Then at iteration s', $E'(i', j')$ is chosen if and only if ($i' < i$ and $j' < j$) or ($i' > i$ and $j' > j$).



(a) Approximate Approach



(b) CNF-SAT

Fig. 1. Running time of minimum vertex cover. Left plot shows running time of APPROX-VC-1 and APPROX-VC-2. Right plot shows running time of CNF-SAT. The x-axis is the number of vertex and the y-axis is the running time(μs)

5 Data

For recording data, which consist of the number of vertices $|V|$ and a list of edges E, are used as input for the three algorithms. Each pair of $|V|$ and E is considered as a graph. All graphs are generated by the given *graphGen* program, which generate graphs for $|V| \in [5, 50]$ using that program, in increments of 5. That is, graphs with $5, 10, 15, ..., 50$ vertices. Graphs with the same number of vertices have the same number of edges but more vertices come with more edges. The running time and approximate ratio are measured by 10 runs for each graph.
The evaluation statistics primarily have two values, the running time and approximation ration. The running time is the time consumption of CPU of each algorithm on Linux system. The unit of running time is microsecond (μs). For each different number of vertices, mean and standard deviation are computed for both running time and approximation ratio based on the 100 runs.

5.1 Running-time

The Figure 1 shows the running time of vertex cover where (a) is the plot of Approx-VC-1 (green line) and Approx-VC-2 (blue line) and (b) is the plot of CNF-SAT (blue line). The x-axis is the number of vertex and the y-axis is the running time(μs). For each value of $|V|$ in graph, we plotted the mean value and the the standard deviation as an errorbar. Since the running time of CNF-SAT is much greater than approximate approaches, so they can not plot in one graph.

5.2 Approximation ratio

The Figure 2 shows the approximation ratio for each value of $|V|$ where the green line is the plot of Approx-VC-1 ratio CNF-SAT and blue line is Approx-VC-2 ratio CNF-SAT. The x-axis is the number of vertex and the y-axis is the approximate ratio. For each value of $|V|$ in graph, we plotted the mean value and the the standard deviation as an errorbar, which is the vertical line on each vertex. Since the CNF-SAT-VC algorithm guarantees the optimal solution, so the approximation ration is always 1. It is more meaningful to analyze the approximation ratio of the other two algorithms.

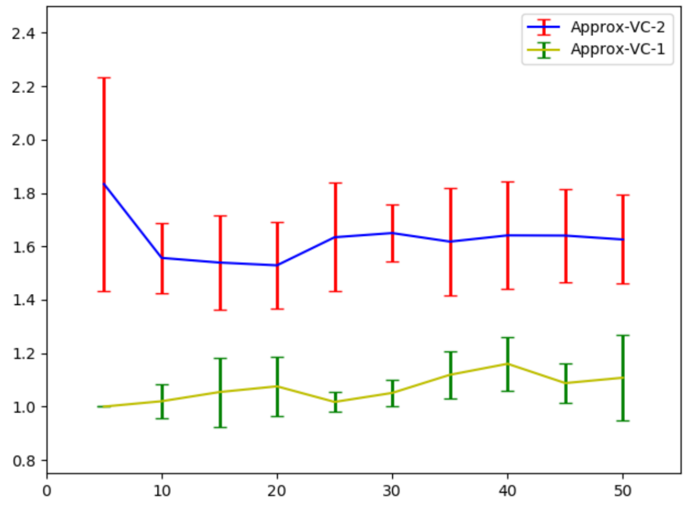


Fig. 2. Approximation Ratio of APPROX-VC-1 ratio CNF-SAT and APPROX-VC-2 ratio CNF-SAT where the x-axis is the number of vertex and the y-axis is the running time(μs)

6 Analysis

The three algorithms used to solve vertex cover problem are explained and analyzed in this section based on their running time and approximation ratio.

6.1 Running time

As Figure 2 (a)shows, the mean running time of APPROX-VC-1 and APPROX-VC-2. Due to the increase of vertices number, the data in vertices and graph would also increase accordingly because the running time for these two approach depend on the size of input.

For the APPROX-VC-1, the time of iterating the graph, the number of comparisons between sizes subset, and the remove of highest degree vertex from its neighbors' vectors would all accumulate and rise. It also indicates the standard deviation have a slightly increase with the increase of running time. This is because when we have more vertex, there are also more subset of vertex we can pick. If we a lucky enough, we only need iterate a few subset, but we also may iterate all subsets, which will cost a high standard deviation.

Compare to the APPROX-VC-1, although the APPROX-VC-2 is increasing, but it does not increase as fast as first approach. The running time of these two approaches are both polynomial time and base on the size of input, but the reason that causes this difference between two running time is APPROX-VC-1 is using fixed solving steps, but the APPROX-VC-2 is randomly pick edge from the graph. For the APPROX-VC-2, the increased number of vertices will cause more edges in the graph.

In the Figure.1 (b), it shows the running time for CNF-SAT. Dose not like the greedy algorithms, the CNF-SAT use the external library miniset to find vertex cover. Therefore, the main part of CNF-SAT is the implementation of reduction between CNF-SAT and vertex cover. According to the graph, we can observe that although we optimized the clause, the running time of CNF-SAT is still mush slower than the others. The number of vertices $|V|$ and the number of edges $|E|$ is related to the running time. As Figure.1 (b) shows, the running time increases when the number of vertices increases. This trend is expected due to the increase of running time of nested for-loops. When the sizes of input become greater, it takes more time for nested for-loops to iterate the vectors proportionally. There is another factor affecting the running time, which is the size of vertex k. Since the size of k is from 1 to $|V|$, so the more number of vertex in the graph, the greater k need to test.

6.2 Approximate ratio

As mentioned before, the approximation ratio of CNF-SAT-VC algorithm is always 1, so the figure only have two plot: APPROX-VC-1 ratio optimal and APPROX-VC-2 ratio optimal.

In the Figure.3, the mean approximate ratio of APPROX-VC-1 is increasing apparently at each vertex. This phenomenon is reasonable because when the number of vertex is small, the vertex have more possible to cover all the edge in the graph, so the number of vertex is close to the optimal answer. If the vertex has the largest size of neighbor vector, but its neighbor vector contains all next highest degree vertices, then this current vertex is extra and can be deleted from final vertex cover. The approximate ratio do not go over $\log n$, which match the previous expectation.

For the APPROX-VC-2, the mean approximate ratio is increasing apparently at each vertex. As the Figure.3 shows, the average approximation ratio fluctuates around 1.6, which is less than $2\cdot$ the minimum verter cover. In addition, there is obviously large standard deviation at each vertex. This is because the selection of an edge is random. Sometimes the two vertices have higher degree, sometimes they do not.

7 Conclusion

In this final course project, it analysis the difference result of three approaches to solve minimum vertex cover problem. Through research and analysis on running time and approximate ratio, the data can be conclude as the APPROX-VC-2 has the minimum running time overall among all three algorithms, but it have greatest approximate ratio, and CNF-SAT have minimum approximate ratio but it is difficult to scale to large number of vertices.

References

[1] Garey M.R., Johnson D.S., 1978. Strong NP-completeness results: motivation, examples, and implications, Journal of ACM, 25, 499-508.

[2] Chen J., Kanj I.A., Jia W., 2001. Vertex cover: further observations and further improvements, Journal of Algorithms 41,280-301.

[3] Chen J., Kani I.A., Xia G., 2010. Improved upper bounds for vertex cover, Theoretical Computer Science, 411,3736-3756.

[4] Cormen, Thomas H.., Charles Eric. Leiserson, Ronald L. Rivest, and Clifford Stein. Introduction to Algorithms. N.p.: n.p., n.d. Print.