Indian Institute of Technology (IIT) Indore

STUDENTS:

Name
Aditya Gupta - 170001002
Ravi Maurya - 170001039

Under the guidance of **Dr. Kapil Ahuja**

NP Complete Problem : “Vertex Cover”

16th April 2019

INDEX

1. INTRODUCTION

The minimum vertex cover problem is the optimization problem of finding a smallest vertex cover in a given graph. Let G be a graph of order n with no isolated vertex. A vertex covering of the graph G is a set of vertices such that every edge of the graph is incident to at least one vertex of the set.

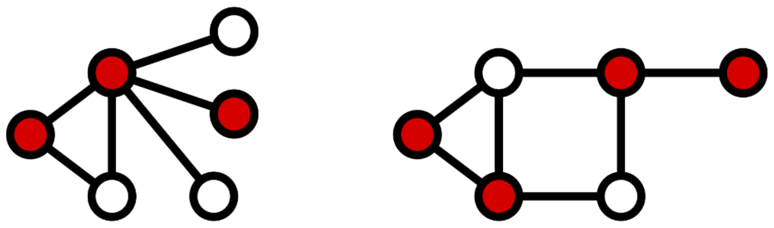
The problem is a know NP Complete problem, i.e., there is no polynomial time solution for this unless P = NP. There are approximate polynomial time algorithms to solve the problem though.

2. GOALS

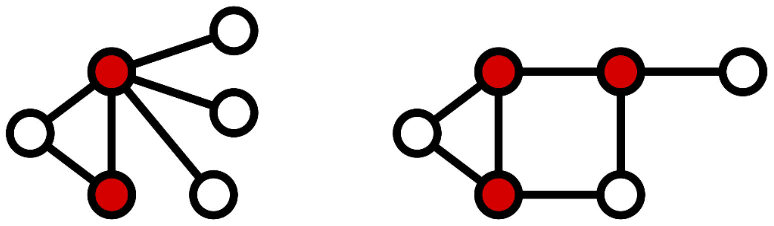
1. To study or analyze the available algorithms in the basis of time and space complexity.
2. To implement the existing algorithms and compare them in terms of efficiency.
3. Intending to find the best/worst case scenarios for various approaches available. Also to try and find out where these algorithms might fail.

3. DEFINITION

Formally, a vertex cover of an undirected graph G=(V,E) is a subset of V such that, that is to say it is a set of vertices V' where every edge has at least one endpoint in the vertex cover V'. Such a set is said to cover the edges of G. The following figure shows two examples of vertex covers, with some vertex cover V' marked in red.



A minimum vertex cover is a vertex cover of smallest possible size. The vertex cover number is the size of a minimum vertex cover, i.e.. The following figure shows examples of minimum vertex covers in the above graphs.



ALGORITHMS IMPLEMENTED/ANALYSED

1. APPROXIMATE ALGORITHMS:

- Approx principle procedure(2-approximation)
- Greedy Technique
- Alom’s Algorithm

2. EXACT ALGORITHMS:

- For Trees
- For Bipartite graphs

4. APPROXIMATE ALGORITHMS

4.1. Approx principle procedure (2-approximation)

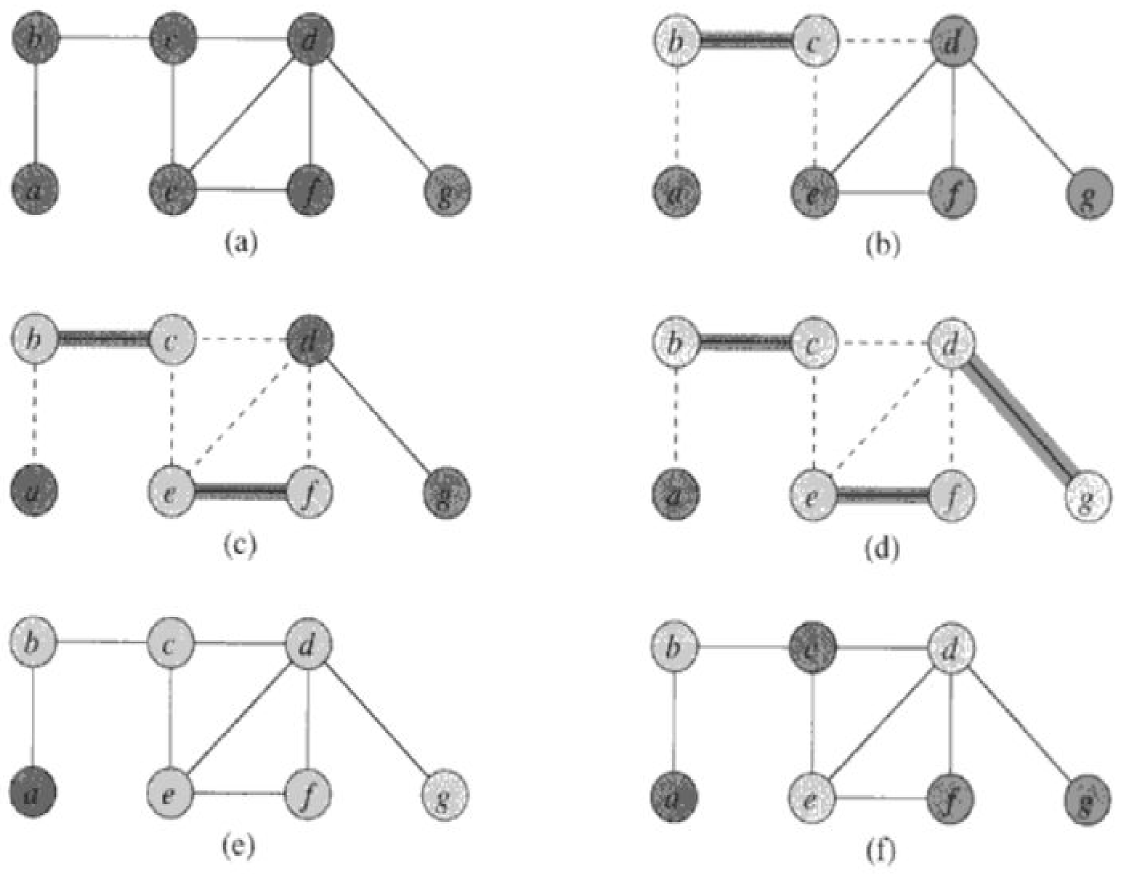
This method involves selecting a random edge,adding both of its vertices to vertex cover and then removing all the edges which has that vertices at one end. The process is repeated until we are left with no edge.

Pseudocode:

```
1. C = ∅
2. E'= G.E
3. while E'≠ ∅:
      a. let (u, v) be an arbitrary edge of E'
      b. C = C ∪ {u, v}
      c. remove from E' every edge incident on either u or
         v
4. return C
```

The above algorithm doesn’t guarantee the optimum solution (because vertex cover problem is NP complete). However, it can be proved that the above approximate algorithm never finds a vertex cover whose size is more than twice the size of minimum possible vertex cover.That why this algorithm is a 2-approximation algorithm.

Time Complexity of above algorithm is **O(V + E).**



Approx Principle procedure

4.2. Greedy Algorithm

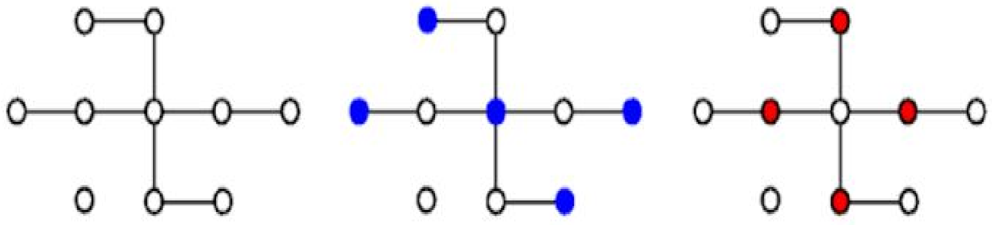
This is a simple method used to solve optimization problems. It involves arbitrarily selecting an edge and adding one of its vertex to vertex cover set then removing all other edges incident on that vertex. The above procedure is repeated until we are left with no edge.

Pseudocode:

```
1. C ← Ø
2. while E ≠ Ø 3.
    a. Pick any edge e ∈E and choose an end-point v of e
    b. C ← C U {v}
    c. E ← E \ {e ∈ E : v ∈ e}
3. return C
```

Time complexity of above algorithm is again **O(V+E)**.

The above algorithm also doesn't guarantee the optimum solution as depicted in the below picture :



A graph instance Vertex cover by greedy Optimal vertex cover

A clever implementation of Greedy Algorithm involves selecting those vertices from graph which have maximum degree. This will reduce the complexity of the algorithm in cases where vertices have larger degree.

If the number of Edges is much larger than the number of vertices i.e, if the graph is of higher degree, then the complexity of clever implementation of Greedy algorithm will be less than linear. Therefore, this method is suitable for graphs of higher degree.

4.4. Alom’s Algorithm

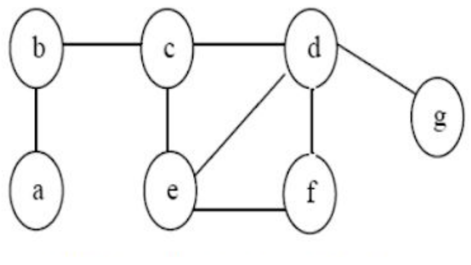
This vertex cover algorithm selects the vertex which has maximum number of edges incident to it. All the edges are discarded incident to that vertex. If more than one vertex have same maximum number of edges, this algorithm select that vertex which have at least one edge that is not covered by other vertices, which has maximum edge. This process is repeated until to cover all the vertices of the graph.

Pseudocode:

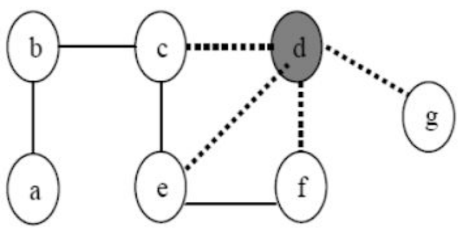
```
1. OPTIMAL_VT_COVER (E, V) {// E is an edge and V is
   an vertex
2. V← Φ;
3. E′  ←E [G]
4. While (E′  ≠ ) {
      a. M ← Choose vertex which has maximum incident
         edge; If (More than one vertex have maximum
         number of edges) then
```

```
        b. M ← Choose that node which has at least one
           edge that is not covered by others Which have
           maximum number of edges.
        c. V← V U M;
        d. Remove the all incident edges at vertex M;
    5. Count incident edge of new graph.}
    6. Return V}
```

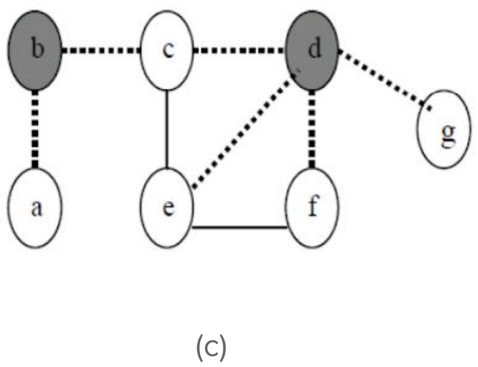
Since the number of iterations of the loop is at most E. So time complexity of this algorithm is **O (V+E)**, where E is total no.of edges.



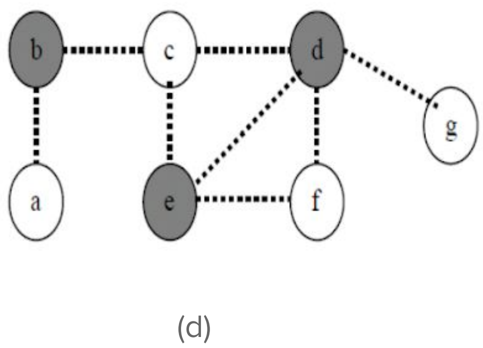
(a)



(b)



(c)



(d)

Alom's Algorithm

Alom’s algorithm is the best algorithm in terms of time complexity among the above listed algorithms. In case of higher degree graphs, the algorithm runs even faster than the clever greedy.

5. EXACT ALGORITHMS

The decision variant of the vertex cover problem (getting the exact size of the minimum vertex cover) is NP-complete, which means it is unlikely that there is an efficient algorithm to solve it exactly. However there exist some special cases when the exact solution of the problem could be found. Those cases are discussed below:

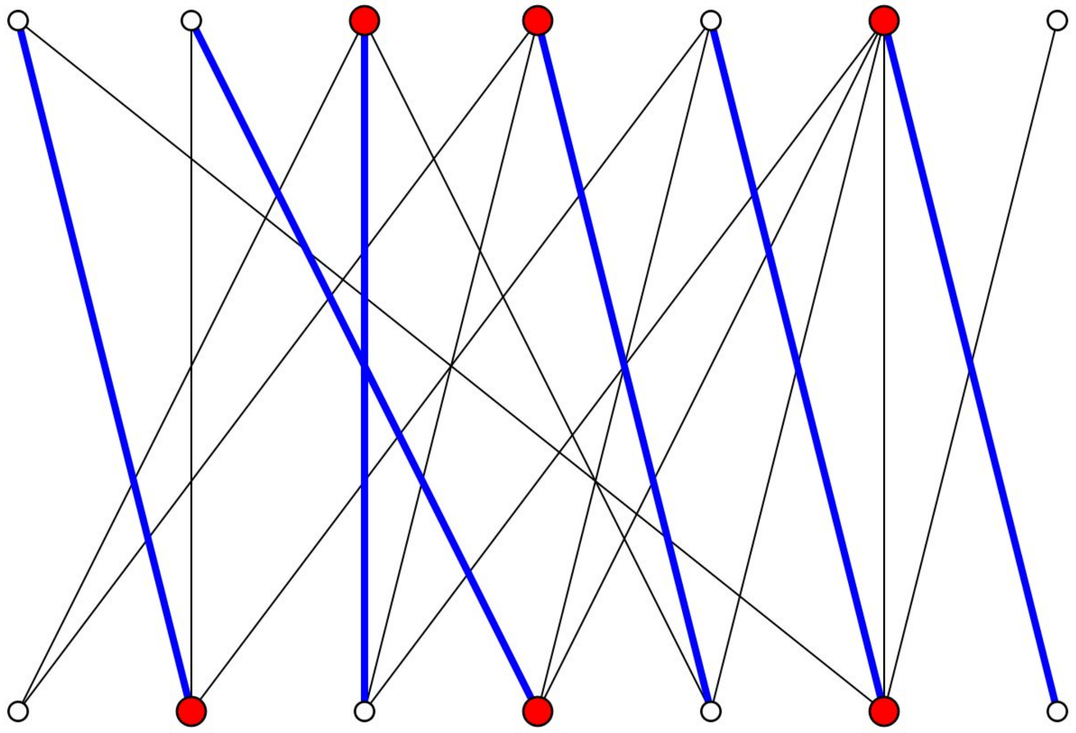
5.1. For Bipartite Graph

Lets first state **Kőnig's theorem**

Statement:

“In any bipartite graph, the number of edges in a maximum matching equals the number of vertices in a minimum vertex cover.”

A **matching** in a Bipartite Graph is a set of the edges chosen in such a way that no two edges share an endpoint. A **maximum matching** is a matching of maximum size (maximum number of edges).



The maximum matching problem for bipartite graph could be solved by **Hopcroft Karp algorithm** in polynomial time.

The equivalence between vertex cover and maximum matching described by the **Kőnig's theorem** theorem allows the bipartite vertex cover problem to be solved in polynomial time.

Some Terminologies:

Free Node or Vertex: Given a matching M, a node that is not part of matching is called free node.

Augmenting path: A path that alternates between matching and not matching edges, and has free vertices as starting and ending points.

Hopcroft Karp algorithm:

1. Initialize Maximal Matching M as empty.
2. While there exists an Augmenting Path p
    a. Remove matching edges of p from M and add not-matching edges of p to M
    b. This increases size of M by 1 as p starts and ends with a free vertex)
3. Return M.

Time complexity of above algorithm is **O(sqrt(V)*E)**.For sparse graphs, it runs in worst-case time.

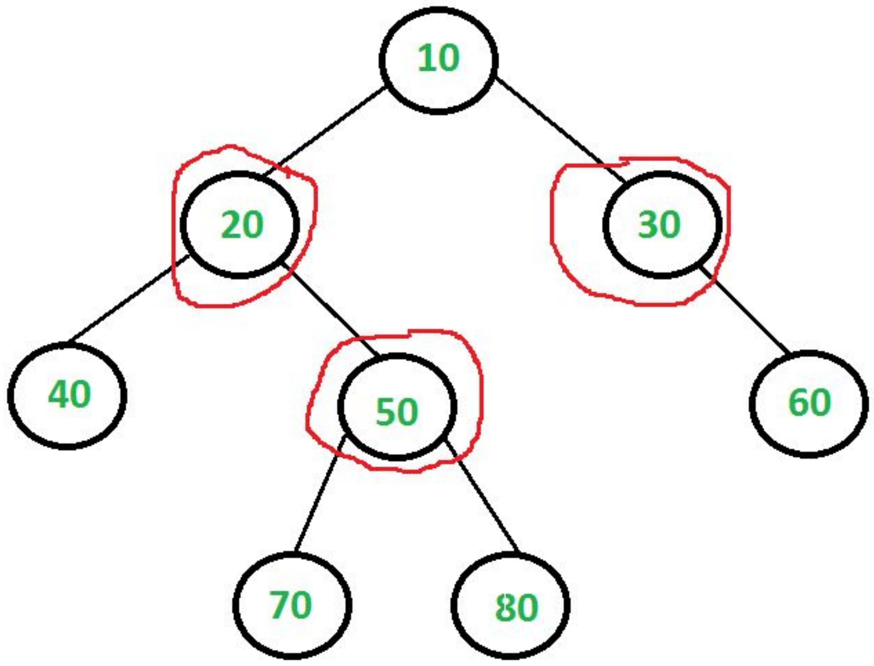
5.2. For Trees

The idea here is to consider following two possibilities for root and recursively for all nodes down the root.

1. Root is part of vertex cover: In this case root covers all children edges. We recursively calculate size of vertex

covers for left and right subtrees and add 1 to the result (for root).
2. Root is not part of vertex cover: In this case, both children of root must be included in vertex cover to cover all root to children edges. We recursively calculate size of vertex covers of all grandchildren and number of children to the result (for two children of root).

The naive implementation of the above algorithm takes exponential time i.e., **O(2^n)**. However an efficient implementation of the above algorithm can be achieved using **memoization** (Dynamic Programming), reducing the complexity to **O(n)** where n is the number of vertices.



6. WORK

1. https://github.com/raviMaurya12/VertexCoverImpleme ntations

7. REFERENCES

1. https://en.wikipedia.org/wiki/Vertex_cover
2. https://pdfs.semanticscholar.org/85ae/36531799292fd e4f705583bcf07145e4ff25.pdf
3. https://www.geeksforgeeks.org/hopcroft-karp-algorith m-for-maximum-matching-set-1-introduction/
4. K V R Kumar, Deepak Garg, –Complete Algorithms on Minimum Vertex Cover ‖ CIIT International Journal of Software Engineering and Technology, Issue May 2009 ISSN 0974 – 9748 & Online: ISSN 0974 – 9632.
5. Reshu Tyagi Muskaan Batra "Implementation and Comparison of Vertex Cover Problem using Various Techniques" International Journal of Computer Applications 144(10):26-31 · June 2016