| | |
|---|---|
| **IT584 Approximation Algorithms** | January 04, 2024 |
| Lecture 2 | |
| *Lecturer: Rachit Chhaya* | *Scribe/s: Sneh Raval (202311013)* |

# 1 The Vertex Cover Problem

**Definition:** A vertex cover of an undirected graph $G = (V, E)$ is a subset $V' \subseteq V$ such that if $(u, v) \in E$, then $u \in V'$ or $v \in V'$ (or both). That is, each vertex "covers" its incident edges, and a vertex cover for $G$ is a set of vertices that covers all the edges in $E$. The size of a vertex cover is the number of vertices in it.

**Example:** The graph in Figure 1 has a vertex cover $\{w, z\}$ of size 2.

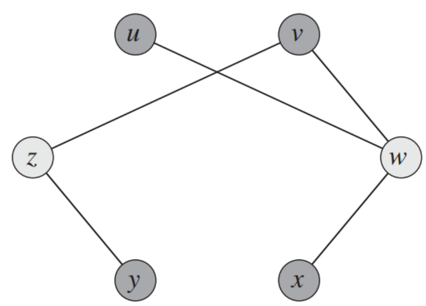


**Figure 1**: Example of vertex cover

**Goal:** To find a vertex cover of minimum size in a given undirected graph.

## 1.1 Greedy Paradigm

Let us first discuss about matching in graph.

### 1.1.1 Matching

**Definition:** Given a graph $G = (V, E)$, a matching $M$ in $G$ is a set of pairwise non-adjacent edges, none of which are loops; that is, no two edges share common vertices.

**Maximal Matching:** A maximal matching is a matching $M$ of a graph $G$ that is not a subset of any other matching. A matching $M$ of a graph $G$ is maximal if every edge in $G$ has a non-empty intersection with at least one edge in $M$. The following figure shows examples of maximal matching (red) in three graphs.

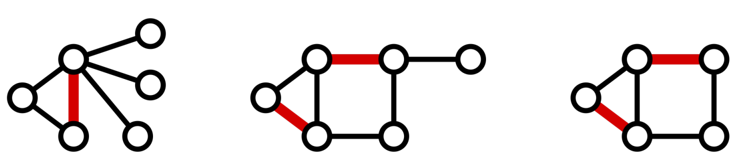

**Figure 2**: Example of maximal matching

**Maximum Matching:** A maximum matching is a matching that contains the largest possible number of edges. There may be many maximum matching. The matching number $v(G)$ of a graph $G$ is the size of a maximum matching. Every maximum matching is maximal, but not every maximal matching is a maximum matching. The following figure shows examples of maximum matching in the same three graphs.

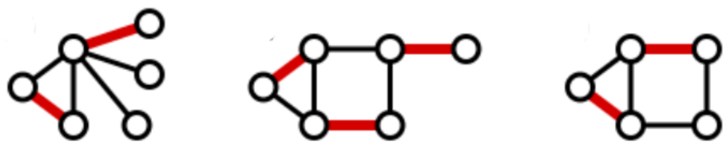

**Figure 3**: Example of maximum matching

We will use the vertices formed from maximal matching to find vertex cover. Why? Because we have a greedy algorithm to find matching. We can then have the following description of the algorithm:

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

#### 1.1.2 Approx-Vertex-Cover Algorithm

1: $C \leftarrow \emptyset$
2: $E' \leftarrow E$
3: **while** $E' \neq \emptyset$ **do**
4:     Let $(u, v)$ be an arbitrary edge of $E'$
5:     $C \leftarrow C \cup \{u, v\}$
6:     Remove from $E'$ every edge incident on either $u$ or $v$
7: **return** $C$

The variable $C$ contains the vertex cover being constructed.

Line 1 initializes $C$ to the empty set. Line 2 sets $E'$ to be a copy of the edge set G:E of the graph. The loop of lines 3–6 repeatedly picks an edge $(u, v)$ from $E'$, adds its endpoints $u$ and $v$ to $C$, and deletes all edges in $E'$ that are covered by either $u$ or $v$. Finally, line 7 returns the vertex cover C.

The set of edges picked by this algorithm is a matching. In fact, it is a maximal matching. The running time of this algorithm is $O(V + E)$, using adjacency lists to represent $E'$.

Figure 4 illustrates how Approx-Vertex-Cover operates on an example graph. shown heavy, are the edge chosen by Approx-Vertex-Cover and shown lightly shaded, are added to the set $C$. The set $C$, which is the vertex cover produced by Approx-Vertex-Cover, contains the six vertices $(b, c)$; $(d, e)$; $(f, g)$; (From the matching). Optimal vertex cover for this problem contains only three vertices: b, d, and e which can be seen in figure (f).

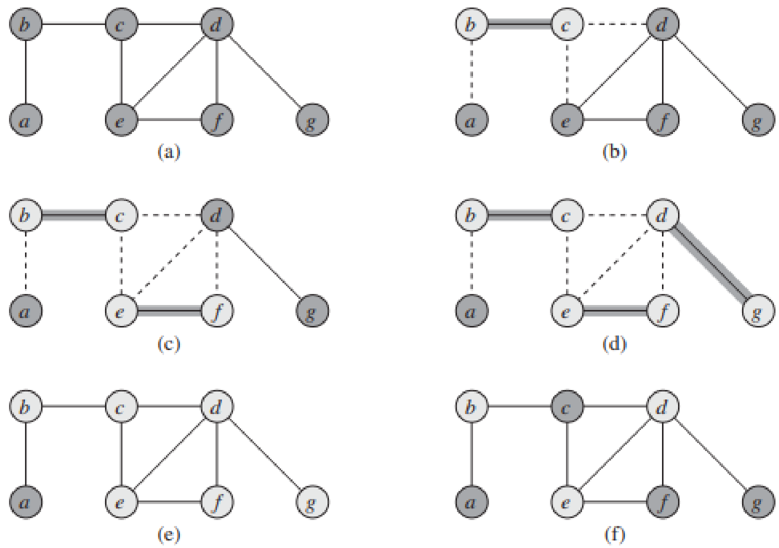


**Figure 4**: The operation of Approx-Vertex-Cover

## 1.2 Analysis of Approximation Algorithm for Vertex Cover

**Claim 1** *Size of maximal matching lower bounds the size of vertex cover.*

$$|M| \leq OPT, \text{ (where } OPT \text{ is the actual minimum vertex cover)}$$

**Proof** Every edge $\in M$ is clearly covered. If an edge, $e \notin M$ is not covered, then $M \cup \{e\}$ is matching, which contradict to maximality of $M$. ■

**Claim 2** *Approx-Vertex-Cover gives 2 - Approximation for the vertex cover problem.*

$$Sol \leq \ \alpha * OPT \ \text{ (where } \alpha = 2\text{)}$$

**Proof** In accordance with Claim 1, we have lower bound on the size of vertex.

$$|M| \leq OPT \tag{1}$$

Now, Each execution of line 4 picks an edge for which neither of its endpoints is already in $C$, yielding an upper bound (an exact upper bound, in fact) on the size of the vertex cover returned:

$$Sol \leq 2|M| \tag{2}$$

Combining equations 1 and 2, we obtain

$$Sol \leq \ 2 * OPT \tag{3}$$

■

## 1.3 Can we do better than 2 - Approximation?

Is it possible that this algorithm can do better than 2-approximation? We can show that 2-approximation is a tight bound by a tight example.

### 1.3.1 Tight Example

Consider a complete bipartite graph $A$ of n black nodes on one side and n red nodes on the other side, denoted $K_{n,n}$.

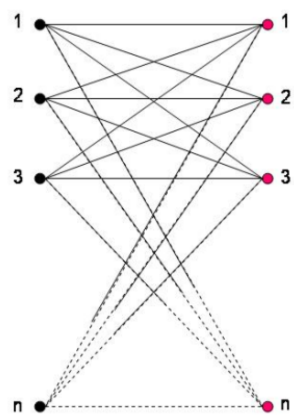


**Figure 5**: $K_{n,n}$ - complete bipartite graph

Notice that size of any maximal matching of this graph equals n (Kőnig's theorem)[1],

$$|M| = n$$

So the Approx-Vertex-Cover algorithm returns a cover of size 2n.

$$A(K_{n,n}) = n$$

But, clearly the optimal solution = n.

$$OPT(K_{n,n}) = n$$

Note that a tight example needs to have arbitrarily large size to prove tightness of analysis, otherwise we can just use brute force for small graphs and $A$ for large ones to get an algorithm that avoid that tight bound. Here, it shows that this algorithm gives 2-approximation no matter what size n is.

[1] Kőnig's theorem - In any bipartite graph, the number of edges in a maximum matching equals the number of vertices in a minimum vertex cover.