MINIMUM VERTEX COVER

Beamer Presentation

1905091,1905099,1905103

Department of Computer Science and Engineering
Bangladesh University of Engineering and Technology

February 26, 2023

Table of contents

Table of Contents

Definition

The vertex-cover problem

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

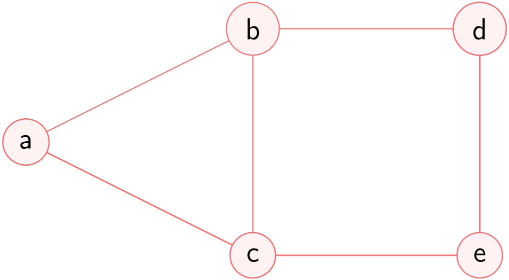


Figure: Vertex Cover Example

Example

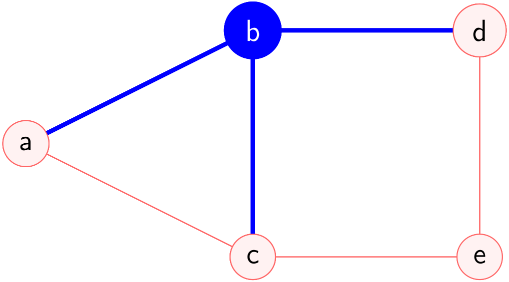


Figure: Vertex Cover Example

Example

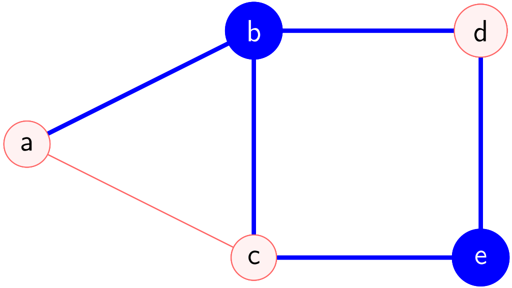


Figure: Vertex Cover Example

Example

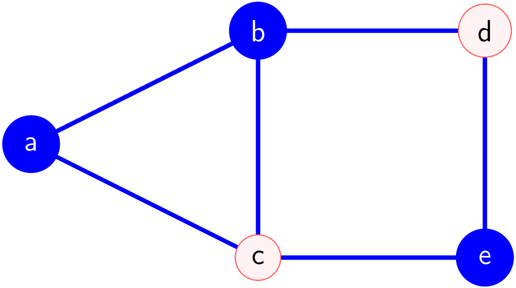


Figure: Vertex Cover Example

Minimum vertex cover

Problem statement

The vertex-cover problem is to find a vertex cover of minimum size in a given graph. Restating this optimization problem as a decision problem, we wish to determine whether a graph has a vertex cover of a given size k. As a language, we define
VERTEX-COVER={<G,K> : graph G has a vertex cover of size k}

Vertex Cover Problem is NP Copmplete

Statement

The vertex cover problem is an NP-Complete problem, which means that there is no known polynomial-time solution for finding the minimum vertex cover of a graph unless it can be proven that P = NP.

Proof

Vertex cover $\in NP$

Suppose we are given a graph G=(V,E) and an integer k. Let $V^{'} \subseteq V$ and $|V^{'}| = k$. Then it checks for each edge $(u, v) \in E$, that $u \in V^{'}$ or $v \in V^{'}$. We can easily verify the certificate in polynomial time.

Proof

Vertex cover $\in NP$ hard

In order to prove that Vertex cover is an NP-hard problem we reduce a known NP-hard problem to vertex cover problem. We chose clique for an instance.

Proof

Complement graph

Given a graph G=(V,E) we define complement of G as $\bar{G}$=(V,$\bar{E}$), Where $\bar{E}$={(u,v):u,v∈ V,u $\not\subseteq v$ and (u,v) ∉ E}

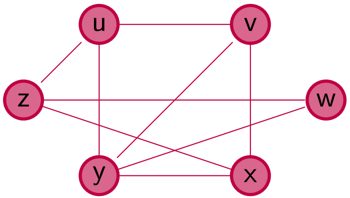


Figure: G

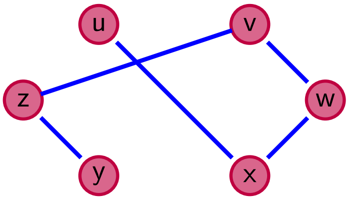


Figure: $\bar{G}$

Proof

Reduction algorithm

The reduction algorithm takes as input an instance (G,K) of the clique problem .
It computes the complement $\bar{G}$ in polynomial time $(\bar{G}, |V| - K)$ is an instance of vertex cover problem.

Reduced problem

The graph G has a clique of size k if and only if the graph $\bar{G}$ has a vertex cover of size |V|-k

Proof

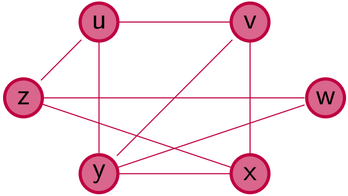


Figure: G

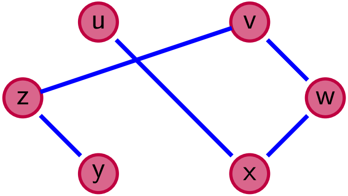


Figure: $\bar{G}$

The graph G has a clique of size k if and only if the graph $\bar{G}$ has a vertex cover of size |V|-k

Table of Contents

Optimal solution?

Naive approach

We can naively solve the problem by iterating over all the subsets of the vertices and using only those vertices, forming a new graph containing all the edges contained by these vertices. Then we can check if this new graph, contains all the edges of the original graph or not based on which it can be a candidate for the vertex cover. Out of all the candidates, we print the set, which has the minimum size.

This naive approach will have an exponential runtime complexity.

Approximation

Some hope?

Even though we do not know how to find an optimal vertex cover in a graph G in polynomial time, we can efficiently find a vertex cover that is near optimal.

Approximation

Definition

An approximation algorithm for a problem is a polynomial-time algorithm that, when given input i, outputs an element of FS(i), which is the set of feasible solutions for i.

Approach

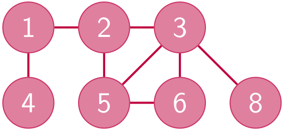
The following approximation algorithm takes as input an undirected graph G and returns a vertex cover whose size is guaranteed to be no more than twice the size of an optimal vertex cover.

Approximate Vertex Cover

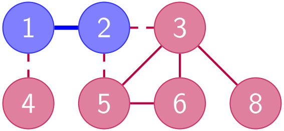
Require: A graph $G = (V, E)$
Ensure: A vertex cover $C \subseteq V$

$C \leftarrow \emptyset$
for $(u, v) \in E$ **do**
  if $u \notin C$ and $v \notin C$ **then**
    $C \leftarrow C \cup \{u\}$
    $C \leftarrow C \cup \{v\}$
  end if
end for
return C

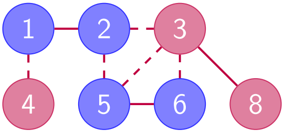
Approx algorithm continued....



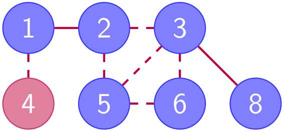
Approx algorithm continued....



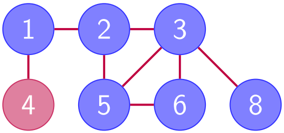
Approx algorithm continued....



Approx algorithm continued....



Approx algorithm continued....



Approx algorithm continued....

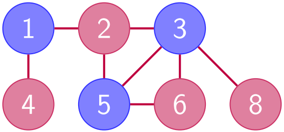


Table of Contents

Efficiency

We can see that approximation does not always give a minimum vertex cover, but it can be proven that the returned vertex cover is at most twice the size of an optimal cover.

Proof

Let A denote the set of edges that line 4 of APPROX VERTEX COVER picked. In order to cover the edges in A,

- an optimal cover C* must include at least one endpoint of each edge in A.
- No two edges of A share an endpoint.

Proof

Lower Bound

$|C^*| >= |A|$

Proof

Each execution of line 4 picks an edge for which neither of its endpoints is already in C, yielding

Upper Bound

$|C| = 2|A|$

Proof

Combining we get,

Conclusion

$|C|=2|A| => |C|<=2|C^*|$

Table of Contents

Key Points

- A set of minimum number of vertices such that each edge of the graph is incident to at least one vertex of the set, is called the vertex cover.

Key Points

- A set of minimum number of vertices such that each edge of the graph is incident to at least one vertex of the set, is called the vertex cover.
- Although minimum vertex cover is a NP hard problem, near optimal solution can be found in polynomial time.

Key Points

- A set of minimum number of vertices such that each edge of the graph is incident to at least one vertex of the set, is called the vertex cover.
- Although minimum vertex cover is a NP hard problem, near optimal solution can be found in polynomial time.
- While greedy algorithm might not be optimal, approximation algorithm guarantees a near optimal solution.