Vertex cover

- Vertex cover
- Deterministic algorithm
- Randomized algorithm

Vertex Cover

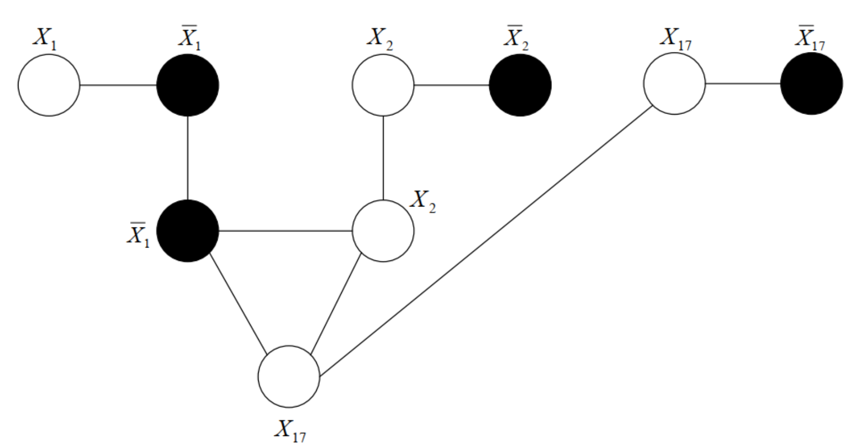
- Given: undirected graph G
- Goal: Find a minimum-cardinality subset $V_0 \subseteq V$ such that if $(u, v) \in E(G)$, then $u \in V_0$ or $v \in V_0$
- Cover edges by picking vertices
- NP-hard

Vertex Cover

- Given: undirected graph G
- Goal: Find a minimum-cardinality subset $V_0 \subseteq V$ such that if $(u, v) \in E(G)$, then $u \in V_0$ or $v \in V_0$
- Cover edges by picking vertices
- NP-hard
- Applications
    - Every edge forms a task, every vertex represents a person that can execute that task
    - Perform all tasks with min resources
    - Extensions: weighted vertices or hypergraphs

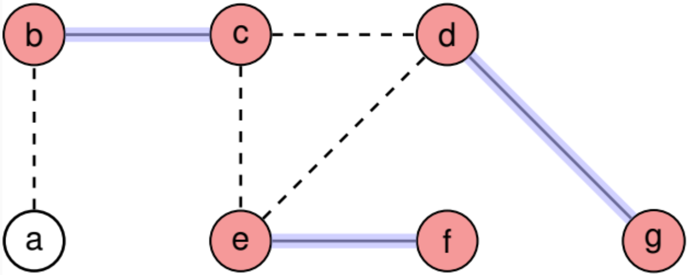
Vertex Cover: NP-harness proof

Establish a mapping between VC and 3SAT



A Greedy Approximation Algorithm

```
APPROX-VERTEX-COVER(G)
C = ∅
E' = G.E
while E' ≠ ∅
    let (u, v) be an arbitrary edge of E'
    C = C ∪ {u, v}
    remove from E' every edge incident on either u or v
return C
```



Analysis of Greedy Approximation Algorithm

Theorem

APPROX-VERTEX-COVER is a poly-time 2–approximation algorithm

Proof

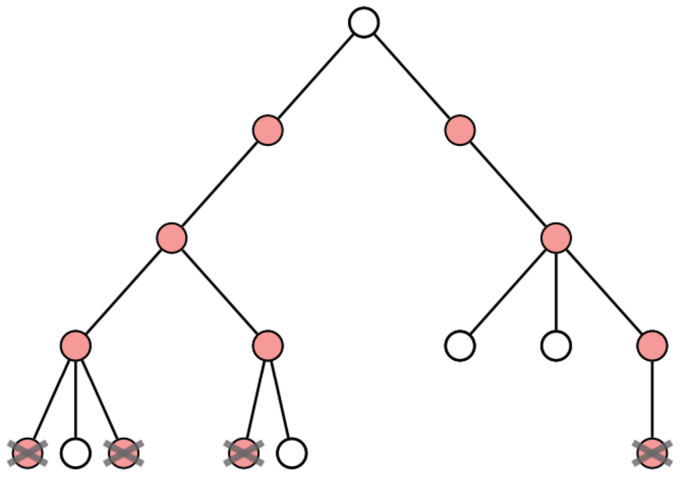
Let $A \subseteq E$ denote the set of chosen edges

- Every optimal cover C^* must include at least one endpoint of edges in A, and edges in A do not share a common endpoint: $|C^*| \geq |A|$
- Every edge in A contributes 2 vertices to $|C|$: $|C| = 2|A| \leq 2|C^*|$

Vertex Cover on Trees

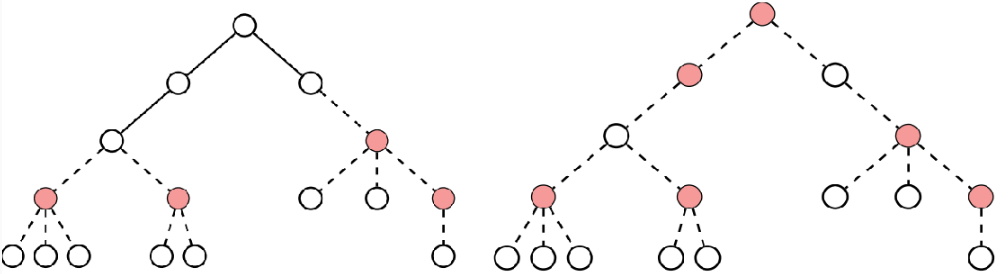
There exists an optimal vertex cover which does not include any leaves

- Replace any leaf in the cover by its parent



Solving Vertex Cover on Trees

```
VERTEX-COVER-TREES(G)
C = ∅
while ∃ leaves in G
    Add all parents to C
    Remove all leaves and their parents from G
return C
```



Can be also solved on bipartite graphs, using Max-Flows and Min-Cuts

Exact Algorithms

Find a vertex cover of size k in any simple graph

- Brute-Force Search takes $\binom{n}{k} = \Theta(n^k)$ time: How to improve?

Theorem

Consider a graph G and its edge uv. Let G_u be the graph obtained by deleting u and its incident edges. G has a vertex cover of size k iff G_u or G_v has a vertex cover of size $k-1$

Proof

$\Longleftarrow$: Assume G_u has a vertex cover C_u of size $k-1$, adding u yields a vertex cover of G

$\Longrightarrow$: Assume G has a vertex cover C of size k, which contains, say u, removing u from C yields a vertex cover of G_u

A More Efficient Search Algorithm

$\textsc{Vertex-Cover-Search}(G, k)$

If $E = \emptyset$ **return** $\emptyset$
If $k = 0$ and $E \neq \emptyset$ **return** $\perp$
Pick an arbitrary edge $(u, v) \in E$
$S_1 = \textsc{Vertex-Cover-Search}(G_u, k-1)$
$S_2 = \textsc{Vertex-Cover-Search}(G_v, k-1)$
if $S_1 \neq \perp$ **return** $S_1 \cup \{u\}$
if $S_2 \neq \perp$ **return** $S_2 \cup \{v\}$
return $\perp$

Running time: $O(2^k)$

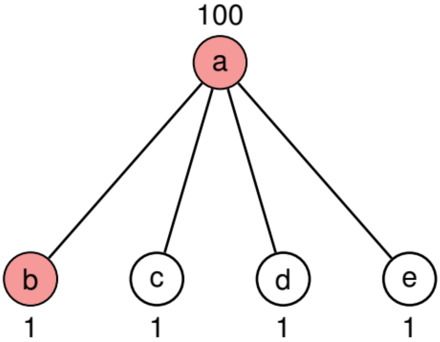
Weighted Vertex Cover

- Given: undirected, vertex-weighted graph G
- Goal: Find a minimum-weight subset $V_0 \subseteq V$ such that if $(u, v) \in E(G)$, then $u \in V_0$ or $v \in V_0$
- Cover edges by picking vertices
- NP-hard
- Applications
    - Every edge forms a task, every vertex represents a person that can execute that task
    - Weight of a vertex could be salary of a person
    - Perform all tasks with the minimal amount of resources

Greedy Algorithm from Unweighted case

APPROX-VERTEX-COVER(G)

$C = \emptyset$
$E' = G.E$
while $E' \neq \emptyset$
    let (u, v) be an arbitrary edge of E'
    $C = C \cup \{u, v\}$
    remove from E' every edge incident on either u or v
return C



Invoking an (Integer) Linear Program

Idea: Round the solution of an associated linear program

0-1 Integer Program

$$\begin{array}{llcll} \text{minimize} & \sum_{v \in V} w(v)x(v) & & & \\ \text{subject to} & x(u) + x(v) & \geq & 1 & \text{for each } (u, v) \in E \\ & x(v) & \in & \{0, 1\} & \text{for each } v \in V \end{array}$$

Linear Program

$$\begin{array}{llcll} \text{minimize} & \sum_{v \in V} w(v)x(v) & & & \\ \text{subject to} & x(u) + x(v) & \geq & 1 & \text{for each } (u, v) \in E \\ & x(v) & \in & [0, 1] & \text{for each } v \in V \end{array}$$

LP-based Algorithm

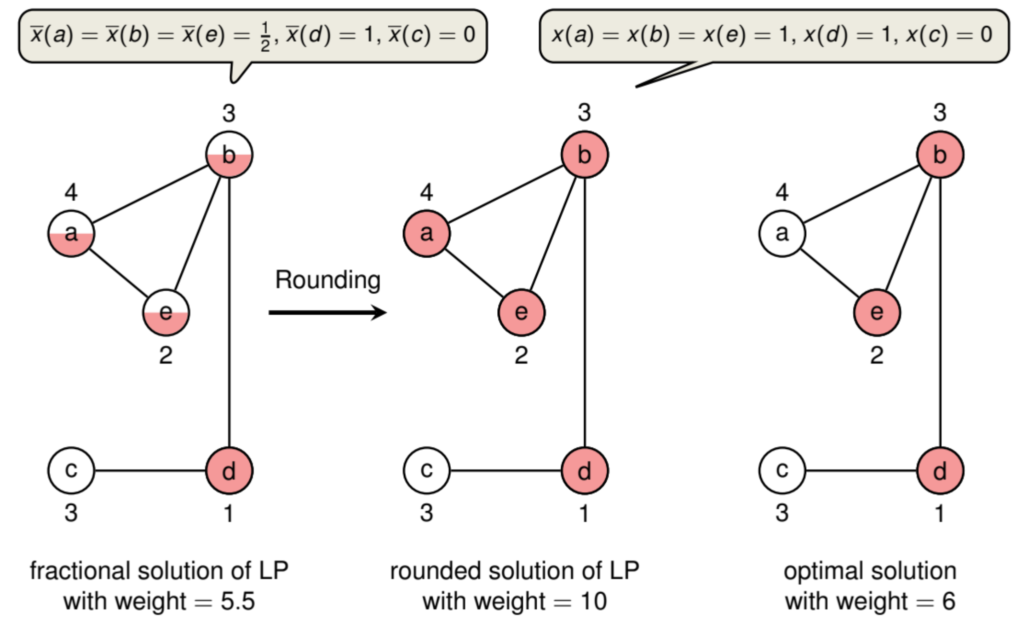
APPROX-MIN-WEIGHT-VC(G, w)

$C = \emptyset$
compute $\bar{x}$, an optimal solution to the linear program
for each $v \in V$
    if $\bar{x}(v) \geq 1/2$
        $C = C \cup \{v\}$
return C

Approximation ratio

APPROX-MIN-WEIGHT-VC is a polynomial-time 2-approximation algorithm for the minimum-weight vertex-cover problem

Example


$\overline{x}(a) = \overline{x}(b) = \overline{x}(e) = \frac{1}{2}, \overline{x}(d) = 1, \overline{x}(c) = 0$
$x(a) = x(b) = x(e) = 1, x(d) = 1, x(c) = 0$
3
b
4
a
e
2
c
d
3
1
Rounding
fractional solution of LP with weight $= 5.5$
rounded solution of LP with weight $= 10$
optimal solution with weight $= 6$

Approximation Ratio

Approximation ratio

- Let C^* denote an optimal solution, z^* the value of an optimal solution of the LP: $z^* \leq w(C^*)$
- We first prove that the computed set C covers all vertices
  - Consider any edge (u, v) with constraint $x(u) + x(v) \geq 1$: at least one of $x(u)$ and $x(v) \geq 1/2$, hence C covers (u, v)
- We then prove that $w(C) \leq 2z^*$

$$w(C^*) \geq z^* = \sum_{v \in V} w(v)\bar{x}(v) \geq \sum_{v \in V : \bar{x}(v) \geq 1/2} w(v) \cdot \frac{1}{2} = \frac{1}{2} W(C)$$

Randomized Algorithm

RAND-VC: For each $e = (u, v)$, if e is not covered, add u to C with probability $w_v/(w_u + w_v)$, otherwise add v

Approximation ratio

RAND-VC is a poly-time 2-approximation algorithm in expectation

Proof

- C_i: C after iteration i; C^*: optimum
- We prove by induction $E[\sum_{v \in C_i \cap C^*} w_v] \geq E[\sum_{v \in C_i \setminus C^*} w_v]$
  - If both $u, v \in C^*$: $>$ holds; if one of $u, v \in C^*$:

$$E\left[\sum_{v \in C_i \cap C^*} w_v\right] = E\left[\sum_{v \in C_{i-1} \cap C^*} w_v\right] + \frac{w_u w_v}{w_u + w_v};$$

$$E\left[\sum_{v \in C_i \setminus C^*} w_v\right] = E\left[\sum_{v \in C_{i-1} \setminus C^*} w_v\right] + \frac{w_u w_v}{w_u + w_v}$$