CS240 Algorithm Design and Analysis

Lecture 25

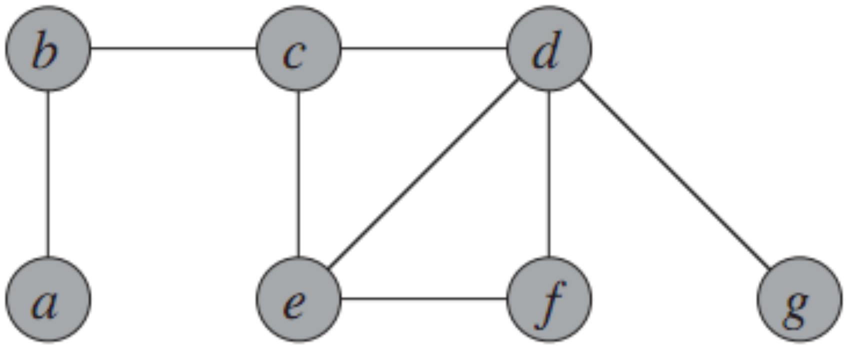
Approximation Algorithms

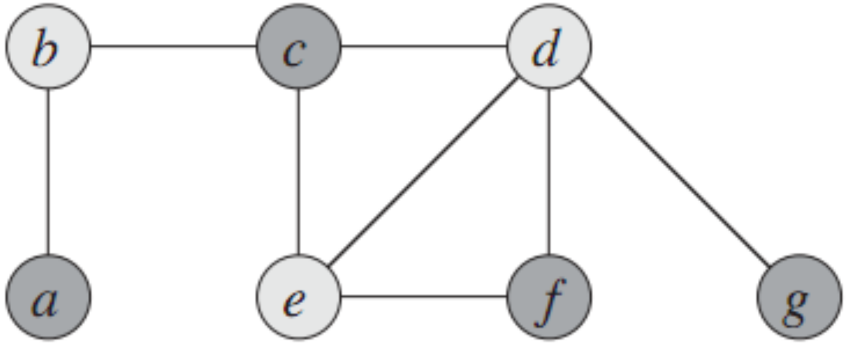
Quan Li
Fall 2022
2022.12.13

Vertex Cover

Vertex Cover

- Input A graph with vertices V and edges E.
- Output A subset V′ of the vertices, so that every edge in E touches some vertex in V′.
- Goal Make |V′| as small as possible.



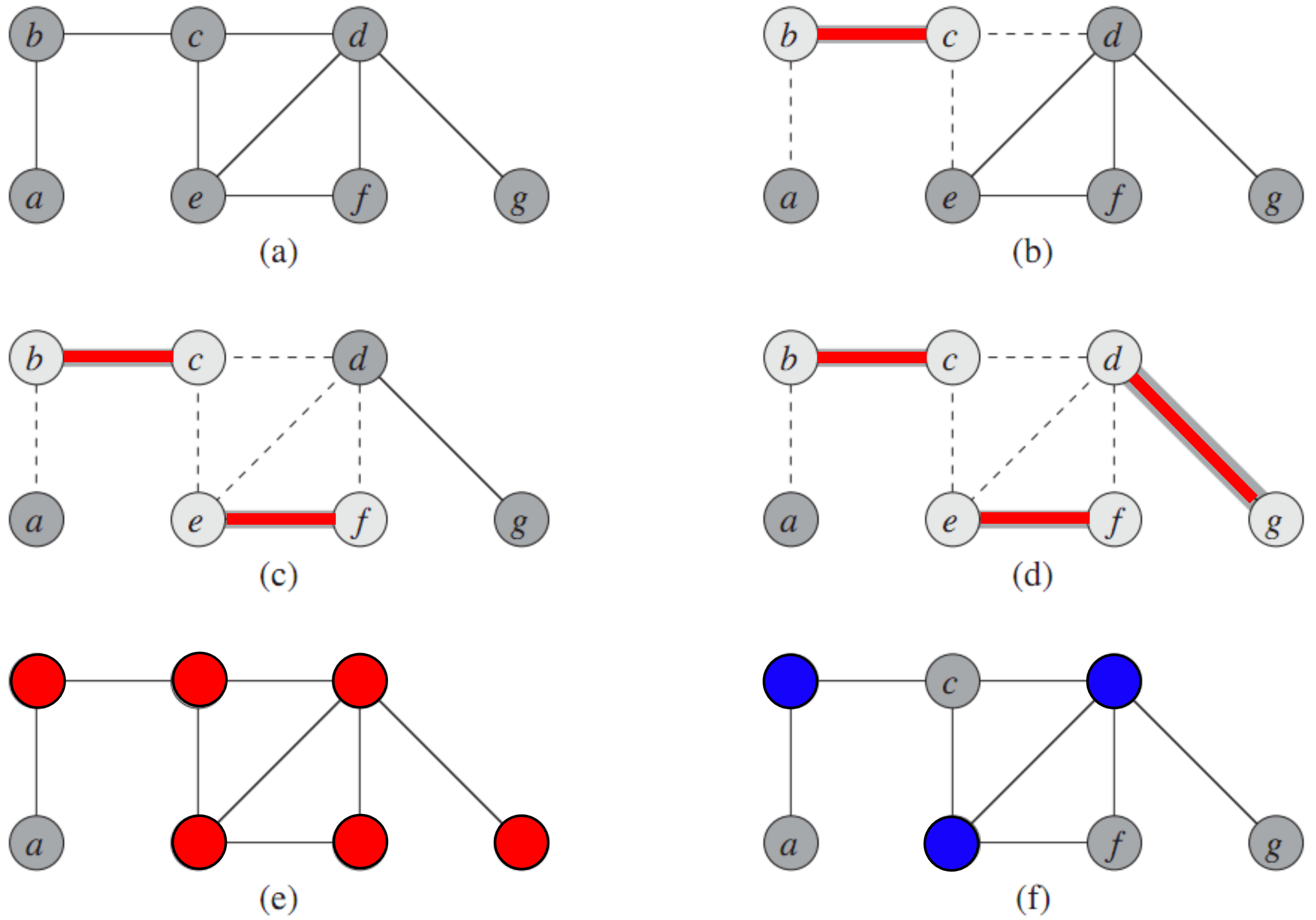


- Finding the minimum vertex cover is NP-complete.
- We'll see a simple 2 approximation for this problem.

A Vertex Cover Algorithm

- Initially, let D be all the edges in the graph, and C be the empty set.
  - C is our eventual vertex cover.
- Repeat as long as there are edge left in D.
  - Take any edge (u,v) in D.
  - Add {u,v} to C.
  - Remove all the edges adjacent to u or v from D.
- Output C as the vertex cover.

Example



Algorithm's vertex cover

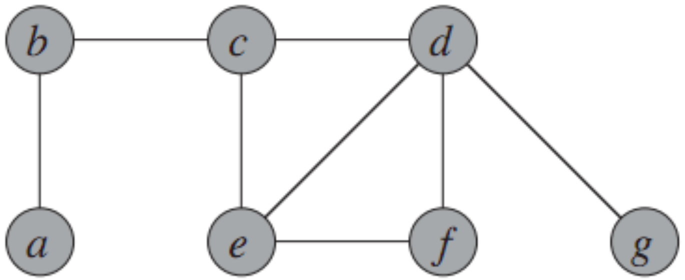
Optimal vertex cover

Proof of Correctness

- The output is certainly a vertex cover.
  - In each iteration, we only take out edges that get covered.
  - We keep adding vertices till all edges are covered.
- Now, we show it's a 2 approximation.
- Let C^* be an optimal vertex cover.
- Let A be the set of edges the algorithm picked.

Proof of Correctness

- None of the edges in A touch each other.
  - Each time we pick an edge, we remove all adjacent edges.
- So each vertex in C* covers at most one edge in A.
  - The edges covered by a vertex all touch each other.
- Every edge in A is covered by a vertex in C*.
  - Because C* is a vertex cover.
- So $|C^*| \geq |A|$.
- The number of vertices the algorithm uses is $2|A|$.
  - If algorithm picks edge (u,v), it uses {u,v} in the cover.
- So (# vertices algorithm uses) / (# vertices in opt cover) = $2|A| / |C^*| \leq 2|A| / |A| = 2$.

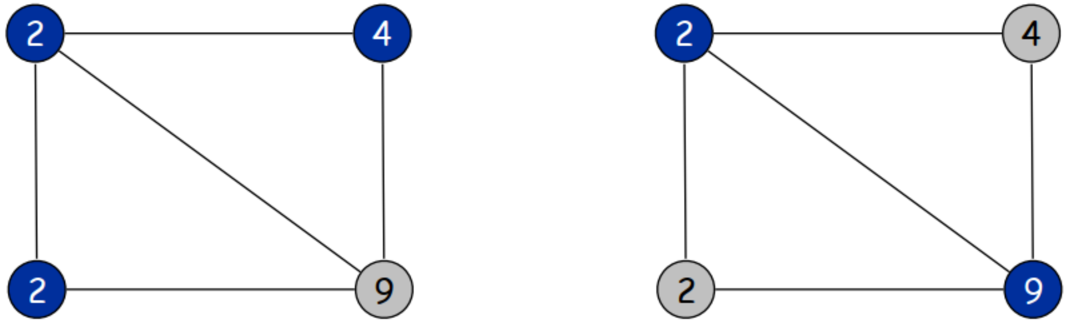


The Pricing Method: Vertex Cover

Weighted Vertex Cover

Weighted vertex cover. Given a graph G with vertex weights, find a vertex cover of minimum weight.

It's a special case of the set cover problem, so the $H(d^*)$ approximation ratio can be achieved by the greedy algorithm, where d^* = max degree



weight = 2 + 2 + 4 = 8

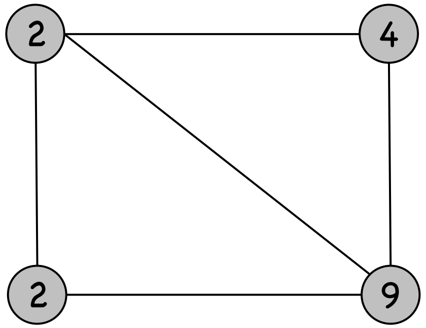
weight = 2 + 9 = 11

Weighted Vertex Cover

Pricing method. Each edge must be covered by some vertex i. Edge e pays price $p_e \geq 0$ to use vertex i.

Fairness. Edges incident to vertex i should pay $\leq w_i$ in total.

$$for\ each\ vertex\ i: \sum_{e=(i,j)} p_e \leq w_i$$



Claim. For any vertex cover S and any fair prices p_e: $\sum_e p_e \leq w(S)$.

Proof.
$$\sum_{e\in E} p_e \leq \sum_{i\in S} \sum_{e=(i,j)} p_e \leq \sum_{i\in S} w_i = w(S) \quad \blacksquare$$

each edge e covered by at least one node in S

sum fairness inequalities for each node in S

立志成才报国裕民

Pricing Method

Pricing method. Set prices and find vertex cover simultaneously.

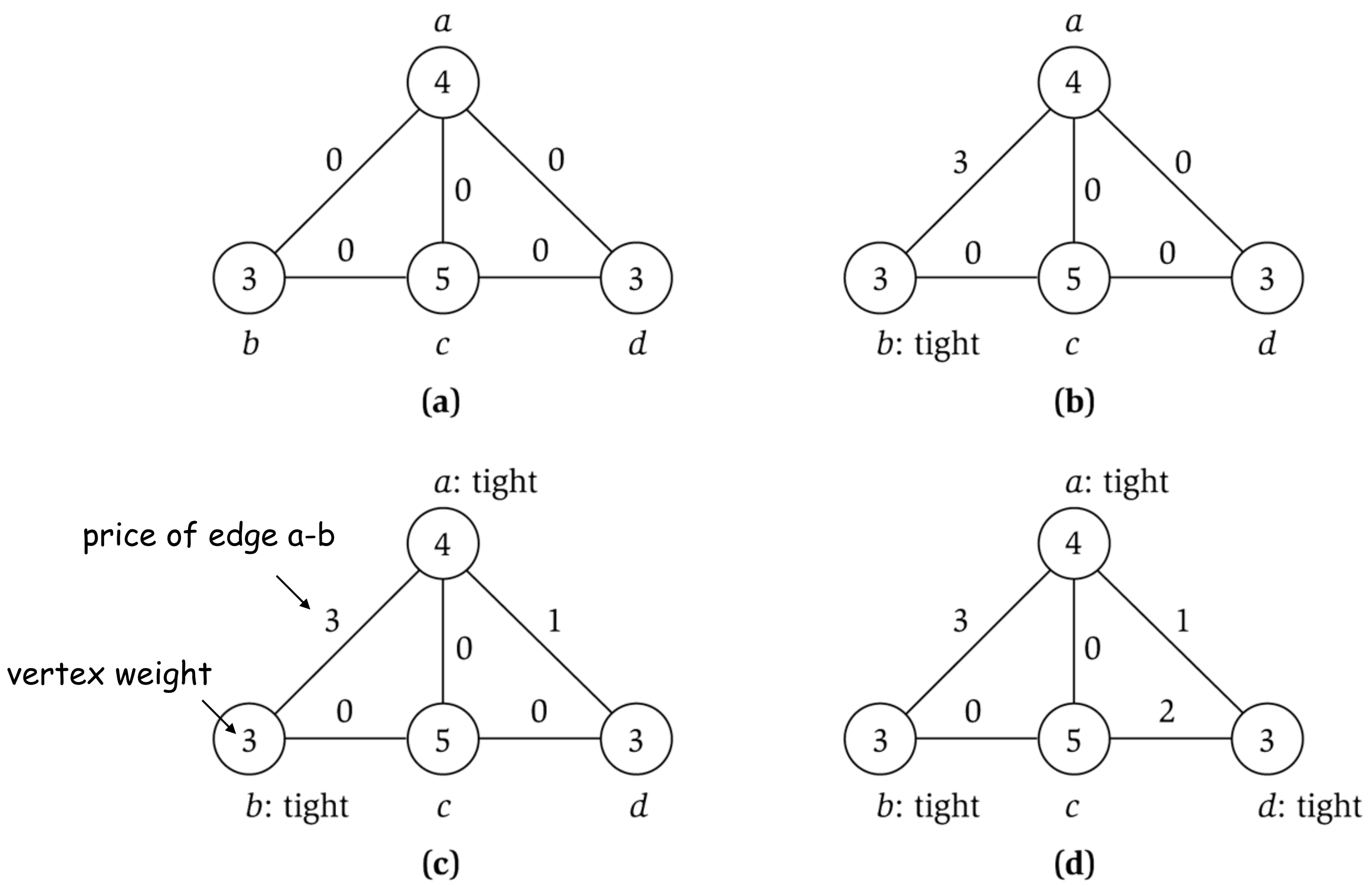
```
Weighted-Vertex-Cover-Approx(G, w) {
   foreach e in E
      pe = 0

   while (∃ edge i-j such that neither i nor j are tight)
      select such an edge e
      increase pe without violating fairness
   }

   S ← set of all tight nodes
   return S
}
```

$$\sum_{e=(i,j)} p_e = w_i$$

Pricing Method: Example



Pricing Method: Analysis

Theorem. Pricing method is a 2-approximation.

Pf.

- Algorithm terminates since at least one new node becomes tight after each iteration of while loop.

- Let S = set of all tight nodes upon termination of algorithm.
- S is a vertex cover: if some edge i-j is uncovered, then neither i nor j is tight. But then while loop would not terminate.

- Let S* be optimal vertex cover. We show $w(S) \leq 2w(S^*)$.

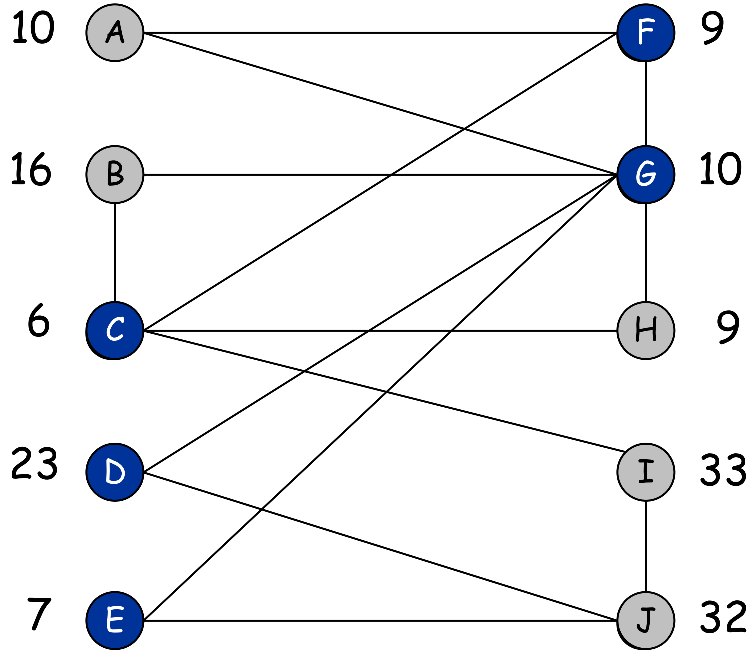
$$w(S) = \underbrace{\sum_{i \in S} w_i = \sum_{i \in S} \sum_{e=(i,j)} p_e}_{\text{all nodes in } S \text{ are tight}} \leq \sum_{i \in V} \sum_{e=(i,j)} p_e = 2 \sum_{e \in E} p_e \leq 2w(S^*). \quad \blacksquare$$

all nodes in S are tight | $S \subseteq V$, prices ≥ 0 | each edge counted twice | fairness lemma

LP Rounding: Vertex Cover

Weighted Vertex Cover

Weighted vertex cover. Given an undirected graph G = (V, E) with vertex weights $w_i \geq 0$, find a minimum weight subset of nodes S such that every edge is incident to at least one vertex in S.



total weight = 55

Weighted Vertex Cover: Integer Linear Programming Formulation

Weighted vertex cover. Given an undirected graph G = (V, E) with vertex weights $w_i \geq 0$, find a minimum weight subset of nodes S such that every edge is incident to at least one vertex in S.

Integer programming formulation.

- Model inclusion of each vertex i using a 0/1 variable x_i.

$$x_i = \begin{cases} 0 & \text{if vertex } i \text{ is not in vertex cover} \\ 1 & \text{if vertex } i \text{ is in vertex cover} \end{cases}$$

- Objective function: minimize $\Sigma_i\, w_i\, x_i$.

- Must take either i or j: $x_i + x_j \geq 1$.

$$\begin{array}{llllll} (ILP) & \min & \sum_{i \in V} w_i x_i & & & \\ & \text{s.t.} & x_i + x_j & \geq & 1 & (i,j) \in E \\ & & x_i & \in & \{0,1\} & i \in V \end{array}$$

Integer Programming

INTEGER-PROGRAMMING. Given integers a_{ij} and b_i, find integers x_j that satisfy:

$$\begin{array}{lrcl} \min & c^t x & & \\ \text{s. t.} & Ax & \geq & b \\ & x & \geq & 0 \\ & x \text{ integral} & & \end{array}$$

$$\begin{array}{rcll} \sum_{j=1}^{n} a_{ij} x_j & \geq & b_i & 1 \leq i \leq m \\ x_j & \geq & 0 & 1 \leq j \leq n \\ x_j & & \text{integral} & 1 \leq j \leq n \end{array}$$

Observation. Vertex cover formulation proves that integer programming is NP-hard search problem.

even if all coefficients are 0/1 and
at most two variables per inequality

Integer Programming

Linear programming. Max/min linear objective function subject to linear inequalities.

- Input: integers c_j, b_i, a_{ij} .
- Output: real numbers x_j.

$$\begin{aligned} \text{(LP)} \quad \min \quad & c^t x \\ \text{s. t.} \quad & Ax \ge b \\ & x \ge 0 \end{aligned}$$

$$\begin{aligned} \text{(LP)} \quad \min \quad & \sum_{j=1}^{n} c_j x_j \\ \text{s. t.} \quad & \sum_{j=1}^{n} a_{ij} x_j \ge b_i \quad 1 \le i \le m \\ & x_j \ge 0 \quad 1 \le j \le n \end{aligned}$$

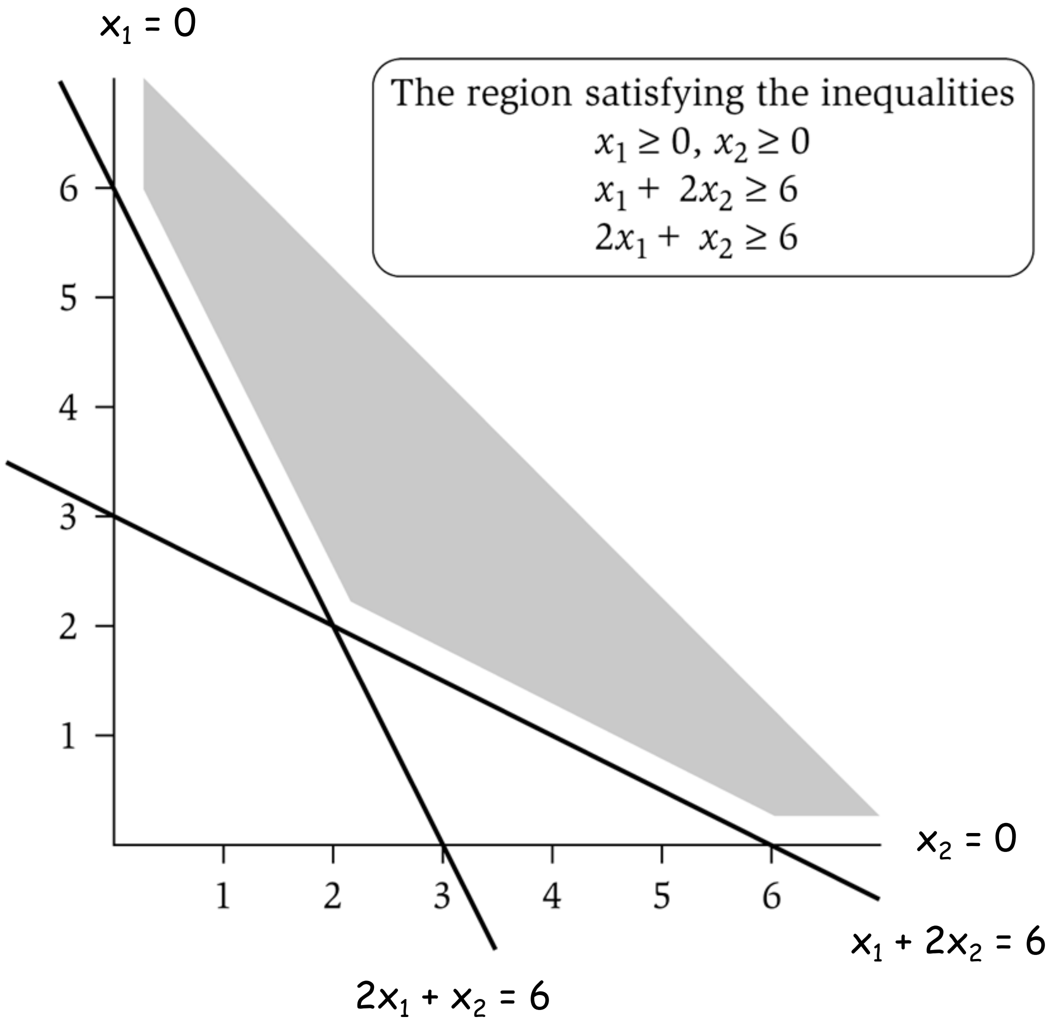
Linear. No x^2, xy, $\arccos(x)$, $x(1-x)$, etc.

Simplex algorithm. [Dantzig 1947] Can solve LP in practice.
Ellipsoid algorithm. [Khachian 1979] Can solve LP in poly-time.

上海科技大学 ShanghaiTech University
立志成才报国裕民

LP Feasible Region

LP geometry in 2D.



上海科技大学 ShanghaiTech University

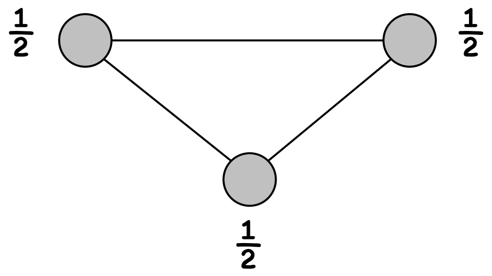
Weighted Vertex Cover: LP Relaxation

Weighted vertex cover. Linear programming formulation.

$$
\begin{array}{llllll}
(LP) & \min & \sum_{i \in V} w_i\, x_i & & & \\
& \text{s.t.} & x_i + x_j & \geq & 1 & (i,j) \in E \\
& & x_i & \geq & 0 & i \in V
\end{array}
$$

Observation. Optimal value of (LP) is $\leq$ optimal value of (ILP).
Pf. LP has fewer constraints.

Note. LP is not equivalent to vertex cover.



Q. How can solving LP help us find a small vertex cover?
A. Solve LP and round fractional values.

立志成才报国裕民

Weighted Vertex Cover

Theorem. If x^* is optimal solution to (LP), then $S = \{i \in V : x^*_i \geq ½\}$ is a vertex cover whose weight is at most twice the min possible weight.

Pf. [S is a vertex cover]

- Consider an edge $(i, j) \in E$.
- Since $x^*_i + x^*_j \geq 1$, either $x^*_i \geq ½$ or $x^*_j \geq ½ \Rightarrow (i, j)$ covered.

Pf. [S has desired cost]

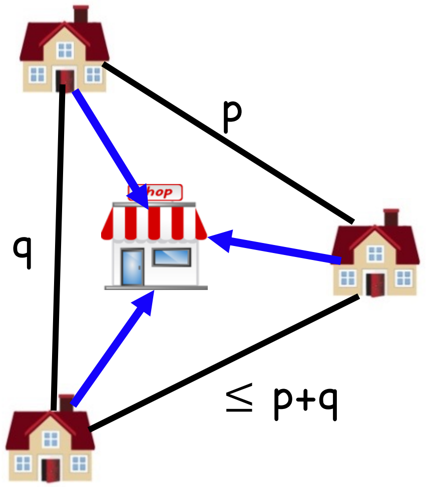
- Let S^* be optimal vertex cover. Then

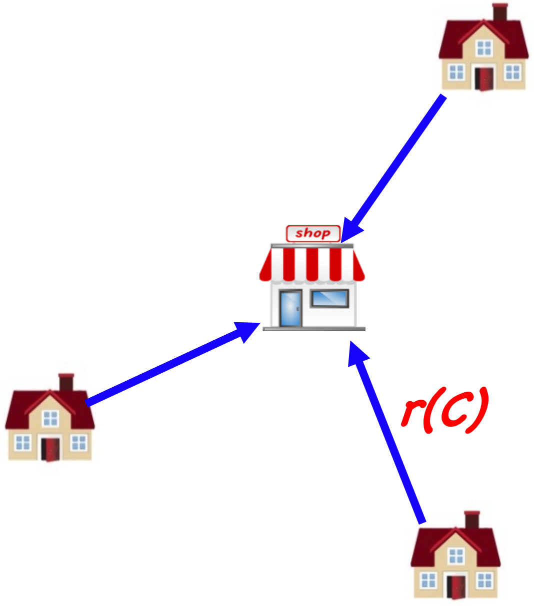
$$\sum_{i \in S^*} w_i \;\underset{\text{LP is a relaxation}}{\geq}\; \sum_{i \in S} w_i x_i^* \;\underset{x_i^* \geq \frac{1}{2}}{\geq}\; \frac{1}{2} \sum_{i \in S} w_i$$

K-Center Problem

K-Center Problem

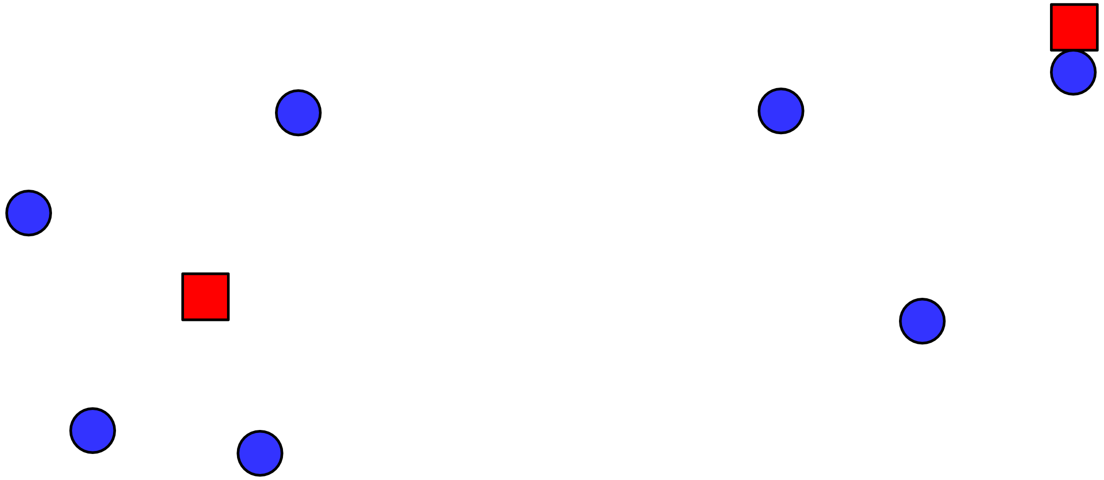
- Given a city with n sites, we want to build k centers to serve them.
  - Let S be set of sites, C be set of centers.
- Each site uses the center closest to it.
  - Distance of site s from the nearest center is $d(s, C) = min_{c \in C} d(s, c)$.
- Goal is to make sure no site is too far from its center.
  - We want to minimize the max distance that any site is from its closest center.
    - Minimize $r(C) = max_{s \in S} min_{c \in C} d(s, c)$.
  - C is called a cover of S, and r is called C's radius.
  - Where should we put centers to minimize the radius?
- Assume distances satisfy triangle inequality.





Gonzalez's Algorithm

- k-Center is NP-complete.
- We'll give a simple 2-approximation for it.
- Idea Say there's one site that's farthest away from all centers. Then it makes the radius large. We'll put a center at that site, to reduce the radius.
  - Note we allow putting center at same location as site.



Gonzalez's Algorithm

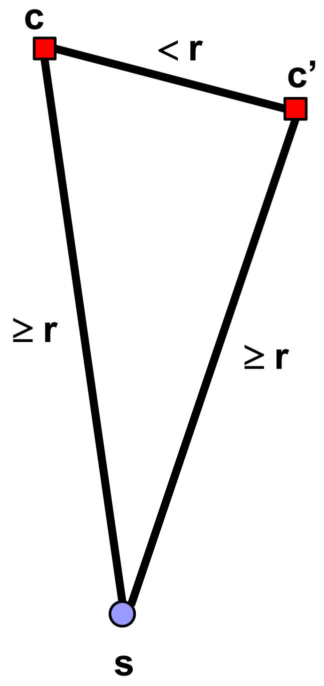
- C is set of centers, initially empty.

- repeat k times
  - choose site s with maximum d(s,C)
  - add s to C
- return C

- Note The centers are located at the sites.

Proof of Correctness

- Let C be the algorithm's output, and r be C's radius.
  - $r = max_{s \in S} min_{c \in C} d(s, c)$
- Lemma 1 For any $c, c' \in C, d(c, c') \geq r$.
- Proof Since r is the radius, there exists a point $s \in S$ at distance ≥ r from all the centers.
  - If there's no such s, then C's radius < r.
  - So s is distance ≥ r from c and c'.
  - Suppose WLOG c' is added to C after c.
  - If d(c,c')<r, then algorithm would add s to C instead of c', since s is farther.



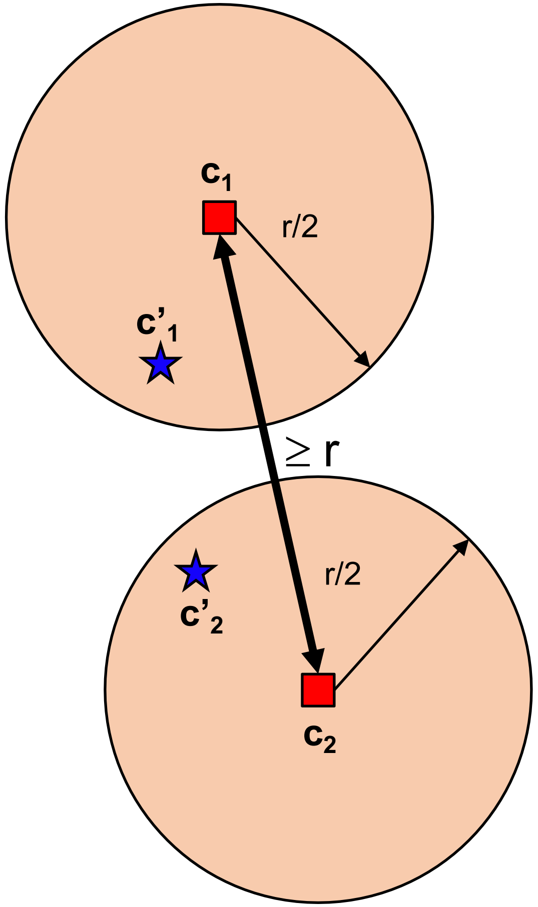
Proof of Correctness

- Cor There exist k+1 points mutually at distance $\geq$ r from each other.
  - By the lemma, the k centers are mutually $\geq$ r distance apart.
  - Also, there's an $s \in S$ at distance $\geq$ r from all the centers.
    - Otherwise, C's covering radius is < r.
  - So, the k centers plus s are the k+1 points.
- Call these k+1 points D.

上海科技大学 ShanghaiTech University
立志成才报国裕民

Proof of Correctness

- Let C^* be an optimal cover with radius r^*.
- Lemma 2 Suppose $r > 2r^*$. Then for every $c \in D$, there exists a corresponding $c' \in C^*$. Furthermore, all these c' are unique.
- Proof Draw a circle of radius $r/2$ around each $c \in D$.
  - There must be a $c' \in C^*$ inside the circle, because
    - c is at most distance r^* away from its nearest center, since r^* is C^*'s radius.
    - $r/2 > r^*$.
  - Given $c_1, c_2 \in D$, let $c'_1, c'_2 \in C^*$ be inside c_1 and c_2's circle, resp.
  - c_1 and c_2's circles don't touch, because $d(c_1, c_2) \geq r$.
  - So $c'_1 \neq c'_2$



Proof of Correctness

- Thm Let C be the output of Gonzalez's algorithm and let C^* be an optimal k-center. Then $r(C) \leq 2r(C^*)$.
- Proof By Lemma 2, if $r(C)>2r(C^*)$, then for every $c\in D$, there is a unique $c'\in C^*$.
  - But there are k+1 points in D, by the corollary.
  - So, there are k+1 points in C^*. This is a contradiction because C^* is a k-center.

上海科技大学 ShanghaiTech University

立志成才报国裕民

Next

Final Review