# Approximation Algorithms

- Center Selection
- Vertex Cover
- Knapsack Problem

# Approximation Algorithms

Q. Suppose I need to solve an NP-hard problem. What should I do?
A. Theory says you're unlikely to find a poly-time algorithm.

Must sacrifice one of three desired features.

- Solve problem to optimality.
- Solve problem in poly-time.
- Solve arbitrary instances of the problem.

$\rho$-approximation algorithm.

- Guaranteed to run in poly-time.
- Guaranteed to solve arbitrary instance of the problem
- Guaranteed to find solution within ratio $\rho$ of true optimum.

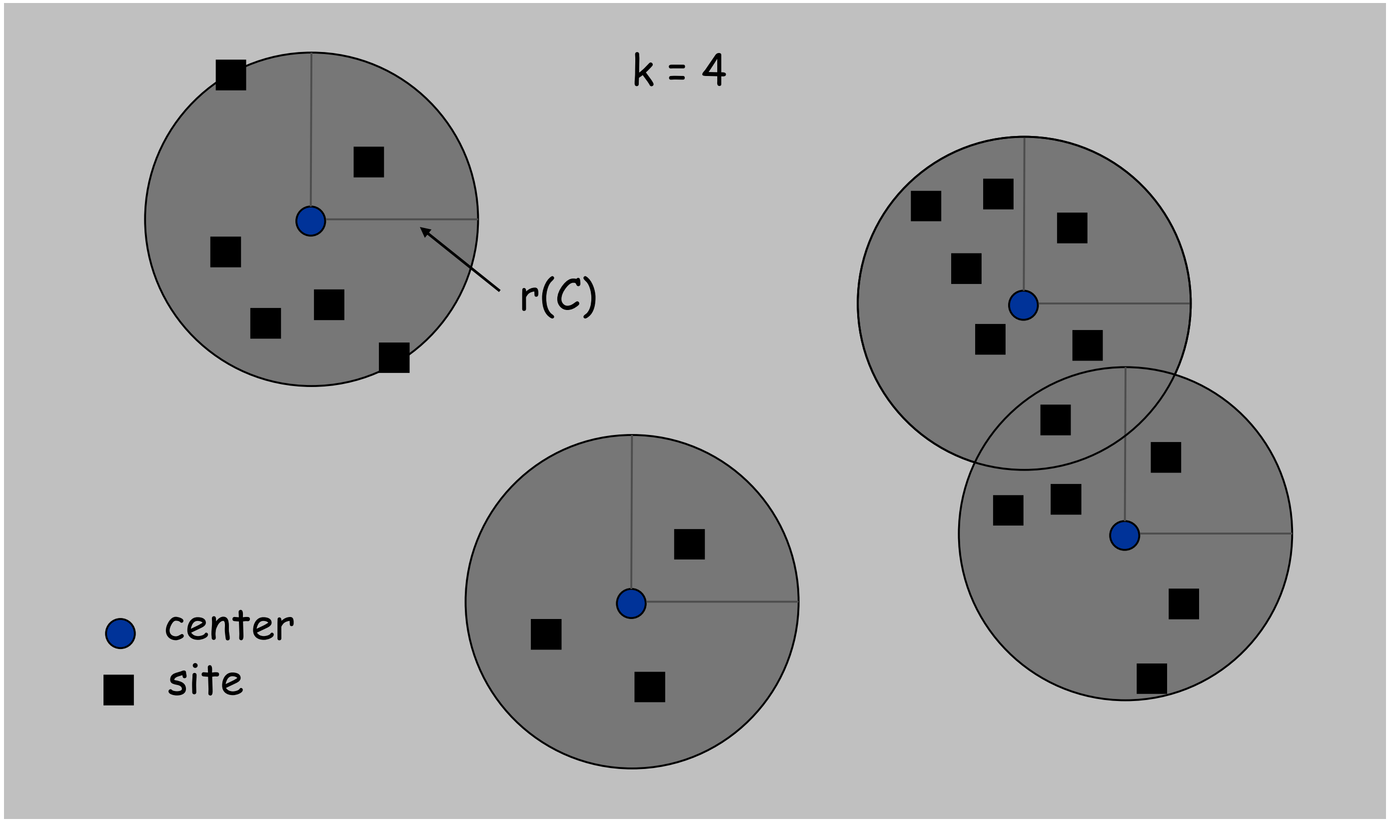
Challenge. Need to prove a solution's value is close to optimum, without even knowing what optimum value is!

# Center Selection

# Center Selection Problem

Input. Set of n sites $s_1, \ldots, s_n$ and integer $k > 0$.

Center selection problem. Select k centers $C$ so that maximum distance from a site to nearest center is minimized.



# Center Selection Problem

Input. Set of n sites $s_1, \ldots, s_n$ and integer $k > 0$.

Center selection problem. Select k centers $C$ so that maximum distance from a site to nearest center is minimized.

Notation.

- $\text{dist}(x, y)$ = distance between x and y.
- $\text{dist}(s_i, C) = \min_{c \in C} \text{dist}(s_i, c)$ = distance from $s_i$ to closest center.
- $r(C) = \max_i \text{dist}(s_i, C)$ = smallest covering radius.

Goal. Find set of centers $C$ that minimizes $r(C)$, subject to $|C| = k$.

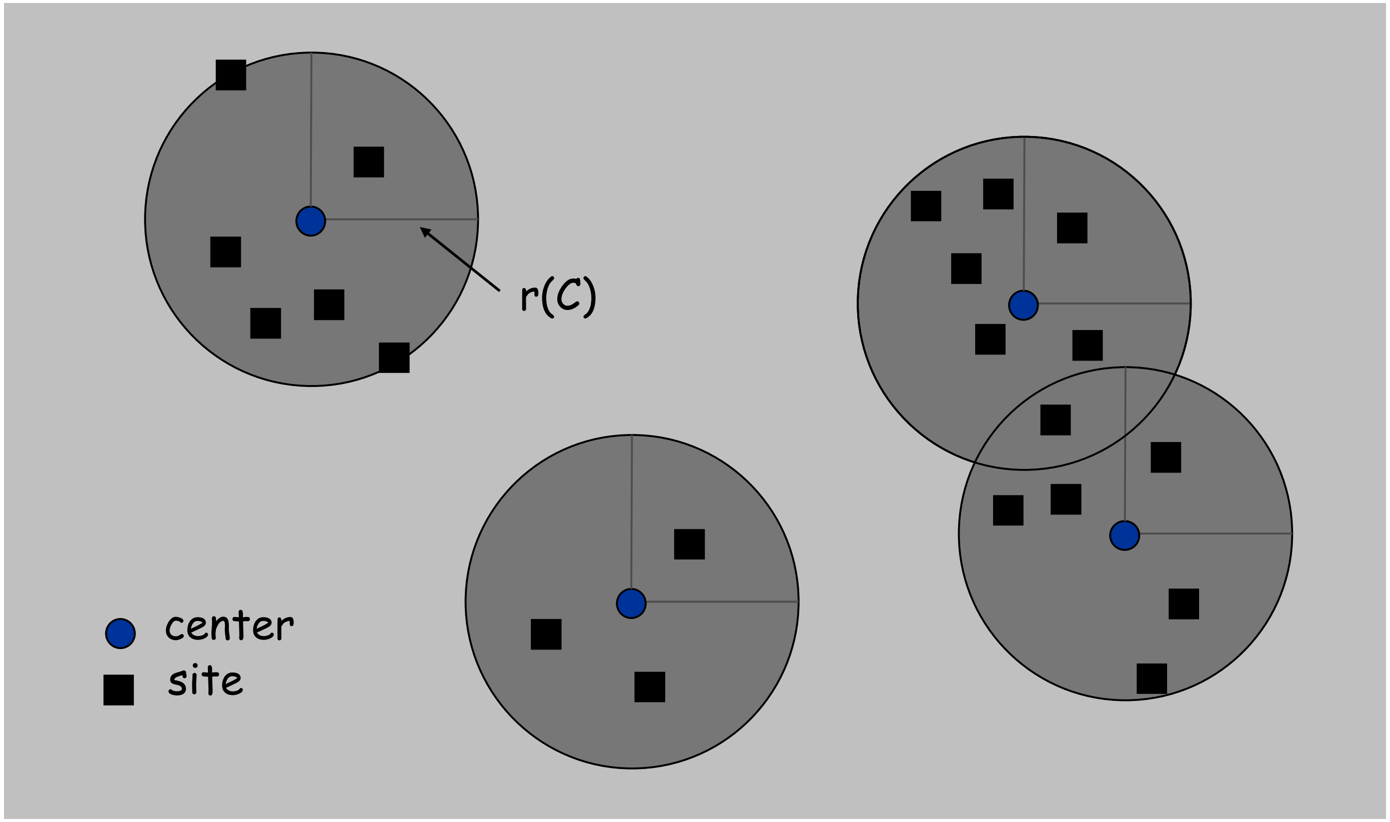
Distance function properties.

- $\text{dist}(x, x) = 0$ (identity)
- $\text{dist}(x, y) = \text{dist}(y, x)$ (symmetry)
- $\text{dist}(x, y) \leq \text{dist}(x, z) + \text{dist}(z, y)$ (triangle inequality)

# Center Selection Example

Ex: each site is a point in the plane, a center can be any point in the plane, dist(x, y) = Euclidean distance.

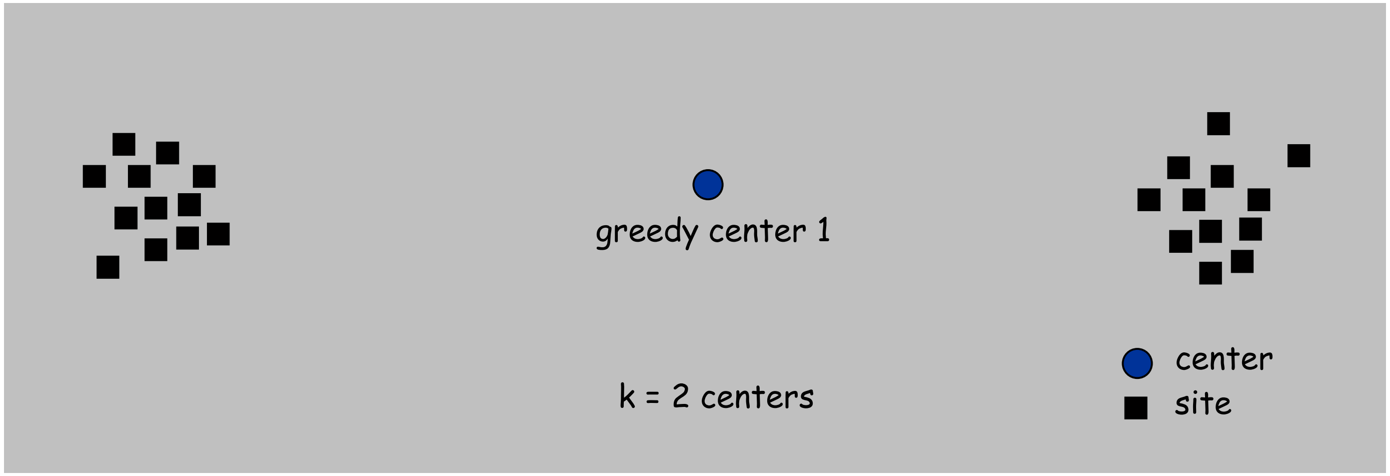
Remark: search can be infinite!



# Greedy Algorithm: A False Start

Greedy algorithm. Put the first center at the best possible location for a single center, and then keep adding centers so as to reduce the covering radius each time by as much as possible.

Remark: arbitrarily bad!



# Center Selection: Greedy Algorithm

Greedy algorithm. Repeatedly choose the next center to be the site farthest from any existing center.

```
Greedy-Center-Selection(k, n, s1,s2,…,sn) {

   C = φ
   repeat k times {
      Select a site si with maximum dist(si, C)
      Add si to C
   }                          site farthest from any center
   return C
}
```

Observation. Upon termination all centers in $C$ are pairwise at least $r(C)$ apart.
Pf. By construction of algorithm.

# Center Selection: Analysis of Greedy Algorithm

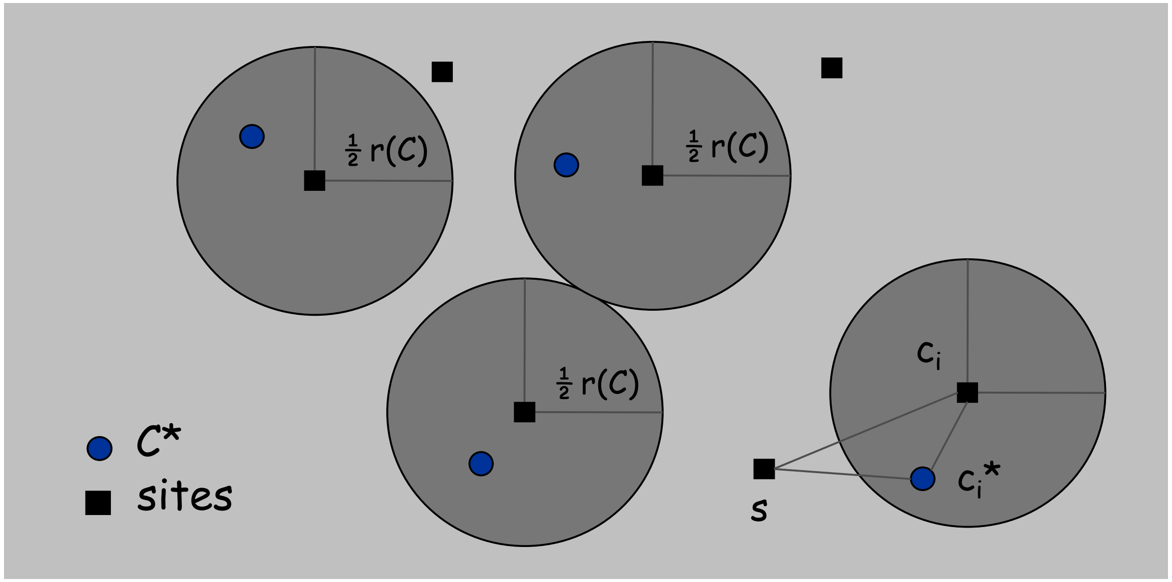
**Theorem.** Let $C^*$ be an optimal set of centers. Then $r(C) \leq 2r(C^*)$.

**Pf.** (by contradiction) Assume $r(C^*) < \frac{1}{2} r(C)$.

- For each site $c_i$ in $C$, consider ball of radius $\frac{1}{2} r(C)$ around it.
- Exactly one $c_i^*$ in each ball; let $c_i$ be the site paired with $c_i^*$.
- Consider any site $s$ and its closest center $c_i^*$ in $C^*$.
- $\text{dist}(s, C) \leq \underbrace{\text{dist}(s, c_i)}_{} \leq \underbrace{\text{dist}(s, c_i^*) + \text{dist}(c_i^*, c_i)}_{} \leq 2r(C^*)$.
- Thus $r(C) \leq 2r(C^*)$. ▪

$\Delta$-inequality

$\leq r(C^*)$ since $c_i^*$ is closest center



# Center Selection

**Theorem.** Let $C^*$ be an optimal set of centers. Then $r(C) \leq 2r(C^*)$.

**Theorem.** Greedy algorithm is a 2-approximation for center selection problem.

**Remark.** Greedy algorithm always places centers at sites, but is still within a factor of 2 of best solution that is allowed to place centers anywhere.

e.g., points in the plane

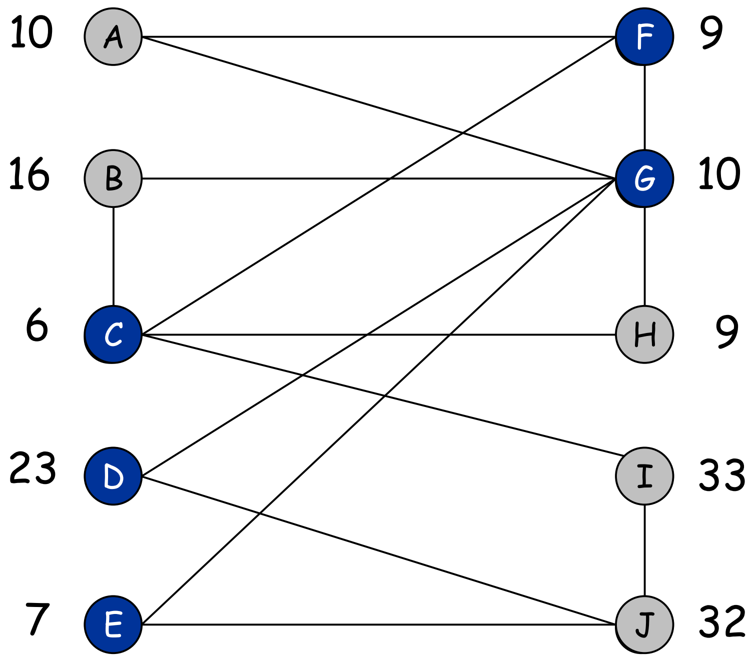
**Question.** Is there hope of a 3/2-approximation? 4/3?

**Theorem.** Unless P = NP, there no $\rho$-approximation for center-selection problem for any $\rho < 2$.

# 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: IP 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 $x_i$.

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

  Vertex covers in 1-1 correspondence with 0/1 assignments:
  $S = \{i \in V : x_i = 1\}$

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

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

# Weighted Vertex Cover: IP Formulation

Weighted vertex cover. Integer programming formulation.

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

Observation. If x* is optimal solution to (ILP), then $S = \{i \in V : x^*_i = 1\}$ is a min weight vertex cover.

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

$$\begin{array}{rrcl} \max & c^t x & & \\ \text{s. t.} & Ax & \geq & b \\ & x & & \text{integral} \end{array}$$

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

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

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

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

$$
\begin{array}{llrcl}
(\mathrm{P}) & \max & c^t x & & \\
& \text{s. t.} & Ax & \geq & b \\
& & x & \geq & 0
\end{array}
$$

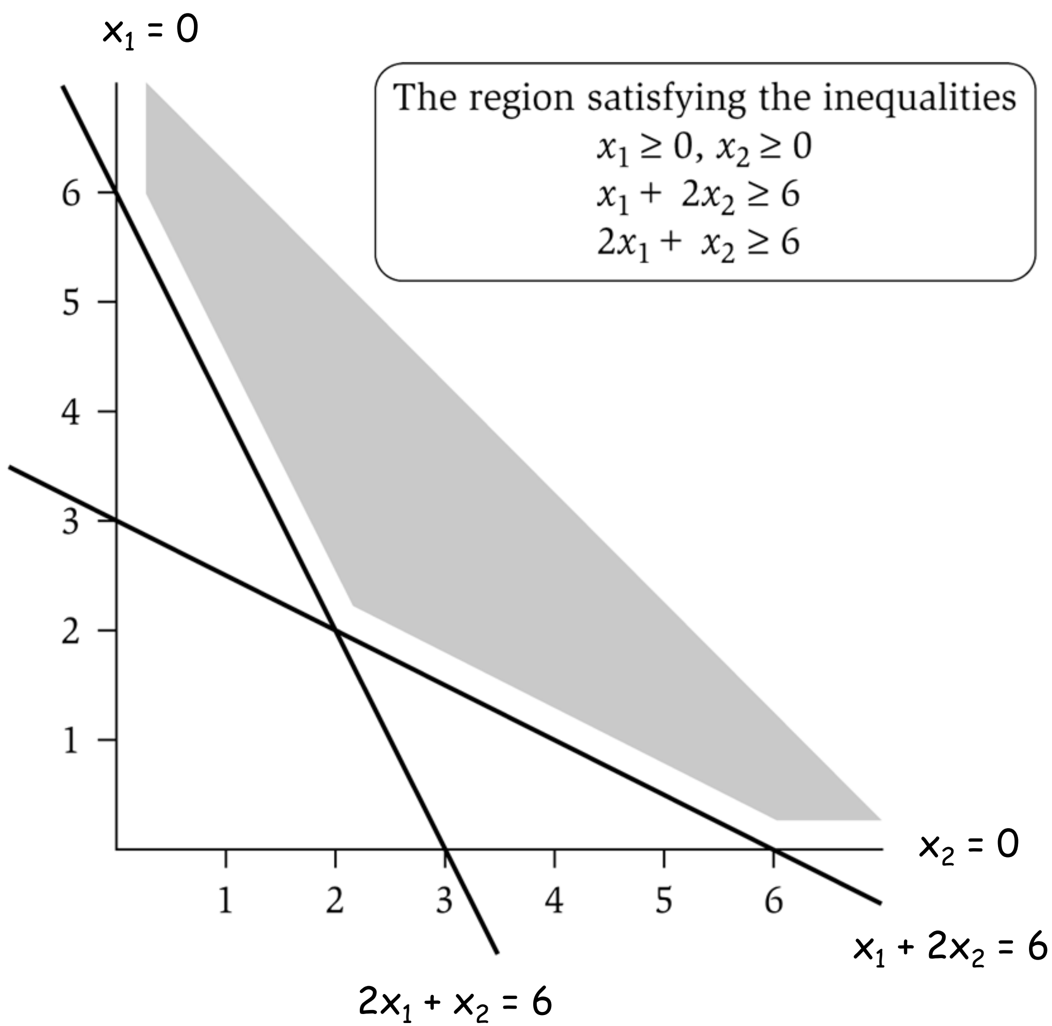
$$
\begin{array}{llrcll}
(\mathrm{P}) & \max & \sum_{j=1}^{n} c_j x_j & & & \\
& \text{s. t.} & \sum_{j=1}^{n} a_{ij} x_j & \geq & b_i & 1 \leq i \leq m \\
& & x_j & \geq & 0 & 1 \leq j \leq n
\end{array}
$$

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.

# LP Feasible Region

LP geometry in 2D.



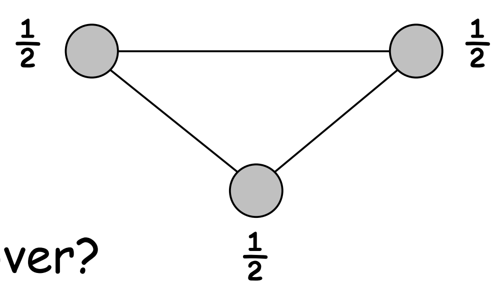
# Weighted Vertex Cover: LP Relaxation

Weighted vertex cover. Linear programming formulation.

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

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.

# Weighted Vertex Cover

**Theorem.** If $x^*$ is optimal solution to (LP), then $S = \{i \in V : x^*_i \geq \frac{1}{2}\}$ 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 $x^*_i \geq \frac{1}{2}$ or $x^*_j \geq \frac{1}{2}$ $\Rightarrow$ $(i, j)$ covered.

**Pf.** [S has desired cost]

- Let $S^*$ be optimal vertex cover. Then

$$\sum_{i \in S^*} w_i \underbrace{\geq}_{\text{LP is a relaxation}} \sum_{i \in S} w_i x_i^* \underbrace{\geq}_{x^*_i \geq \frac{1}{2}} \frac{1}{2} \sum_{i \in S} w_i$$

# Weighted Vertex Cover

Theorem. 2-approximation algorithm for weighted vertex cover.

Theorem. [Dinur-Safra 2001] If $P \neq NP$, then no $\rho$-approximation for $\rho < 1.3607$, even with unit weights.

$10\sqrt{5} - 21$

Open research problem. Close the gap.

# Knapsack Problem

# Polynomial Time Approximation Scheme

PTAS. $(1 + \varepsilon)$-approximation algorithm for any constant $\varepsilon > 0$.

- Euclidean TSP. [Arora 1996]

Consequence. PTAS produces arbitrarily high quality solution, but trades off accuracy for time.

This section. PTAS for knapsack problem via rounding and scaling.

# Knapsack Problem

Knapsack problem.

- Given n objects and a "knapsack."
- Item i has value $v_i > 0$ and weighs $w_i > 0$. ← we'll assume $w_i \leq W$
- Knapsack can carry weight up to W.
- Goal: fill knapsack so as to maximize total value.

Ex: { 3, 4 } has value 40.

W = 11

| Item | Value | Weight |
|---|---|---|
| 1 | 1 | 1 |
| 2 | 6 | 2 |
| 3 | 18 | 5 |
| 4 | 22 | 6 |
| 5 | 28 | 7 |

# Knapsack is NP-Complete

KNAPSACK: Given a finite set X, nonnegative weights $w_i$, nonnegative values $v_i$, a weight limit W, and a target value V, is there a subset $S \subseteq X$ such that:

$$\sum_{i \in S} w_i \leq W$$
$$\sum_{i \in S} v_i \geq V$$

SUBSET-SUM: Given a finite set X, nonnegative values $u_i$, and an integer U, is there a subset $S \subseteq X$ whose elements sum to exactly U?

Claim. SUBSET-SUM $\leq_P$ KNAPSACK.
Pf. Given instance $(u_1, \ldots, u_n, U)$ of SUBSET-SUM, create KNAPSACK instance:

$$v_i = w_i = u_i \qquad \sum_{i \in S} u_i \leq U$$
$$V = W = U \qquad \sum_{i \in S} u_i \geq U$$

# Knapsack Problem: Dynamic Programming 1

**Def.** OPT(i, w) = max value subset of items 1,..., i with weight limit w.

- Case 1: OPT does not select item i.
  - OPT selects best of 1, …, i–1 using up to weight limit w
- Case 2: OPT selects item i.
  - new weight limit = $w - w_i$
  - OPT selects best of 1, …, i–1 using up to weight limit $w - w_i$

$$OPT(i, w) = \begin{cases} 0 & \text{if } i = 0 \\ OPT(i-1, w) & \text{if } w_i > w \\ \max\{\, OPT(i-1, w),\ \ v_i + OPT(i-1, w-w_i) \,\} & \text{otherwise} \end{cases}$$

**Running time.** O(n W).

- W = weight limit.
- Not polynomial in input size!

# Knapsack Problem: Dynamic Programming II

Def. OPT(i, v) = min weight subset of items 1, …, i that yields value exactly v.

- Case 1: OPT does not select item i.
  - OPT selects best of 1, …, i-1 that achieves exactly value v
- Case 2: OPT selects item i.
  - consumes weight $w_i$, new value needed = v – $v_i$
  - OPT selects best of 1, …, i-1 that achieves exactly value v

$$OPT(i,v) = \begin{cases} 0 & \text{if } v = 0 \\ \infty & \text{if } i = 0,\ v > 0 \\ OPT(i-1, v) & \text{if } v_i > v \\ \min\{\, OPT(i-1, v),\ \ w_i + OPT(i-1, v - v_i) \,\} & \text{otherwise} \end{cases}$$

$V^* \le n\, v_{max}$

Running time. $O(n\, V^*) = O(n^2\, v_{max})$.

- $V^*$ = optimal value = maximum v such that $OPT(n, v) \le W$.
- Not polynomial in input size!

# Knapsack: FPTAS

Intuition for approximation algorithm.

- Round all values up to lie in smaller range.
- Run dynamic programming algorithm on rounded instance.
- Return optimal items in rounded instance.

| Item | Value | Weight |
|---|---|---|
| 1 | 934,221 | 1 |
| 2 | 5,956,342 | 2 |
| 3 | 17,810,013 | 5 |
| 4 | 21,217,800 | 6 |
| 5 | 27,343,199 | 7 |

W = 11

original instance

| Item | Value | Weight |
|---|---|---|
| 1 | 1 | 1 |
| 2 | 6 | 2 |
| 3 | 18 | 5 |
| 4 | 22 | 6 |
| 5 | 28 | 7 |

W = 11

rounded instance

# Knapsack: FPTAS

**Knapsack FPTAS.** Round up all values: $\bar{v}_i = \left\lceil \dfrac{v_i}{\theta} \right\rceil \theta, \quad \hat{v}_i = \left\lceil \dfrac{v_i}{\theta} \right\rceil$

- $v_{max}$ = largest value in original instance
- $\varepsilon$ = precision parameter
- $\theta$ = scaling factor = $\varepsilon\, v_{max} / n$

**Observation.** Optimal solution to problems with $\bar{v}$ or $\hat{v}$ are equivalent.

**Intuition.** $\bar{v}$ close to v so optimal solution using $\bar{v}$ is nearly optimal; $\hat{v}$ small and integral so dynamic programming algorithm is fast.

**Running time.** $O(n^3 / \varepsilon)$.

- Dynamic program II running time is $O(n^2\, \hat{v}_{\max})$, where

$$\hat{v}_{\max} = \left\lceil \frac{v_{\max}}{\theta} \right\rceil = \left\lceil \frac{n}{\varepsilon} \right\rceil$$

# Knapsack: FPTAS

**Knapsack FPTAS.** Round up all values: $\bar{v}_i = \left\lceil \frac{v_i}{\theta} \right\rceil \theta$

**Theorem.** If $S$ is solution found by our algorithm and $S^*$ is any other feasible solution then $(1+\varepsilon) \sum_{i \in S} v_i \geq \sum_{i \in S^*} v_i$

**Pf.** Let $S^*$ be any feasible solution satisfying weight constraint.

$$
\begin{aligned}
\sum_{i \in S^*} v_i &\leq \sum_{i \in S^*} \bar{v}_i && \text{always round up} \\
&\leq \sum_{i \in S} \bar{v}_i && \text{solve rounded instance optimally} \\
&\leq \sum_{i \in S} (v_i + \theta) && \text{never round up by more than } \theta \\
&\leq \sum_{i \in S} v_i + n\theta && |S| \leq n \\
&\leq (1+\varepsilon) \sum_{i \in S} v_i && n\theta = \varepsilon v_{max},\ v_{max} \leq \Sigma_{i \in S} v_i
\end{aligned}
$$

DP alg can take $v_{max}$

# References

# References

- Sections 11.2, 11.6 and 11.8 of the text book “algorithm design” by Jon Kleinberg and Eva Tardos
- The original slides were prepared by Kevin Wayne. The slides are distributed by Pearson Addison-Wesley.