Chapter 8
Approximation Algorithms

Algorithm Theory
WS 2015/16

Fabian Kuhn

Minimum (Weighted) Set Cover

Minimum Set Cover:

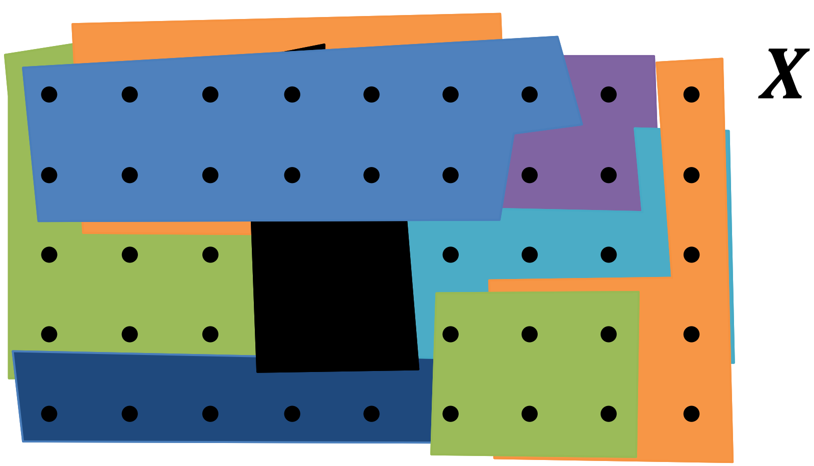
- **Goal:** Find a set cover $\mathcal{C}$ of smallest possible size
  - i.e., over X with as few sets as possible

Minimum Weighted Set Cover:

- Each set $S \in \mathcal{S}$ has a weight $w_S > 0$
- **Goal:** Find a set cover $\mathcal{C}$ of minimum weight

$X,\ \mathcal{S} \subseteq 2^X$

Example:



Weighted Set Cover: Greedy Algorithm

Greedy Weighted Set Cover Algorithm:

- Start with $\mathcal{C} = \emptyset$
- In each step, add set $S \in \mathcal{S} \setminus \mathcal{C}$ with the best weight per newly covered element ratio (set with best efficiency):

$$S = \arg\min_{S \in \mathcal{S} \setminus \mathcal{C}} \frac{w_S}{\left|S \setminus \bigcup_{T \in \mathcal{C}} T\right|}$$

Analysis of Greedy Algorithm:

- Assign a price $p(x)$ to each element $x \in X$:
  The efficiency of the set when covering the element
- If covering x with set S, if partial cover is $\mathcal{C}$ before adding S:

$$p(x) = \frac{w_S}{\left|S \setminus \bigcup_{T \in \mathcal{C}} T\right|}$$

S

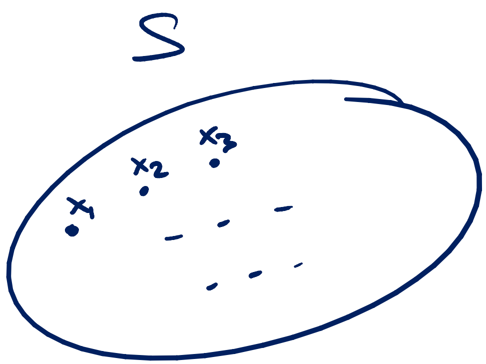
ω_S

Weighted Set Cover: Greedy Algorithm

Corollary: The total price of a set $S \in \mathcal{S}$ of size $|S| = k$ is

$$\sum_{x \in S} p(x) \le w_S \cdot H_k, \qquad \text{where } H_k = \sum_{i=1}^{k} \frac{1}{i} \le 1 + \ln k$$

Theorem: The approximation ratio of the greedy minimum (weighted) set cover algorithm is at most $\boldsymbol{H_s \le 1 + \ln s}$, where s is the cardinality of the largest set ($s = \max_{S \in \mathcal{S}} |S|$).



$$\sum_{x \in S} p(x) \le w_S \cdot H_{|S|}$$

price for covering S

$$H_{|S|} = 1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{|S|}$$

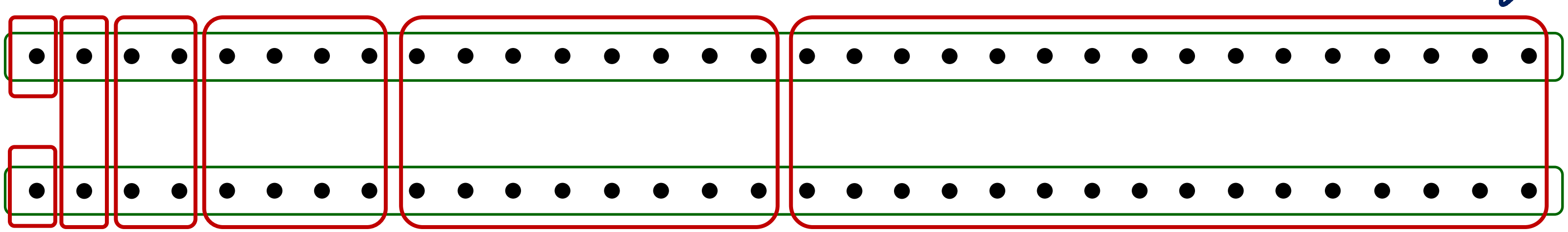
Set Cover Greedy Algorithm

Can we improve this analysis?

No! Even for the unweighted minimum set cover problem, the approximation ratio of the greedy algorithm is $\geq \big(1 - o(1)\big) \cdot \ln s$.

- if s is the size of the largest set… (s can be linear in n)

Let's show that the approximation ratio is at least $\Omega(\log n)$…



$$\mathbf{OPT = 2}$$

$$\mathbf{GREEDY \geq \log_2 n}$$

Set Cover: Better Algorithm?

An approximation ratio of $\ln n$ seems not spectacular...

Can we improve the approximation ratio?

$n^{O(1)}$ $\quad$ $n^{O(\frac{n}{\log n})}$

No, unfortunately not, unless $\mathrm{P} \approx \mathrm{NP}$

Feige showed that unless NP has deterministic $n^{O(\log\log n)}$-time algorithms, minimum set cover cannot be approximated better than by a factor $\big(1 - o(1)\big) \cdot \ln n$ in polynomial time.

- Proof is based on the so-called PCP theorem
  - PCP theorem is one of the main (relatively) recent advancements in theoretical computer science and the major tool to prove approximation hardness lower bounds
  - Shows that every language in NP has certificates of polynomial length that can be checked by a randomized algorithm by only querying a constant number of bits (for any constant error probability)

Connection to Linear Programming

Linear Programming:

- minimize/maximize a linear function (over $\mathbb{R}$)
- subject to linear side constraints

min $a_1 x_1 + a_2 x_2 + \dots + a_n x_n$

$2x_1 - 3x_3 + x_5 \le 7$

$x_i \in \mathbb{R}$

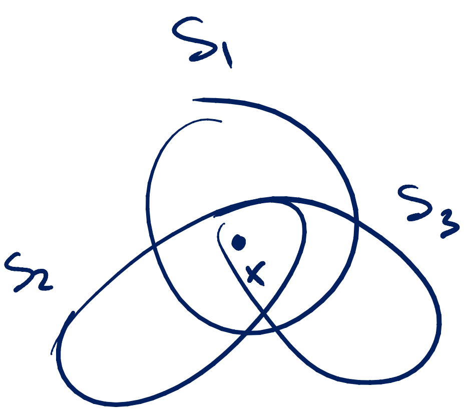
$x_i \in \mathbb{Z}$

Set Cover as an Integer Linear Program (LP):

- Given elements X and sets $\mathcal{S} \subseteq 2^X$
- Define a variable $\alpha_S \in \{0,1\}$ for each set $S \in \mathcal{S}$

$$\min \sum_{S \in \mathcal{S}} w_S \cdot \alpha_S$$

$\sum w_S \alpha_S^* \le |w(C^*)|$



$$\forall x \in X : \sum_{S \in \mathcal{S} : x \in S} \alpha_S \ge 1$$

$$\forall S \in \mathcal{S} : \boldsymbol{\alpha_S \ge 0}$$

LP relaxation

$\alpha_S = 0.5$

LP Duality

Strong LP Duality: Every linear program (LP) has a dual LP with the same optimal value of the objective function.

LP:

variables

$$\begin{aligned} &\min \boldsymbol{c}^T \boldsymbol{x} \\ &\text{s.t.} \quad A \cdot \boldsymbol{x} \geq \boldsymbol{b} \\ &\qquad\quad \boldsymbol{x} \geq \boldsymbol{0} \end{aligned}$$

x, y feasible:

x^*, y^* optimal sol.

Dual LP:

$$\begin{aligned} &\max \boldsymbol{b}^T \boldsymbol{y} \\ &\text{s.t.} \quad A^T \cdot \boldsymbol{y} \leq \boldsymbol{c} \\ &\qquad\quad \boldsymbol{y} \geq \boldsymbol{0} \end{aligned}$$

$b^T y \leq c^T x$ (weak duality)

$b^T y^* = c^T x^*$

Dual Set Cover LP

$\frac{1}{H_n}\sum p(x) = \sum_x \beta_x \leq \sum_S w_S \alpha_S^* \leq |w(C^*)|$

size of greedy solution

(Fractional) Set Cover LP:

$$\min \sum_{S \in \mathcal{S}} w_S \cdot \alpha_S$$

$$\forall x \in X: \sum_{S \in \mathcal{S}: x \in S} \alpha_S \geq 1$$

$$\forall S \in \mathcal{S}: \alpha_S \geq 0$$

(Fractional) Set Cover Dual LP:

$$\max \sum_{x \in X} \beta_x$$

$$\forall S \in \mathcal{S}: \sum_{x \in S} \beta_x \leq w_S$$

$$\forall x \in X: \beta_x \geq 0$$

variable for each element $x \in X$

β_x

max $\sum \beta_x$

β_{x_2}: x_2

β_{x_1}: x_1

$\sum \leq w_S$

price

$p(x)$ for each element

we showed:

$\forall S \subset \mathcal{S}: \sum_{x \in S} p(x) \leq w_S \cdot H_{|S|}$

$\leq w_S \cdot H_n$

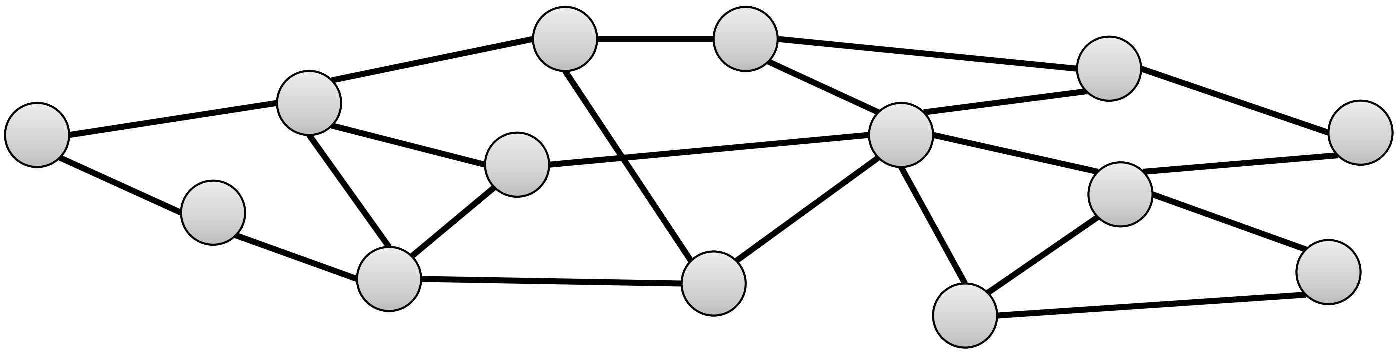
dual fitting

$\rightarrow \beta_x := \frac{p(x)}{H_n} \rightarrow$ feasible dual sol.

Set Cover: Special Cases

Vertex Cover: set $S \subseteq V$ of nodes of a graph $G = (V, E)$ such that

$$\forall \{u, v\} \in E, \qquad \{u, v\} \cap S \neq \emptyset.$$



2-approx.

Minimum Vertex Cover:

- Find a vertex cover of minimum cardinality

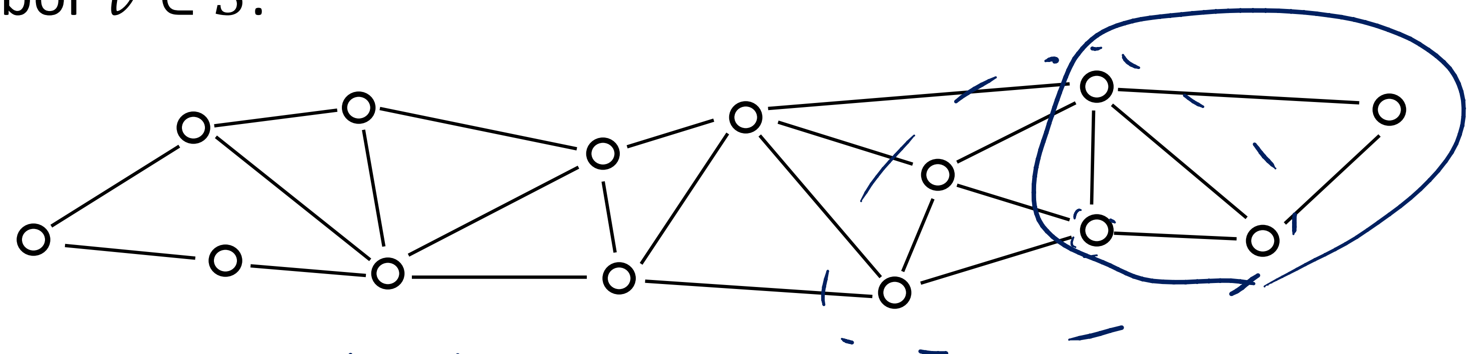
Minimum Weighted Vertex Cover:

- Each node has a weight
- Find a vertex cover of minimum total weight

Set Cover: Special Cases

Dominating Set:

Given a graph $G = (V, E)$, a dominating set $S \subseteq V$ is a subset of the nodes V of G such that for all nodes $u \in V \setminus S$, there is a neighbor $v \in S$.



unweighted / weighted

greedy: approx. ratio : $H_{\Delta+1} \approx \ln \Delta$

Minimum Hitting Set

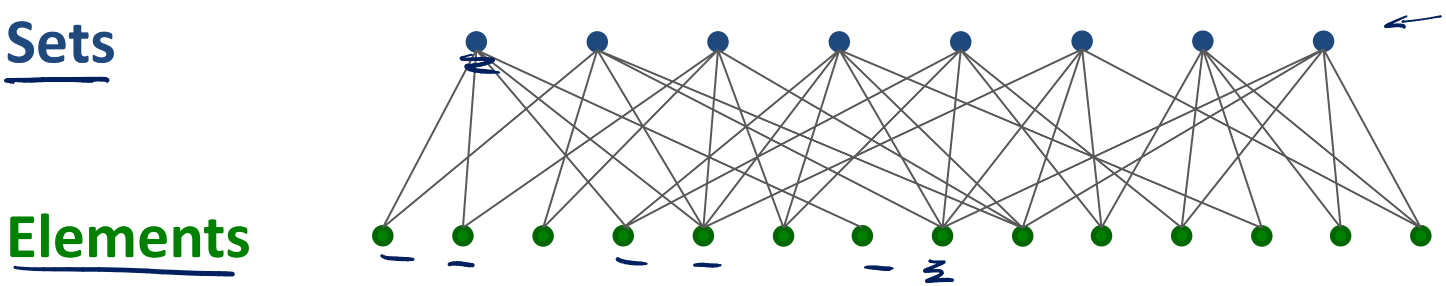
Given: Set of elements X and collection of subsets $\mathcal{S} \subseteq 2^X$

– Sets cover X: $\bigcup_{S \in \mathcal{S}} S = X$

Goal: Find a min. cardinality subset $H \subseteq X$ of elements such that

$$\forall S \in \mathcal{S} : S \cap H \neq \emptyset$$

Problem is equivalent to min. set cover with roles of sets and elements interchanged



Knapsack

- n items $1, \dots, n$, each item has weight $w_i > 0$ and value $v_i > 0$
- Knapsack (bag) of capacity W

- Goal: pack items into knapsack such that total weight is at most W and total value is maximized:

$$\max \sum_{i \in S} v_i$$

$$\text{s.t.} \quad S \subseteq \{1, \dots, n\} \text{ and } \sum_{i \in S} w_i \leq W$$

$S \subseteq \{1, \dots, n\}$

$x_i \in \{0, 1\}$

- E.g.: jobs of length w_i and value v_i, server available for W time units, try to execute a set of jobs that maximizes the total value

Knapsack: Dynamic Programming Alg.

We have shown:

- If all item weights w_i are integers, using dynamic programming, the knapsack problem can be solved in time $O(nW)$
- If all values v_i are integers, there is another dynamic progr. algorithm that runs in time $O(n^2V)$, where V is the max. value.

Problems:

- If W and V are large, the algorithms are not polynomial in n
- If the values or weights are not integers, things are even worse (and in general, the algorithms cannot even be applied at all)

Idea:

- Can we adapt one of the algorithms to at least compute an approximate solution?

Approximation Algorithm

- The algorithm has a parameter $\varepsilon > 0$
- We assume that each item alone fits into the knapsack (w.l.o.g.)
- We define:

$$V := \max_{1\leq i\leq n} v_i, \qquad \forall i: \hat{v}_i := \left\lceil \frac{v_i n}{\varepsilon V} \right\rceil, \qquad \hat{V} := \max_{1\leq i\leq n} \hat{v}_i$$

- We solve the problem with integer values $\hat{v}_i$ and weights w_i using dynamic programming in time $O(n^2 \cdot \hat{V})$
- If solution value $< V$, we take item with value V instead

Theorem: The described algorithm runs in time $O(n^3/\varepsilon)$.

$\varepsilon = \frac{1}{\text{poly}(n)}$

Proof:

$$\hat{V} = \max_{1\leq i\leq n} \hat{v}_i = \max_{1\leq i\leq n} \left\lceil \frac{v_i n}{\varepsilon V} \right\rceil = \left\lceil \frac{\cancel{V} n}{\varepsilon \cancel{V}} \right\rceil = \left\lceil \frac{n}{\varepsilon} \right\rceil$$

$1+\varepsilon$