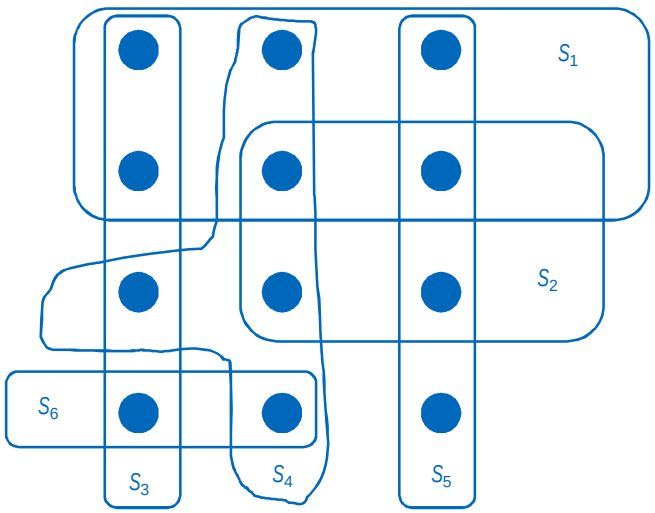
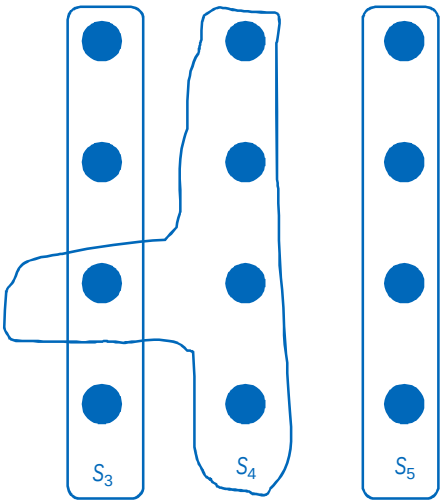
35.3 The set-covering problem

- An instance $(X, \mathcal{F})$ of the set-covering problem consists of a finite set X (universe) and a family $\mathcal{F}$ (cover) of subsets of X, such that every element of X belongs to at least one subset in $\mathcal{F}$:
  $$X = \bigcup_{S \in \mathcal{F}} S$$
- A subset $S \in \mathcal{F}$ covers its elements
- The problem is to find a minimum-size subset (subcover) $\mathcal{C} \subseteq \mathcal{F}$ whose members cover all of X:
  $$X = \bigcup_{S \in \mathcal{C}} S$$
- Any $\mathcal{C}$ satisfying this equation covers X



Optimal: 3 subsets



- The set-covering problem abstracts many commonly arising combinatorial problems
- Let X represents a set of skills that are needed to solve a problem and we are given set of people available to work on the problem
- Form a committee, containing as few people as possible, s.t. for every requisite skill in X, at least one member of the committee has that skill
- In the decision version of the set-covering problem, we ask whether a covering exists with size at most k, where k is an additional parameter specified in the problem instance

- The corresponding decision problem
  - **Given**: a ground set X, cover $\mathcal{F}$ and a natural number k
  - **Question**: Does X have a subcover $\mathcal{C} \subseteq \mathcal{F}$ s.t. $|\mathcal{C}| \leq k$?

Theorem *The decision version of minimum set cover (minSC) problem is NP-complete.*

Proof. Obviously minSC ∈ NP: Let us guess from the given cover $\mathcal{F}$ a subcover $\mathcal{C}$ containing k subsets and verify deterministically in polynomial time that we really have a subcover.

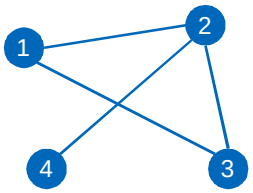
- Polynomial time reduction VC $\leq_m^p$ minSC is easy to give. Let $\langle G, k \rangle$ be an instance of the vertex cover in which $G = (V, E)$. We choose the mapping f:

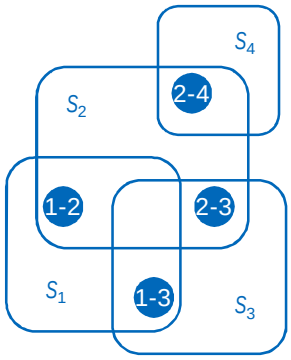
$$f(\langle (V, E), k \rangle) = \langle E, V_E, k \rangle,$$

- where V_E is the collection of edges connected to the nodes of G. In other words, each $v \in V$ has a corresponding set

$$\{e \in E \mid e = (v, w)\}$$

- Clearly f is computable in polynomial time and is a reduction. □



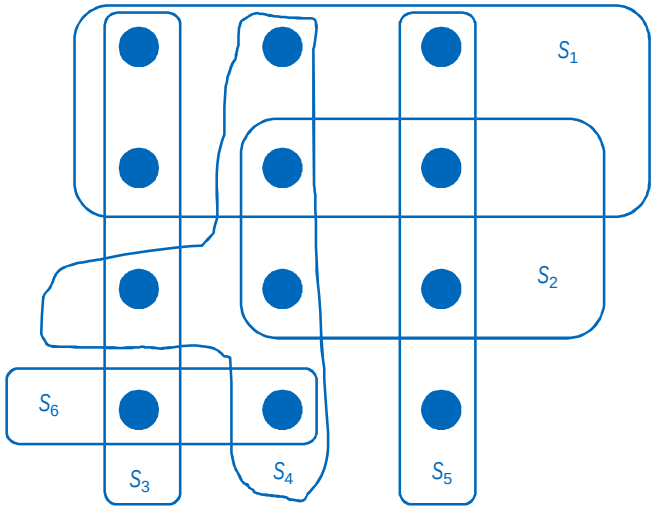


- Hence, minSC is an intractable problem – we do not know of a polynomial time algorithm for solving it
- Therefore, we attempt to find a polynomial time approximation algorithm for it
  - does not necessarily give the best possible (optimal) solution, but
  - can be shown always to be at most a function of the input length worse than the optimal solution
- Let Opt be the cost of the solution given by an optimal algorithm and App that of the solution given by an approximation algorithm

GREEDY-SET-COVER$(X, \mathcal{F})$

1. $U \leftarrow X$
2. $\mathcal{C} \leftarrow \emptyset$
3. **while** $U \neq \emptyset$
4. select an $S \in \mathcal{F}$ that maximizes $|S \cap U|$
5. $U \leftarrow U - S$
6. $\mathcal{C} \leftarrow \mathcal{C} \cup \{S\}$
7. **return** $\mathcal{C}$

Greedy: 4 subsets



- The greedy algorithm can quite easily be implemented to run in polynomial time in the length of the input $|X|$ and $|\mathcal{F}|$
  - The loop in row 3 is executed at most $\min(|X|, |\mathcal{F}|)$ times and the body of the loop itself can be implemented to require time $O(|X| \cdot |\mathcal{F}|)$
  - Altogether the time requirement thus is
    $$O(|X| \cdot |\mathcal{F}| \min(|X|, |\mathcal{F}|))$$

  It is also possible to give a linear time implementation for the greedy approximation algorithm for set cover
- The collection $\mathcal{C}$ returned by the algorithm is obviously a set cover, because the loop of row 3 is executed until there are no more elements to cover

- In order to relate the cost of the set cover returned by the greedy algorithm, we set cost 1 to each of the chosen sets
- Let S_i be the set selected by the greedy algorithm at round i
- We distribute the cost of S_i evenly among all those elements in it that now become covered for the first time
- Let c_x denote the cost assigned on element $x \in X$
- Each element gets assigned a cost only once, the first time it is covered by some set

- If x is first covered by the set S_i, the cost assigned to it is:

$$c_x = \frac{1}{|S_i - (S_1 \cup S_2 \cup \cdots \cup S_{i-1})|}$$

- Each step of the algorithm assigns 1 unit of cost:

$$\text{App} = |\mathcal{C}| = \sum_{x \in X} c_x$$

- Each element $x \in X$ is in at least one set in the optimal cover $\mathcal{C}^*$, and so we have

$$\sum_{S \in \mathcal{C}^*} \sum_{x \in S} c_x \geq \sum_{x \in X} c_x$$

- Combining previous equation/inequality, yields

$$\text{App} = |\mathcal{C}| \leq \sum_{S \in \mathcal{C}^*} \sum_{x \in S} c_x$$

- Let $H(k)$ denote the k-th harmonic number

$$H(k) = \sum_{j=1}^{k} \frac{1}{j} = 1 + \frac{1}{2} + \cdots + \frac{1}{k}$$

- We define $H(0) = 0$
- For any set S belonging to the family $\mathcal{F}$,

$$\sum_{x \in S} c_x \leq H(|S|)$$

- Then by previous inequality

$$\text{App} = |\mathcal{C}| \leq \sum_{S \in \mathcal{C}^*} H(|S|)$$
$$\leq |\mathcal{C}^*| \cdot H(\max\{|S| : S \in \mathcal{F}\})$$
$$= \text{Opt} \cdot H(\max\{|S| : S \in \mathcal{F}\})$$

Lemma *For each $S \in \mathcal{F}$ it holds*

$$\sum_{x \in S} c_x \leq H(|S|)$$

Proof. Let $S \in \mathcal{F}$ be arbitrary and $i = 1, 2, \dots, |\mathcal{C}|$. Furthermore, let

$$n_i = |S - (S_1 \cup S_2 \cup \cdots \cup S_i)|$$

be the number of those elements of S that have not yet been covered when the greedy algorithm has chosen sets $S_1, S_2, \dots, S_i$ to the set cover.

Let $n_0 = |S|$. Let k be the smallest index s.t. $n_k = 0$; i.e., every element of S belongs to at least one of the sets $S_1, S_2, \dots, S_k$.

Then $n_{i-1} \geq n_i$ and $S_i, i = 1, 2, \ldots, k$, covers $n_{i-1} - n_i$ elements for the first time.

$$\sum_{x \in S} c_x = \sum_{i=1}^{k} (n_{i-1} - n_i) \frac{1}{|S_i - (S_1 \cup S_2 \cup \cdots \cup S_{i-1})|}$$

Since S_i is chosen greedily, it must cover at least as many elements as the set S (or otherwise S should have been selected). Hence

$|S_i - (S_1 \cup \cdots \cup S_{i-1})| \geq |S - (S_1 \cup \cdots \cup S_{i-1})| = n_{i-1}$

Which further yields

$$\sum_{x \in S} c_x \leq \sum_{i=1}^{k} (n_{i-1} - n_i) \frac{1}{n_{i-1}}$$

$$\sum_{x \in S} c_x \leq \sum_{i=1}^{k} (n_{i-1} - n_i) \frac{1}{n_{i-1}}$$

$$= \sum_{i=1}^{k} \sum_{j=n_i+1}^{n_{i-1}} \frac{1}{n_{i-1}}$$

$$\leq \sum_{i=1}^{k} \sum_{j=n_i+1}^{n_{i-1}} \frac{1}{j}$$

because $j \leq n_{i-1}$. Moreover,

$$\leq \sum_{i=1}^{k} \left(\sum_{j=1}^{n_{i-1}} \frac{1}{j} - \sum_{j=1}^{n_i} \frac{1}{j} \right)$$

$$= \sum_{i=1}^{k} (H(n_{i-1}) - H(n_i))$$
$$= H(n_0) - H(n_k)$$

We have chosen $n_k = 0$ and defined $H(0) = 0$.
Therefore, further

$$= H(n_0) - H(0)$$
$$= H(n_0)$$
$$= H(|S|)$$

and we have proved the lemma. □

- For the harmonic number $H(k)$ it holds $\ln k < H(k) \leq \ln k + 1$
- From the above results it follows:

Theorem *For the greedy algorithm of the set cover problem it holds that*

$$\frac{\text{App}}{\text{Opt}} \leq H(\max\{|S| : S \in \mathcal{F}\}) \leq \ln|X| + 1$$

- In some applications $\max\{|S| : S \in \mathcal{F}\}$ is a small constant
- Then the solution returned by the greedy algorithm is only a small constant away from the optimal one
- In particular, if subsets S have an upper bound d for their size,

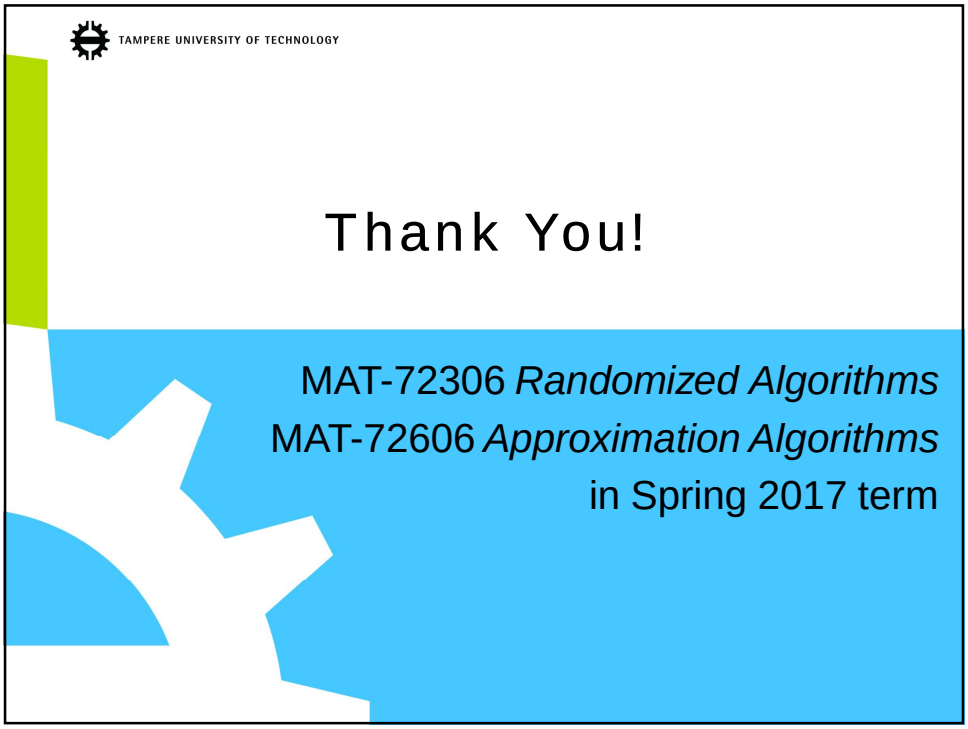
$$\text{App}/\text{Opt} \leq H(d)$$

- E.g., when the nodes of the graph of vertex cover have maximum degree 3, then
  - the solution returned by the greedy set cover algorithm is at most $H(3) = 11/6 < 2$ times as large as the optimal cover

- Feige, 1996: no polynomial-time algorithm can approximate minSC within $(1-\varepsilon)\ln m$, for any $\varepsilon > 0$, unless NP $\subseteq$ DTIME($n^{\log\log n}$)
- Hence, it is not possible to find an approximation algorithm for minSC that would be significantly better than the greedy one
- Slavík, 1996: A more exact upper bound for the approximation ratio of the greedy algorithm is

$$\ln m - \ln\ln m + \Theta(1)$$

- In fact this is also a lower bound for the approximation ratio of the greedy algorithm
- $\ln m - \ln\ln m + \Theta(1)$ is thus the asymptotically exact approximation ratio of the greedy algorithm


TAMPERE UNIVERSITY OF TECHNOLOGY
Thank You!
MAT-72306 *Randomized Algorithms*
MAT-72606 *Approximation Algorithms*
in Spring 2017 term