Technique 1: Round the LP solution.

We first solve the LP-relaxation and then we round the fractional values so that we obtain an integral solution.

Set Cover relaxation:

$$\begin{array}{llll} \min & & \sum_{i=1}^{k} w_i x_i & \\ \text{s.t.} & \forall u \in U & \sum_{i:u\in S_i} x_i & \geq & 1 \\ & \forall i \in \{1,\dots,k\} & x_i & \in & [0,1] \end{array}$$

Let f_u be the number of sets that the element u is contained in (the frequency of u). Let $f = \max_u \{f_u\}$ be the maximum frequency.

Technique 1: Round the LP solution.

Rounding Algorithm:
Set all x_i-values with $x_i \geq \frac{1}{f}$ to 1. Set all other x_i-values to 0.

Technique 1: Round the LP solution.

Lemma 2
The rounding algorithm gives an f-approximation.

Proof: Every $u \in U$ is covered.

- We know that $\sum_{i:u\in S_i} x_i \geq 1$.
- The sum contains at most $f_u \leq f$ elements.
- Therefore one of the sets that contain u must have $x_i \geq 1/f$.
- This set will be selected. Hence, u is covered.

Technique 1: Round the LP solution.

The cost of the rounded solution is at most $f \cdot \text{OPT}$.

$$\begin{aligned} \sum_{i\in I} w_i &\leq \sum_{i=1}^{k} w_i (f \cdot x_i) \\ &= f \cdot \text{cost}(x) \\ &\leq f \cdot \text{OPT} \ . \end{aligned}$$

Technique 2: Rounding the Dual Solution.

Relaxation for Set Cover

Primal:

$$\begin{array}{lrl} \min & \sum_{i\in I} w_i x_i & \\ \text{s.t. } \forall u & \sum_{i:u\in S_i} x_i & \geq 1 \\ & x_i & \geq 0 \end{array}$$

Dual:

$$\begin{array}{lrl} \max & \sum_{u\in U} y_u & \\ \text{s.t. } \forall i & \sum_{u:u\in S_i} y_u & \leq w_i \\ & y_u & \geq 0 \end{array}$$

Technique 2: Rounding the Dual Solution.

Rounding Algorithm:
Let I denote the index set of sets for which the dual constraint is tight. This means for all $i \in I$

$$\sum_{u:u\in S_i} y_u = w_i$$

Technique 2: Rounding the Dual Solution.

Lemma 3
The resulting index set is an f-approximation.

Proof:
Every $u \in U$ is covered.

- Suppose there is a u that is not covered.
- This means $\sum_{u:u\in S_i} y_u < w_i$ for all sets S_i that contain u.
- But then y_u could be increased in the dual solution without violating any constraint. This is a contradiction to the fact that the dual solution is optimal.

Technique 2: Rounding the Dual Solution.

Proof:

$$\begin{aligned} \sum_{i\in I} w_i &= \sum_{i\in I} \sum_{u:u\in S_i} y_u \\ &= \sum_u |\{i \in I : u \in S_i\}| \cdot y_u \\ &\leq \sum_u f_u y_u \\ &\leq f \sum_u y_u \\ &\leq f \operatorname{cost}(x^*) \\ &\leq f \cdot \mathrm{OPT} \end{aligned}$$

Let I denote the solution obtained by the first rounding algorithm and I' be the solution returned by the second algorithm. Then

$$I \subseteq I' \ .$$

This means I' is never better than I.

- Suppose that we take S_i in the first algorithm. I.e., $i \in I$.
- This means $x_i \geq \frac{1}{f}$.
- Because of Complementary Slackness Conditions the corresponding constraint in the dual must be tight.
- Hence, the second algorithm will also choose S_i.

Technique 3: The Primal Dual Method

The previous two rounding algorithms have the disadvantage that it is necessary to solve the LP. The following method also gives an f-approximation without solving the LP.

For estimating the cost of the solution we only required two properties.

1. The solution is dual feasible and, hence,

$$\sum_u y_u \leq \mathrm{cost}(x^*) \leq \mathrm{OPT}$$

   where x^* is an optimum solution to the primal LP.
2. The set I contains only sets for which the dual inequality is tight.

Of course, we also need that I is a cover.

Technique 3: The Primal Dual Method

Algorithm 1 PrimalDual

1: $y \leftarrow 0$
2: $I \leftarrow \emptyset$
3: **while** exists $u \notin \bigcup_{i \in I} S_i$ **do**
4: increase dual variable y_u until constraint for some new set S_ℓ becomes tight
5: $I \leftarrow I \cup \{\ell\}$

Technique 4: The Greedy Algorithm

Algorithm 1 Greedy

1: $I \leftarrow \emptyset$
2: $\hat{S}_j \leftarrow S_j$ for all j
3: **while** I not a set cover **do**
4: $\ell \leftarrow \arg\min_{j:\hat{S}_j \neq 0} \frac{w_j}{|\hat{S}_j|}$
5: $I \leftarrow I \cup \{\ell\}$
6: $\hat{S}_j \leftarrow \hat{S}_j - S_\ell$ for all j

In every round the Greedy algorithm takes the set that covers remaining elements in the most cost-effective way.

We choose a set such that the ratio between cost and still uncovered elements in the set is minimized.

Technique 4: The Greedy Algorithm

Lemma 4
Given positive numbers $a_1, \dots, a_k$ and $b_1, \dots, b_k$, and $S \subseteq \{1, \dots, k\}$ then

$$\min_i \frac{a_i}{b_i} \le \frac{\sum_{i\in S} a_i}{\sum_{i\in S} b_i} \le \max_i \frac{a_i}{b_i}$$

Technique 4: The Greedy Algorithm

Let n_ℓ denote the number of elements that remain at the beginning of iteration ℓ. $n_1 = n = |U|$ and $n_{s+1} = 0$ if we need s iterations.

In the ℓ-th iteration

$$\min_j \frac{w_j}{|\hat{S}_j|} \le \frac{\sum_{j\in \mathrm{OPT}} w_j}{\sum_{j\in \mathrm{OPT}} |\hat{S}_j|} = \frac{\mathrm{OPT}}{\sum_{j\in \mathrm{OPT}} |\hat{S}_j|} \le \frac{\mathrm{OPT}}{n_\ell}$$

since an optimal algorithm can cover the remaining n_ℓ elements with cost OPT.

Let $\hat{S}_j$ be a subset that minimizes this ratio. Hence, $w_j/|\hat{S}_j| \le \frac{\mathrm{OPT}}{n_\ell}$.

Technique 4: The Greedy Algorithm

Adding this set to our solution means $n_{\ell+1} = n_\ell - |\hat{S}_j|$.

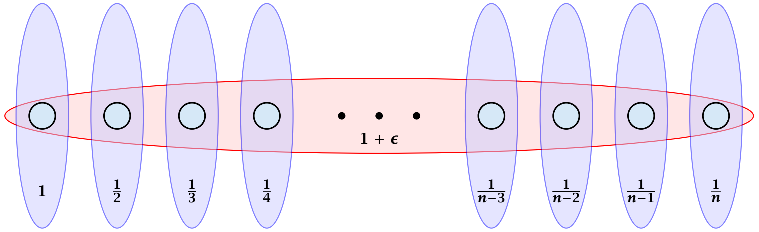
$$w_j \le \frac{|\hat{S}_j|\mathrm{OPT}}{n_\ell} = \frac{n_\ell - n_{\ell+1}}{n_\ell} \cdot \mathrm{OPT}$$

Technique 4: The Greedy Algorithm

$$\begin{aligned}
\sum_{j\in I} w_j &\le \sum_{\ell=1}^{s} \frac{n_\ell - n_{\ell+1}}{n_\ell} \cdot \mathrm{OPT} \\
&\le \mathrm{OPT} \sum_{\ell=1}^{s} \left(\frac{1}{n_\ell} + \frac{1}{n_\ell - 1} + \dots + \frac{1}{n_{\ell+1} + 1} \right) \\
&= \mathrm{OPT} \sum_{i=1}^{k} \frac{1}{i} \\
&= H_n \cdot \mathrm{OPT} \le \mathrm{OPT}(\ln n + 1) \ .
\end{aligned}$$

Technique 4: The Greedy Algorithm

A tight example:



Technique 5: Randomized Rounding

One round of randomized rounding:
Pick set S_j uniformly at random with probability $1 - x_j$ (for all j).

Version A: Repeat rounds until you have a cover.

Version B: Repeat for s rounds. If you have a cover STOP. Otherwise, repeat the whole algorithm.

Probability that $u \in U$ is not covered (in one round):

$$\begin{aligned}\Pr[u \text{ not covered in one round}] &= \prod_{j:u\in S_j}(1-x_j) \le \prod_{j:u\in S_j} e^{-x_j} \\ &= e^{-\sum_{j:u\in S_j} x_j} \le e^{-1} \ .\end{aligned}$$

Probability that $u \in U$ is not covered (after ℓ rounds):

$$\Pr[u \text{ not covered after } \ell \text{ round}] \le \frac{1}{e^{\ell}} \ .$$

$$\begin{aligned}&\Pr[\exists u \in U \text{ not covered after } \ell \text{ round}] \\ &\quad= \Pr[u_1 \text{ not covered} \lor u_2 \text{ not covered} \lor \ldots \lor u_n \text{ not covered}] \\ &\quad\le \sum_i \Pr[u_i \text{ not covered after } \ell \text{ rounds}] \le ne^{-\ell} \ .\end{aligned}$$

Lemma 5
With high probability $\mathcal{O}(\log n)$ rounds suffice.

With high probability:
For any constant α the number of rounds is at most $\mathcal{O}(\log n)$ with probability at least $1 - n^{-\alpha}$.

Proof: We have

$$\Pr[\#\text{rounds} \geq (\alpha+1)\ln n] \leq ne^{-(\alpha+1)\ln n} = n^{-\alpha} .$$

Expected Cost

- Version A.
  Repeat for $s = (\alpha+1)\ln n$ rounds. If you don't have a cover simply take for each element u the cheapest set that contains u.

$$E[\text{cost}] \leq (\alpha+1)\ln n \cdot \text{cost}(LP) + (n \cdot \text{OPT})n^{-\alpha} = \mathcal{O}(\ln n) \cdot \text{OPT}$$

Expected Cost

- Version B.
  Repeat for $s = (\alpha+1)\ln n$ rounds. If you don't have a cover simply repeat the whole process.

$$E[\text{cost}] = \Pr[\text{success}] \cdot E[\text{cost} \mid \text{success}] + \Pr[\text{no success}] \cdot E[\text{cost} \mid \text{no success}]$$

This means

$$\begin{aligned} & E[\text{cost} \mid \text{success}] \\ &= \frac{1}{\Pr[\text{succ.}]}\Big(E[\text{cost}] - \Pr[\text{no success}] \cdot E[\text{cost} \mid \text{no success}]\Big) \\ &\leq \frac{1}{\Pr[\text{succ.}]}E[\text{cost}] \leq \frac{1}{1-n^{-\alpha}}(\alpha+1)\ln n \cdot \text{cost}(\text{LP}) \\ &\leq 2(\alpha+1)\ln n \cdot \text{OPT} \end{aligned}$$

for $n \geq 2$ and $\alpha \geq 1$.

Randomized rounding gives an $\mathcal{O}(\log n)$ approximation. The running time is polynomial with high probability.

Theorem 6 (without proof)

There is no approximation algorithm for set cover with approximation guarantee better than $\frac{1}{2}\log n$ unless NP has quasi-polynomial time algorithms (algorithms with running time $2^{\text{poly}(\log n)}$).

Integrality Gap

The integrality gap of the SetCover LP is $\Omega(\log n)$.

- $n = 2^k - 1$
- Elements are all vectors $\boldsymbol{i}$ over $GF[2]$ of length k (excluding zero vector).
- Every vector $\boldsymbol{j}$ defines a set as follows

$$S_{\boldsymbol{j}} := \{\boldsymbol{i} \mid \boldsymbol{i} \cdot \boldsymbol{j} = 1\}$$

- each set contains 2^{k-1} vectors; each vector is contained in 2^{k-1} sets
- $x_i = \frac{1}{2^{k-1}} = \frac{2}{n+1}$ is fractional solution.

Integrality Gap

Every collection of $p < k$ sets does not cover all elements.

Hence, we get a gap of $\Omega(\log n)$.

Techniques:

- Deterministic Rounding
- Rounding of the Dual
- Primal Dual
- Greedy
- Randomized Rounding
- Local Search
- Rounding Data + Dynamic Programming