Outline

Weighted Set-Cover

NP Completeness
Set-Cover

Problem:
- Input: A collection $C = \{S_1, S_2, \ldots, S_m\}$ of subsets of $U = \bigcup_{S \in C} S$ and weights $w : C \to \R^+$
- Output: Find $C' \subset C$ such that

$$U = \bigcup_{S \in C'} S$$

that minimizes $\sum_{S \in C'} w_S$. 
Greedy Set-Cover Algorithm

Algorithm

1. Let $R \leftarrow U$, $C' \leftarrow \emptyset$
2. While $R \neq \emptyset$:
   2.1 Pick $S \in \{S_1, \ldots, S_m\}$ be the set minimizing $\frac{w_S}{|S \cap R|}$
   2.2 $R \leftarrow R - S$ and $C' \leftarrow C' \cup \{S\}$
3. Return $C'$

Theorem
Approx ratio is $\frac{1}{d^*} + \frac{1}{d^*-1} + \ldots + \frac{1}{1} \approx \ln d^*$ where $d^* = \max_i |S_i|$

Definition
When $S$ is chosen, say $e \in S \cap R$ is covered at cost $c_e = \frac{w_S}{|S \cap R|}$. Note

$$\sum_{S \in C'} w_S = \sum_{e \in U} c_e .$$

Ex: If $S_1 = \{1, 2\}$, $S_2 = \{1, 2, 3\}$, $S_3 = \{3, 4\}$ where $w_{S_1} = 4$, $w_{S_2} = 7$, $w_{S_3} = 20$ then $c_1 = c_2 = 2$, $c_3 = 7$, and $c_4 = 20.$
Analysis 1/2

- Let the optimal solution $C_{\text{OPT}}$ have cost $w_{\text{OPT}}$.
- **Claim:** For each $S \in C_{\text{OPT}},$
  \[
  w_S \geq \sum_{e \in S} \frac{c_e}{H(|S|)}
  \]
  where $H(|S|) = \frac{1}{|S|} + \frac{1}{|S|-1} + \ldots + \frac{1}{1}$.
- Then,
  \[
  w_{\text{OPT}} \geq \sum_{S \in C_{\text{OPT}}} \sum_{e \in S} \frac{c_e}{H(|S|)} \geq \frac{1}{H(d^*)} \sum_{S \in C_{\text{OPT}}} \sum_{e \in S} c_e \geq \frac{1}{H(d^*)} \sum_{e \in U} c_e
  \]
- But $\sum_{e \in U} c_e = \sum_{S \in \mathcal{C}'} w_S$ and so $w_{\text{OPT}} \geq \frac{1}{H(d^*)} \sum_{S \in \mathcal{C}'} w_S$
Analysis 2/2

Claim

For all $S \in C$, $\sum_{e \in S} c_e \leq H(|S|) \cdot w_S$.

Proof.

- Suppose $S = \{e_1, \ldots, e_d\}$ be ordered according to order in which elements are covered (ties broken arbitrarily).
- Suppose $S'$ is chosen to cover $e_j$. Because algorithm is greedy,

$$c_{e_j} = \frac{w_{S'}}{|S' \cap R|} \leq \frac{w_S}{|S \cap R|}$$

- Before $e_j$ was covered $e_{j+1}, \ldots, e_d$ were also uncovered,

$$|S \cap R| \geq (d - j + 1)$$

- Therefore

$$\sum_{j=1}^{d} c_{e_j} \leq \sum_{j=1}^{d} \frac{w_S}{d - j + 1} = \frac{w_S}{d} + \frac{w_S}{d - 1} + \cdots + \frac{w_S}{1}$$
Outline

Weighted Set-Cover

NP Completeness
Recap: Clique and 3-SAT

CLIQUE:

- **Input**: Given graph $G = (V, E)$ and integer $k$.
- **Question**: Does $G$ contain a clique of size $k$, i.e., a subgraph with $k$ nodes in which all $\binom{k}{2}$ edges are present.

3-SAT:

- **Input**: A 3-SAT formula $\phi(x_1, \ldots, x_n)$, e.g.,

  $$(x_1 \lor \overline{x_2} \lor x_3) \land (\overline{x_1} \lor x_2 \lor \overline{x_3})$$

- **Question**: Is there a setting of each $x_i$ to TRUE or FALSE such that the formula is satisfied.

We showed that in polynomial time it is possible to transform any instance $\phi$ of 3-SAT into an instance $(G_\phi, k_\phi)$ of CLIQUE such that $\phi$ was a “yes” instance of 3-SAT iff $(G_\phi, k_\phi)$ was a “yes” instance of CLIQUE.

Hence, if there’s a polynomial time algorithm for CLIQUE then there’s also a polynomial time algorithm for 3-SAT.
Polynomial Time Reduction

Definition
Π is a decision problem if it only has a “yes” or “no” answer.

Definition
Given two decision problems Π₁, Π₂ we say Π₂ is polynomial time reducible to Π₁ iff there exists a polynomial time algorithm f that transforms any instance X of Π₂ to an instance f(X) of Π₁ such that:

\[(X \text{ is a “yes” instance of } Π₂) \iff (f(X) \text{ is a “yes” instance of } Π₁)\]

We write Π₂ ≤ₚ Π₁ to denote “Π₂ is polynomial time reducible to Π₁”.

Some Examples:
- INDEPENDENT-SET ≤ₚ CLIQUE
- VERTEX-COVER ≤ₚ SET-COVER
- VERTEX-COVER ≤ₚ INDEPENDENT-SET
P and NP Definitions

Definition (P)
Π ∈ P iff there exists a polynomial time algorithm A such that:

\[(X \text{ is a "yes" instance of } \Pi) \iff (A(X) = \text{"yes"})\]

Definition (NP)
Π ∈ NP iff there exists a polynomial time algorithm A such that:

\[(X \text{ is a "yes" instance of } \Pi) \implies (\exists Y : |Y| = \text{poly}(|X|), A(X, Y) = \text{"yes"})\]
\[(X \text{ is a "no" instance of } \Pi) \implies (\nexists Y : |Y| = \text{poly}(|X|), A(X, Y) = \text{"yes"})\]

We call Y a witness.
Example: Clique

- **Input**: Given graph $G = (V, E)$ and integer $k$.
- **Question**: Does $G$ contain a clique of size $k$?

**Lemma**

*Clique is in NP.*

**Proof.**

1. Suppose the witness $Y$ encodes a set of $k$ nodes in $V$ and $A(G, Y)$ checks if the induced graph on $Y$, $G[Y]$ is a clique.
2. $A$ is a polynomial time algorithm.
3. If there exists a clique of size $k$, there exists $Y$ of size $k$ such that $A(G, Y)$ outputs “yes”
4. If there doesn’t exist a clique of size $k$, there doesn’t exist $Y$ of size $k$ such that $A(G, Y)$ outputs “yes”

Example for a problem that is not known to be in NP: Is a quantified boolean formula, e.g., $\forall x \exists y \exists z , ((x \lor z) \land y)$, true?
NP-Completeness

Definition
A decision problem $\Pi$ is NP-Hard iff for all $\Pi' \in NP$, $\Pi' \leq_P \Pi$.

Definition
A decision problem $\Pi$ is NP-Complete iff it is both NP-Hard and in NP.

Remark 1: If $\Pi$ is NP-Complete and $\Pi \in P$ then $P = NP$

Remark 2: In 1971, Cook showed 3-SAT is NP-Complete. Because

\[
\text{CLIQUE} \in NP \text{ and } 3\text{-SAT} \leq_P \text{CLIQUE}
\]

we now know CLIQUE is NP-Complete.