CSCI3160 Design and Analysis of Algorithms (2025 Fall)

Approximation Algorithms 3: Set Cover and Hitting Set

Instructor: Xiao Liang[1]

Department of Computer Science and Engineering
The Chinese University of Hong Kong

[1] These slides are primarily based on materials prepared by Prof. Yufei Tao (please refer to Prof. Tao's version from 2024 Fall for the original content). Some modifications have been made to better align with this year's teaching progress, incorporating student feedback, in-class interactions, and my own teaching style and research perspective.

Set Cover

We are given a collection[2] $\mathcal{S}$ where each member of $\mathcal{S}$ comes from a certain domain (which is not important).

Define the **universe** $U = \bigcup_{S \in \mathcal{S}} S$.

A sub-collection $\mathcal{C} \subseteq \mathcal{S}$ is a **set cover** (of U) if every element of U appears in at least one set in $\mathcal{C}$.

The set cover problem:
Find a set cover with the smallest size.

[2] I.e., a set of sets.

Example: $U = \{1, 2, ..., 12\}$ and $\mathcal{S} = \{S_1, S_2, ..., S_6\}$ where

$$\begin{aligned} S_1 &= \{1, 2, 3\} \\ S_2 &= \{4, 5, 6\} \\ S_3 &= \{2, 3, 4, 5\} \\ S_4 &= \{7, 8, 9, 10\} \\ S_5 &= \{10, 11, 12\} \\ S_6 &= \{8, 9, 10\} \end{aligned}$$

An optimal solution is $\mathcal{C} = \{S_1, S_2, S_4, S_5\}$.

Application 1: Facility Location

Problem:

- Choose locations to place facilities (e.g., hospitals, warehouses)
- Each facility serves a subset of cities
- Goal: Cover all cities using the minimum number of facilities

Set Cover Mapping:

- Universe: All cities
- Subsets: Each facility covers a subset of cities
- Find minimum number of facilities to cover all cities

Application 2: Sensor Placement

Problem:

- Place sensors to monitor an area (e.g., a building, a field)
- Each sensor covers a region
- Goal: Use as few sensors as possible to cover the entire area

Set Cover Mapping:

- Universe: All regions that need monitoring
- Subsets: Each sensor covers a region
- Find the smallest set of sensors that covers all regions

Application 3: Test Case Minimization

Problem:

- Each test case detects a subset of possible bugs
- Goal: Run as few tests as possible to detect all bugs

Set Cover Mapping:

- Universe: All bugs
- Subsets: Each test case covers certain bugs
- Choose minimal test cases covering all bugs

Set Cover is NP-Hard

The input size of the set cover problem is $n = \sum_{S \in \mathcal{S}} |S|$.

The problem is NP-hard.

- No one has found an algorithm solving the problem in time polynomial in n.
- Such algorithms cannot exist if $\mathcal{P} \neq \mathcal{NP}$.

$\mathcal{A}$ = an algorithm that, given any legal input $\mathcal{S}$ with universe U, returns a set cover $\mathcal{C}$.

Denote by $OPT_{\mathcal{S}}$ the smallest size of all set covers when the input collection is $\mathcal{S}$.

$\mathcal{A}$ is a **ρ-approximate algorithm** for the set cover problem if, for any legal input $\mathcal{S}$, $\mathcal{A}$ can return a set cover with size at most $\rho \cdot OPT_{\mathcal{S}}$.

The value ρ is the **approximation ratio**.
We say that $\mathcal{A}$ achieves an approximation ratio of ρ.

We will show a greedy algorithm achieving $\rho = 1 + \ln(|U|)$.

Greedy Algorithm for Set Cover

Input: A collection $\mathcal{S}$

1. $\mathcal{C} = \emptyset$
2. **while** U still has elements not covered by any set in $\mathcal{C}$
3. $\quad F \leftarrow$ the set of elements in U not covered by any set in $\mathcal{C}$
   $\quad$/* for each set $S \in \mathcal{S}$, define its **benefit** to be $|S \cap F|$ */
4. $\quad$add to $\mathcal{C}$ a set in $\mathcal{S}$ with the largest benefit
5. **return** $\mathcal{C}$

It is easy to show:

- The $\mathcal{C}$ returned is a set cover;
- The algorithm runs in time polynomial to n.

We will prove later that the algorithm is $(1 + \ln |U|)$-approximate.

Example: $U = \{1, 2, ..., 12\}$.
$S_1 = \{1, 2, 3\}$, $S_2 = \{4, 5, 6\}$, $S_3 = \{2, 3, 4, 5\}$, $S_4 = \{7, 8, 9, 10\}$, $S_5 = \{10, 11, 12\}$, and $S_6 = \{8, 9, 10\}$.

- In the beginning, $\mathcal{C} = \emptyset$ and $F = \{1, 2, ..., 12\}$.
- Next, we can add S_3 or S_4 to $\mathcal{C}$ (benefit 4). The choice is arbitrary; suppose we add S_3. Now, $F = \{1, 6, 7, 8, 9, 10, 11, 12\}$.
- Next, we can add S_4 (benefit 4). Now, $F = \{1, 6, 11, 12\}$.
- Next, we can add S_5 (benefit 2). Now, $F = \{1, 6\}$.
- Next, we can add S_1 or S_2 (benefit 1). The choice is arbitrary; suppose we add S_1. Now, $F = \{6\}$.
- Finally, we add S_2. Now, $F = \emptyset$.

The algorithm terminates with $\mathcal{C} = \{S_1, S_2, S_3, S_4, S_5\}$.

Theorem 1: The algorithm returns a set cover with size at most $1 + (\ln |U|) \cdot OPT_{\mathcal{S}}$.

Note that this theorem implies the $(1 + \ln |U|)$-approximation ratio we claimed earlier, because

$$1 + (\ln |U|) \cdot OPT_{\mathcal{S}} \leq (1 + \ln |U|) \cdot OPT_{\mathcal{S}}.$$

$\mathcal{C}$ = the set cover returned.
$t = |\mathcal{C}|$.

Denote the sets in $\mathcal{C}$ as $S_1, S_2, ..., S_t$, picked in the order shown.

For each $i \in [1, t]$, define z_i as the size of F after S_i is picked.
/* Recall that F denotes the set of elements in U that are not covered yet */

Specially, define $z_0 = |U|$.

$z_t = 0$ and $z_{t-1} \geq 1$. **Think:** why?

Denote by $\mathcal{C}^*$ an optimal set cover, namely, $OPT_{\mathcal{S}} = |\mathcal{C}^*|$.

We will prove later:

Lemma 1: For $i \in [1, t]$, it holds that

$$z_i \le z_{i-1} \cdot \left(1 - \frac{1}{OPT_{\mathcal{S}}}\right).$$

From Lemma 1, we get:

$$\begin{aligned} z_{t-1} &\le z_{t-2}\cdot\left(1-\frac{1}{OPT_{\mathcal{S}}}\right) \le z_{t-3}\cdot\left(1-\frac{1}{OPT_{\mathcal{S}}}\right)^2 \le \ldots \le z_0\cdot\left(1-\frac{1}{OPT_{\mathcal{S}}}\right)^{t-1} \\ &= |U|\cdot\left(1-\frac{1}{OPT_{\mathcal{S}}}\right)^{t-1} \le |U|\cdot e^{-\frac{t-1}{OPT_{\mathcal{S}}}} \end{aligned}$$

where the last inequality used the fact $1+x \le e^x$ for any real value x.

Recall the Maclaurin expansion of e^x (where the expansion converges for all $x \in \mathbb{R}$)

$$e^x = \sum_{k=0}^{\infty}\frac{x^k}{k!} = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots,$$

As $z_{t-1} \ge 1$, we have

$$1 \le |U|\cdot e^{-\frac{t-1}{OPT_{\mathcal{S}}}} \tag{1}$$

which resolves to $t \le 1 + (\ln|U|)\cdot OPT_{\mathcal{S}}$. This proves Theorem 1.

Proof of Lemma 1

Before S_i is chosen, F has z_{i-1} elements.

At this moment, at least one set $S^* \in \mathcal{C}^*$ has a benefit at least

$$\frac{z_{i-1}}{|\mathcal{C}^*|} = \frac{z_{i-1}}{OPT_{\mathcal{S}}} > 0$$

Think: Why? - Averaging argument.

The set S^* cannot have been chosen (every chosen set has benefit 0) and is thus a candidate for S_i. It thus follows that S_i must have a benefit at least $\frac{z_{i-1}}{OPT_{\mathcal{S}}}$ (greedy). Therefore:

$$\begin{aligned} z_i &= |F \setminus S_i| = |F| - |F \cap S_i| \\ &\leq z_{i-1} - \frac{z_{i-1}}{OPT_{\mathcal{S}}} \\ &= z_{i-1}\left(1 - \frac{1}{OPT_{\mathcal{S}}}\right) \end{aligned}$$

$\square$

An Alternative Proof

The previous proof shows that the algorithm is $(1 + \ln |U|)$-approximate.

Next, by a different proof strategy, we will show that the **same algorithm** is also h-approximate, where $h = \max_{S \in \mathcal{S}} |S|$.

> **Theorem 1:** The algorithm returns a universe cover with cost at most $h \cdot OPT_{\mathcal{S}}$.

This bound would be tighter under certain cases where

$$\max_{S \in \mathcal{S}} |S| < 1 + \ln |U|.$$

For example, consider the case where $|U| = 1024$ and $\max_{S \in \mathcal{S}} |S| \leq 9$.

Proof of Theorem 1

Suppose that our algorithm picks t sets. Every time the algorithm picks a set, at least one **new** element is covered. For each $i \in [1, t]$, denote by e_i an arbitrary element that is **newly** covered when the i-th set is picked.

Let $\mathcal{C}^*$ be an optimal universe cover. Then, we have:

$$t = \sum_{i=1}^{t} 1 \le \sum_{i=1}^{t} \#\text{ sets in } \mathcal{C}^* \text{ containing } e_i \;\; \le \;\; \sum_{e \in U} \#\text{ sets in } \mathcal{C}^* \text{ containing } e,$$

where

- The first $\le$ symbol is because each e_i exists in at least one set of $\mathcal{C}^*$.
- The second $\le$ is because $|U| \ge t$ (or, there might be more than one element covered by the i-th picked set).

Proof of Theorem 1

Next, we claim that (see the following figure for a proof)

$$\sum_{e \in U} \# \text{ sets in } \mathcal{C}^* \text{ containing } e = \sum_{S \in \mathcal{C}^*} |S| \qquad (2)$$

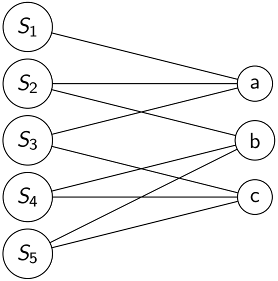


Figure: An example illustrating the claim

In the example shown left:

- there are 5 sets $\{S_1, \ldots, S_5\}$ in the optional over $\mathcal{C}^*$;
- there are three element $\{a, b, c\}$ in the universe U;
- it is then clear that both the left-hand side and the right-hand side of Equation (2) are counting the **number of edges** in the left figure.

Proof of Theorem 1

In summary, we have

$$t \leq \sum_{e \in U} \#\text{ sets in } \mathcal{C}^* \text{ containing } e = \sum_{S \in \mathcal{C}^*} |S| \leq |\mathcal{C}^*| \cdot h.$$

This finishes the proof of Theorem 1. $\square$

Our set cover algorithm can be used to solve many problems with approximation guarantees. Next, we will see two examples.

Example I: Vertex Cover

Recall the Vertex Cover problem: $G = (V, E)$ is an undirected graph. We want to find a small subset $V^* \subseteq V$ such that every edge of E is incident to at least one vertex in V^*. The optimization goal is to minimize $|V^*|$.

Vertex Cover can be reduced to Set Cover:

- For every $v \in V$, define $S_v =$ the set of edges incident on v.
- Apply our algorithm on the set-cover instance: $\mathcal{S} = \{S_v \mid v \in V\}$.

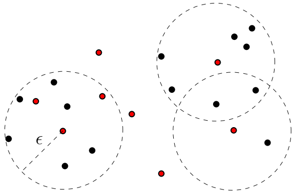
This gives an $\min\{O(\ln|V|), h\}$-approximate solution, where $h = \max_{v \in V} |S_v|$.

Remark: This algorithm is not as competitive as the 2-approximate vertex-cover algorithm we discussed in the lecture. But the point here is to demonstrate the usefulness of set cover, rather than improving the approximation ratio.

Example II: Facility Location

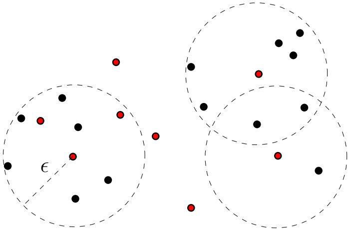
R = a set of n 2D red points, each called a **facility**
B = a set of n 2D black points, each called a **customer**
ϵ = a positive integer.

A subset $S \subseteq R$ is a **feasible facility set** if, for every black point $b \in B$, there is at least one point $r \in S$ with $dist(r, b) \leq \epsilon$.



OPT = the smallest size of all feasible facility sets.

Example II: Facility Location



Convert the problem to set cover:

- For every $r \in R$, define S_r = the set of black points b satisfying $dist(r, b) \leq \epsilon$.
- Apply our algorithm on the set-cover instance: $\mathcal{S} = \{S_r \mid r \in R\}$.

This gives an $O(\log n)$-approximate solution.

Next, we will introduce a closely related problem called the **hitting set problem**.

Hitting Set

Let U be a finite set called the **universe**.

We are given a collection $\mathcal{S}$ where each member of $\mathcal{S}$ is a set $S \subseteq U$.

A subset $H \subseteq U$ **hits** a set $S \in \mathcal{S}$ if $H \cap S \neq \emptyset$.
A subset $H \subseteq U$ is a **hitting set** (of $\mathcal{S}$) if it hits all the sets in $\mathcal{S}$.

The hitting set problem:
Find a hitting set H of the minimize size.

Example: $U = \{1,2,3,4,5,6\}$ and $\mathcal{S} = \{S_1, S_2, \ldots, S_{12}\}$ where

$$
\begin{aligned}
S_1 &= \{1\} \\
S_2 &= \{1,3\} \\
S_3 &= \{1,3\} \\
S_4 &= \{2,3\} \\
S_5 &= \{2,3\} \\
S_6 &= \{2\} \\
S_7 &= \{4\} \\
S_8 &= \{4,6\} \\
S_9 &= \{4,6\} \\
S_{10} &= \{4,5,6\} \\
S_{11} &= \{5\} \\
S_{12} &= \{5\}
\end{aligned}
$$

An optimal solution is $H = \{1,2,4,5\}$.

The input size of the set cover problem is $n = \sum_{S \in \mathcal{S}} |S|$.

The problem is NP-hard.

- No one has found an algorithm solving the problem in time polynomial in n.
- Such algorithms cannot exist if $\mathcal{P} \neq \mathcal{NP}$.

$\mathcal{A}$ = an algorithm that, given any legal input $\mathcal{S}$ with universe U, returns a hitting set.

Denote by $OPT_{\mathcal{S}}$ the smallest size of all hitting sets.

$\mathcal{A}$ is a ρ-**approximate algorithm** for the hitting set problem if, for any legal input $\mathcal{S}$, $\mathcal{A}$ can return a hitting set with size at most $\rho \cdot OPT_{\mathcal{S}}$.

The value ρ is the **approximation ratio**.

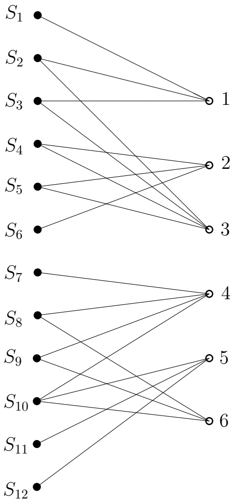
We say that $\mathcal{A}$ achieves an approximation ratio of ρ.

Hitting set and set cover are essentially the same problem.

Let $\mathcal{S}$ be the input to the hitting set problem (recall that $\mathcal{S}$ is a collection of sets). By converting the problem to an instance of set cover, we can obtain a polynomial-time hitting-set algorithm that guarantees an approximation ratio of

$$1 + \ln |\mathcal{S}|.$$

The proof is left as a regular exercise, but the next slide illustrates the key idea behind the conversion.



Consider the hitting set example on Slide 27. Let us create a bipartite graph G (shown left).

Each set $S \in \mathcal{S}$ corresponds to a vertex on the left of G.
Each element $e \in U$ corresponds to a vertex on the right of G.
An edge exists between vertex S and vertex e if and only if $e \in S$.

> Solving the hitting set problem is equivalent to finding a smallest set R of **right** vertices such that every left vertex is adjacent to at least one vertex in R.

This gives rise to the set cover example on Slide 3.