**601.435/635 Approximation Algorithms** **Lecturer:** Michael Dinitz
**Topic:** Set Cover and Max $k$-Cover **Date:** 1/30/24
**Scribe:** Michael Dinitz

Today we're going to talk about perhaps the most fundamental covering problem: Set Cover. We'll also talk about a very related problem known as Max $k$-Cover, or sometimes as Maximum Coverage. Our real goal, though, is going to be understanding the use of *greedy algorithms* in approximation algorithms. Greedy algorithms are used extensively in approximation algorithms, so we're just going to see some examples, but the type of analysis that we're going to do is very common, and Set Cover is most famous example of it. So it's a good first problem to start with.

## 3.1 Set Cover

**Definition 3.1.1** *The input in the Set Cover problem is a universe $U$, with $|U| = n$, and a family of sets $S_1, S_2, \ldots, S_m$ with $S_i \subseteq U$ for each $i$. Feasible solutions are index sets $I \subseteq [m]$ such that $\bigcup_{i \in I} S_i = U$, and the objective is to minimize $|I|$.*

There's an obvious greedy algorithm for Set Cover.

**Algorithm 1** A greedy algorithm for Set Cover

**Input**: Universe $U$ of $n$ elements, family $\{S_i\}_{i=1}^m$ of subsets of $U$.
**Output**: A minimum-size index set $I \subseteq [m]$ satisfying $\bigcup_{i \in I} S_i = U$.

```
I ← ∅, X ← U
while X ≠ ∅ do
    Let i be the index maximizing |X ∩ S_i|
    I ← i, X ← X \ S_i
end while
return I
```

**Theorem 3.1.2** *If OPT contains $k$ sets, the greedy algorithm uses at most $k(1 + \ln \frac{n}{k})$ sets.*

**Proof:** Let $I_t$ be the sets selected by the greedy algorithm in the first $t$ iterations. Let $n_t$ be the number of uncovered elements after iteration $t$. Then $n_t = n - |\bigcup_{i \in I_t} S_i|, n_0 = n, I_0 = \varnothing$.

To prove this we will first prove the following claim.

**Claim 3.1.3** $n_t \leq (1 - \frac{1}{k}) n_{t-1}$

**Proof:** Let $J_t = U \backslash (\bigcup_{i \in I_t} S_i)$, then OPT covers $J_{t-1}$ with $\leq k$ sets.

Because $|J_{t-1}| = n_{t-1}$, we know that OPT covers $n_{t-1}$ uncovered elements with $\leq k$ sets. Therefore there exists a set in OPT which covers at least $\frac{n_{t-1}}{k}$ uncovered elements.

Because the greedy algorithm always chooses the set which covers most uncovered elements, the greedy algorithm covers at least $\frac{n_{t-1}}{k}$ uncovered elements at iteration $t$.

Therefore $n_t \leq n_{t-1} - \frac{n_{t-1}}{k} = (1 - \frac{1}{k}) n_{t-1}$ ■

Now, by induction, $n_t \leq (1-\frac{1}{k})^t n$. Consider $t = k \ln \frac{n}{k}$,

$$n_t \leq \left(1 - \frac{1}{k}\right)^{k \ln \frac{n}{k}} n \leq e^{-\ln \frac{n}{k}} \cdot n \leq \frac{k}{n} \cdot n = k$$

Note that this uses the inequality $1 + x \leq e^x$ for all $x \in \mathbb{R}$, which is an incredibly useful inequality that we will use regularly.

The greedy algorithm covers the remaining $k$ elements using at most $k$ sets, so the greedy algorithm uses at most $k + k \ln \frac{n}{k} = k(1 + \ln \frac{n}{k})$ sets overall. ■

**Corollary 3.1.4** *The greedy algorithm is an $O(\log n)$-approximation for Set Cover*

**Proof:** By Theorem 3.1.2 we know that $ALG/OPT \leq 1 + \ln \frac{n}{OPT} \leq O(\log n)$. ■

**Corollary 3.1.5** *If $|S_i| \leq \alpha$ for all $i \in [m]$, then the greedy algorithm is an $O(\log \alpha)$ approximation.*

**Proof:** Clearly in this case we have that $k = OPT \geq n/\alpha$, since every set covers at most $\alpha$ elements and there are $n$ elements to cover in total. Thus Theorem 3.1.2 implies that $ALG/OPT \leq 1 + \ln \frac{n}{n/\alpha} = O(\log \alpha)$. ■

**Is the analysis tight?** Yes. Consider the following example.

$U = \{a_1, \ldots, a_n, b_1, \ldots, b_n, c_1, \ldots, c_n\}$, $S_1 = \{a_1, \ldots, a_n\}$, $S_2 = \{b_1, \ldots, b_n\}$, $S_3 = \{c_1, \ldots, c_n\}$, $S'_i = \{a_j, b_j, c_j \mid \frac{n}{2^i} < j \leq \frac{n}{2^{i-1}}\}$, $i = 1, \ldots, \log n + 1$.

The greedy algorithm will choose all the set $S'_i$, $i = 1, \ldots, \log n + 1$, since each one covers exactly half of the remaining elements. But the optimal solution is $S_1, S_2, S_3$.

**Is there any better algorithm?** No, due to a recent result of Dinur and Steurer [DS14]:

**Theorem 3.1.6** *Unless $P = NP$, there is no $C \cdot \ln n$-approx for set cover problem with constant $C < 1$.*

Feige [Fei98] showed a weaker result: Unless $NP \subseteq DTIME(n^{poly \log n})$, there is no $C \cdot \ln n$-approx for set cover problem with constant $C < 1$.

## 3.2 Weighted Set Cover

Consider the following generalization, in which we assign weights to sets and try to minimize the total weight of the sets chosen.

- Valid instances : Universe $U$, $|U| = n$. Family of sets $F = \{S_1, \ldots, S_m\}$, $S_i \subseteq U$ for all $i$. Each set $S_i$ has a cost $c_i$.
- Feasible solutions : A set $I \subseteq [m]$ such that $\bigcup_{i \in I} S_i = U$.
- Objective function : Minimize $\sum_{i \in I} c_i$.

In this context, the natural greedy algorithm is the following: In each iteration, pick a set which maximizes $\frac{\text{number of uncovered elements in the set}}{\text{cost of the set}}$ (this is called the *density* of the set), until all the elements are covered.

**Theorem 3.2.1** *The greedy algorithm is an $H_n = \Theta(\log n)$-approximation algorithm. Here $H_n = 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n}$.*

**Proof:** Let $I_t$ be the sets selected by the greedy algorithm up to $t$ iterations. Let $n_t$ be the number of uncovered elements at iteration $t$. Let $C^* = \sum_{i \in OPT} c_i$. Then $n_t = n - |\bigcup_{i \in I_t} S_i|, n_0 = n, I_0 = \varnothing$.

We first prove that in iteration $t$ there is a set of reasonably large density remaining.

**Claim 3.2.2** *In iteration $t$, there exists a set in OPT with $\frac{\text{cost of the set}}{\text{number of uncovered elements}} \leq \frac{c^*}{n_{t-1}}$.*

**Proof:** Let $J_t = U \setminus (\bigcup_{i \in I_t} S_i)$, then OPT covers $J_{t-1}$ with cost $\leq c^*$.

Suppose the claim is false. We have:

$$c^* = \sum_{i \in OPT} c_i = \sum_{i \in OPT} \frac{c_i}{|S_i \cap J_{t-1}|} |S_i \cap J_{t-1}| > \sum_{i \in OPT} \frac{c^*}{n_{t-1}} |S_i \cap J_{t-1}| \geq \frac{c^*}{n_{t-1}} |J_{t-1}| = c^*$$

This is a contradiction, and hence the claim is true. ■

Thus the greedy algorithm picks a set with density $= \frac{\text{number of uncovered elements}}{\text{cost of the set}} \geq \frac{n_{t-1}}{c^*}$ in iteration $t$, since some such set exists and the greedy algorithm chooses the set with highest density.

Now, assume the greedy algorithm picks $S'_1, \ldots, S'_k$, then $\frac{c(S'_t)}{|S'_t \cap J_{t-1}|} \leq \frac{c^*}{n_{t-1}}$. Let $x_t = |S'_t \cap J_{t-1}|$ be the number of elements that greedy covers in iteration $t$. Then $c(S'_t) \leq x_t \frac{c^*}{n_{t-1}}$

$$\begin{aligned}
c(Greedy) &= \sum_{i=1}^{k} c(S'_i) \leq \sum_{i=1}^{k} x_i \tfrac{c^*}{n_{i-1}} \\
&= x_1 \tfrac{c^*}{n} + x_2 \tfrac{c^*}{n - x_1} + x_3 \tfrac{c^*}{n - x_1 - x_2} + \ldots + x_k \tfrac{c^*}{n - x_1 - x_2 - \ldots - x_{k-1}} \\
&= c^*(\underbrace{\frac{1}{n} + \ldots + \frac{1}{n}}_{x_1} + \underbrace{\frac{1}{n - x_1} + \ldots + \frac{1}{n - x_1}}_{x_2} + \ldots \\
&\quad + \underbrace{\frac{1}{n - x_1 - \ldots - x_{k-1}} + \ldots + \frac{1}{n - x_1 - \ldots - x_{k-1}}}_{x_k}) \\
&\leq c^*(\tfrac{1}{n} + \tfrac{1}{n-1} + \ldots + 1) = c^* H_n
\end{aligned}$$

■

**Remark:** In some scenarios it is natural to consider problem instances where $n$ is small, but $m$ (the number of sets in the family) is extremely large (exponential in $n$). It is worth noting that in order to function, the greedy algorithm just needs to be able to pick the set of maximum density. Even when $m$ is exponential, sometimes it is reasonable to assume that we can do this, i.e. we can find the set of maximum density in polynomial time despite an exponential number of

sets. This is known as a "density oracle", and if one exists then the greedy algorithm still gives an $H_n$-approximation in polynomial time.

In fact, it suffices to simply be able to approximate the density:

**Theorem 3.2.3** *If there exists an $\alpha$-approximation for the max density problem, then there exists an $\alpha H_n$-approximation for the original problem.*

## 3.3 Small Frequencies

We can see the vertex cover problem as a special set cover problem: the universe $U$ is the edge set $E$, and the family of sets is $\{S_u \mid u \in V\}$ where $S_u = \{\{u, v\} \mid \{u, v\} \in E\}$. But this view naturally leads to the following question: why does vertex cover have a 2-approximation, when the best possible for set cover is $\ln n$?

**Definition 3.3.1** *The frequency of $e \in U$ is $f_e = |\{i \in [m] \mid e \in S_i\}|$, the number of sets in $F$ that contain $e$.*

**Theorem 3.3.2** *If $f_e \leq f$ for all $e \in U$, then there is an $f$-approximation algorithm for set cover problem.*

**Algorithm:** In each iteration, arbitrarily choose an uncovered element and select all the sets that contain this element. Repeat until all elements covered.

This is an $f$-approximation for the same reason that our algorithm for vertex cover was a 2-approximation. Informally, for every two elements $e, e' \in U$ considered by this algorithm, there are no sets which cover both $e$ and $e'$ (or else whichever was covered first would have caused this set to be included, so the algorithm would not consider the second element). Then $OPT$ has to be at least as large as the number of iterations of this algorithm. On the other hand, in each iteration this algorithm only picks $f$ sets. Hence it includes at most $f \cdot OPT$ sets.

## References


[DS14] Irit Dinur and David Steurer. Analytical approach to parallel repetition. In *Proceedings of the Forty-sixth Annual ACM Symposium on Theory of Computing*, STOC '14, pages 624–633, 2014.

[Fei98] Uriel Feige. A threshold of ln $n$ for approximating set cover. *J. ACM*, 45(4):634–652, 1998.