CS-E400204: Approximation Algorithms **Spring (Fifth Period) 2022**

Lecture 3 (April 27): Set Cover and Layering

Lecturer: Kamyar Khodamoradi *Scribe: Kamyar Khodamoradi*

3.1 Set Cover Problem

- Input:
  - A set U of items
  - A collection $\mathcal{S}$ of subsets of U
  - (In the general case:) A cost function $c : \mathcal{S} \to \mathbb{Q}_{\geq 0}$
- Output:
  - The minimum cardinality (cost) cover $\mathcal{S}' \subseteq \mathcal{S}$: every element is covered or $\bigcup \mathcal{S}' = U$.

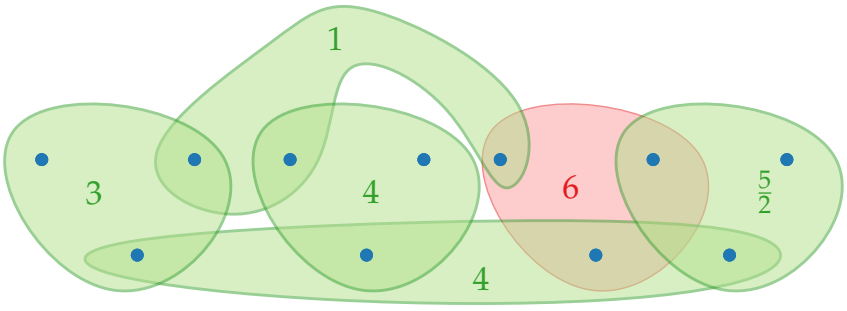

Figure 3.1: A Set Cover instance and solution

We can think of an iterative "purchasing" of the elements. We can evenly distribute the cost of a set to the (new) elements that it covers:

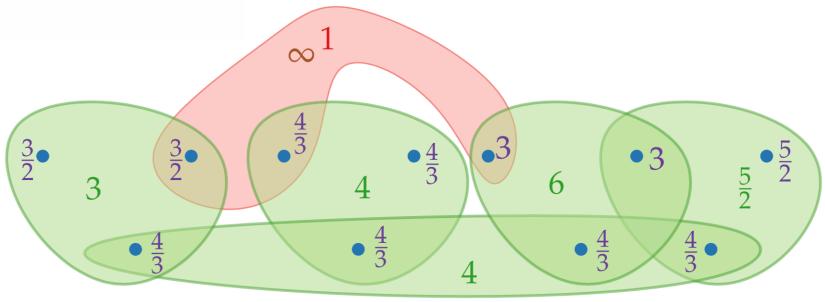

Figure 3.2: A Set Cover instance and iterative purchasing

What would be the greedy approach to solve this question?

Buy the set with the best value for money!

Algorithm 1 A greedy approximation for the Set Cover problem

Input: A set U of items, collection $\mathcal{S}$ of subsets of U, and costs $c(S) \geq 0, \forall S \in \mathcal{S}$.
Output: A collection $\mathcal{S}' \subseteq \mathcal{S}$ that covers U

```
S' ← ∅
C ← ∅
while C ≠ U do
    S ← a set from S which minimizes c(S)/|S−Y|
    for all u ∈ S\C do
        price(u) ← c(S)/|S\C|
    end for
    S' ← S' ∪ {S}
    C ← C ∪ S
end while
return S'
```

Theorem 1 *Algorithm 1 is an H_k-approximation of the* Set Cover *problem, where:*

- $k = \max_{S \in \mathcal{S}} \{|S|\}$
- $H_k = \sum_{i=1}^{k} \frac{1}{i} = \ln(k) + \mathcal{O}(1) \, (\approx \ln k + 0.577)$

Proof. First, we seek an upper bound on the price of every element at the moment the greedy algorithm purchases them. When Algorithm 1 adds a set that cover multiple new items, we fix an arbitrary order on how it purchases items in the set.

Lemma 1 *Let $S \in \mathcal{S}$ and assume Algorithm 1 buys the items in S in the following order: $u_1, u_2, \ldots, u_\ell$. Then, $price(u_j) \leq \frac{c(S)}{\ell - j + 1}$.*

Proof. Note that $j-1$ elements of S are already bought, so at this time, at least $\ell - j + 1$ elements are still not bought (covered). So, the uniformly-split cost of S over the remaining elements is at most $\frac{c(S)}{\ell-j+1}$ (Can it be lower even?). The algorithm either picks S to cover u_j or something even better (with better value for money) since it's greedy. So, the price that it sets for u_j is at most $\frac{c(S)}{\ell-j+1}$. ■

Note that this lemma upper bounds the cost that Algorithm 1 assigns to an item with respect to any set S that contains it. Next, we can use this upper bound to also upper bound the cost of every set in out Algorithm 1:

Corollary 1 $price(S) \triangleq \sum_{j=1}^{\ell} u_j \leq c(S) \cdot H_\ell$.

The proof is immediate from the previous lemma. Now that we have an upper bound on the cost of our proposed approximate solution, we use the old trick of assuming to know OPT and analyzing its cost. Let $\{S_1, S_2, \ldots, S_m\}$ be the optimum set cover. First, note that $\text{OPT} = \sum_{i=1}^{m} c(S_i)$. Also note that both OPT and the cover produced by Algorithm 1 cover every element of U. We can write:

$$\begin{aligned} price(U) &= \sum_{u \in U} price(u) \\ &\leq \sum_{i=1}^{m} price(S_i) \\ &\leq \sum_{i=1}^{m} c(S_i) \cdot H_k \\ &= H_k \cdot \text{OPT}. \end{aligned}$$

Here, the last inequality uses Corollary 1. Note that this corollary is about the price that the greedy algorithm assigns to any set, and that $price(U)$ is exactly the cost of the returned approximate solution. ■

3.1.1 Final remarks about Set Cover

First, the question of tightness of the algorithm. Is the greedy algorithm tight? It turns out that the answer is yes.

Example 1 *Imagine the instance given in Figure 3.3. This example shows an instance on which the greedy algorithm returns a solution with factor $\frac{H_n}{1+\varepsilon} = \ln n + \mathcal{O}(1)$.*

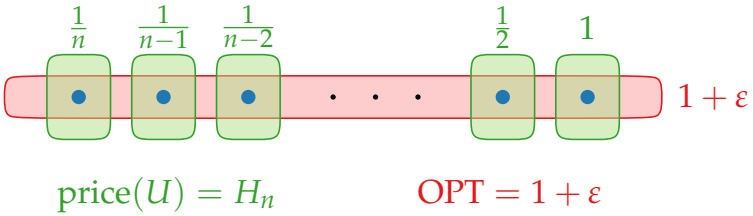

Figure 3.3: The tight instance for the greedy Set Cover algorithm

Regarding the tightness, we have the following theorem:

Theorem 2 *Unless* P = NP*, there is no* $c \cdot \log n$*-approximation for* Set Cover *for any constant* $c < 1$*.*

Theorem 1 was first proven in [C79] and Theorem 2 in [DS14], though the same hardness lower bound under the stronger assumption that NP $\not\subseteq$ DTIME$(2^{O(\log \log n)})$ was earlier proven in [F98].

3.2 Layering

This technique can help us deal with some weighted cases of the problems. In this section, we give a 2-approximation algorithm for weighted Vertex Cover via a greedy approach and layering. Remember that our greedy approximation from the first lecture only worked for the cardinality case of Vertex Cover.

The main idea here is to decompose a given weight function into a family of nested weight types that we know how to handle. For the case of Vertex Cover, we peal off some portions of the weight function on the vertices to form *degree-weighted* function on a nested sequence of subgraphs of the given instance G.

Reminder. In an instance of the Vertex Cover problem, we are given a graph $G = (V, E)$ and a weight function on the vertices $c : V \to \mathbb{Q}_{\geq 0}$. We wish to calculate the minimum cost vertex cover of the graph.

Definition 1 *We say a weight function* c *is* degree-weighted *if there exists a constant* $\delta \geq 0$ *such that* $\forall v \in V$: $c(v) = \delta \cdot deg(v)$*.*

Now, we can show the following lemma:

Lemma 2 *Let $c : V \to \mathbb{Q}_{\geq 0}$ be a degree-weighted function. Then, $c(V) \leq 2 \cdot \mathrm{OPT}$.*

Proof. Let δ be the constant in Definition 1, and let C^* be the optimal vertex cover. Observe that $\sum_{v \in C^*} deg(v) \geq |E|$ since C^* must cover all the edges. Therefore, $\mathrm{OPT} = c(C^*) \geq \delta \cdot |E|$. On the other hand, note that if we take the entirety of V as a vertex cover, $c(V) = \sum_{v \in v} c(v) = \sum_{v \in V} \delta \cdot deg(v) = 2\delta \cdot |E|$. This implies the claim of the lemma. ■

How do we *peal-off* some weights out of an arbitrary vertex weight function?

Algorithm 2 A layering function for VERTEX COVER

Input: A graph $G = (V, E)$, an arbitrary weight function $c : V \to \mathbb{Q}_{\geq 0}$.
Output: Sets $W_0, \ldots, W_{k-1}$,$D_0, \ldots, D_k$, and $\delta_0, \ldots, \delta_{k-1}$.

```
i ← 0
G_0 ← G
while V(G_i) ≠ ∅ do
    D_i ← all vertices with degree 0 in G_i
    δ_i ← min_{v ∈ V(G_i \ D_i)} {c(v)/deg_{G_i}(v)}
    for all v ∈ V(G_i \ D_i) do
        t_i(v) ≜ δ_i · deg_{G_i}(v)
        c(v) ← c(v) − t_i(v)
    end for
    W_i ← all vertices whose new weight is 0
    G_{i+1} ← G_i \ (W_i ∪ D_i)
    i ← i + 1
end while
k ← i − 1
return W_0, W_1, ..., W_{k−1}, D_0, D_1, ..., D_k, δ_0, δ_1, ..., δ_{k−1}
```

Theorem 3 *The set $C \triangleq W_0 \cup W_1 \cup \ldots \cup W_{k-1}$ is a 2-approximation for* VERTEX COVER *with arbitrary weights.*

Proof. We need to show:

1. C is a valid vertex cover of the graph
2. $c(C) \leq 2 \cdot \mathrm{OPT}$

To see 1, assume C is not a feasible vertex cover for G. So, there must exist an edge $e = (u, v)$ with $u \in D_i$ and $v \in D_j$. W.l.o.g. assume $i \leq j$. This means that when Algorithm 2 was at iteration i, the edge e still existed, which means that $deg_{G_i}(u) \geq 0$, which is a contradiction since u is a degree zero vertex in G_i.

To see 2, we first make the following observation:

Observation 1 *For the weight of a vertex v, we have*

$$\bullet \quad v \in W_j \Rightarrow c(v) = \sum_{i=0}^{j} t_i(v) \tag{3.1}$$

$$\bullet \quad v \in D_j \Rightarrow c(v) \geq \sum_{i=0}^{j-1} t_i(v) \tag{3.2}$$

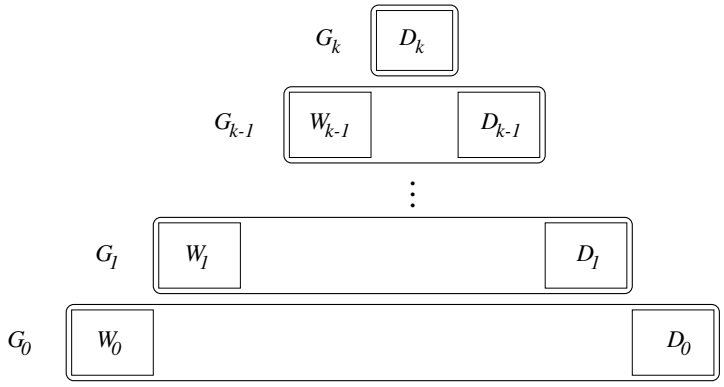

Figure 3.4: Layering for a graph G

Assume C^* is a optimal vertex cover.

Claim 1 *$C^* \cap G_i \backslash D_i$ is the optimal vertex cover for G_i.*

Proof.

- Why is it a vertex cover? (Note that G_i is a vertex-induced subgraph of G).
- Why is it optimal? (Assume it's not and the optimal for G_i is $\hat{C}$. Replace $C^* \cap G_i$ with $\hat{C}$).

■

Claim 2 *For any $i \in [k-1]$, $t_i(C \cap G_i) \leq 2 \cdot t_i(C^* \cap G_i)$.*

Proof. Note that $C \cap G_i$ is W_i. We now invoke Lemma 2. Assume we had an instance induced on G_i when all the weights were reduced to t_i. Then, by the lemma and the observation that $C^* \cap G_i$ is the optimal vertex cover for G_i, $t_i(C \cap G_i) \leq t_i(G_i) \leq 2 \cdot t_i(C^* \cap G_i)$. ■

Finally, we bound the total cost of the solution C. First, note that any vertex that is not in C must belong to $D = \cup_{i=0}^{k} D_i$. Then,

$$c(C) = \sum_{i=0}^{k-1} t_i(C \cap G_i) \leq 2 \cdot \sum_{i=0}^{k-1} t_i(C^* \cap G_i \backslash D_i).$$

Next, we break the right-hand side of the above inequality as follows: Let W and D denote $W_0 \cup W_1 \cup \ldots \cup W_{k-1}$ and $D_1 \cup D_2 \cup \ldots \cup D_k$ respectively. For a vertex $v \in C^* \cap W$, let $W(v)$ denote the level j such that W_j contains v. We define $D(v)$ for $v \in c^* \cap D$ similarly. We note that:

$$\begin{aligned}
c(C) &\leq 2 \cdot \textstyle\sum_{i=0}^{k-1} t_i(C^* \cap G_i \backslash D_i) \\
&= 2 \left(\textstyle\sum_{v \in C^* \cap W} \sum_{i=0}^{W(v)} t_i(v) + \sum_{v \in C^* \cap D} \sum_{i=0}^{D(v)-1} t_i(v) \right) && \forall v \in C^* \cap D_i, \text{ last level not counted in } t_i(C^* \cap G_i \backslash D_i) \\
&= 2 \left(\textstyle\sum_{v \in C^* W} c(v) + \sum_{v \in C^* \cap D} \sum_{i=0}^{D(v)-1} t_i(v) \right) && \text{By Equality (3.1)} \\
&\leq 2 \left(\textstyle\sum_{v \in C^* W} c(v) + \sum_{v \in C^* \cap D} c(v) \right) && \text{By Inequality (3.2)} \\
&= 2\text{OPT}
\end{aligned}$$

■

References

[1] I. DINUR AND D. STEURER, An analytic approach to parallel repetition, *in proceedings of ACM Symposium on the Theory of Computing*, 2014.

[2] U. FEIGE, A threshold of $\ln n$ for set cover, *Journal of the ACM*, 45:634–652, 1998.