CS 6200: Algorithmics II **Fall 2020**
Instructor: Avah Banerjee, S&T.
Lectures: 26 **Dates: 10/26**

1 Set-Cover problem

Theorem 1.1 *Suppose there exist $\mathcal{X} = \{x_1, x_2, ..., x_n\}$, and $\mathcal{F} = \{S_1, S_2, ..., S_m\}$, an instance $(\mathcal{X}, \mathcal{F})$of the set-covering problem consists of a finite set $\mathcal{X}$ and a family $\mathcal{F}$ of subsets of $\mathcal{X}$, such that every element of $\mathcal{X}$ belongs to at least one subset in $\mathcal{F}$:*

$$\mathcal{X} = \bigcup_{\mathcal{S} \in \mathcal{F}} \mathcal{S}$$

We say that a subset $\mathcal{S} \in \mathcal{F}$ covers its elements. The problem is to find a minimum-size subset $\mathcal{H} \in \mathcal{F}$ whose members cover all of $\mathcal{X}$:

$$\mathcal{X} = \bigcup_{\mathcal{S} \in \mathcal{H}} \mathcal{S}$$

There is an equivalent problem is named as Hitting-set problem. It is to find a subset $\mathcal{H} \subseteq \mathcal{X}$ of minimum size such that $\forall \mathcal{S}_i \in \mathcal{F},\ \mathcal{S} \bigcap \mathcal{X} \neq \emptyset$

Consider the bipartite graph $\mathcal{G}(\mathcal{X}, \mathcal{F})$, there is an edge $(x, \mathcal{S})$ if $x \in \mathcal{S}$

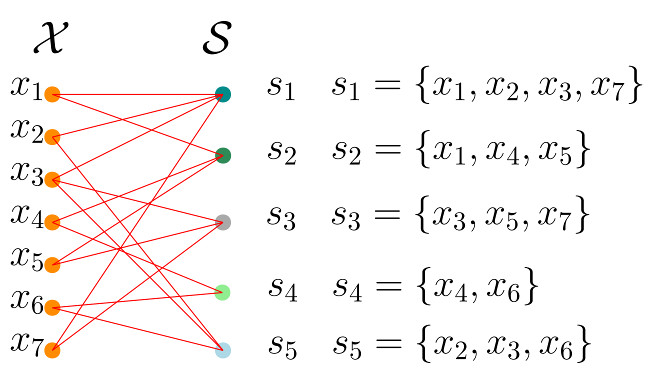

Fig. 1: The graph in theorem 1.1.

In Fig. 1, the set-cover problem is to find the a subset of the right subsets $\mathcal{S}_i$ such that every element is part of at least one subset. The hitting-set problem is to find a subset of the left vertices x_i such that all the sets are covered.

A greedy approximation algorithm for set-cover problem

As shown in Algorithm 1, The greedy method works by picking, at each stage, the set $\mathcal{S}$ that covers the greatest number of remaining elements that are uncovered.

The algorithm works as follows. The set $\mathcal{U}$ contains, at each stage, the set of remaining uncovered elements. The set $\mathcal{C}$ contains the cover being constructed. Line 4 is the greedy decision-making step,

Algorithm 1 Greedy-Set-Cover $(\mathcal{X}, \mathcal{F})$

1: $\mathcal{U} = \mathcal{X}$ (uncovered elements)
2: $\mathcal{C} = \mathcal{X}$ (selected sets)
3: **while** $\mathcal{U} \neq \emptyset$ **do**
4: select an $\mathcal{S} \in \mathcal{F}$ that maximizes $|\mathcal{S} \bigcap \mathcal{U}|$
5: $\mathcal{U} = \mathcal{U} - \mathcal{S}$
6: $\mathcal{C} = \mathcal{C} \bigcup \mathcal{S}$
7: **end while**
8: **return** $\mathcal{C}$

choosing a subset $\mathcal{S}$ that covers as many uncovered elements as possible (breaking ties arbitrarily). After $\mathcal{S}$ is selected, line 5 removes its elements from $\mathcal{U}$, and line 6 places $\mathcal{S}$ into $\mathcal{C}$. When the algorithm terminates, the set $\mathcal{C}$ contains a subfamily of $\mathcal{F}$ that covers $\mathcal{X}$.

Theorem 1.2 *GREEDY-SET-COVER is a polynomial-time $\rho(n)$-approximation algorithm, where $\rho(n) = H(max|\mathcal{S}| : \mathcal{S} \in \mathcal{F})$.*

Proof: Let $\mathcal{S}_i$ denote the i th subset selected by GREEDY-SET-COVER; the algorithm incurs a cost of 1 when it adds $\mathcal{S}_i$ to $\mathcal{C}$, where the $\mathcal{C}$ is the total cost. We spread this cost of selecting $\mathcal{S}_i$ evenly among the elements covered for the first time by $\mathcal{S}_i$. Let c_x denote the cost allocated to element x, for each $x \in \mathcal{X}$. Each element is assigned a cost only once, when it is covered for the first time. If x is covered for the first time by $\mathcal{S}_i$, then:

$$c_x = \frac{1}{|\mathcal{S}_i - (\mathcal{S}_1 \cup \mathcal{S}_2 \cup \ \cdots \ \cup \mathcal{S}_{i-1})|}$$

Each step of the algorithm assigns 1 unit of cost, and so:

$$|\mathcal{C}| = \sum_{x \in \mathcal{X}} c_x$$

Each element $x \in \mathcal{X}$ is in at least one set in the optimal cover $\mathcal{C}^*$, and so we have:

$$\sum_{\mathcal{S} \in \mathcal{C}^*} \sum_{x \in \mathcal{S}} c_x \geq \sum_{x \in \mathcal{X}} c_x$$

Combining equations above, we have that:

$$|\mathcal{C}| \leq \sum_{\mathcal{S} \in \mathcal{C}^*} \sum_{x \in \mathcal{S}} c_x$$

Then we need to proof that for any set $\mathcal{S}$ belonging to the family $\mathcal{F}$, $\sum_{x \in \mathcal{S}} c_x \leq H(|\mathcal{S}|)$, which is in the lemma 1.3 proof 1

Then we have:

$$|\mathcal{C}| \leq \sum_{\mathcal{S} \in \mathcal{C}^*} H(|\mathcal{S}|) \leq |\mathcal{C}^*| \cdot H(max\{|\mathcal{S}| : \mathcal{S} \in \mathcal{F}\})$$

thus proving the theorem that GREEDY-SET-COVER is a polynomial-time $\rho(n)$-approximation algorithm . ■

Lemma 1.3 *For any set $\mathcal{S}$ belonging to the family $\mathcal{F}$, $\sum_{x\in\mathcal{S}} c_x \leq H(|\mathcal{S}|)$.*

Proof: Consider any set $\mathcal{S} \in \mathcal{F}$ and any $i = 1, 2, ..., |\mathcal{C}|$, and let $\rho_i = |\mathcal{S}_i - (\mathcal{S}_1 \cup \mathcal{S}_2 \cup \cdots \cup \mathcal{S}_{i-1})|$ be the number of elements in $\mathcal{S}$ that remain uncovered after the algorithm has selected sets $\mathcal{S}_1, \mathcal{S}_2, ..., \mathcal{S}_i$. Define $\rho_0 = \mathcal{S}$. Suppose $\mathcal{S}_i$ is the set picked by the GREEDY-SET-COVER and $\mathcal{S}$ is any set. Due to the greedy-choice property, we have:

$$|\mathcal{S}_i - (\mathcal{S}_1, \mathcal{S}_2, ..., \mathcal{S}_i)| \leq |\mathcal{S} - (\mathcal{S}_1, \mathcal{S}_2, ..., \mathcal{S}_i)| = \rho_{i-1}$$

because the greedy choice of $\mathcal{S}_i$ guarantees that $\mathcal{S}$cannot cover more new elements than $\mathcal{S}_i$ does (otherwise, the algorithm would have chosen $\mathcal{S}$ instead of $\mathcal{S}_i$).

Also, Since $\mathcal{S}_i$ is covered monotonically, we have $\rho_{i-1} \geq \rho_i$, we have:

$$\sum_{x\in\mathcal{S}} c_x \leq \sum_{i=1}^{k} (\rho_{i-1} - \rho_i) \cdot \frac{1}{\rho_{i-1}} = \sum_{i=1}^{k} \sum_{j=\rho_{i+1}}^{\rho_{i-1}} \frac{1}{\rho_{i-1}} \leq \sum_{i=1}^{k} \sum_{j=\rho_{i+1}}^{\rho_{i-1}} \frac{1}{j} \qquad (\textit{because } j \leq \ \rho_{i-1})$$

$$= \sum_{i=1}^{k} (\sum_{j=1}^{\rho_{i-1}} \frac{1}{j} - \sum_{j=1}^{\rho_i} \frac{1}{j}) = \sum_{i=1}^{k} (H(\rho_{i-1}) - H(\rho_i)) = H(\rho_0) - H(\rho_k) \qquad (\textit{because the sum telescopes})$$

$$= H(\rho_0) - H(0) = H(\rho_0) \qquad (\textit{because } H(0) = 0)$$

$$= \sum_{j=1}^{\rho_0} \frac{1}{j} = H(|\mathcal{S}|)$$

Therefore, there exists $\mathcal{S}$, $\sum_{x\in\mathcal{S}} c_x \leq H(|\mathcal{S}|)$ ■

Lemma 1.4 *Lower Bound: For every $1 \geq \alpha > 0$, there are no $((1-\alpha)\ln n)$-approximation schemes unless $P = NP$.*

Proof: Following is the proof of : fully polynomial-time approximation scheme (FPTAS) $\Rightarrow$ pseudo-poly exact algorithms.

Suppose the *ALG* (algorithm) be an FPTAS for some minimization problem P, and which is integral valued. For all instances of P of size n, let W be largest value of any solutions. For any numeric value in any instance. Let $\varepsilon = \frac{1}{W}$. Cost $\mathcal{C}$ of the solutions returned by the ALG, we have:

$$(1+\varepsilon) \cdot C^* \ < C^* \ + \varepsilon \cdot C^* < C^* \ + \varepsilon \cdot W \ = \ C^* \ + 1$$

Then, $C = C^*$. Further, the *ALG* runs in a polynomial time, i.e. $O(poly(n, W))$ ■