Network Design Foundation
Fall 2011
Lecture 3

Instructor: Mohammad T. Hajiaghayi
Scribe: Avinash Das

September 14, 2011

1 Summary

In the notes, we will explore three problems: a) set cover, b) unique-coverage and c) maximum coverage problems.

2 Maximum coverage

In the previous lectures, we defined maximum coverage problem. Let us briefly summarize the definition and greedy algorithm to solve the problem.
Definition
Input: Given a universe set $U = \{U_1, \cdots, U_n\}$, a collection $\mathcal{S} = \{S_1 \dots S_k\}$ of subsets of U, a cost function $c : \mathcal{S} \to \mathbb{Q}^+$ and a weight function $w : \mathcal{U} \to \mathbb{Q}^+$ and a budget $L \in \mathbb{Q}^+$.

Goal: Find a $\mathcal{S}' \subseteq \mathcal{S}$, so that the total weight of $\mathcal{S}$ i.e, $w(\mathcal{S}')$ is maximized and cost of $\mathcal{S}$ is less than the budget L ($c(\mathcal{S}') < L$).

In the unit cost version of the problem, cost of all set is one. The maximum-coverage problem, even the unit cost version, is NP-hard. The problem can be trivially reduced to the set-cover problem: Try minimum L that can cover everything.

Now, we give a greedy algorithm to find an approximate solution of the maximum-coverage problem.

Theorem 1 *There is a* $\left(1 - \frac{1}{e}\right)$ *approximation greedy algorithm for the maximum-coverage problem.*

Unit cost version: At each step pick a set maximizing the weight of the uncovered elements. The output will be a collection of sets that will cover all the

elements of U.
Proof: Let opt denote the optimal solution for the unit cost maximum-coverage problem. Let $S_1 \cdots S_L$ be the sets that we pick respectively in iteration $1 \ldots L$ where L is the budget. Let $G_i = \bigcup_{n=1}^{i} S_n$ and let $w(F)$ denote the total weight of the elements in F.

Lemma 1

$$w(G_i) - w(G_{i-1}) \geq \frac{1}{l}\left(w(\mathsf{opt}) - w(G_{i-1})\right) \qquad (1)$$
$$\textit{for } i \geq 1$$

Proof:

Let w'_j be the total weight of the elements in S_j not covered in G_{i-1}. We note that for set S_i we have $w'_i = w(G_i) - w(G_{i-1})$. In i^{th} iteration we are choosing S_i, such that w_i is maximum. Thus for any set S_j in the optimum solution, we have $w'_j \leq w'_i$. Therefore since opt has at most L sets, we have: $w(\mathsf{opt}) - w(G_{i-1}) \leq L\, w'_i$. ■

Lemma 2 $w(G_i) \geq (1 - (1 - \frac{1}{L})^i) w(\mathsf{opt})$

Proof: We prove the lemma using induction. The case for $i = 1$ follows directly from lemma 1: $w(G_1) \geq w(\mathsf{opt})/L$.

Suppose the inequality holds for $1, \cdots, i-1$. We have

$$\begin{aligned}
W(G_i) &= w(G_{i-1}) + [w(G_i) - w(G_{i-1})] \\
&\geq w(G_{i-1}) + \frac{1}{L}[w(\mathsf{opt}) - w(G_{i-1})] \quad \textit{by lemma 1} \\
&= (1 - \frac{1}{L})\left(1 - (1 - \frac{1}{L})^{i-1}\right) w(\mathsf{opt}) + \frac{1}{L} w(\mathsf{opt}) \quad \textit{by induction hypothesis} \\
&= (1 - \frac{1}{L}) w(\mathsf{opt}) - (1 - \frac{1}{L})^i w(\mathsf{opt}) + \frac{1}{L} w(\mathsf{opt}) \\
&= (1 - (1 - \frac{1}{L})^i) w(\mathsf{opt})
\end{aligned}$$

■

Using the above 2 lemmas, we can easily prove the theorem.

$$\begin{aligned}
w(G_i) &\geq (1 - (1 - \frac{1}{L})^L) w(\mathsf{opt}) \\
&\geq (1 - \frac{1}{e}) w(\mathsf{opt}) \qquad (2)
\end{aligned}$$

■

Lets us apply the greedy algorithm of adding the most cost effective set, to the maximum coverage problem when sets have different costs. Suppose $U = \{X_1, X_2\}$ and $S_1 = \{X_1\}, S_2 = \{X_2\}, c(S_1) = 1, c(S_2) = p, w(X_1) = 1, w(X_2) = p + 1$and budget p. So, $\mathsf{opt} = \{S_2\}$. The greedy algorithm will choose S_1 and it will get stuck. Therefore, greedy algorithm does not work when each set has a different cost.

The greedy algorithm can be improved to give a $\left(1 - \frac{1}{e}\right)$ approximation algorithm for the maximum-coverage problem with sets of unequal costs. We simply fix a constant k and we find the best subset of size k of S at the beginning. The output of the algorithm would be either this collection or the output of the greedy algorithm, whichever has the higher total weight.

In the next section, we will show that maximum coverage problem even with unit cost cannot be approximated to $\left(1 - \frac{1}{e} - \varepsilon\right)$ unless, $\mathsf{NP} \subseteq \mathsf{DTIME}(n^{\log\log n})$.

Theorem 2 (Hardness theorem) *The maximum coverage with unit cost cannot be approximated with a factor better than* $(1 - \frac{1}{e})$ *unless,*

$$\mathsf{NP} \subseteq \mathsf{DTIME}(n^{\log(\log n)}) \tag{3}$$

Proof:

To prove this theorem we will use Feige theorem that we covered in last lecture.

Theorem 3 (Feirge) *If a set cover problem is approximable within a factor of* $1 - \varepsilon \ln(n)$*, for* $\varepsilon > 0$*. Then,*

$$\mathsf{NP} \subseteq \mathsf{DTIME}(n^{\log\log n}) \tag{4}$$

Consider a unit cost set-cover. Weight of each element is one.
Say there is an algorithm A with an approximation factor $\alpha > 1 - 1/e$.
We guess the number of the sets in the optimal solution of set-cover. Let it be k. Since the number of sets in the optimal solution is k, it is possible to cover all the elements using at most k sets.

New algorithm for the set-cover:

- Run A with limit $L = k$. It will cover at least $\alpha\ n$ elements.
- Choose sets for this cover C of αn elements.
- Remove C and elements which are covered.
- Iterate by running A in the reduced set.

Analysis: Let n_i denote the number of the uncovered elements at the start of the i^{th} iteration. Since, A covers at least αn_i at iteration i^{th} we have $n_{i+1} \le n_i(1 - \alpha)$.

Suppose we iterate $L + 1$ times for $n_L \ge 1$
$1 \le n_L \le n_{L-1}(1 - \alpha) \le n_{L-2}(1 - \alpha)^2 \le \cdots \le n(1 - \alpha)^L$

Therefore,

$$\begin{aligned} L &\leq \frac{\ln n}{\ln\left(\frac{1}{1-\alpha}\right)} \\ kL &\leq k\frac{\ln n}{\ln\left(\frac{1}{1-\alpha}\right)} \\ &= opt\frac{\ln n}{\ln\left(\frac{1}{1-\alpha}\right)} \end{aligned}$$

If $\alpha > 1 - \frac{1}{e}$ then, $\frac{1}{1-\alpha} > e$ therefore, $\ln\frac{1}{1-\alpha} > 1$. This is contradiction to the Feirge theorem. ■

Till now we have covered mostly the minimization problems. Maximization problems are usually harder. This is because of many reasons, including there are few standard functions in which a maximization can reduces to.

3 Unique-Coverage

Definition:
Given a universe U of n elements and given a collection S of subsets of U, we want to find a sub-collection $S' \subseteq S$ which maximizes the number of elements uniquely covered, i.e., appeared in exactly one set in S'.

One can apply a normal greedy algorithm to add a set that maximizes number of uniquely covered elements. But it can get stuck at some step. Therefore normal greedy algorithm does not works for this problem.

3.1 Budgeted unique-coverage

Definition:
Given profits for the elements and costs for the subsets, given also a budget B, find a sub-collection $S' \subseteq S$, such that total cost is at most B and that maximizes the total cost of uniquely covered elements.

Budgeted unique-coverage is a special case of the coverage problem, motivated by an application at Bell-labs of low coverage problem described described below.

Low coverage problem: Find a subset of base stations and options within the total budget that maximizes the total satisfaction weighted by client densities, where satisfaction S_k for covering a region by k base stations, is given in the form $S_0 = 0, S_1 \geq S_2 \geq \cdots \geq 0$.
So, $S_1 = 1$ and $S_2 = S_3 \cdots = 0$ will be the budgeted unique coverage problem.

Though the budgeted unique-coverage is similar to maximum-coverage problem, it is a relatively harder problem. The problem is generalization of max-cut problem. It has also similarity with radio broadcast coverage problem. We will cover this in later parts of the lecture.

For simplicity we will only focus on the unique coverage, although the generalization is not very hard. This problem has important applications in envy-free pricing in computational economics.

Simple $O(\log(n))$ algorithm:

- Partition the elements into $\log(n)$ classes according to their degrees, i.e., the number of sets that cover the element. So class i contains all the elements which are covered with at least 2^i and at most 2^{i+1} sets.
- Let i be the class of the maximum cardinality.
- Choose any set with probability $\frac{1}{2^i}$.

Lemma 3 *The expected number of the elements uniquely covered from class i is $\frac{1}{e^2}$ fraction of the element in class i.*

Proof: Consider an element X in class i and say its degree is d in S', where $2^i \leq d \leq 2^{i*+1}$:

$$\begin{aligned} P(X \text{ covered uniquely in } S') &= d\frac{1}{2^i}(1-\frac{1}{2^i})^{d-1} \\ &\geq \frac{2^i}{2^i}(1-\frac{1}{2^i})^{2^{i+1}} \\ &= \left(\left(1-\frac{1}{2^i}\right)^{2^i}\right)^2 \\ &\geq \frac{1}{e^2}. \end{aligned}$$

■

Therefore, the total profit of uniquely covered elements is at least $\frac{1}{e^2 \log n} \times \mathsf{opt}$.
If the maximum degree of the sets is d then the approximation ratio is $O(1/\log(d))$.
If we can get by some way $O(\log n)$ approximation for budgeted unique-coverage we can also get $\log B$, where B is the size of the set.

Theorem 4 *: The unique coverage is hard to approximate within a factor better than $O(\log^c n)$ for $0 < c < 1$ unless NP has a sub-exponential algorithm (it implies there is no $2^{o(n)}$ algorithm to solve it, it is also called, exponential time hypothesis (ETH)).*

Remark:$O(\log^{\frac{1}{3}} n)$ or even $o(\log n)$ approximation is hard under stronger but still plausible complexity assumptions.

Is there any subset such that cannot be uniquely covered $\frac{1}{\log n}$ fraction of all the elements. **A bad instance**: A bad instance that cannot be covered is shown in the Figure 1. A $\log n$ collection of subsets B_i is made from U. Each of the elements of U can belong to the set B_i with probability 2^{-i}. In this example, at most $o(\log n)$ elements can be uniquely covered by sets of A according to $|A|$.

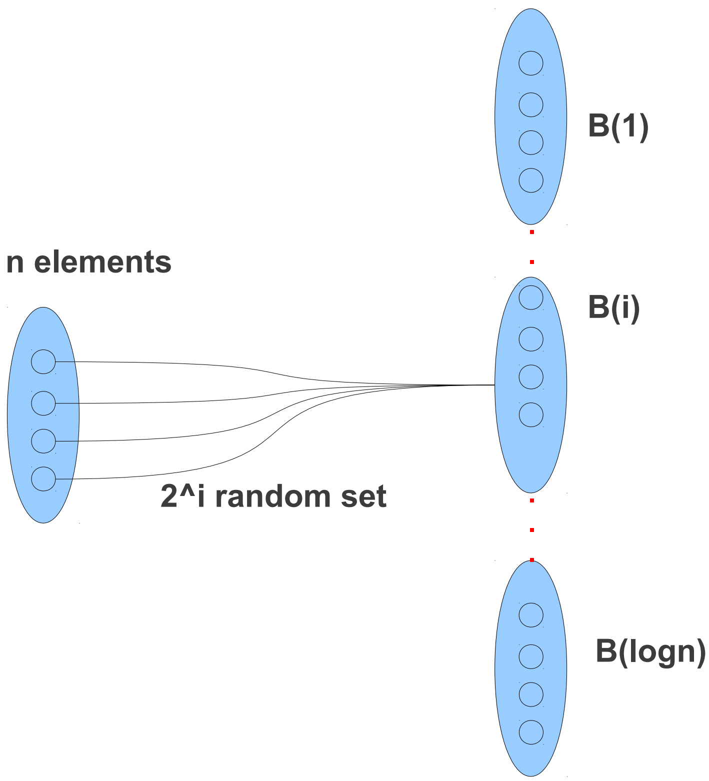

Figure 1: Bad instance where we cannot cover $\frac{1}{\log n}$ fraction