Universität Bremen

FB3: Mathematik/Informatik

Prof. Nicole Megow
Dr. Syamantak Das

Summer 2017

Approximation Algorithms

Lecture Notes 3:Greedy Algorithms

1 Weighted Set Cover

In the set cover problem, we are given a universe of n elements $U = \{e_1, e_2, \cdots e_n\}$ and a family of m subsets of U, $F = \{S_1, S_2, \cdots S_m\}$. There is a cost function (or weight function) associated with the family F, $c : F \to \mathbb{R}^+$. $c(S_j)$ denotes the cost of the subset S_j. The goal is to find the minimum cost subset of $F' \subseteq F$, or in simpler terms, a collection of sets of the minimum total cost such that

$$\forall i = 1, 2, \cdots n, e_i \in \bigcup_{S_j \in F'} S_j$$

1.1 Greedy Algorithm

We shall adopt a greedy algorithm for this problem. Let U_t be the set of uncovered elements at the beginning of iteration t and let $u_t = |U_t|$. Note that $U_1 = U$, the whole universe. Then, we shall pick the set from F which minimizes the ratio $\frac{c(S_j)}{|S_j \cap U_t|}$. We continue till all elements are covered. The algorithm runs for t' iterations such that $U_{t'} = \phi$.

Theorem 1. *The above Greedy algorithm is an* $(\ln(n) + 1)$*-approximation algorithm for weighted set cover*

In order to prove the above theorem, we shall be using the following crucial lemma.

Lemma 2 (Greedy Ratio Lemma)**.** *At any iteration* t, $\frac{c(S_t)}{|S_t \cap U_t|} \leq \frac{OPT}{u_t}$

Proof. (sketch). Consider the beginning of iteration t and recall that U_t is the set of elements still uncovered in Greedy. Now, consider the sets that the optimal solution uses to cover these elements. Let the set of these sets be O_t. Surely, we have not yet selected any of the sets in O_t till the iteration $t-1$, since otherwise, at least one of the elements in U_t would have been already covered. Hence, the minimum ratio set that greedy picks at iteration t must have a ratio which is at most the minimum among the sets in O_t. Formally,

$$\frac{c(S_t)}{|S_t \cap U_t|} \leq min_{S_j \in O_t} \frac{c(S_j)}{|S_j \cap U_t|}$$

Hence,

$$\frac{c(S_t)}{|S_t \cap U_t|} \leq min_{S_j \in O_t} \frac{c(S_j)}{|S_j \cap U_t|} \leq \frac{\sum_{S_j \in O_t} c(S_j)}{\sum_{S_j \in Ot} |S_j \cap U_t|} \leq \frac{OPT}{|U_t|} \tag{1}$$

The second least inequality follows from the following simple algebraic fact

Fact 3. *Let a_i, b_i be two sets of positive real numbers. Then* $\min_i \frac{a_i}{b_i} \leq \frac{\sum_i a_i}{\sum_i b_i}$

The above Fact follows from the simple averaging argument (please try to verify).
Now,

$$u_{t+1} = u_t - |S_t \cap U_t|$$

Using (1)

$$\begin{aligned} u_{t+1} &= u_t(1 - \frac{c(S_t)}{OPT}) \\ &\leq u_t e^{-\frac{c(S_t)}{OPT}} \end{aligned}$$

The above implies that for any t, $u_t \leq u_1 \Pi_{i=1}^{t-1} e^{-\frac{c(S_t)}{OPT}} = u_1 e^{-\sum_{i=1}^{t-1} \frac{c(S_t)}{OPT}}$. Taking natural logarithms (to the base e) on both sides and a simple algebraic manipulation gives

$$\sum_{i=1}^{t-1} c(S_i) \leq \ln(\frac{u_1}{u_t}) OPT \leq \ln(n) OPT, \text{ since } u_1 = n \text{ and } u_t \geq 1$$

Note that, we have not considered the cost of S_t in the above calculation. However, it is easy to see that this can be bounded by at most OPT using (1).

□

Remarks

- Note that the above analysis can yield stronger guarantees for other applications. For example, if we stop the algorithm at the iteration t (we choose t subsets). Then the approximation ratio is actually $\ln(u_1/u_t)$ and this could be much better than $\ln(n)$. Of course this is not a set cover solution. However, we shall see an application of this in the exercises in a slightly different context.

- For some applications, one might need to come up with an approximate version of the Greedy Ratio Lemma. For example, finding the object that minimizes the ratio of cost over number of new elements covered might itself be an NP-hard problem. In such cases, if one can guarantee that the Greedy Ratio Lemma holds with a factor of α, then using the above analysis, one can achieve a factor $\alpha \ln(n)$ approximation for the problem.

2 Tight Example

The following example shows that the analyssi of greedy is essentially tight, up to some constant factors. There are n elements and assume they are partitioned in to ℓ levels, where

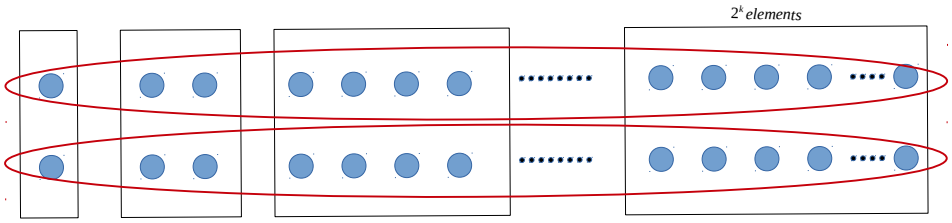

Figure 1: Tight example for Greedy

$\ell = O(\log n)$ and level k contains 2^{k-1} elements. Each of the square sets cover eveything in one level, while each of the two red sets can cover $n/2$ elements. Every set has cost 1. Greedy can potentially choose all the square sets, making its cost $O(\log n)$ while OPT chooses the two red sets incurring a cost of 2.

3 Hardness

Theorem 4. *For any $c < 1$, if there is a $c\log(n)$-approximation for set cover, then there exists algorithms that run in time $n^{O(\log \log n)}$ for each problem in NP.*

We shall see a weaker version of this result in a subsequent lecture.