COMPSCI 590.1: Graph Algorithms March 25, 2015

Lecture 20 : Online Set Cover

Lecturer: Debmalya Panigrahi *Scribe: Abhinandan Nath*

1 Overview

In this lecture, we discuss the online set cover problem [AAA$^+$03]. The ideas introduced in this lecture will be useful while discussing the online node-weighted Steiner tree problem.

2 The Set Cover Problem

In the set cover problem, we are given a universe U of elements and a collection of subsets $\mathbb{S} \subseteq 2^U$. We are also given a cost function $c : \mathbb{S} \to \mathbb{R}_{>0}$. The goal is to find a subset of $\mathbb{S}$ of minimum total cost such that it covers all the elements of U.

In the online version of the problem, the sets $\mathbb{S}$ are offline and the elements arrive online. Once a new element arrives the algorithm has to cover the element (if it is not already covered) by choosing a set containing the element, i.e., at each step the algorithm has to maintain a cover for the elements already seen. Once a set has been chosen it cannot be put back. Note that the algorithm might not see all the elements of U. The goal is to minimize the total cost of the sets chosen. Letting $|U| = n$ and $|\mathbb{S}| = m$, the next algorithm will achieve a competitive ratio of $O(\ln m \ln n)$.

Outline of the algorithm. We will write an LP for the problem, then produce a fractional solution to the LP which is $\ln m$ factor away from the optimal solution. Finally, we round the fractional solution in an online manner which leads to a loss of an extra factor of $\ln n$.

3 The Algorithm

The natural LP relaxation for the set cover problem is given below

$$\begin{aligned} \min \sum_S c(S) x_S \\ s.t. \sum_{S: e \in S} x_S \geq 1 \ \ \forall e \in U \\ x_S \geq 0 \ \ \forall S \in \mathbb{S} \end{aligned}$$

3.1 A Fractional Solution

Let OPT denote the cost of an optimal set cover. We do an initialization step where we set $x_S = 1/m$ after discarding all sets S with $c(S) > OPT$. Thus the initial value of the fractional solution is at most OPT.

Note. We can guess the value of OPT, by running the algorithm below for different values of OPT (which are powers of 2) in parallel and choosing the cover of the smallest value.

Next, when an element e arrives, we look at the quantity $\sum_{S:e\in S} x_S$ — if this is less than 1, we increase the value of x_S for all S containing e at the following rate (think of t as a continuous time variable)

$$\frac{dx_S}{dt} = \frac{x_S}{c(S)}$$

until $\sum_{S:e\in S} x_S = 1$. Now at any time t, let e be the current element. We then have

$$\frac{d}{dt}(\sum_S c(S)x_S) = \sum_{S:e\in S} c(S)\frac{dx_S}{dt} = \sum_{S:e\in S} x_S \leq 1$$

Thus the rate of change of the fractional solution value is at most 1, which means that the final value of the fractional solution is at most the total time for which the algorithm runs. The following lemma guarantees that the total time is at most $OPT \ln m$.

Lemma 1. *Total time for which the algorithm runs is at most $OPT \ln m$.*

Proof. We will bound the maximum amount of time for which a variable x_S can increase for any set S. The value of x_S starts at $1/m$ and grows to at most 1. We thus have

$$\begin{aligned}
&\frac{dx_S}{dt} = \frac{x_S}{c(S)} \\
\Rightarrow &\int_{1/m}^{1} \frac{dx_S}{x_S} = \int_0^{t_{max}(S)} \frac{dt}{c(S)} \\
\Rightarrow &t_{max}(S) = c(S)\ln m
\end{aligned}$$

When the current element e is being covered, at least one of the sets in the optimal solution contains e and hence its value is being increased. Thus, the sum of the times for which the x_S values of a set in an optimal solution increases is an upper bound on the total time for which the algorithm runs. Thus, the total time is at most $\sum_{S\in OPT} t_{max}(S) = \sum_{S\in OPT} c(S)\ln m = OPT \ln m$. □

Theorem 2. *The value of the fractional solution is at most $OPT \ln m$.*

3.2 Rounding the Fractional Solution

We need to round the above fractional solution in an online manner. If we can devise a rounding scheme such that at any step of the algorithm, the probability of a set S being picked is $x_S \ln n$, then we get an integral solution at the end whose expected value is $\sum_S x_S \ln n \leq OPT \ln m \ln n$.

The rounding scheme is as follows – each variable x_S is rounded independently of other sets. For a particular set S, let x_{S_i} be the values of the x_S variable after the ith element's primal constraint is satisfied. Suppose the probability that the set S is picked after the ith element arrives is $x_{S_i} \log n$. When the $(i+1)$th element arrives, if S is not yet picked we pick S with probability p_{i+1}. We then have

$$\begin{aligned}
&\Pr[S \text{ is picked after the } (i+1)\text{th element arrives}] \\
=&\Pr[S \text{ is picked after the } (i+1)\text{th element arrives} | S \text{ is picked after the } i\text{th element arrives}] \\
&\Pr[S \text{ is picked after the } i\text{th element arrives}] + \\
&\Pr[S \text{ is picked after the } (i+1)\text{th element arrives} | S \text{ is not picked after the } i\text{th element arrives}] \\
&\Pr[S \text{ is not picked after the } i\text{th element arrives}] \\
=&x_{S_i}\ln n + (1 - x_{S_i}\ln n)p_{S_{i+1}}
\end{aligned}$$

We want this value to be equal to $x_{S_{i+1}} \log n$ and we thus get $p_{S_{i+1}} = \frac{(x_{S_{i+1}} - x_{S_i}) \ln n}{1 - x_{S_i} \ln n}$

We thus have a Monte-Carlo algorithm with a competitive ratio of $\ln m \ln n$. This means that an element might remain uncovered after rounding. If this happens, we pick the cheapest set containing the element. The cost of this set is at most OPT. The probability that an element e is not picked is $\prod_{S:e\in S}(1 - x_S \log n) \leq \prod_{S:e\in S} \exp(-x_S \log n) = 1/n$. Hence, the expected value of the sets picked to cover the uncovered elements after rounding is at most OPT. We thus have a Las Vegas algorithm with a competitive ratio of $O(\ln m \ln n)$.

4 Summary

We presented an algorithm for the online set cover problem with a competitive ratio of $O(\ln m \ln m)$

References

[AAA+03] Noga Alon, Baruch Awerbuch, Yossi Azar, Niv Buchbinder, and Joseph Naor. The online set cover problem. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing, June 9-11, 2003, San Diego, CA, USA*, pages 100–105, 2003.