COMPSCI 530: Design and Analysis of Algorithms Nov 26, 2013

Lecture #25

Lecturer: Debmalya Panigrahi *Scribe: Nisarg Raval*

1 Overview

In this lecture we show how to solve online algorithms using linear programing (LP) and rounding mechanism. Specifically, we describe a method to solve online version of the "Set Cover" problem. Since, the entire input is not known a priori, writing an LP for the problem itself is challenging. We present an LP based fractional solution with $(\log m)OPT$ bounds on cost where OPT is the cost of the optimal solution. We also give a rounding scheme to construct an integer solution which has at most $(\log n \log m)OPT$ cost.

2 Online Set Cover

The offline set cover problem is defined as follows. Given a universe set U, a set system $\mathbb{S} = \{(S, C_s) : S \subseteq U\}$ such that $\bigcup_{S\in\mathbb{S}} S = U$ and C_s is the cost of set S, find a subset system $\mathbb{S}' \subseteq \mathbb{S}$ to minimize $\sum_{s\in\mathbb{S}'} C_s$ such that $\bigcup_{S\in\mathbb{S}'} S = U$. In a special case where all costs C_s are 1 (unweighted set cover) the goal is to find minimum number of subsets that cover all the elements in U.

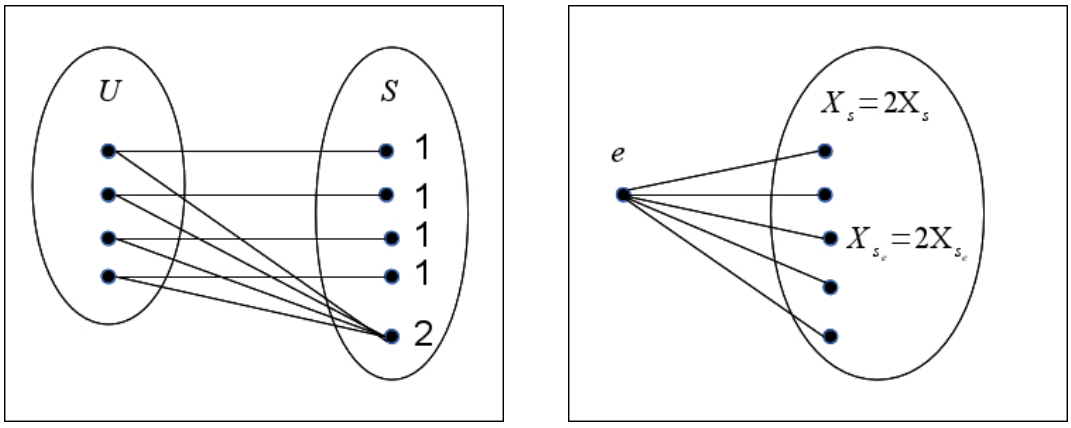
The online set cover problem is similar to the offline version except instead of covering all the elements of U we need to only cover elements which arrive online. Usually, the elements arrive one at a time and in each iteration the solution is updated to cover the newly arrived element. But the catch here is that, once a subset is included in the solution we can not remove it from the solution. In other words the solution should satisfy the monotone property.

The LP for the set cover problem can be written as follows:

$$\begin{aligned} \min & \sum_{S\in\mathbb{S}} C_s X_s \\ \text{s.t.} & \sum_{S:e\in S} X_s \geq 1 \qquad \forall e \in U \\ & X_s \geq 0 \qquad\qquad \forall S \in \mathbb{S} \end{aligned}$$

Here, X_s is the variable corresponding to each set $S \in \mathbb{S}$. Note that we can not directly use the above LP in online version because we don't know the entire input at once. More specifically, we don't know what elements need to be covered before hand. However, we can think of the stream of input as evolving LP. The idea is to start with the basic LP and on every iteration, change the LP according to the new input. In case of set cover this is equivalent of adding one constraint (corresponding to the new element) on every iteration. In case of offline version, we can simply run LP solver and then perform rounding to get an integer solution. However, in case of online version we can not simply run LP solver on every iteration because the resulting solution may violate the monotone property described before. To solve the online set cover problem using LP, we use greedy algorithm to get the fractional solution and apply rounding technique (similar to the offline version) to get an integer solution.

Competitive Ratio. Before describing the algorithm we briefly talk about the competitive ratio of the online set cover problem. Remember that the approximation factor for the offline set cover problem is $\log n$.



(a) Competitive Ratio Example. (b) Illustrating cost of Greedy Algorithm.

Figure 1: Example Illustration of cost of Online Set Cover Problem.

However, in case of online set cover problem the competitive ratio is polynomial in n ($\Omega(n)$). This can be shown using a simple example illustrated in Figure 1a. Consider a universe $U = \{e_1, e_2, \ldots e_n\}$ and a set system $\mathbb{S} = \{S_1, S_2, \ldots S_n, S_{n+1}\}$ such that $\forall i = 1 \rightarrow n, S_i = \{e_i\}$ and $C_{s_i} = 1$. Also, $S_{n+1} = U$ and $C_{s_{n+1}} = 2$. In this case at each step the algorithm will choose the set S_i for the new element e_i since that is the set with least cost. This leads to the cost of n. However, the optimal solution is S_{n+1} which has cost of only 2.

2.1 Greedy Algorithm

We need to make sure that on every iteration we get fractional solution which should satisfy the new constraint and monotone property. Although the described algorithm is applicable in general case, for simplicity, assume all sets are of cost 1. Hence, the cost of the solution is the total number of subsets in it. Initialize each X_s by $1/m$, where m is the number of subsets ($|\mathbb{S}|$). In first iteration, the cost of the total solution is $\sum_{S \in \mathbb{S}} X_s = \sum_{S \in \mathbb{S}} (1/m) = 1$, which is equal to the cost of optimal solution as optimal algorithm needs to choose at least one subset. Hence, for any one input $ALGO \leq OPT$, where $ALGO$ and OPT are the costs of greedy algorithm and optimal algorithm respectively. For any new element e, if its already covered do nothing otherwise double the value of all subsets containing e until e is covered. Formally, these steps are described below:

---

while e is not covered (i.e. $\sum_{S:e \in S} X_s < 1$) **do**
  $X_s = 2X_s \quad \forall S \subseteq \mathbb{S} : e \in S$ //double the value of each set containing e
end while

Lemma 1. *In any iteration of the greedy algorithm, the cost (ALGO) increases by at most* 1.

Proof. We only increase the value of X_s if $\sum_{S:e \in S} X_s < 1$. Hence, the change in the cost in one iteration is $\sum_{S:e \in S} (X_s^{new} - X_s^{old}) = \sum_{S:e \in S} (2X_s^{old} - X_s^{old}) = \sum_{S:e \in S} X_s^{old} < 1$. □

Theorem 2. *The cost of a fractional solution of the greedy algorithm is at most* $(\log m)OPT$.

Proof. Since, in each iteration $ALGO$ is increased by at most 1, we only need to count the number of iterations taken by the greedy algorithm to compute $ALGO$. Consider a new element e which belongs to

some subsets of $\mathbb{S}$ as shown in Figure 1b. The variables X_s corresponding to each of these subsets will be doubled. In particular, the set X_{s_e} will also be doubled, where X_{s_e} is the variable corresponding to the subset S_e which in turn is the subset used by the optimal algorithm to cover e. Hence, the algorithm can not have an iteration where no subset of OPT participate. Therefore, the question is how many iterations a subset S can participate in? Since, we initialize each subset with $1/m$ and at each iteration (in which subset participates) we double its value, a subset can participate in at most $\log m$ iterations because it can have value of at max 1. In worst case each subset in OPT participates in $\log m$ iterations. Hence, the total number of iterations is at most $(\log m)OPT$. In other words, $ALGO \leq (\log m)OPT$. □

2.2 Rounding

In this subsection we show how to construct an integer solution from the fractional solution using rounding technique. In case of offline set cover we perform rounding simply by setting $X_s = 1$ with probability $X_s^*(\log n)$ and 0 otherwise [BSN09], where X_s^* is the fractional optimal solution. However, we can not apply the same rounding technique in online version because it doesn't ensure the monotonicity of the solution. For example, some X_s may rounded to 1 in one iteration and 0 in later iterations. In order to ensure monotonicity, we need to ensure that once X_s is rounded to 1, in any subsequent iterations it must be rounded to 1. This can be achieved by monotone rounding scheme with $Pr[\mathbb{X}_s = 1] = X_s(\log n)$, where $\mathbb{X}_s$ is the boolean variable indicating whether set X_s is picked in the solution or not. The bound on the integer solution is $(\log n \log m)OPT$ which can be proved as follows:

$$\begin{aligned}
\mathbb{E}[IntegerSolution] &= \mathbb{E}[\sum_{S\in\mathbb{S}} \mathbb{X}_s] \\
&= \sum_{S\in\mathbb{S}} \mathbb{E}[\mathbb{X}_s] \qquad \text{// by linearity of expectation} \\
&= \sum_{S\in\mathbb{S}} X_s \log n \\
&\leq (\log n \log m)OPT
\end{aligned}$$

Next, we show with what probability we should perform rounding in every iteration to achieve the overall probability of $X_s(\log n)$. Let $\mathbb{X}_s^{-e}$ and $\mathbb{X}_s^{+e}$ be the respective values of S before and after e arrives. Suppose for element e, X_s increases by ΔX_s, then there are two possible cases.

1. If set S is already in the solution then $\mathbb{X}_s^{-e} = 1$, we don't have any choice but to pick S.
2. Otherwise, $\mathbb{X}_s^{-e} = 0$, in which case we pick S with probability, $Pr[\mathbb{X}_s^{+e} = 1] = \frac{\Delta X_s \log n}{1 - X_s \log n}$.

This procedure ensures that $Pr[\mathbb{X}_s^{-e} = 1] = X_s \log n$ (before e arrives) and $Pr[\mathbb{X}_s^{+e} = 1] = (X_s + \Delta X_s)\log n$ (after e arrives).

$$\begin{aligned}
Pr[\mathbb{X}_s^{+e} = 1] &= Pr[\mathbb{X}_s^{+e} = 1/\mathbb{X}_s^{-e} = 1]Pr[\mathbb{X}_s^{-e} = 1] + Pr[\mathbb{X}_s^{+e} = 1/\mathbb{X}_s^{-e} = 0]Pr[\mathbb{X}_s^{-e} = 0] \\
&= X_s \log n + \frac{\Delta X_s \log n}{1 - X_s \log n}(1 - X_s \log n) \\
&= (X_s + \Delta X_s)\log n
\end{aligned}$$

Since, $Pr[\mathbb{X}_s = 1] = X_s \log n$, we can show that all elements are covered in integer solution with high probability using the similar analysis performed in an offline case. The given algorithm is a Monte Carlo algorithm which can be easily converted into Las Vegas algorithm. The idea is at each step, look at the elements which are not covered and cover them using greedy approach (select the subset which cover most uncovered elements with minimum cost).

3 Summary

We introduced online set cover problem and gave an LP based technique to find fractional solution. We proved $(\log m)OPT$ bound on cost of the fractional solution. We also gave monotone rounding scheme to find an integer solution having cost of at most $(\log n \log m)OPT$. It is possible to de-randomized the given algorithm to find a deterministic algorithm for online set cover problem [BSN09]. The method describe in this lecture is more general and can be applied to many online problems like Stiner Forest, Metric/Non-metric Facility Location, Weighted Paging etc.

References

[BSN09] Niv Buchbinder and Joseph (Seffi) Naor. The design of competitive online algorithms via a primal: Dual approach. *Found. Trends Theor. Comput. Sci.*, 3:93–263, February 2009.