Online Set Cover

CS 599I Online Algorithms

February 27

1 The Set Cover Problem

We are given a universe of elements $\mathcal{U}$, where $|\mathcal{U}| = n$, and m subsets of $\mathcal{U}$: $S_1, \ldots, S_m$, such that $\bigcup_i S_i = \mathcal{U}$. A *set cover* for this instance is defined as any $I \subseteq [m]$ such that $\bigcup_{i \in I} S_i = \mathcal{U}$. The goal of the problem is to find the set I of minimum size.

This is the *unweighted* version of set cover, since each set has an equal weight of 1. It is known that the offline instance of Set Cover is NP-hard.

2 Offline Approximation Algorithms

We present two algorithms for the offline version of Set Cover that both give a $O(\log n)$ approximation ratio.

2.1 The Greedy Algorithm

The first algorithm, GREEDY, works by greedily choosing the set that covers most of the elements that are still not covered by any set. More formally:

1. $I \leftarrow \{\}$
2. While $\mathcal{U} \neq \emptyset$:
   (a) Pick $j = \arg\max_i |S_i|$ and let $I \leftarrow I \cup \{j\}$.
   (b) $\mathcal{U} \leftarrow \mathcal{U} \setminus S_j$.
   (c) $\forall i \in [m] : S_i \leftarrow S_i \setminus S_j$.

Theorem 2.1. GREEDY *is an* $O(\log n)$ *approximation algorithm for Set Cover.*

Proof. (sketch) Let $T_1, \ldots, T_r$ be the sets picked by GREEDY. Let us also denote by t_i the number of elements covered for the first time by the set T_i, where $i = 1, \ldots, r$. Finally, assume that the optimum solution (OPT) has size O.

At the first iteration, since GREEDYchooses the set of maximum size, t_1 will be at least as large as the average set size chosen by OPT. Hence, $t_1 \geq n/O$. Following similar reasoning, for

$i = 1, \ldots, r$, we have that $t_i \geq (n - \sum_{j=1}^{i-1} t_j)/O$. After some calculations, we conclude that indeed $r \leq O(\log n)O$. □

2.2 The LP Algorithm

The second algorithm is based on linear programming and randomized rounding. First, we express Set Cover as an integer linear program. For this, we use an indicator variable x_i which denotes whether set S_i is chosen in the cover (value 1) or not (value 0).

$$\begin{aligned} \text{minimize} \quad & \sum_{i \in [m]} x_i \\ \text{subject to} \quad & \forall e \in \mathcal{U} : \sum_{i: e \in S_i} x_i \geq 1 \\ & \forall i \in [m] : x_i = \{0, 1\} \end{aligned}$$

We obtain the relaxed version of the program by allowing x_i to take any real value between $[0, 1]$; hence, for the LP the last condition becomes $x_i \geq 0$. By solving the relaxed LP, we obtain a fractional solution x^*, where $\sum_{i \in [m]} x_i^* \leq OPT$. We then round the fractional solution to obtain an integer one. The algorithm, Randomized Rounding, can be summarized as follows:

1. Solve the relaxed LP to obtain the optimum solution x^*. Let $I \leftarrow \emptyset$.
2. For $4 \log n$ times repeat:
   (a) $\forall i \in [m]$, let $I \leftarrow I \cup \{i\}$ with probability x_i^*.

Theorem 2.2. Randomized Rounding *gives:*

1. *a feasible solution with probability* $\geq 1 - 1/n^2$.
2. $\mathbb{E}[I] = O(\log n) \sum_i x_i^* \leq O(\log n) OPT$.

Proof. We first show (2). Fix some iteration $k = 1, \ldots, 4 \log n$. Let X_i be the indicator random variable that denotes whether we pick set S_i during iteration k. We have that $Pr[X_i = 1] = x_i^*$ and $Pr[X_i = 1] = 1 - x_i^*$. If $X = \sum_{i \in [m]} X_i$, by linearity of expectation:

$$\mathbb{E}[X] = \sum_{i \in [m]} \mathbb{E}[X_i] = \sum_{i \in [m]} x_i^*$$

Summing over all rounds, we obtain that the expected size of I will be at most $(4 \log n) \sum_{i \in [m]} x_i^*$.

In order to show (1), we fix some element $f \in \mathcal{U}$ and some iteration k. We then compute the probability that the element is not covered in this iteration:

$$Pr[f \text{ not covered at } k] = \prod_{i: f \in S_i} (1 - x_i^*) \leq \prod_{i: f \in S_i} e^{-x_i^*} = e^{-\sum_{i: f \in S_i} x_i^*} \leq e^{-1}$$

where the first inequality comes from the fact that for any $x \geq 0, 1 - x \leq e^{-x}$ and the second inequality from the fact that x^* is a feasible solution for the LP. Now, since every iteration has independent choices, the probability that f is not covered at all is:

$$Pr[f \text{ not covered}] \leq (1/e)^{4 \log n} \leq 1/n^3$$

We have proven this bound for a fixed element f. We next apply the union bound to conclude that the probability that there exists some element f not covered at all is at most $n(1/n^3) = 1/n^2$. □

3 Online Algorithms

In the online setting of Set Cover, we are initially given the m sets, but we do not know which elements they contain. At any time t, we get a new element e_t and learn which sets contain e_t. We then have to irrevocably pick a set that will cover e_t if it is not already covered. The goal is again to minimize the number of sets we pick.

We first give a lower bound on the competitive ratio of Online Set Cover. Consider an instance where $\mathcal{U} = \{1, \ldots, n\}$ and $S_1, \ldots, S_m$ are all the sets of size $\sqrt{n}$. The adversary will repeat the following strategy for $\sqrt{n}$ iterations: pick any element that is not covered by the current solution. This implies that any deterministic algorithm will construct a solution with cost $\sqrt{n}$. On the other hand, some set S_j will cover all the elements chosen by the adversary and thus $OPT = 1$. This gives a lower bound of $\sqrt{n}$. However, notice that the number of sets in this case is $m = \binom{n}{\sqrt{n}}$.

We next present an online algorithm with competitive ratio that depends on both m, n. As a first step, we will give an algorithm for the fractional version of Set Cover.

1. For every $i \in [m]$, initialize $x_i \leftarrow 1/m^2$.
2. At time t, when e_t arrives
   (a) Set $y_e \leftarrow 0$.
   (b) While $\sum_{i:e_t \in S_i} x_i < 1$:
   - $\forall i$ s.t.$e_t \in S_i : x_i \leftarrow 2x_i$
   - $y_{e_t} \leftarrow y_{e_t} + 1$.

The lines in blue explain how we construct in parallel with the primal solution for the LP a solution for the dual LP, which is the following:

$$\begin{aligned} \text{maximize} \quad & \sum_{e \in \mathcal{U}} y_e \\ \text{subject to} \quad & \forall i \in [m] : \sum_{e \in S_i} y_e \leq 1 \\ & \forall e \in \mathcal{U} : y_e \geq 0 \end{aligned}$$

The dual variables will only be used for the analysis of the algorithm.

Theorem 3.1. *The Online Fractional algorithm has a $O(\log m)$ competitive ratio.*

Proof. The analysis is based on a primal-dual argument. In particular, we will show how to construct a feasible dual solution y^* such that the total cost of the algorithm is at most $O(\log m) \sum_e y_e^*$. Since the cost of a feasible dual is at most the cost of the optimal primal solution, this proves the theorem. To show this, we will prove two claims. Let $\mathcal{U}_t$ be the set of elements that appear through time t.

Claim 1. At any time t, $\sum_{i=1}^m x_i \leq \sum_{e \in \mathcal{U}_t} y_e + 1/m$.

We will show this using induction. Before the first iteration, we have that $\sum_{i=1}^m x_i = \sum_{i=1}^m 1/m^2 = 1/m$, whereas $\sum_{e \in \mathcal{U}_t} y_e = 0$; so this holds with equality. Now, consider some time t. If the constraint is satisfied, then nothing changes. Otherwise, we have that $\sum_{i:e_t \in S_i} x_i^{old} < 1$, where x_i^{old} is the value of the variable before time t. In this case, the new value will be $x_i = 2x_i^{old}$.

Hence, the change on the LHS of the inequality will be $\sum_{i=1}^{m}(x_i - x_i^{old}) = \sum_{i=1}^{m} x_i^{old} < 1$, whereas the change on the RHS is exactly 1.

Claim 2. The solution $y_e^* = y_e/(2\log m)$ is a feasible dual solution.

To show this claim, we have to prove that for any set S_i, $\sum_{e \in S_i} y_e \leq 2\log m$. Indeed, notice that one of the y_e increases only in the case where the variable x_i doubles. Moreover, once x_i reaches 1, it does not need to double again. Since the starting value of x_i is $1/m^2$, x_i can be doubled at most $O(\log m)$ times. □

Thus, we can compute online a fractional solution with competitive ratio $O(\log m)$. To obtain an online integer solution for Set Cover, we can use the randomized rounding technique from the previous section, which will give an additional $O(\log n)$ factor in the competitive ratio.

Theorem 3.2. *The competitive ratio for Online Set Cover is* $O(\log n \log m)$.