1 Overview

In the lecture we introduce a different approach for designing approximation algorithms. So far, we introduced combinatorial optimization problems, and solved the problems via combinatorial algorithms. We phrased the problems as mathematical and integer programs, but did not use these formulations directly. In this lecture we will give examples of approximation algorithms that rely on rounding solutions to linear programs. As we've seen before, the integer programming problems we have seen can be relaxed to a linear program. The problem however is that a linear program produces *fractional solutions* where variables can take on values in $[0,1]$ and not in $\{0,1\}$. In this lecture we will show how to take a solution of a linear program and round it in a meaningful way. To do so we discuss the set cover problem, which is the dual problem of the max-cover problem we studied in previous lectures.

2 The Set Cover Problem

The minimum set cover problem can be formalized as follows. We are given sets $T_1, \ldots, T_m$ that cover some universe with n elements, and the goal is to find a family of sets with minimal cardinality whose union covers all the elements in the universe. We assume that the number of sets m and the number of elements in the universe n are polynomially related, i.e. $m \approx n^c$, for some constant $c > 0$. We will express the approximation ratio in terms of n (the number of elements in the universe), but the results are asymptotically equivalent to in terms of m. [1]

2.1 The greedy algorithm for set cover

Notice that the decision version of the min-set cover is exactly the decision version of max-cover: *given a family of sets $T_1, \ldots, T_m$ is there a family of sets of size k that covers at least d elements in the universe?* We know that this decision problem is NP-complete, and hence min-set cover is an NP-hard optimization problem. Our goal will therefore be to design a good approximation algorithm for this problem. A natural candidate is the greedy algorithm presented in Algorithm 1, which is a straightforward adaptation of the greedy algorithm for max-cover.

Theorem 1. *The greedy algorithm is a $\Theta(\log n)$ approximation.*

[1] In the max cover problem we used n to denote the number of sets, and the results we expressed were independent of the size of the universe. For convenience, we used n here to denote the number of elements in the universe since our analysis will be in terms of the elements and not the sets that cover them, but as discussed above this choice is insignificant for the results we will show.

Algorithm 1 Greedy algorithm for Min Set Cover

```
S ← ∅
while not all elements in the universe are covered do
    T set that covers the most elements that are not yet covered by S
    S ← S ∪ {T}
end while
return S
```

Proof. First, observe that the algorithm terminates after at most m stages. Since there are m sets and in each iteration of the `while` loop we add a set to the solution, we will terminate after at most m steps.

Let u_j denote the number of elements in the universe that are still *not* covered at iteration j of the while loop, and let k denote the number of sets in the optimal solution, i.e. $k = \texttt{OPT}$. In each iteration j we can use all the k sets in the optimal solution to cover the entire universe, and in particular to cover u_j. Therefore, there must exist at least one set in the optimal solution that covers at least u_j/k elements. Since we select the set whose marginal contribution is largest at each iteration, this implies that in every iteration we include at least u_j/k elements into the solution. Put differently, we know that after iteration j we are left with at most $u_j - u_j/k$ elements. That is:

$$u_{j+1} \leq \left(u_j - \frac{u_j}{k}\right) \leq \left(1 - \frac{1}{k}\right) u_j \leq \left(1 - \frac{1}{k}\right)\left(1 - \frac{1}{k}\right) u_{j-1} \leq \ldots \leq \left(1 - \frac{1}{k}\right)^{j+1} u_0 = \left(1 - \frac{1}{k}\right)^{j+1} n$$

where the last equality is due to the fact that $u_0 = n$ since there are exactly n elements in the universe before the first iteration. Notice also that once we get to stage i for which $u_i \leq 1$ we're done, since this implies that we need to select at most one more set and obtain a set cover. So the question is how large does i need to be to guarantee that $u_i \leq 1$? A simple bound shows that whenever $i \geq k \cdot \ln n$ we have that $u_i \leq 1$:

$$\left(1 - \frac{1}{k}\right)^i = \left(\left(1 - \frac{1}{k}\right)^k\right)^{\frac{i}{k}} \leq e^{-\frac{i}{k}}$$

we can approximate the number of iterations as if the size is reduced by a factor of $1/e$:

$$n \cdot e^{-\frac{i}{k}} \leq 1 \iff e^{-\frac{i}{k}} \leq n^{-1} \iff -i/k \leq -\ln n \iff i \geq k \ln n$$

and we therefore have that after $i = k \cdot \ln n$ steps the remaining number of elements u_i is smaller or equal to 1. Thus, after at most $k \cdot \ln n + 1 = \texttt{OPT} \ln n + 1$ iterations the algorithm will terminate with a set cover whose size is at most $k \cdot \ln n + 1 = \texttt{OPT} \ln n + 1 \in \Theta(\ln n \cdot \texttt{OPT}) = \Theta(\log n \cdot \texttt{OPT})$. □

Until this point we have been fortunate to find constant-factor approximation algorithms, which makes the $\log n$ approximation factor seem somewhat disappointing. It turns out however that improving over the $\log n$ approximation ratio is impossible in polynomial time unless P=NP[1]. We can look at the glass as half-full: we have an optimal (unless P=NP) approximation algorithm.

A note about modeling. Notice the stark difference between the guarantees obtainable for min-set cover and max-cover. In one we cannot hope to do better than $\Theta(\log n)$ where as in the other we can obtain an approximation ratio of $1-1/e \approx 63\%$ of the optimal solution. In some cases, we really want to solve a min-set cover problem and the $\Theta(\log n)$ is unavoidable. However, there are many cases where we have the freedom to choose the models we work with. In this case, the choice to cover as many elements in the universe as possible under some budget as opposed to covering all the elements under a minimal cost is the difference between desirable and not-so-desirable guarantees.

3 An LP-based Approach

We will now introduce a different approach for designing approximation algorithms. This approach involves solving a linear program which is a *relaxation* of the integer program that defines the problem as a first step. Then, it uses various methods to take a fractional solution of the linear program and interpret it as an integral solution.

Min set cover as a mathematical program. For the min-set cover problem, we can associate a variable x_i with each set T_i, and formulate the problem as the following integer program:

$$\min \sum_{i=1}^{m} x_i \tag{1}$$

$$\text{s.t.} \sum_{i:j \in T_i} x_i \geq 1 \qquad \forall j \in [n] \tag{2}$$

$$x_i \in \{0,1\} \qquad \forall i \in [m] \tag{3}$$

We know that the problem is NP-hard (and following the above discussion that its solution cannot be approximated within a factor better than $\log n$ unless P=NP), and therefore we do not know how to solve the above integer program in polynomial-time. However, if we relax condition (3) to:

$$x_i \in [0,1] \quad \forall i \in [m]$$

we have a linear program which we can solve in polynomial time. One key observation is that the optimal integral solution $\texttt{OPT}$ is always an upper bound on the fractional solution $\texttt{OPT}_{LP}$. This is simply due to the fact that the optimal integral solution is a feasible solution to the LP, and therefore since we can solve the LP optimally, its solution has to be at least as good as that of the optimal integral solution.

An approximation algorithm through randomized rounding. The first LP-based algorithm we will introduce uses a technique called *randomized rounding*. This techniques naturally interprets the fractional solution of the linear program as a probability distribution on its variables, and then selects a solution using the probability distribution. For this problem let d be a constant that satisfies the following condition:

$$e^{-d\log n} \leq \frac{1}{4n}$$

This choice may seem a bit mystical now, but it'll become clear as we analyze the algorithm below.

Theorem 2. *The randomized rounding algorithm returns a set S which is a set cover and a $\Theta(\log n)$ approximation to $\texttt{OPT}$, with probability at least 1/2.*

Algorithm 2 Randomized-rounding algorithm

$S \leftarrow \emptyset$
$\mathbf{p} \leftarrow$ Solution to LP
while $i \le d \log n$ **do**
  $S^{(i)} \leftarrow$ select every set $i \in [m]$ with probability p_i
  $S \leftarrow S \cup S^{(i)}$
end while
return S

Proof. Consider an element a in the universe, and w.l.o.g. assume that this element is covered by the k sets $T_1, \dots, T_k$. We will first analyze the probability that this element a is covered by the set S returned by the algorithm.

$$\mathbb{P}\left[a \text{ is not covered by } S^{(i)}\right] \le \prod_{j=1}^{k}(1-p_j) \le \left(1-\frac{1}{k}\right)^k \le \frac{1}{e}$$

The above inequality $\prod_{j=1}^{k}(1-p_j) \le \left(1-\frac{1}{k}\right)^k$ is due to the fact that for any $x_1, \dots, x_k \in \mathbb{R}$ the function $h(\mathbf{x}) = \prod_{j=1}^{k}\left(1 - \frac{x_i}{\sum_{i \in [k]} x_i}\right)$ achieves its maximum when $x_1 = x_2 = \ldots = x_k = \sum_{i \in [k]} x_i / k$.

Therefore:

$$\begin{aligned}
&\mathbb{P}\left[a \text{ is not covered by } S = \cup_{i=1}^{d \log n} S^{(i)}\right] \\
&= \mathbb{P}\left[a \text{ not covered by } S^{(1)}\right] \cdot \mathbb{P}\left[a \text{ not covered by } S^{(2)}\right] \cdots \mathbb{P}\left[a \text{ not covered by } S^{(d \log n)}\right] \\
&\le \left(\frac{1}{e}\right)^{d \log n} \\
&\le \frac{1}{4n}
\end{aligned}$$

We can now bound the probability that S is not a set cover by a union bound:

$$\mathbb{P}\left[S \text{ is not a set cover}\right] \le \sum_{j=1}^{n} \mathbb{P}\left[a_j \text{ is not covered by } S\right] \le n \cdot \frac{1}{4n} \le \frac{1}{4}$$

We therefore have that that with probability at least $1/4$ the solution S is a set cover. This is the first step. We now need to argue about the quality of the set cover, that is, the approximation ratio.

We will first bound the *expected* number of sets that are in S:

$$\begin{aligned}
\mathbb{E}\left[|S|\right] &= \mathbb{E}\left[\sum_{i=1}^{d\log n} |S^{(i)}|\right] \\
&= \sum_{i=1}^{d\log n} \mathbb{E}\left[|S^{(i)}|\right] \\
&= \sum_{i=1}^{d\log n} \sum_{i=1}^{m} p_i \\
&= \sum_{i=1}^{d\log n} \texttt{OPT}_{LP} \\
&= d\log n \cdot \texttt{OPT}_{LP} \\
&\leq d\log n \cdot \texttt{OPT}
\end{aligned}$$

We therefore have the expected number of sets selected is at most $d\log n \cdot \texttt{OPT}$, which tells us that *in expectation* we have a $d\log n$ approximation. To strengthen this guarantee we can ask what is the probability that we will have more than $|S| \geq 4 \cdot d\log n$? Recall Markov's inequality:

> ***Markov's inequality:*** *Let X be a non-negative random variable and $t \geq 0$. Then:*
>
> $$\mathbb{P}[X \geq t] \leq \frac{\mathbb{E}[X]}{t}$$

Using Markov's inequality we get:

$$\mathbb{P}\left[|S| \geq 4d\log n\texttt{OPT}\right] \leq \frac{\mathbb{E}\left[|S|\right]}{4d\log n\texttt{OPT}} = \frac{1}{4} \tag{4}$$

We therefore get that with probability at least 3/4 the size of the solution is at most $4d\log n\texttt{OPT}$, or in other words, with probability 3/4 we have a $4d\log n$ approximation. Since d is constant, this is a $\Theta(\log n)$ approximation.

Now notice that with probability at most $\frac{1}{4} + \frac{1}{4}$ we either don't have a set cover or the number of sets exceeds $4d\log n$. Therefore with probability at least 1/2 we have that S is a set cover (it covers all the elements in the universe) *and* $|S| \leq 4d\log n$, i.e. we have a $\Theta(\log n)$ approximation. $\square$

Notice that the guarantee we have from the randomized rounding algorithm is quite strong. After the algorithm terminates we can check whether it returns a set cover, and also whether the total cost exceeds $4d\log n\texttt{OPT}_{LP}$ (we can do that since we have the solution of the LP). Thus, if the algorithm failed, we can run it again and hope that it succeeds. In expectation, we will need to run the algorithm twice after which we will have a set cover whose cost is at most $4d\log n \cdot \texttt{OPT}_{LP}$.

3.1 A rounding algorithm for small frequencies

The minimum set cover problem is a generalization of a simpler problem we studied. Recall the *vertex cover problem* (problem set 9) where we have an undirected graph, and wish to find the fewest nodes that cover all the edges in the graph. We can model this as a minimum set cover problem by associating each vertex v_i with a set T_i and each edge e_j becomes an element a_j in the universe that we aim to cover. While this is an instance of the minimum set cover problem, it is an easier one. In particular, the *frequency* of elements in sets is bounded.

Definition. *For a given instance of set cover we say that an element $j \in [n]$ appears with **frequency** c_j if it is covered by at most c_j sets. The **frequency of the instance** is $c = \max_{j \in [n]} c_j$.*

In general, the frequency of the instance can be as large as m which is the number of sets. In some cases however, it is very reasonable to assume that we have instances with bounded frequencies. In the vertex cover problem, for example, the frequency is 2 since each edge can belong to at most two vertices. We have already seen in the problem set (or at least in the solution) that one can achieve an approximation ratio of 2 for this problem, which is far better than the $O(\log n)$ we have for the general minimum set cover case. Can we generally obtain better guarantees when the frequency is bounded? The following algorithm gives an approximation ratio that depends on c.

Algorithm 3 Rounding algorithm

$S \leftarrow \emptyset$
$\mathbf{p} \leftarrow$ Solution to LP
$S \leftarrow$ all sets $i \in [m]$ for which $p_j \geq 1/c$
return S

Theorem 3. *For any instance of the minimum set cover problem whose frequency is c, the rounding algorithm above returns a solution which is a c-approximation to* $\texttt{OPT}$.

Proof. Let a be an element in the universe, which, w.l.o.g. is covered by sets $T_1, \ldots, T_k$, the requirement of the LP is that the corresponding variables $p_1, \ldots, p_k$ respect: $p_1 + \ldots + p_k \geq 1$. Therefore, there must be at least one variable p_i s.t. $p_i \geq 1/k$. Since the instance frequency is c we know that $k \leq c$ and therefore $p_i \geq 1/k \geq 1/c$. By the condition of the algorithm this implies that we selected at least one set that covers a. Since this holds for all elements in the universe we have a set cover.

Regarding the approximation ratio, consider the cost of the LP: $\sum_{i=1}^{m} p_i$, and let the I denote the set of indices for which $p_i \geq 1/c$, notice that the solution that we selected has cost $\sum_{i \in I} c \cdot p_i$ since we turned each set whose cost was greater or equal to $1/c$ to 1. Thus:

$$|S| = \sum_{i \in I} c \cdot p_i = c \cdot \sum_{i \in I} p_i \leq c \cdot \sum_{i=1}^{m} p_i = c \cdot \texttt{OPT}_{LP} \leq c \cdot \texttt{OPT}. \qquad \square$$

4 Discussion and Further Reading

Rounding linear and convex programs is power technique for designing approximation algorithms. In this course, we will soon explore other rounding methods, and see how we can design approximation algorithms and by using more involved rounding techniques. To read more about rounding techniques please see [2] and [3].

References

[1] Uriel Feige. A threshold of ln n for approximating set cover. *Journal of the ACM (JACM)*, 45(4).

[2] Vijay V. Vazirani. *Approximation algorithms*. Springer, 2001.

[3] David P. Williamson and David B. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, 2011.