COMPSCI 632: Approximation Algorithms	September 27, 2017
Lecture #10	
Lecturer: Debmalya Panigrahi	*Scribe: Eli Margolin*

1 Overview

This lecture is the first in a series focusing on rounding in approximation algorithms. We introduce the concept of randomized rounding, apply it to the Set Cover problem, and then start a new problem, congestion minimization, which will be finished next lecture.

2 A Note on Separating Hyperplanes

A common method for solving Linear Programs is with SIMPLEX. However, this is worst-case exponential. Instead, we are going to use a polynomial time method to prove feasibility given a certain solution. This method relies on using *separating hyperplanes*.

Given a polytope P and some solution x, we create an algorithm that tells us if $x \in P$. If $x \notin P$, then we create a seperating hyperplane, such that the entirely of P is on one side and x on the other. This hyperplane represents some constraint of P that has been violated by x.

3 Randomized Rounding

3.1 Set Cover

We start with the set cover problem. As a reminder, our linear program for the optimal fractional solution OPT_f consists of the following:

$$min \sum_s x_s \tag{1}$$

$$\sum_{s:e\in s} x_s \geq 1 \tag{2}$$

$$x_s \geq 0 \tag{3}$$

Our approach is to round OPT_f to an integral solution. We are going to use x_s, the value from OPT_f, as the probability that subset s will be used in our integral solution. We repeat this process of picking s until we have a set cover.

First, we need to figure out the cost of such a solution. Let:

$$X_s = \begin{cases} 1 & \text{w/ probability } x_s \\ 0 & \text{w/ probability } 1 - x_s \end{cases}$$

Thus the cost c for each round of picking elements is given by the following

$$c = E[\sum X_s] \tag{4}$$

For all elements

$$P(\text{element e is covered by set s}) = 1 - \prod_{s:e \in s} (1 - x_s) \tag{5}$$

Thus for each of the k sets in which e appears:

$$P(\text{element e is covered}) \geq 1 - \prod_{s:e \in s} (1 - x_s)^k \tag{6}$$

Given than x_s is some fractional value derived from the fractional values derived from our LP, we can rewrite the above equation as $(1 - \frac{1}{x_s})^k$. Since this function is increasing, we can upper bound it by the following: Let $y = k$ such that $y \geq$ the largest x_s

$$(1 - \frac{1}{y})^y = e^{ln(1-\frac{1}{y})^y} \tag{7}$$

$$ln(1 - \frac{1}{y})^y = yln(1 - \frac{1}{y}) = yln(\frac{y-1}{1}) \tag{8}$$

$$lim_{x \to \infty} yln(\frac{y-1}{1}) = \infty * 0 \tag{9}$$

$$\tag{10}$$

Applying L'Hopital's rule we get:

$$lim_{x \to \infty} - \frac{x}{x-1} = -1 \tag{11}$$

Plugging this back in we get the following:

$$(1 - \frac{1}{y})^y \leq e^{-1} \tag{12}$$

Therefore:

$$P(\text{element e is covered}) \geq 1 - \frac{1}{e} \tag{13}$$

So for a single iteration of rounding, we have a non-zero constant possibility that an element isn't covered in our rounding solution.

In order get a feasible solution we repeat this rounding process t times to get the following solution:

$$X = \bigcup^{t} X_i \tag{14}$$

where X_i is just the ith iteration of the rounding. Thus, in this case:

$$P(\text{element e is covered in the final solution}) \geq 1 - (\frac{1}{e})^t \tag{15}$$

Thus the total cost of our algorithm is the cost for each iteration time the number of iterations necessary to reduce the probability of not obtaining a set cover to a suitably small quantity.

If we want this to be an inverse polynomial likelihood that our solution is not feasible:

$$P(\text{every element covered}) \geq 1 - \frac{1}{e^{log(n)}} = 1 - \frac{1}{n} \tag{16}$$

Thus, if we apply $log(n^m)$ iterations, we get a polynomial time solution that outputs a feasible solution with $P \geq 1 - \frac{1}{n^x}$

3.2 Congestion Minimization

In the congestion minimization problem we are given the following: a graph G and a set of source destination pairs (s_i, t_i) Our goal is to calculate a path P_i for each source-destination pair such that the maximum congestion of any edge (i.e. the number of paths using a given edge) is minimized.

In this lecture we introduce the setup for this problem, discuss the creation of the separation oracle, and present an initial algorithm. Analysis of costs and feasibility will occur next lecture

For variable x_p which indicates whether path p_i from s_i to t_i is used, and p_i which is the set of all paths from s_i to t_i:

$$\sum_{p \in p_i} x_p \geq 1 \tag{17}$$

$$x_p \geq 0 \tag{18}$$

$$\sum_i \sum_{p \in p_i : e \in p} x_{p_{ij}} \leq \lambda \tag{19}$$

where λ is the maximum congestion

Since the number of variables is exponential, a separation oracle cannot be found in polynomial time. So instead, we convert this primal LP to a dual.

Dual: Variables: $\alpha_i = \sum_{p \in p_i} x_p$, $y_e = \sum_i \sum_{p \in p_i : e \in p} x_p \leq \lambda$

$$max \sum \alpha_i \tag{20}$$

$$\sum y_e \leq 1 \tag{21}$$

$$\alpha_i - \sum_{e \in p} y_e \leq 0 \tag{22}$$

We obtain our fractional LP solution through the following method: Given as solution with a set of values, we check if $\sum y_e \leq 1$. If not we have our separation oracle. Otherwise, we set y_e as the shortest path for each pair $(s_i, t_i$. If $\sum y_e \leq \alpha$ then that becomes our separation oracle.

Our Algorithm to round our fractional solution to an integral one is as follows: Assume an optimal solution to the fractional LP, obtained through the use of a separation oracle. Much like with set cover, we will round this solution to an integral one. For each (s_i, t_i) we pick path P_i with probability x_{p_i}, the value from our fractional solution.

The expected congestion on any one edge = $E[\sum_i \sum_{p \in p_i : e \in p} x_p]$.

Given that this is an optimal fractional solution, we know that:

$$E[\sum_i \sum_{p \in p_i : e \in p} x_p] \leq \lambda \tag{23}$$

This gives us $maxE[\lambda_e], e \in E$.

However, this isn't very informative. Consider the example of a graph where for a single edge, $\lambda_e = n$, but for all other edges $\lambda_e = 0$. In this case $maxE[\lambda_e]$ will be low, even though there is a significant amount of congestion on a single edge.

Instead, we want to find $E(max\lambda_e)$. We will cover how to do this using tail bounds in the next lecture.

4 Summary

In this lecture we introduced the concept of using randomized rounding to create an approximation of optimal integral solutions from the the optimal fractional solutions obtained through linear programming. This process consists of three basic steps: 1. Create an integer programming formulation of the problem. 2. Relax to a fractional solution, solved through the use of a linear program. 3. Use the fractional solution values to round to an integer solution.