Approximation Algorithms and Hardness of Approximation March 8, 2013

Lecture 6

Lecturer: Ola Svensson *Scribes: Christos Kalaitzis*

1 Introduction to Linear Programming Relaxations

The topic of our next discussion will be the use of Linear Programs for the design of good approximation algorithms for various combinatorial problems. These algorithms work by taking a solution to a relaxed version of our problem and rounding it to get a solution for its integral version. Such methods have played a central role in the theory of Approximation Algorithms; they offer a very natural and general framework to work with, and in many cases they have produced some of the most powerful results.

Our discussion will include an introduction to the use of Linear Programs for such problems, a brief overview of their history, an introduction to the notion of Integrality Gap, and some examples of how these techniques are used on some very well-known problems. Finally, we will conduct an introduction to the exploitation of the Extreme Point Structure of certain relaxations.

1.1 Basic Definitions and History of Linear Programming

Let us start with describing what a Linear Program is; generally, this term is used to describe the problem of optimizing a linear function over some linear constraints. More formally, a linear program of n variables over m constraints is expressed in the following form:

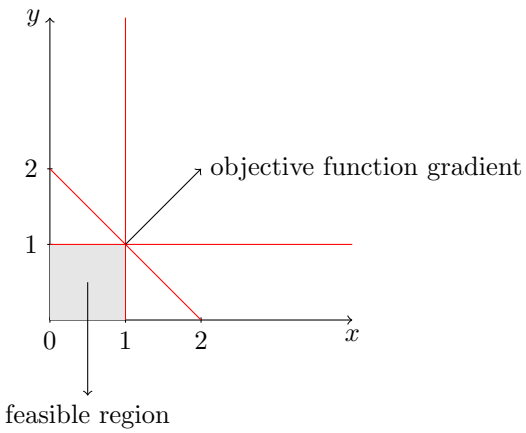
$$\begin{aligned} \text{minimize/maximize} \quad & \sum_{j=1}^{n} c_j x_j & \\ \text{s.t.} \quad & \sum_{j=1}^{n} a_{i,j} x_j \geq b_j & \forall i \in \{1...m\} \end{aligned}$$

We would like to use this tool to express various combinatorial problems; therefore, one natural way to do this is to correspond the variables of a linear program to some sort of decision variable which is related to our problem, and restrict these variables to be integers. The resulting program is called an Integer Linear Program, and is generally NP-hard to solve optimally:

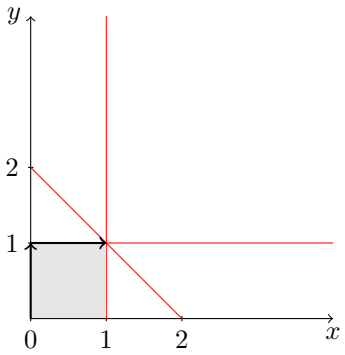
$$\begin{aligned} \text{minimize/maximize} \quad & \sum_{j=1}^{n} c_j x_j & \\ \text{s.t.} \quad & \sum_{j=1}^{n} a_{i,j} x_j \geq b_j & \forall i \in \{1...m\} \\ & x_j \in \{0,1\} & \forall j \in \{1...n\} \end{aligned}$$

Linear Programs were first formulated by the Russian scientist Leonid Vitaliyevich Kantorovich, who used it to optimize certain aspects of the Soviet economy. For this contribution, he was awarded the Nobel Prize in Economics at 1975.

The first efficient algorithm to solve Linear Programs was provided by American scientist George Dantzig, who published the famous simplex algorithm in the 1950's. The main idea of the algorithm relies upon the fact that the linear inequalities form a convex polytope:



The algorithm then proceeds to select one of its vertices and greedily walk towards another one, constantly trying to improve the objective function:



The algorithm is known to run in exponential time in the worst case; however, it is still the most practical algorithm for solving linear programs. Variants of the algorithm which choose the next vertex which will be visited in a random way, are known to run in at least $2^{\Omega(\sqrt{n})}$ time.

In the 1970-s, Leonid Khachiyan analyzed an algorithm which is known as the ellipsoid algorithm, and which is guaranteed to run in polynomial time(although it is not considered practical). The main idea of the algorithm is to iteratively reduce the area in which the feasible region may reside, until either a feasible point is found or the program is decided to be infeasible. Later, other practical algorithms like the interior-point methods, firstly introduced by Indian scientist Narendra Karmarkar, also achieved polynomial running time. In this case, the algorithm, instead of traveling along the vertices of the polytope, travels through the feasible region in order to find an optimum.

2 Linear Programming in Approximation Algorithms

A general framework to design approximation algorithms using linear programs can be described as follows:

(a) We begin by finding a formulation of our problem as an integer linear program.

(b) Then, we relax the integrality constraints, and solve the resulting linear program in polynomial time.

(c) Finally, we round the fractional solution to an integral one, hoping it will be good compared to the optimal one.

The first problem we will apply linear programming relaxations to is the Vertex Cover problem: given a graph $G = (V, E)$ and a weight function $w : V \to \mathbb{R}^+$, we would like to find a subset C of V such that every edge $e \in E$ is covered(i.e. $e \cap C \neq \emptyset$) and $w(C) = \sum_{v \in C} w(v)$ is minimized. Let us follow the general framework to derive a reasonable approximation algorithm:

(a) First, we should formulate the Vertex Cover problem as an integer linear program. For this purpose, we will define indicator variables x_v for all $v \in V$ such that $x_v = 1$ iff $v \in C$. Then, Vertex Cover can be formulated as such:

$$\begin{aligned} \text{minimize} \quad & \sum_{v \in V} w_v x_v & \\ \text{s.t.} \quad & x_u + x_v \geq 1 & \forall e = (u, v) \in E \\ & x_v \in \{0, 1\} & \forall v \in V \end{aligned}$$

(b) Afterwards, we should relax the integrality constraints, and solve the resulting linear program in polynomial time:

$$\begin{aligned} \text{minimize} \quad & \sum_{v \in V} w_v x_v & \\ \text{s.t.} \quad & x_u + x_v \geq 1 & \forall e = (u, v) \in E \\ & 0 \leq x_v \leq 1 & \forall v \in V \end{aligned}$$

Hence, we get a fractional solution $\mathbf{x}$ of value $\text{OPT}_f \leq \text{OPT}$(this is implied by the fact that the set of optimal integral solutions is a subset of the set of feasible points).

(c) Finally, we round the fractional solution $\mathbf{x}$ to an integral one $\mathbf{x}^*$: $C = \{v : x_v \geq \frac{1}{2}\}$.

Now, observe the following facts:

Fact 1 *C is a feasible solution to the Vertex Cover problem.*

Proof This is true since for every edge $e = (u, v) \in E$ we know that $x_u + x_v \geq 1$, which implies that at least one of x_u, x_v will be at least $\frac{1}{2}$. Hence, e will be covered by C. ■

Fact 2 *$w(C) \leq 2 \cdot OPT_f \leq 2 \cdot OPT$.*

Proof $w(C) = \sum_{v \in C : x_v \geq \frac{1}{2}} w(v) \leq 2 \cdot \sum_{v \in V : x_v \geq \frac{1}{2}} x_v w_v \leq 2 \cdot \text{OPT}_f \leq 2 \cdot \text{OPT}$. ■

It is easy to see that the above proof defines a general pattern for proving approximation guarantees for rounding algorithms; specifically, in order to achieve an α-approximate solution, it is sufficient to prove that our rounding returns a solution of cost at most $\alpha \cdot \text{OPT}_f$.

2.1 Integrality Gap

A key notion of linear programming relaxations is the concept of Integrality Gap. Given a problem as a set $\mathcal{I}$ of instances, the Integrality Gap is defined as

$$\sup_{I \in \mathcal{I}} \frac{\text{OPT}(I)}{\text{OPT}_f(I)}$$

for minimization problems and

$$\inf_{I \in \mathcal{I}} \frac{\text{OPT}(I)}{\text{OPT}_f(I)}$$

for maximization problems.

The integrality gap enforces a limit on the approximation power of our relaxation; specifically, using OPT_f as a bound towards which we will compare our algorithm, we cannot achieve an approximation algorithm with a guarantee better than the integrality gap.

In the case of Vertex Cover, our simple rounding algorithm is a proof that the integrality gap is at most 2; we would like to see if that bound is tight.

Theorem 3 *The integrality gap of Vertex Cover is at least* $2 - \frac{2}{n}$.

Proof Consider the complete graph on n vertices K_n. For this instance, the solution where for all $v \in V$ $x_v = \frac{1}{2}$ is feasible, returning a fractional optimum of $\text{OPT}_f = \frac{n}{2}$. However, the integral optimum is clearly $n - 1$. Hence, the integrality gap of Vertex Cover is at least $\frac{n-1}{\frac{n}{2}} = 2 - \frac{2}{n}$. ■

Finally, in order to complete the analysis of our simple rounding algorithm for Vertex Cover, observe that there is an instance which returns a solution of cost twice the optimum: that is the complete bipartite graph, in which case our algorithm, given a solution in which for all $v \in V$ $x_v = \frac{1}{2}$, will output the whole graph as a solution.

2.2 Set Cover and Randomized Rounding

Let us revisit the Set Cover problem: given a universe $U = \{e_1 ... e_n\}$, a family of subsets $T = \{S_1 ... S_m\}$, and a cost function $c : T \to \mathbb{R}^+$, we would like to find a collection of subsets that covers all the elements and has minimum cost. Let us use the general framework again to design an approximation algorithm:

(a) We begin by formulating Set Cover as an integer linear program, using indicator variables x_S for every $S \in T$, such that x_S iff S belongs to the set cover:

$$\begin{aligned} \text{minimize} \quad & \sum_{S \in T} c(S) x_S & \\ \text{s.t.} \quad & \sum_{S : e \in S} x_S \geq 1 & \forall e \in U \\ & x_S \in \{0, 1\} & \forall S \in T \end{aligned}$$

It is worth to note that the above LP formulation is generally known as a covering LP.

(b) We relax the integrality constraints, and solve the resulting linear program in polynomial time:

$$\begin{aligned} \text{minimize} \quad & \sum_{S \in T} c(S) x_S & \\ \text{s.t.} \quad & \sum_{S : e \in S} x_S \geq 1 & \forall e \in U \\ & 0 \leq x_S \leq 1 & \forall S \in T \end{aligned}$$

(c) In order to round the fractional solution $\mathbf{x}$, let us define the frequency f of an instance of Set Cover to be the maximum number of sets some element can be covered by. Then, the set cover the algorithm returns is the set $\{S : x_S \geq \frac{1}{f}\}$.

Using arguments similar to the ones we used for the Vertex Cover problem, one can prove that this algorithm is an f-approximation algorithm; moreover, if $f \in O(1)$, then this is the best guarantee we can hope for; on the other hand, we already know an $\ln n$-approximation algorithm for Set Cover, implying that if $f \in \omega(\ln n)$, this algorithm is inferior.

However, instead of this rounding, we would like to try an alternative method; specifically, we would like to retrieve a rounded solution using randomization. A first attempt could be to include every set S

in the Set Cover with probability x_S; then, the expected cost of the solution is $\mathbb{E}[\text{cost}] = \sum_{S \in T} c(S)x_S = \text{OPT}_f$. Of course, this approximation guarantee looks too good to be true: indeed, we have no guarantee that the solution that we will get this way will be feasible. In order to determine this, let's estimate the probability that each element will be covered by this algorithm.

Lemma 4 *The randomized rounding algorithm covers each element with probability at least* $1 - \frac{1}{e}$.

Proof Consider element a which can be covered by k sets, each of which is selected by the algorithm with probabilities $p_1 ... p_k$. Then, since $\sum_{1 \le i \le k} p_i \ge 1$,

$$\Pr[a \text{ is not covered}] = \prod_{1 \le i \le k} (1 - p_i) \le (1 - \frac{1}{k})^k \le \frac{1}{e}$$

Hence, $\Pr[a \text{ is covered}] \ge 1 - \frac{1}{e}$. ■

Therefore, we have a procedure which we can run and cover each element with some constant probability; all we have to do now is repeat the same procedure $d \log n$ times, for some appropriately chosen d, and output the union of all the set covers each procedure returned. This way, for every element $a \in U$ $\Pr[a \text{ is not covered}] \le (\frac{1}{e})^{d \ln n} \le \frac{1}{4n}$. Applying Union Bounds, we get that $\Pr[\text{every element is covered}] \ge \frac{3}{4}$, for a total expected cost of $\mathbb{E}[\text{cost}] = d \cdot \ln n \cdot \text{OPT}_f$.

Now, we can upper-bound the probability that we get a very bad solution; using Markov's inequality, we get that $\Pr[\text{cost} > 4 \cdot \text{OPT}_f \cdot d \cdot \ln n] \le \frac{1}{4} \implies \Pr[\text{cost} \le 4 \cdot \text{OPT}_f \cdot d \cdot \ln n] \ge \frac{3}{4}$.

Now, we have two undesirable events: the solution might be infeasible, or the solution might have extremely high cost. The probability of both these events is upper-bounded by $\frac{1}{4}$, therefore the probability of their union is uper-bounded by $\frac{1}{2}$, which leads to the following theorem:

Theorem 5 *With probability at least* $\frac{1}{2}$*, the above algorithm using* $d \cdot \ln n$ *repetitions of the simple randomized rounding algorithm will return a* $4 \cdot d \cdot \ln n$*-approximate solution.*

Repeating the above procedure until both conditions are met will eventually return a solution with reasonable cost. However, as one might have already noticed, this algorithm has no guarantee that it will terminate in polynomial time, although its expected running time is polynomial; specifically, this algorithm belongs to the complexity class ZPP, which consists of all languages which can be decided in expected polynomial time.

As for the integrality gap of the above relaxation, this algorithm shows that it is upper bounded by $4 \cdot d \cdot \ln n$. On the other hand, we will state the following theorem without proof:

Theorem 6 *The integrality gap of the LP-relaxation for Set Cover we described is*

$$\Omega(\ln n) \le \sup_G \frac{OPT(G)}{OPT_f(G)} \le 4 \cdot d \cdot \ln n$$

3 Extreme Point Structure

Next, we will discuss how we are able to exploit the special structure some linear programs have. The focus of our attention will be the so-called extreme points:

Definition 7 *An extreme point solution of a linear program is a feasible point that cannot be expressed as the convex combination of other feasible points.*

Fact 8 *If the feasible region defined by a linear program is bounded, there always exists an optimum which is an extreme point.*

In order to exhibit the use of extreme points, let us return to the Vertex Cover problem and the relaxation we used previously:

$$\begin{aligned} \text{minimize} \quad & \sum_{v \in V} w_v x_v & \\ \text{s.t.} \quad & x_u + x_v \geq 1 & \forall e = (u,v) \in E \\ & 0 \leq x_v \leq 1 & \forall v \in V \end{aligned}$$

Then, we are able to show that every extreme point solution of this LP has certain properties:

Lemma 9 *Every extreme point solution to the Vertex Cover LP is half-integral, i.e. for all $v \in V$, $x_v \in \{0, \frac{1}{2}, 1\}$.*

Proof Contrary to what we want to prove, assume towards contradiction that there is an extreme point solution which is not half-integral. Then, let $V^+ = \{v \in V : \frac{1}{2} < x_v < 1\}$ and $V^- = \{v \in V : 0 < x_v < \frac{1}{2}\}$. By our assumption, $V^+ \cup V^- \neq \emptyset$. Consider a pair of vertices u, v such that $x_u + x_v = 1$. Then, $x_u \in V^+$ implies that $x_v \in V^-$.

Given a solution $\mathbf{x}$ to the LP, consider the following solutions $\mathbf{y}, \mathbf{v}$ for every vertex $v \in V$:

$$y_v = \begin{cases} x_v + \epsilon & \text{if } v \in V^+ \\ x_v - \epsilon & \text{if } v \in V^- \\ x_v & \text{otherwise} \end{cases}$$

$$z_v = \begin{cases} x_v + \epsilon & \text{if } v \in V^- \\ x_v - \epsilon & \text{if } v \in V^+ \\ x_v & \text{otherwise} \end{cases}$$

Then, $\mathbf{x} = \frac{\mathbf{y}+\mathbf{z}}{2}$, which means that $\mathbf{x}$ is the convex combination of $\mathbf{y}$ and $\mathbf{z}$. All we need to prove in order to get a contradiction is that $\mathbf{y}$ and $\mathbf{z}$ are feasible. By selecting ϵ to be small enough, we can concern ourselves only with the edges $e = (u, v)$ for which it holds $x_u + x_v = 1$. Then, we can distinguish three cases:

- $x_v = \frac{1}{2}$: then, $x_u = \frac{1}{2}$ and therefore $y_u + y_v = x_u + x_v = 1$ and $z_u + z_v = x_u + x_v = 1$
- $x_v > \frac{1}{2}$: then, $x_u < \frac{1}{2}$ and therefore $y_u + y_v = x_u - \epsilon + x_v + \epsilon = 1$ and $z_u + z_v = x_u + \epsilon + x_v - \epsilon = 1$
- $x_v < \frac{1}{2}$: then, $x_u > \frac{1}{2}$ and $y_u + y_v = x_u + \epsilon + x_v - \epsilon = 1$ and $z_u + z_v = x_u - \epsilon + x_v + \epsilon = 1$

Therefore, $\mathbf{y}$ and $\mathbf{z}$ are feasible and we have acquired a contradiction. ■

An interesting application of this property is the following:

Theorem 10 *Given a k-coloring of a graph G, we can find a vertex cover which is a $2 - \frac{2}{k}$ approximation.*

Proof Given a solution $\mathbf{x}$ to the Vertex Cover LP, consider the sets $V_0 = \{v \in V : x_v = 0\}$, $V_{\frac{1}{2}} = \{v \in V : x_v = \frac{1}{2}\}$ and $V_1 = \{v \in V : x_v = 1\}$. Then, there are no edges between two vertices in V_0, or between a vertex in V_0 and a vertex in $V_{\frac{1}{2}}$. Furthermore, we know that $V_{\frac{1}{2}}$ can be colored with k colors, which means that there are k independent sets in $V_{\frac{1}{2}}$. Let I be the independent set of $V_{\frac{1}{2}}$ which has the maximum sum of weights. Rounding down x_v for every $v \in I$, and rounding up x_v for every $v \notin I$, will give us a feasible solution of cost at most $2 \cdot \sum_{v \in V_{\frac{1}{2}} \setminus I} w_v + \sum_{V_1} w_v \leq 2(1 - \frac{1}{k}) \cdot \sum_{v \in V_{\frac{1}{2}}} w_v + \sum_{V_1} w_v \leq 2(1 - \frac{1}{k}) \cdot \text{OPT}_f$, which is $2(1 - \frac{1}{k})$-approximate solution. ■