Topics in Theoretical Computer Science April 08, 2014

Lecture 8

Lecturer: Ola Svensson *Scribes: David Leydier and Samuel Grütter*

1 Introduction

In this lecture we will introduce Linear Programming.
It was first formalized and applied to problems in economics in the 1930s by Leonid Kantorovich. Kantorivich's work was hidden behind the Iron Curtain (where it was largely ignored) and therefore unknown in the West. Linear programming was rediscovered and applied to shipping problems in the early 1940s by Tjalling Koopmans.

2 Linear Programming

We begin by giving a formal definition to a Linear Program (LP).

Definition 1 *A linear programming problem is the problem of finding values for n variables $x_1, x_2, \ldots, x_n$ that minimize (or equivalently, maximize) a given linear objective function, subject to m linear constraints*

$$\begin{aligned} \textbf{minimize:} \quad & \sum_{i=1}^{n} c_i x_i & \\ \textbf{Subject to:} \quad & \sum_i e_{i,j} x_i = b_j & \text{for } j = 1, \ldots, m_1 \\ & \sum_i d_{i,k} x_i \geq g_k & \text{for } k = 1, \ldots, m_2 \\ & \sum_i f_{i,p} x_i \leq l_p & \text{for } p = 1, \ldots, m_3 \end{aligned}$$

where $m_1 + m_2 + m_3 = m$.

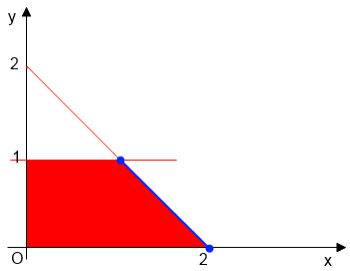
Definition 2 *An extreme point is a feasible point that cannot be written as a convex combination of other feasible points.*

Remark If the region defined by the program constraints is bounded then there always exists an optimum which is an extreme point.

Example: Let's take two variables $x, y \in \mathbb{R}$. We define the following linear program:

$$\begin{aligned} \textbf{Maximize:} \quad & x + y \\ \textbf{Subject to:} \quad & x + y \leq 2 \\ & y \leq 1 \\ & x, y \geq 0 \end{aligned}$$

These constraints define the physical region in red, bounded, with four extreme points. In this case, the whole segment in blue is solution of the problem, so the two extreme points in blue are both OPT solutions.



2.1 Resolution Methods

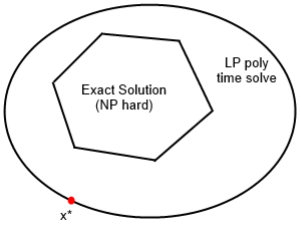
Simplex Method was published by George Dantzig in 1947: it is the first complete algorithm to solve linear programming problems.
The principle is the following: we start from an extreme point and then we look at its neighbors. If one of these is better we move on it and continue in the same way, else we stop. Once we stop, we can be sure that we have an optimal solution, since we're in a convex polytope. Even if it's usually extremely fast, we know some bad examples where this method visits an exponential number of extreme points before to reach a solution, so this method does not always run in polynomial time.
Ellipsoid Method was studied by Leonid Khachiyan in the seventies.
This method is guaranteed to run in poly time (exactly $poly(n, m, \log(u))$ where u is the largest constant) but it is slow in practice.
We do a binary search for the optimal value of the objective function, so we can add the objective function as a constraint, and just need to decide if there exists a feasible point. We start by taking an ellipsoid surrounding the feasible area. We then check the center of the ellipsoid, if it's inside the area then we have our solution, else we identify a violated constraint, and cut our ellipsoid in two parts using this constraint, and construct a new ellipsoid around the part of the old ellipsoid in which the constraint is satisfied. We repeat this process until we find a feasible point or can be sure that the feasible area is empty.
Interior point Method developed by Narendra Karmarkar in 1984.
In this Method we move in the region to find an OPT solution.

Remark In what follows, the different LP used are solved using a polynomial-time algorithm, i.e. we can get the optimal value in polynomial time.

3 How to use Linear Programming

So, now we want to use Linear Programming to solve some combinatorial problems. In that way, we will use the following general recipe:

1. Formulate the problem as Integer LP – usually with binary variables ($x_i \in \{0, 1\}$).
2. Relax to LP – $x_i \in [0, 1]$



3. Solve LP to get an OPT solution x^*. Then round x^* to an integral solution "without loosing too much".

3.1 Min st-cut

Given: $G = (V, E)$ and two vertices $s, t \in V$
Output: $S \subseteq V$ such that $s \in S$, $t \notin S$ and $|E(S, \overline{S})|$ is minimized.

We now define a corresponding integer linear program (ILP).
We define a variable x_v:

$$x_v = \begin{cases} 1 & \text{if } v \in S \\ 0 & \text{otherwise} \end{cases}$$

We need to define the objective function and the constraints. On the one hand, if an edge $(u, v) \in E$ is cut, then one of its vertices has the value 1 and the other 0. On the other hand, if an edge isn't cut then the two vertices have the same value. So we can deduce our objective function:

$$\min\Big(\sum_{(u,v)\in E} |x_u - x_v|\Big)$$

With two constraints: $x_s = 1$ and $x_t = 0$.
As we want a linear objective function, we have to modify it using the following lemma.

Lemma 3 $\min|a - b| \Leftrightarrow \min(y)$ *where* y *verifies* $y \geq a - b$ *and* $y \geq b - a$

So we obtain this new program:

$$\min \sum_{e\in E} y_e$$

With constraints: $x_s = 1$, $x_t = 0$ and

$$\left.\begin{array}{rcl} y_e & \geq & x_u - x_v \\ y_e & \geq & x_v - x_u \end{array}\right\} \forall e = (u, v) \in E$$

Last we have to relax to LP:

$$0 \leq x_v \leq 1 \quad \forall v \in V \setminus \{s, t\}$$

Now we can run the following algorithm:

1. Solve LP to obtaine an OPT solution (x^*, y^*).
2. Select a threshold $\theta \in [0, 1]$ uniformly at random.

3. Output $S = \{v | x_v \geq \theta\}$.

$$\Pr_\theta[e = (u,v) \text{ is cut}] = |x^*_u - x^*_v| \leq y^*_e$$

$$\begin{array}{rcl} \mathbb{E}[\sharp \text{ edges cut}] & = & \sum_{e \in E} \Pr_\theta[e \text{ is a cut}] \\ & \leq & \sum_{e \in E} y_e \\ & = & OPT_{LP} \text{ value} \\ & \leq & OPT \text{ value} \end{array}$$

As OPT is the optimal solution – so we can't find a better one (all $\leq$ here could be replaced by $=$). The solution found in this case is not just the OPT for the ILP, but the OPT solution for the general problem. When we've done the relaxation, we didn't loose any information. One can see that an integer solution for the above LP is an extreme point.

Exercise 1 For which weights on edges does the st-cut LP give an optimal solution ?

Solution Our new objective function is:

$$\min(\sum_{(u,v) \in E} w_e |x_u - x_v|)$$

As we need to linearize it, if the weight is positive then we can do it as we've done before. But if the weights are negative we're not able to model the min.

4 Vertex Cover

Now let us apply this technique to get an approximation of the *Vertex Cover* problem. We will use a generalized version of the Vertex cover problem, where nodes have weights:

Definition 4 (VC: Vertex Cover Problem) *Given a graph $G = (V, E)$ and a weight function for the vertices $w : V \to \mathbb{R}_+$, output a set $C \subseteq V$ of minimum weight such that for all $\{u, v\} \in E$, $u \in C$ or $v \in C$.*

First, let us define an ILP solving VC: For all $v \in V$, we introduce a variable x_v, which is 1 if $v \in C$, and 0 otherwise. The objective function is

$$\min \sum_{v \in V} w(v) x_v$$

and for each $\{u, v\} \in E$, we introduce the constraint $x_u + x_v \geq 1$. This ensures that for each edge, at least one endpoint is in C. Additionally, we require that $\forall v \in V,\ x_v \in \{0, 1\}$.

Second, we relax the ILP to an LP by allowing that $x_v \in [0, 1]$. Actually, it's sufficient to require that $x_v \geq 0$, because having an x_v greater than 1 will not satisfy any additional constraint, but only increase the value of the objective function, so this will not happen.

Third, we have to do the rounding: Suppose we've solved the LP and got an optimal solution x^*. We will return $C = \{v : x^*_v \geq \frac{1}{2}\}$ as our solution to VC.

Claim 5 *C is a feasible solution.*

Proof Consider any edge $\{u, v\}$ and its constraint. Since $x_u^* + x_v^* \geq 1$, at least one of x_u^*, x_v^* is $\geq \frac{1}{2}$ and thus in C. ■

Claim 6 *The weight of C is at most twice the value of the optimal solution of VC.*

Proof We have

$$\sum_{v \in C} w(v) = \sum_{v \in V : x_v^* \geq \frac{1}{2}} w(v) \leq \sum_{v \in V : x_v^* \geq \frac{1}{2}} 2x_v^* w(v) \leq \sum_{v \in V} 2x_v^* w(v) = 2 \sum_{v \in V} x_v^* w(v) = 2LP_{OPT} \leq 2VC_{OPT}$$

where the first inequality holds because $2x_v^* \geq 1$. ■

So, we have found a way to approximate VC withing a factor of 2.

5 Integrality Gap

The notion of the *integrality gap* allows us to bound the power that linear programming approximations can have. Let $\mathcal{I}$ be the set of all instances of a given problem. In the case of a minimization problem, the integrality gap g is defined as

$$g = \max_{I \in \mathcal{I}} \frac{OPT(I)}{OPT_{LP}(I)}$$

As an example, suppose $g = 2$, and that LP found that $OPT_{LP} = 70$. Then, since our problem instance might be the one which maximizes the expression for g, all we can guarantee is that $OPT(I) \leq 2 \cdot OPT_{LP}(I) = 140$, so it's not possible to find a linear programming approximation algorithm that approximates better than within a factor of $g = 2$.

5.1 Integrality Gap for Vertex Cover

Claim 7 *The integrality gap for vertex cover is at least $2 - \frac{2}{n}$.*

Proof Consider the complete graph on n vertices. We have $OPT = n - 1$, because if we there are 2 vertices that we don't choose, the edge between them is not covered. $LP_{OPT} \leq \frac{n}{2}$, because assigning $\frac{1}{2}$ to each vertex is a feasible solution of cost $\frac{n}{2}$, so the optimum can only be smaller. Now

$$g \geq \frac{n-1}{\frac{n}{2}} = 2 - \frac{2}{n}$$

■

6 Set Cover

Let us apply this technique to the *Set Cover* problem. It can be seen as a generalization of the vertex cover problem: Instead of having edges each containing exactly two vertices, we now have sets, each containing any number of vertices.

Definition 8 (Set Cover Problem) *Given a universe $U = \{e_1, e_2, \ldots e_n\}$, and a family of subsets $T = \{S_1, S_2, \ldots S_m\}$ and a cost function $c : T \to \mathbb{R}_+$, find a collection C of subsets of minimum cost that cover all elements.*

Exercise 2 Formulate an LP for Set Cover and round it as good as you can.

Solution For each $i \in \{1, \dots m\}$, define x_i, which is 1 if $S_i \in C$, and 0 otherwise. The objective function is

$$\min \sum_{i=1}^{m} x_i \cdot c(S_i)$$

and for each element $e \in U$, we add the constraint $\sum_{S_i \,:\, e \in S_i} x_i \geq 1$. This ensures that each element is covered by at least one set in C. And for each x_i, we require that $x_i \in \{0, 1\}$ in the ILP, and $x_i \in [0, 1]$ in the LP.

Now suppose that each element belongs to at most f sets. Then we can do the following rounding: $C = \{S_i : x_i^* \geq \frac{1}{f}\}$. In each constraint, there's at least one x_i^* which is at least $\frac{1}{f}$, so each constraint is satisfied. Using the same reasoning as in the proof of claim 6, we can show that this approximation is within a factor of f.

6.1 A better approximation for Set Cover

If we introduce randomness and allow our algorithm to output non-feasible solutions with some small probability, we can get much better results (in expectation).

We use the same LP as in the solution of exercise 2, and will run the following algorithm:

1. Solve the LP to get an optimal solution x^*.
2. Choose some positive integer constant d (we will see later how d affects the guarantees we get). Start with an empty result set C, and repeat step 3 $d \cdot \ln(n)$ times.
3. For $i = 1, \dots m$, add set S_i to the solution C with probability x_i^*, choosing independently for each set.

Now let us analyze what guarantees we can get:

Claim 9 *The expected cost of all sets added in one execution of step 3 is*

$$\sum_{i=1}^{m} x_i^* c(S_i) = LP_{OPT}$$

Proof

$$\mathbb{E}[\text{rounded cost}] = \sum_{i=1}^{m} c(S_i) \Pr[S_i \text{ is added}] = \sum_{i=1}^{m} c(S_i) x_i^* = LP_{OPT}$$

■

From this, we can immediately derive

Corollary 10 *The expected cost of C after $d \cdot \ln(n)$ executions of step 3 is at most*

$$d \cdot \ln(n) \cdot \sum_{i=1}^{m} c(S_i) x^* \leq d \cdot \ln(n) \cdot LP_{OPT} \leq d \cdot \ln(n) \cdot OPT$$

Note that we have $LP_{OPT} \leq OPT$ because LP is a relaxation of the original problem, so its optimum can only be better.

That sounds good, but we should also worry about feasibility:

Claim 11 *The probability that a constraint remains unsatisfied after an execution of step 3 is at most $\frac{1}{e}$.*

Proof Suppose our constraint contains k variables, and let us write it as $x_1 + x_2 + \cdots + x_k \geq 1$. Then,

$$\begin{aligned}
\Pr[\text{constraint unsat.}] &= \Pr[S_1 \text{ not taken}] \ldots \Pr[S_k \text{ not taken}] \\
&= (1 - x_1^*) \ldots (1 - x_k^*) \\
&\leq \left(1 - \frac{1}{k}\right)^k \qquad (1) \\
&\leq \frac{1}{e} \qquad (2)
\end{aligned}$$

where (1) follows from the fact that $\sum_i x_i^* \leq 1$ and (2) from the inequality $1 - x \leq e^{-x}$. ■

Claim 12 *The output C is a feasible solution with probability at least $1 - \frac{1}{n^{d-1}}$.*

Proof Using claim 11, we find that the probability that a given constraint is unsatisfied after $d \cdot \ln(n)$ executions of step 3 is at most

$$\left(\tfrac{1}{e}\right)^{d \cdot \ln(n)} = \frac{1}{n^d}$$

and by union-bound, the probability that there exists any unsatisfied constraint is at most

$$n \cdot \frac{1}{n^d} = \frac{1}{n^{d-1}}$$

■

Now we have an expected value for the cost, and also a bound on the probability that an unfeasible solution is output, but we still might have a bad correlation between the two: It could be that all feasible outputs have a very high cost, and all unfeasible outputs have a very low cost.

To reduce this risk, we add the following claim:

Claim 13 *The algorithm outputs a feasible solution of cost at most $4d\ln(n)OPT$ with probability greater than $\frac{1}{2}$.*

Proof Let μ be the expected cost, which is $d\ln(n) \cdot OPT$ by corollary 10. We can upper-bound the bad event that the actual cost is very high: By Markov's inequality, we have $\Pr[cost > 4\mu] \leq \frac{1}{4}$. The other bad event that we have to upper bound is that the output is unfeasible, and by claim 12, we know that this happens with probability at most $\frac{1}{n^{(d-1)}} \leq \frac{1}{n}$. Now in the worst case, these two bad events are completely disjoint, so the probability that no bad event happens is at least $1 - \frac{1}{4} - \frac{1}{n}$, and if we suppose that n is greater than 4, this probability is indeed greater than $\frac{1}{2}$. ■

Note that this approximation does not only give us a general approximation factor, but it also allows better per-instance guarantees. Suppose we have an instance where $LP_{OPT} = 100$, and our algorithm found a solution of cost 110. Since we know that $LP_{OPT} \leq OPT$, we can say that our solution on this instance is at most 10% away from the optimal solution.