COMPSCI 330: Design and Analysis of Algorithms 11/6/2014

Lecture #20

Lecturer: Debmalya Panigrahi *Scribe: Yilun Zhou*

1 Overview

This lecture introduces linear programming (LP), a formulation for a large set of problems. The definition of linear programming is given, several examples presented, geometric interpretations given, and algorithms for solving linear programming briefly talked about.

2 Introductory Example

Instead of present formally the definition of linear programming, we introduce the general idea of linear programming through an example.

2.1 Scenario

Suppose a political election candidate want to win an election. The voters are divided into three sections, R_1, R_2, R_3 and in order to win the election, 5000 voters need to vote for him in R_1, 2000 in R_2, and 1000 in R_3. To win the support of voters, he can conduct three kinds of campaign, A, B, C. Note that campaigns are conducted over all three regions and there is no way of targeting a specific kind of campaign to a specific region. While campaigns can generally win support from voters, they can lose support (e.g. by advertising a perspective that voters from a particular region dislike). The following table summarizes the effect on voter support of spending 1$ on each type of campaign.

	A	B	C
R_1	-2	0	5
R_2	1	0	-1
R_3	1	2	3

2.2 Inequality Notation

The above table means that spending \$1 on campaign A will lose two voters from R_1, spending \$1 on campaign B will have no influence on voters from R_1, spending \$1 on campaign C will gain five voters from R_1, etc.

We can write this table as a system of linear equations. Specifically, let x_A, x_B, x_C be the amount of money spent on three campaigns. Then we have:

$$\begin{aligned}-2x_A + 0x_B + 5x_C &\geq 5000\\ 1x_A + 0x_B - 1x_C &\geq 2000\\ 1x_A + 2x_B + 3x_C &\geq 1000\end{aligned}$$

From a practical viewpoint, we cannot spend negative amount of money on each campaign. So additionally we have:

$$\begin{aligned}x_A &\geq 0\\ x_B &\geq 0\\ x_C &\geq 0\end{aligned}$$

In addition, our goal is to minimize the total amount of money spent, which is equivalent to:

$$\min(x_A + x_B + x_C)$$

3 LP Formulation

3.1 Matrix Notation

The above example can be made even more succinct in matrix form. Specifically, the 6 constraints (3 required by problem, 3 by common sense) and the optimization objective can be written as:
Maximize:

$$\begin{pmatrix}1 & 1 & 1\end{pmatrix}\begin{pmatrix}x_A\\ x_B\\ x_C\end{pmatrix}$$

s.t.

$$\begin{bmatrix}-2 & 0 & 5\\ 1 & 0 & -1\\ 1 & 2 & 3\\ 1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix}\begin{pmatrix}x_A\\ x_B\\ x_C\end{pmatrix} \geq \begin{pmatrix}5000\\ 2000\\ 1000\\ 0\\ 0\\ 0\end{pmatrix}$$

”s.t.” stands for ”subject to (constraints)” and is used widely in optimization problem formulation.

3.2 LP General Matrix Form

The above set of equations can be generalized to:
Maximize/minimize:

$$c \cdot \mathbf{x}$$

s.t.

$$A\mathbf{x} \geq \mathbf{b}$$

In a problem with m constraints (including both constraints enforced by problem and common sense) and n variables, A is a $m \times n$ matrix denoting the coefficients of the constraints; $\mathbf{x}$ is a $n \times 1$ column vector of variables; $\mathbf{b}$ is a $m \times 1$ column vectors specifying the numerical value of each constraint; $\mathbf{c}$ is a $1 \times n$ row vector representing the coefficients of variables in optimization objective.

4 Examples of LP Formulation

In this section we introduce many examples that can be formulated in LP. For simplicity we are going to present constraints in systems of equations instead of matrix form, though it is important to realize that these two forms can be interchanged easily and it is usually the matrices of coefficients that are fed to LP solvers.

4.1 Shortest Path

Given graph $G(V,E)$ with edge $(u,v) \in E$ having length l_{uv}, find the shortest path from $s \in V$ to $t \in V$.
Let d_v denote the distance of a given path from s to v. Thus, we have:
Minimize d_t s.t.

$$\begin{aligned} &d_s = 0 \\ &d_v \leq d_u + l_{uv} \ \forall (u,v) \in E \end{aligned}$$

Note that in the above example d_v is variable and l_{uv} is constant. The first equation gives the distance from s to s, which is 0. The second equation is actually a set of $|E|$ equations and for each edge there should be one equation. It says that the distance of v should never be greater than the distance of u plus the length of (u,v), or otherwise you can just take the edge to go from u to v and decreases the distance of v.
However, the above equations are not complete and it is relatively easy to identify a trivial solution: $d_v = 0 \ \forall v \in V$. Having a complete formulation of shortest path problem is not discussed in the lecture because it can be very tricky. Next we present several other problems whose complete LP formulation can be easily derived.

4.2 Maximum Flow

Let f_{uv} (this is our variable) denote the flow on $(u,v) \in E$ and c_{uv} (this is our constant) be the capacity of the edge (u,v). We want to find the maximum flow between s and t. Thus, we want:
Maximize:

$$\sum_u f_{su} - \sum_v f_{vs}$$

s.t.

$$\begin{aligned} \sum_u f_{uv} - \sum_w f_{vw} &= 0 \ \forall v \neq s,t \\ f_{uv} &\leq c_{uv} \ \forall (u,v) \in E \\ f_{uv} &\geq 0 \ \forall (u,v) \in E \end{aligned}$$

The objective function maximizes the net flow (flow in - flow out) at s. The first equation (which should be expanded into $|V|-2$ equations) guarantees that each vertex other than s,t should have 0 net flow. The second equation (which should be expanded into $|E|$ equations) guarantees that the edge capacities are not exceeded. The third equation (which again should be expanded into $|E|$ equations) guarantees that each edge should have positive flow.

4.3 Vertex Cover

Given $G(V,E)$, find a subset $V' \subseteq V$ of minimum size such that $\forall (u,v) \in E, u \in V' \vee v \in V'$.
Define

$$x_v = \begin{cases} 1 \text{ if } v \in V' \\ 0 \text{ if } v \notin V' \end{cases}$$

Thus, the LP formulation is:
Minimize

$$\sum_{v \in V} x_v$$

s.t.

$$\begin{aligned} x_u + x_v \geq 1 \ \forall (u,v) \in E \\ x_v \in \{0,1\} \ \forall v \in V \end{aligned}$$

The first equation guarantees that at least one of u,v needs to be in V' and the second constraint specifies that x_v must be an integer of 0 or 1.
Strictly speaking, the second equation makes this problem a special kind of linear programming, integer linear programming (ILP). ILP problems have the additional constraint that variables need to be integers. This constraint is actually a very strong one as it turns LP from efficiently solvable (in polynomial time) to efficiently unsolvable (in exponential time, as of now).

4.4 Minimum Spanning Tree

Given $G(V < E)$ with edge length l_e, find the minimum spanning tree T.
Again, we define something similar:

$$x_e = \begin{cases} 1 \text{ if } e \in E \\ 0 \text{ if } e \notin E \end{cases}$$

Thus, we have our formulation:
Minimize:

$$\sum_{e \in E} l_e x_e$$

s.t.

$$\begin{aligned} \sum_{e \in (S,\bar{S})} x_e \geq 1 \ \forall S \subset V \\ x_e \in \{0,1\} \ \forall e \in E \end{aligned}$$

The first equation uses the property that in a spanning tree, there must be at least 1 cut between any subset of vertices and its complement. The second constraint similarly renders this problem as an ILP problem.

4.5 Minimum Cut

Given $G(V,E)$, source s, sink t, find the minimum cut such that s and t are on different sides. For simplicity we assume unit capacity. So the size of the cut is just the number of edges. We define:

$$x_e = \begin{cases} 1 \text{ if } e \in (S,\bar{S}) \\ 0 \text{ if } e \notin (S,\bar{S}) \end{cases}$$

Thus, we have:
Minimize

$$\sum_{e \in E} x_e$$

s.t.

$$\sum_{e \in p} x_e \geq 1 \ \forall \text{path } p \text{ from } s \text{ to } t$$
$$x_e \in \{0,1\} \ \forall e \in E$$

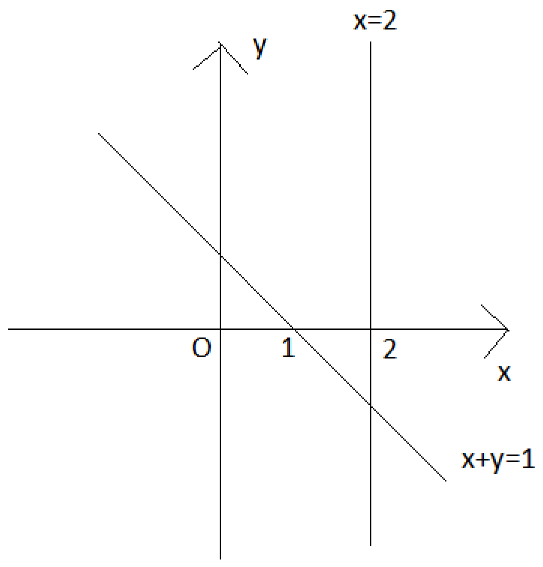
The first equation guarantees that for *any* path from s to t, at least one edge belongs to the cut and thus its removal will cause break the path. The second constraint makes this problem an ILP problem.

5 Geometric Interpretation of LP

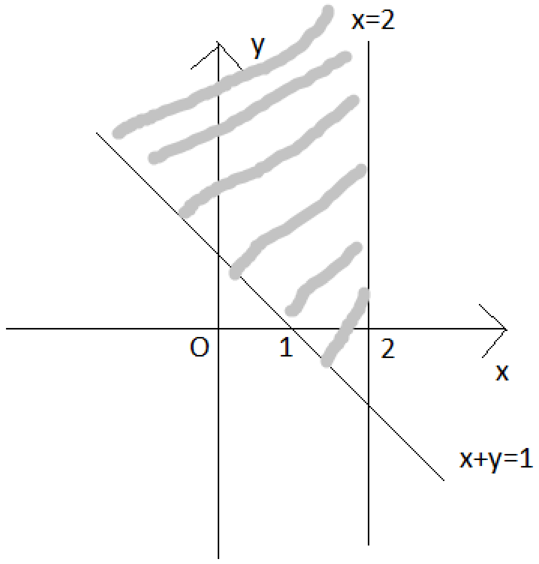
In this section we give an intuitive explanation of what LP is doing. For ease of illustration, we restrict ourselves to two variables x and y. Extending into more variables requires no technical difficulties but becomes hard to visualize.
Assume our LP formulation is: Minimize y s.t.

$$x + y \geq 1$$
$$x \leq 2$$

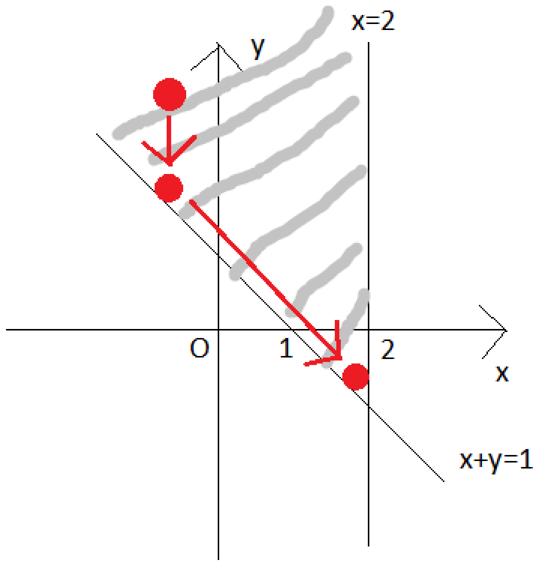
To visualize this problem, we first set up a rectangular coordinate and plot $x+y=1$ and $x=2$. So we have:



The constraint $x+y \geq 1$ restricts our attention to the upper half of the slanted line. The constraint of $x \leq 2$ restricts us to only consider the left half of the vertical line. Thus, taking the intersection of the two, the region of constraint satisfaction is shaded.



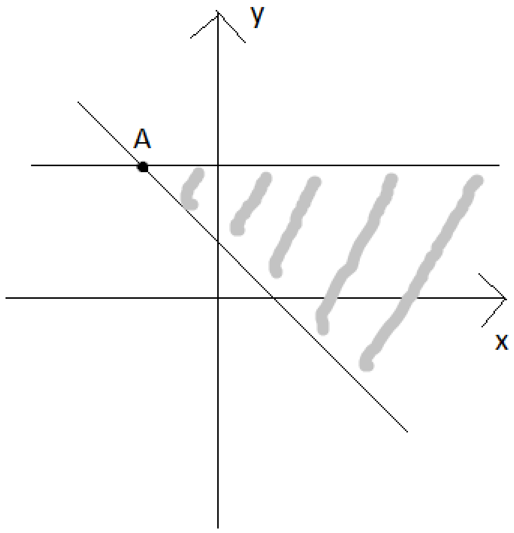
Then assume we place a ball in the shaded area. The optimization objective function acts like gravity that pulls the ball to a more preferable position, until hitting constraint limit. In case of minimizing y, the gravity is downward (because a lower y value is better than a higher one). Thus, the ball will be pulled to the bottom of the region of satisfaction and the solution is $(2,-1)$.



Next, we use the following constraint to illustrate several points:

$$\begin{aligned} x+y &\geq 1 \\ y &\leq 2 \end{aligned}$$

The constraint satisfaction region is shown below:



If the optimization objective is to minimize x, then the "gravity" is pointing leftward and the ball will be attracted to point A.
If the optimization objective is to minimize $x - y$, then the "gravity" is pointing upper left (because travelling in this direction x decreases and y increases). Thus, the ball will also be attracted to point A.
If the optimization objective is to maximize y, then the "gravity" is pointing upward and after the ball hits the ceiling of $y = 2$, it can drift left or right without changing the the value of the objective function. Thus, the solution is not unique.
If the optimization objective is to maximize x, then the "gravity" is pointing rightward and the ball can actually travel as far as it like. Thus, the solution is infinity.

6 Algorithms for Solving LP

There are several algorithms to solve LP but the details of all of them are beyond the scope of this course. Thus, we will only introduce the high-level approach of them.

6.1 Simplex Algorithm

Simplex algorithm is the oldest LP algorithm but it is still widely used. It has worst-case exponential running time but in practice it finishes pretty fast.
Simplex algorithm begins by constructing a convex polygon that is similar to what we have in the previous section (if there are enough constraints, the constraint satisfaction region will be a convex polygon of finite area. Then, it starts from any vertex and determine the objective function value at this vertex and neighboring vertices. If a neighboring vertex has a higher (or lower, depending on maximizing or minimizing) value, it moves to that vertex. This process terminates until it reaches a vertex with highest (or lowest) value.

6.2 Ellipsoid Algorithm

Ellipsoid algorithm was invented by Khachiyan in 1979. It is the first polynomial time algorithm for solving LP, though it is only weakly polynomial. It iteratively builds ellipses of smaller and smaller size to approach the convex polygon.

6.3 Interior Point Algorithm

Interior point algorithm was invented by Karmarkar in 1983. Although it is still weakly polynomial, it has much better performance than ellipsoid algorithm and is often used by LP solvers along with simplex algorithm.

7 Summary

This lecture presents linear programming. The focus on this lecture is to transform a given problem into an LP formulation, instead of algorithms that solve LP. The formal definition of LP is given, several examples worked out, a geometric explanation illustrated, and several algorithms mentioned. LP is still a field full of unknowns and the existence of strongly polynomial algorithm is a major open question in computer science.