Khoury College of Computer Sciences
Northeastern University
CS 7870: Seminar in Theoretical Computer Science

Fall 2023
22 September 2023
Scribe: Rajmohan Rajaraman

Lecture Outline:

- Linear Programming: An Overview
- Linear Programming: The Forms
- Linear Programming: Applications
- Linear Programming: Rounding

1 Introduction

We define the linear programming (LP) problem of minimizing a linear function subject to linear inequality constraints. We then present the general, standard, and canonical forms of the linear programming problem in summation and matrix form. We will then review some applications of linear programming and the technique of rounding linear programs for solving combinatorial optimization problems. There are many excellent expositions for linear programming and its applications, including [Kar91, Goe94, Rou16].

2 Equivalent Forms

2.1 General Form

This is the general form of the linear programming problem. The c_i's can be interpreted as costs. In this case, the objective is to minimize the total cost subject to the linear constraints.

$$\text{Minimize } \sum_{i=1}^{n} c_i x_i \qquad \text{(objective function)} \tag{1}$$

$$\begin{array}{lrcll} \text{subject to} & \sum_{j=1}^{n} a_{ij}x_j & \geq & b_i \quad , i = 1, \ldots, m_1 & \text{(inequality)} \\ & \sum_{j=1}^{n} a_{ij}x_j & = & b_i \quad , i = m_1 + 1, \ldots, m_1 + m_2 & \text{(equality)} \\ & x_j & \geq & 0 \quad , j = 1, \ldots, n_1 & \text{(non} - \text{negativity)} \\ & x_j & \lessgtr & 0 \quad , j = n_1 + 1, \ldots, n & \text{(unconstrained)} \end{array}$$

The general form of the LP can be written more compactly in matrix notation.

$$\text{Minimize } \mathbf{c}^T\mathbf{x} \tag{2}$$

$$\begin{array}{lrcl} \text{subject to} & \mathbf{A}\mathbf{x}_1 & \geq & \mathbf{b} \\ & \mathbf{A}'\mathbf{x}_2 & = & \mathbf{b}' \\ & \mathbf{x}_1 & \geq & 0 \\ & \mathbf{x}_2 & <=> & 0 \end{array}$$

where

$\mathbf{c}$ and $\mathbf{x}$ are n x 1 vectors,

$\mathbf{A}$ is an m_1 x n matrix,

$\mathbf{A}'$ is an m_2 x n matrix,

$\mathbf{b}$ is an m_1 x 1 vector,

$\mathbf{b}'$ is an m_2 x 1 vector,

$\mathbf{x}_1$ is an n_1 x 1 vector,

$\mathbf{x}_2$ is an n_2 x 1 vector, and

$\mathbf{x} = \left\{ \begin{array}{c} \mathbf{x}_1 \\ \mathbf{x}_2 \end{array} \right\}$.

2.2 Standard Form

This is the standard form of the linear programming problem. Here the constraints take the form of linear equality constraints, plus non-negativity constraints on the independent variables.

$$\text{Minimize } \sum_{i=1}^{n} c_i x_i \qquad \text{(objective function)} \tag{3}$$

$$\begin{array}{rrcll} \text{subject to} & \sum_{j=1}^{n} a_{ij} x_j & = & b_i & , i = 1, \ldots, m \quad \text{(equality)} \\ & x_j & \geq & 0 & , j = 1, \ldots, n \quad \text{(non} - \text{negativity)} \end{array}$$

The standard form of the LP can also be written more compactly in matrix notation.

$$\text{Minimize } \mathbf{c}^T \mathbf{x} \tag{4}$$

$$\begin{array}{rrcl} \text{subject to} & \mathbf{Ax} & = & \mathbf{b} \\ & \mathbf{x} & \geq & 0 \end{array}$$

where

$\mathbf{c}$ and $\mathbf{x}$ are n x 1 vectors,

$\mathbf{A}$ is an m x n matrix, and

$\mathbf{b}$ is an m x 1 vector.

Definition 1. If x satisfies $\mathbf{Ax} = \mathbf{b},\ \mathbf{x} \geq \mathbf{0}$, then x is **feasible**.

2.3 Canonical Form

This is the canonical form of the linear programming problem. Here the constraints take the form of linear inequality constraints, plus non-negativity constraints on the independent variables.

$$\text{Minimize } \sum_{i=1}^{n} c_i x_i \qquad (\text{objective function}) \tag{5}$$

$$\begin{array}{rrcll} \text{subject to} & \sum_{j=1}^{n} a_{ij} x_j & \geq & b_i & , i = 1, \ldots, m \quad (\text{inequality}) \\ & x_j & \geq & 0 & , j = 1, \ldots, n \quad (\text{non} - \text{negativity}) \end{array}$$

Once more, the canonical form of the LP can be written more compactly in matrix notation.

$$\text{Minimize } \mathbf{c}^T \mathbf{x} \tag{6}$$

$$\begin{array}{rrcl} \text{subject to} & \mathbf{Ax} & \geq & \mathbf{b} \\ & \mathbf{x} & \geq & 0 \end{array}$$

where

$\mathbf{c}$ and $\mathbf{x}$ are n x 1 vectors,

$\mathbf{A}$ is an m x n matrix, and

$\mathbf{b}$ is an m x 1 vector.

2.4 Constraint Conversion

It is possible to convert between equality and inequality constraints by the following method.

1. To convert a less than or equal inequality constraint,

   $$Ax \leq b$$

   to an equality constraint, add the vector of slack variables, s.

   $$Ax + s = b,\ s \geq 0$$

2. To convert a greater than or equal inequality constraint,

   $$Ax \geq b$$

   to an equality constraint, add the vector of surplus variables, t.

   $$Ax - t = b,\ t \geq 0$$

3. Finally, to convert an equality constraint,

   $$Ax = b$$

   to an inequality constraint, add two inequality constraints.

   $$Ax \leq b,\ -Ax \leq -b$$

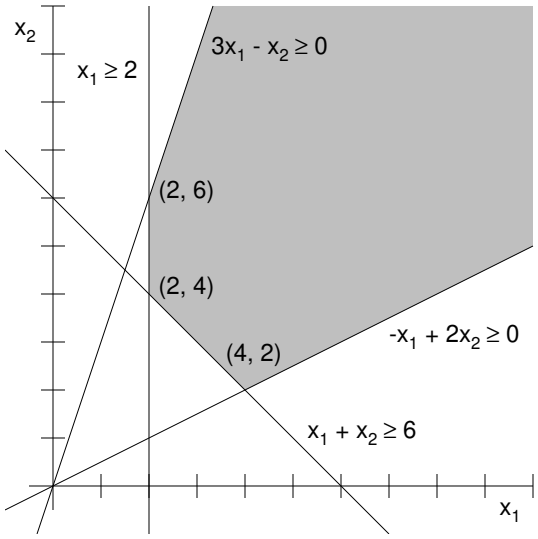

Figure 1: Feasible Region

3 Example

Example 1. Consider the following linear program:

$$\begin{array}{lrcrcl} \text{Minimize} & & & x_2 & & \\ \text{subject to} & x_1 & & & \geq & 2 \\ & 3x_1 & - & x_2 & \geq & 0 \\ & x_1 & + & x_2 & \geq & 6 \\ & -x_1 & + & 2x_2 & \geq & 0 \\ & x_1 & & & \geq & 0 \\ & & & x_2 & \geq & 0 \end{array}$$

The optimal solution is (4, 2) of cost 2 (See Figure 1). If we were maximizing x_2, instead of minimizing under the same feasible region, the resulting linear program would be unbounded, since x_2 can increase arbitrarily. From this picture, the reader should be convinced that, for any objective function for which the linear program is bounded, there exists an optimal solution which is a "corner" of the feasible region. This notion will be formalized in the next class.

4 Applications of LP

Linear Programming was developed during World War II to solve logistics problems for military resources. The simplex method for solving linear programming problems was developed in 1947 by George Dantzig, an American mathematician. Although the simplex method performs well in practice, in 1972 Klee and Minty demonstrated an example in which the simplex method takes an exponential number of steps. In

1979, L. G. Khachiyan produced the ellipsoid method, which was shown to solve any linear program in polynomial time [Kha79]. However, when implemented, his method was not competitive with the simplex method, and lost favor. Then, in 1984, Narender K. Karmarkar presented a method based on the projective transformation of a simplex. Karmarkar showed his method solves any linear program in time that is a polynomial function of the data of the problem [Kar84]. In contrast to Khachiyan's method, implementations of various generalizations of Karmarkar's method are very competitive with the simplex method, and generally outperform it for large problems.

Modern industry-strength LP solvers are extensively used in logistics, operations management, and optimized resource allocation in a variety of industries. These solvers routinely solve linear programs involving millions of variables and constraints. Here we review some applications in more recognizable contexts, including graph optimization problems and machine learning.

Network Flows. In the maximum flow problem, we are given a flow network $G = (V, E)$ with non-negative integer capacity $c(u, v)$ for each edge (u, v) of G, a source s and a sink t. The goal is to find a maximum flow from s to t, subject to capacity and flow conservation constraints. It is straightforward to represent the maximum flow problem as a linear program.

$$\begin{aligned} \max \quad & \sum_{(s,v)\in E} f(s,v) \\ \sum_{(u,v)\in E} f(u,v) &= \sum_{(v,u)\in E} f(v,u) \qquad v \neq s,t \\ f(u,v) &\leq c(u,v) \qquad (u,v) \in E \\ f(u,v) &\geq 0 \qquad (u,v) \in E \end{aligned}$$

One can solve the linear program to obtain a maximum flow. The algorithms you have studied earlier for network flow (Ford-Fulkerson and variants) are much more efficient (since they are specialized) than using the LP hammer for the problem. Furthermore, using those variants, one immediately gets integral flow solutions when all capacities are integers; this fact is useful for many applications. Solving the above linear program may not immediately yield an integral solution.

One significant advantage of using LPs for flow is the flexibility it provides to include additional constraints on the flow. For instance, suppose there is a cost $p(u, v)$ associated with each edge (u, v) and we are interested in finding a flow from s to t of a certain value, say F, at minimum cost. We can capture the associated minimum cost flow problem by replacing the objective function by

$$\min \sum_{(u,v)\in E} p(u,v) f(u,v)$$

and adding a new constraint

$$\sum_{(s,v)\in E} f(s,v) \geq F$$

Additional constraints such that a lower bound on the flow on certain edges can also be easily accommodated.

There are many compelling applications of linear programming in data science and machine learning, some of which we will cover later in the course. For a quick review of the use of linear programming in line fitting, linear classification, and even classification using non-linear separators, see [Rou16].

5 Combinatorial Optimization and Rounding LPs

5.1 Minimum Spanning Trees

In the minimum spanning tree problem, the set of instances is the set of all weighted undirected graphs. For a given instance graph G, the set $S(G)$ of solutions for G is the set of all trees that span every vertex of G. Finally, the value of a solution tree T is simply the sum of the weights of the edges of T.

One LP representation of the problem is the following.

$$\begin{array}{lll} \min & \sum_e w_e x_e & \\ \sum_{e \in C} x_e & \geq 1 & \text{for each cut } C \\ x_e & \in \{0,1\} & \text{for each edge } e \end{array}$$

Note that since there are an exponential number of cuts, the number of constraints is exponential. Another issue with the linear program is the integrality gap. It is not difficult to come up with an example gap where the optimal LP solution is almost half of the optimal integral solution, even in the unweighted case where the MST problem is trivial. (Can you think of one?)

An alternative LP formulation is the following, called the spanning tree polytope, which is a convex hull of the characteristic vectors of all the spanning trees of the graph. For a subset S of vertices, $E(S)$ is the set of all edges with both end-points in S. For any subset E' of edges, $x(E')$ equals $\sum_{e \in E'} x(e)$. The spanning tree polytope $\mathcal{P}$ is given by the following linear inequalities:

$$\{x(E(S)) \leq |S| - 1 \text{ for } S \subseteq V; x(E) = n - 1; x \geq 0\}$$

An LP relaxation for MST with edge weights given by the function w is simply $\min_e \sum w(e)x(e)$ subject to $x \in \mathcal{P}$. A classic result shows that the spanning tree polytope is integral; that is every vertex of the polytope is a 0-1 vector. As we will see in the next class, for every LP, there exists an vertex of the associated polytope that is optimal. We thus obtain that the above LP for MST has no integrality gap, and exactly captures the MST problem.

5.2 Knapsack Problem

Recall that in the knapsack problem we are given n items of sizes w_1 through w_n and profits p_1 through p_n. The goal is to build the items with total size $\leq B$ such that the total profit is maximized.

The LP representation of the problem can be shown as:

$$\begin{array}{ll} \max & \sum x_j p_j \\ \sum_{j=1}^{n} x_j w_j & \leq B \\ x_j & \in \{0,1\} \end{array}$$

$$c^T = \begin{bmatrix} p_1 \\ \vdots \\ p_n \end{bmatrix}$$

$$A = \begin{bmatrix} w_1 & w_2 \ldots w_n \end{bmatrix}$$

$$b = \begin{bmatrix} B \end{bmatrix}$$

Note that the above program is an integer linear program owing to the constraint $x_j \in \{0,1\}$. Replacing it by $x_j \leq 1$ will yield a linear programming relaxation.

5.3 Weighted Vertex Cover

Recall that a vertex cover of a given undirected graph $G = (V, E)$ is a subset $S \subseteq V$ such that for each edge $(u, v) \in E$, either $u \in S$ or $v \in S$ (or both). That is, each vertex covers its incident edges, and a vertex cover for G is a set of vertices that covers all the edges in E. The LP representation of the problem is:

$$\begin{array}{lll} \min & \sum_u x_u w_u & \\ x_u + x_v & \geq 1 & \forall (u,v) \in E \\ x_u & \in \{0,1\} & \forall u \in V \end{array}$$

To obtain an LP relaxation, we replace the constraint $x_u \in \{0,1\}$ by $x_u \leq 1$.

Rounding algorithm: Weighted vertex cover admits arguably the simplest rounding algorithm: Solve the LP to obtain solution x^*, and select every vertex u with $x_u^* \geq 1/2$. We now prove that the resulting solution (say S) is feasible and has cost at most twice the optimal. Let the optimal cost for the instance and the LP optimal cost be given by OPT_{LP}.

Since each edge (u, v) has $x_u + x_v \geq 1$, either $x_u \geq 1/2$ or $x_v \geq 1/2$, implying that either u or v is in S. Hence S is a feasible vertex cover. For the cost, we have

$$\text{cost of } S = \sum_{u: x_u \geq 1/2} w_u \leq 2 \sum_{u: x_u \geq 1/2} w_u x_u \leq 2 \sum_u w_u x_u = 2\text{OPT}_{\text{LP}} \leq 2\text{OPT}.$$

5.4 Set Cover

In the set cover problem, we are given a universe $\mathcal{U}$ of elements, a collection $\mathcal{C}$ of subsets of $\mathcal{U}$, and a cost $c(S)$ for each set S in $\mathcal{C}$. The goal is to determine the minimum-cost collection X of sets from $\mathcal{C}$ such that every element is in some set in X.

We present an integer linear program for the set cover problem. We use $x(S)$ as an indicator variable for set $S \in \mathcal{C}$:

$$x(S) = \begin{cases} 1 & \text{if } S \text{ is selected} \\ 0 & \text{otherwise} \end{cases}$$

Since each element u should be in some chosen subset, we need to select at least one set S that contains u: that is, $\sum_{S:u \in S} x(S) \geq 1$ for all $u \in U$. We obtain the following integer linear program.

$$\begin{array}{lll} \min & \sum_{S \in \mathcal{C}} x(S)c(S) & \\ \sum_{S \in \mathcal{C}: u \in S} x(S) & \geq 1 & \forall u \in U \\ x(S) & \in \{0,1\} & \forall S \in \mathcal{C} \end{array}$$

If $x(S) \in \{0,1\}$ is replaced by $x(S) \geq 0$ then we obtain a linear program (referred to as an LP relaxation of set cover), which is solvable in polynomial time.

We will discuss approaches to round an optimal solution for the above linear program to obtain an approximate solution to the set cover problem.

References

[Goe94] M. Goemans. Introduction to Linear Programming. Lecture notes, available from `http://www.cs.cmu.edu/afs/cs/user/glmiller/public/Scientific-Computing/F-11/RelatedWork/Goemans-LP-notes.pdf`, October 1994.

[Kar84] N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4:373–395, 1984.

[Kar91] H. Karloff. *Linear Programming*. Birkhäuser, Boston, MA, 1991.

[Kha79] L. G. Khachiyan. A polynomial algorithm for linear programming. *Soviet Math. Doklady*, 20:191–194, 1979.

[Rou16] T. Roughgarden. Linear programming: Introduction and applications. Lecture notes, available from `https://timroughgarden.org/w16/l/l7.pdf`, 2016.