DS 574 Algorithmic Mechanism Design	**Lecture #8 Worksheet**
Fall 2023	Prof. Kira Goldner

Introduction to Linear Programming and Duality

Why Linear Programming rocks:

- Incredibly general: Almost all problems from undergrad algorithms can be formulated as a linear program.
- Computationally tractable
  - In theory: Can be solved in polynomial time
  - In practice: Fast with input sizes up into the millions!
- Contains many properties that can be turned into useful algorithmic paradigms and analysis:
  - Duality:
    * Solve an easier equivalent problem.
    * How do we know when we're done?
  - Complementary Slackness and Strong Duality: something is optimal!

How to Think About Linear Programming

Comparison to Systems of Linear Equations

Think back to linear systems of equations. Such a system consists of m linear equations in real-valued variables $x_1, \ldots, x_n$:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{12}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{22}x_n &= b_2 \\ &\vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{m2}x_n &= b_m. \end{aligned}$$

The a_{ij}'s and the b_i's are given; the goal is to check whether or not there are values for the x_j's such that all m constraints are satisfied. We used Gaussian elimination; "solved" meant that the algorithm returns a feasible solution, or correctly reports that no feasible solution exists.

Linear programming is coming up with the "best" solution when instead of equations, we have inequalities.

Ingredients of a Linear Program

Using the language of linear programming, we can express many of the computational problems that we know.

> **Ingredients of a Linear Program**
>
> **a.** *Decision variables* $x_1, \ldots, x_n \in \mathbb{R}$.
>
> **b.** *Linear constraints*, each of the form
>
> $$\sum_{j=1}^{n} a_j x_j \quad (*) \quad b_i,$$
>
> where $(*)$ could be $\leq, \geq$, or $=$.
>
> **c.** A *linear objective function* of the form
>
> $$\max \sum_{j=1}^{n} c_j x_j$$
>
> or
>
> $$\min \sum_{j=1}^{n} c_j x_j.$$

Comments:

- The a_{ij}'s, b_i's, and c_j's are *constants*, part of the input.
- The x_j's are *variables*, what the algorithm is trying to set.
- When specifying constraints, there is no need to make use of both "$\leq$" and "$\geq$"inequalities—one can be transformed into the other just by multiplying all the coefficients by -1 (the a_{ij}'s and b_i's are allowed to be positive or negative).
- Equality constraints can be turned into two inequalities.
- min and max can easily be converted from one to another.

What's not allowed in a linear program? Non-linear variables—terms like $x_j^2, x_j x_k, \log(1 + x_j)$, etc. So whenever a decision variable appears in an expression, it is alone, possibly multiplied by a constant (and then summed with other such terms).

A Simple Example

To make linear programs more concrete and develop your geometric intuition about them, let's look at a toy example. (Many "real" examples of linear programs are coming shortly.) Suppose

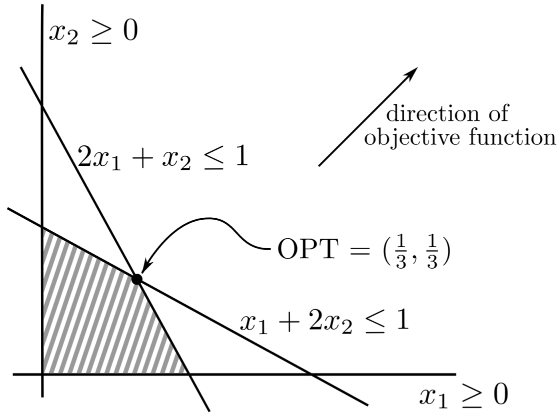

Figure 1: A toy example of a linear program.

there are two decision variables x_1 and x_2—so we can visualize solutions as points (x_1, x_2) in the plane. See Figure 1. Let's consider the (linear) objective function of maximizing the sum of the decision variables:

$$\max x_1 + x_2.$$

We'll look at four (linear) constraints:

$$\begin{aligned} x_1 &\geq 0 \\ x_2 &\geq 0 \\ 2x_1 + x_2 &\leq 1 \\ x_1 + 2x_2 &\leq 1. \end{aligned}$$

The feasible region is shaded in Figure 1. Geometrically, the objective function asks for the feasible point furthest "northeast" in the direction of the coefficient vector $(1, 1)$. Eyeballing, this point is $(\frac{1}{3}, \frac{1}{3})$, for an optimal objective function value of $\frac{2}{3}$.

Geometric Intuition

In higher dimensions, a linear constraint in n dimensions corresponds to a halfspace in $\mathbb{R}^n$. Thus a feasible region is an intersection of halfspaces, the higher-dimensional analog of a polygon.[1]

When there is a unique optimal solution, it is a vertex (i.e., "corner") of the feasible region.

Edge cases occur when the feasible region is unbounded, empty, or the objective function is unbounded.

[1] A finite intersection of halfspaces is also called a "polyhedron;" in the common special case where the feasible region is bounded, it is called a "polytope."

Writing Problems as Linear Programs

Example 1: Grain Nutrients

Suppose BU has hired you to optimize nutrition for campus dining. There are two possible grains they can offer, grain 1 and grain 2, and each contains the macronutrients found in the table below, plus cost per kg for each of the grains.

Macros	Starch	Proteins	Vitamins	Cost ($/kg)
Grain 1	5	4	2	0.6
Grain 2	7	2	1	0.35

The nutrition requirement per day of starch, proteins, and vitamins is 8, 15, and 3 respectively. Determine how much of each grain to buy such that BU spends as little but meets its nutrition requirements.

Decision variables:

Objective:

Constraints:

Example 2: Transportation

You're working for a company that's producing widgets among two different factories and selling them from three different centers. Each month, widgets need to be transported from the factories to the centers. Below are the transportation costs from each factory to each center, along with the monthly supply and demand for each factory and center respectively. Determine how to route the widgets in a way that minimizes transportation costs.

Transit Cost	Center 1	Center 2	Center 3
Factory 1	5	5	3
Factory 2	6	4	1

- The supply per factory is 6 and 9 respectively.
- The demand per center is 8, 5, and 2 respectively.

Decision variables:

Objective:

Constraints:

Converting to Normal Form

The "Normal Form" of a Linear Program looks like:

$$\begin{aligned} \max \quad & \mathbf{c}^T\mathbf{x} \\ \text{subject to} \quad & \mathbf{A}\mathbf{x} \leq \mathbf{b} \end{aligned}$$

Exercise: Convert the Transportation LP to normal form.

A Case Study: Maximum-Weight Matching

The Maximum-Weight Matching Problem

Given a graph $G = (V, E)$ choose a maximum weight matching—a set of edges S with maximum weight such that no vertex is covered by more than one edge.

a. *Decision variables:* What are we try to solve for?

b. *Constraints:*

c. *Objective function:*

Maximum-Weight Matching as an Integer Program

Maximum-Weight Matching as a Linear Program

The Dual of a Linear Program

Every linear program has a *dual* linear program. We call the original linear program the *primal*. There are a bunch of amazing properties that come from LP duality.

Going back to our nutrition example, we want to find the dual linear program. A maximization problem's dual is a minimization problem. Here, we have a minimization problem, so the dual will be a maximization problem.

To take the dual: Label each primal constraint with a new dual variable. In our new linear program, each dual constraint will correspond to a primal variable. For the left-hand side, count up the appearances of this constraint's primal variable (e.g., x_1) in each of the primal constraints and multiply them by the dual variable for those constraints. That is, if x_1 appears 5 times ($5x_1$) in constraint for y_1, then add $5y_1$ to x_1's constraint. Don't forget to include its appearance in the primal's objective function, but this will be the right-hand side of the constraint. Finally, the dual objective function is given by the right-hand side coefficients and their correspondence to the dual variables via the constraints in the primal. (See below).

Primal:

$$
\begin{array}{rll}
\min & 0.6y_1 + 0.35y_2 & \\
\text{subject to} & 5y_1 + 7y_2 \geq 8 & \text{(starch)} \\
& 4y_1 + 2y_2 \geq 15 & \text{(proteins)} \\
& 2y_1 + 1y_2 \geq 3 & \text{(vitamins)} \\
& y_1, y_2 \geq 0 & \text{(non-negativity)}
\end{array}
$$

Dual:

Sometimes, the dual can even be interpreted as a related problem.

The following is the normal form for a maximization problem primal and its dual:

$$
\begin{array}{rl}
\max & \mathbf{c}^T\mathbf{x} \\
\text{subject to} & \mathbf{A}\mathbf{x} \leq \mathbf{b} \\
& \mathbf{x} \geq 0
\end{array}
\qquad\qquad
\begin{array}{rl}
\min & \mathbf{y}^T\mathbf{b} \\
\text{subject to} & \mathbf{A}^T\mathbf{y} \geq \mathbf{c} \\
& \mathbf{y} \geq 0
\end{array}
$$

For the above example:

$\mathbf{A} =$ $\mathbf{b} =$ $\mathbf{c} =$

Example 3: Unweighted Maximum Matching

Given a graph $G = (V, E)$ choose a maximum size matching—a set of edges S such that no vertex is covered by more than one edge.

Decision variables: x_e indicating whether edge e is in the matching.

Primal Linear Program:

$$\begin{array}{rlrl} \max & \sum_{e \in E} x_e & & \\ \text{subject to} & \sum_{e : v \in e} x_e \leq 1 & \forall v & \text{(vertex matched at most once)} \quad (y_v) \\ & x_e \geq 0 & \forall e & \text{(non-negativity)} \end{array}$$

Taking the dual of the above primal, we get the following linear program:

What problem is this?

Conditions for Optimality

Weak Duality

Theorem 1. *If* $\mathbf{x}$ *is feasible in (P) and* $\mathbf{y}$ *is feasible in (D) then* $\mathbf{c}^T\mathbf{x} \leq \mathbf{b}^T\mathbf{y}$.

Proof.

What is not trivial (or by definition) is *strong duality*, and in fact, it is so involved that we will not even prove the hard direction: that an optimal solution always exists.

Strong Duality

Theorem 2 (Strong Duality)**.** *A pair of solutions* $(\mathbf{x}^*, \mathbf{y}^*)$ *are optimal for the primal and dual respectively if and only if* $\mathbf{c}^T\mathbf{x}^* = \mathbf{b}^T\mathbf{y}^*$.

Proof. ($\Rightarrow$) Skip.
($\Leftarrow$)

Complementary Slackness

Primal (P):

$$\begin{aligned} \max \quad & \mathbf{c}^T\mathbf{x} \\ \text{subject to} \quad & \sum_i a_{ji}x_i \le b_j \quad \forall j \quad (y_j) \\ & x_i \ge 0 \quad \forall i \end{aligned}$$

Dual (D):

$$\begin{aligned} \min \quad & \mathbf{y}^T\mathbf{b} \\ \text{subject to} \quad & \sum_i a_{ij}y_i \ge c_i \quad \forall i \quad (x_i) \\ & y_j \ge 0 \quad \forall j \end{aligned}$$

Theorem 3 (Complementary Slackness)**.** *A pair of solutions* $(\mathbf{x}^*, \mathbf{y}^*)$ *are optimal for the primal and dual respectively if and only if the following complementary slackness conditions (1) and (2) hold:*

Proof.

Maximizing Welfare in the Unit Demand Setting

Given n unit-demand bidders and m items, determine the allocation rule that maximizes welfare. Do this by formulating a linear program.

Determine your objective, decision variables, and constraints.

Formulate the dual.

Do you see an interpretation of this dual?

Separation Oracles

Fact 1 (Ellipsoid Algorithm)**.** Every linear program that admits a polynomial-time separation oracle can be solved in polynomial time.

Consider a linear program such that:

a. There are n decision variables.

b. There are any number of constraints, for example, exponential in n. These constraints are not provided explicitly as input.

c. There is a polynomial-time *separation oracle* for the set of constraints. By "polynomial-time," we mean running time polynomial in n and the maximum number of bits of precision required. A separation oracle (Figure 1) is a subroutine that takes as input an alleged feasible solution to the LP, and either (i) correctly declares the solution to be feasible, or (ii) correctly declares the solution to be infeasible, and more strongly provides a proof of infeasibility in the form of a constraint that the proposed solution violates.

(The ellipsoid algorithm is not actually practical, but there are other algorithms that *are* often practically useful that rely on a separation oracle, such as cutting plane methods.)

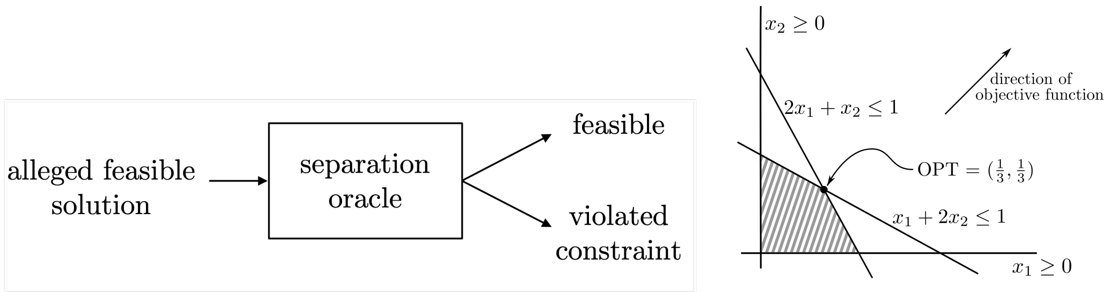

Figure 2: Left: A sketch of a separation oracle. For example, in the toy example on the right, on the alleged feasible solution $(\frac{1}{3}, \frac{1}{2})$, the separation oracle may return the violated constraint $x_1 + 2x_2 \leq 1$.

Revenue Maximization

Consider the single-item revenue-maximization setting with n bidders. Formulate the LP to maximize revenue. Formulate its dual.