Recitations for ORIE 6300: Mathematical Programming I

Benjamin Grimmer[*]

[*]School of Operations Research and Information Engineering, Cornell University, Ithaca, NY 14850, USA; people.orie.cornell.edu/bdg79/.

1 Recitation 1: Linear Images of Polyhedra

Today's recitation provides a proof for the following fact that was stated in class:

Proposition 1.1 (Linear maps preserve polyhedrality). *Let $\mathcal{P} \subseteq \mathbb{R}^n$ be a polyhedral set, which is a set defined by a finite number of linear inequalities. Consider $A \in \mathbb{R}^{m \times n}$. Then, the set*

$$\{Ax \mid x \in \mathcal{P}\}$$

is a polyhedral set.

Before proceeding with the proof, note that assuming $\mathcal{P}$ is defined by linear inequalities is without loss of generality; indeed, we have the following equivalence:

$$a_i^T x = b_i \Leftrightarrow \begin{cases} a_i^T x \le b_i, \\ a_i^T x \ge b_i \end{cases},$$

which allows us to convert any linear equality into two inequalities. The proof is **constructive**, i.e. we can identify an explicit way to start from the inequalities defining $\mathcal{P}$ and construct linear inequalities that define $\{Ax \mid x \in \mathcal{P}\}$.

The next step will make our lives much easier when working towards the proof of Proposition 1.1:

Claim 1. It suffices to prove Proposition 1.1 for projection matrices $P \in \mathbb{R}^{(n-1) \times n}$, which eliminate one coordinate when applied to a vector $x \in \mathbb{R}^n$.

Proof of Claim. Let's take a look at an example of such a matrix:

$$P = \begin{bmatrix} 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ \vdots & & & & \vdots \\ 0 & 0 & \dots & 1 & 0 \end{bmatrix} \in \mathbb{R}^{n-1 \times n}.$$

This projection "eliminates" the variable x_n from a point x.

Now let us see why the Claim holds. First, we prove by induction that we can eliminate as many variables as we want and keep the set polyhedral, as long as we've proved the base step.

- Base step: the set $\{Px, x \in \mathcal{P}\}$, where P is matrix that eliminates one of the coordinates and $\mathcal{P}$ is a polyhedron, is itself polyhedral.
- Now, assume we've proved that eliminating k variables from the set $\{x \in \mathcal{P}\}$ results in a polyhedral set.
- Since we have proved the result for k variables, we have a set in $\mathbb{R}^{n-k}$ which is polyhedral. Then, by the base step, we can apply a projection again to result in a set that is polyhedral and in $\mathbb{R}^{n-k-1}$, i.e. we have eliminated one more variable.

Now, consider the case of an arbitrary linear map A: if you examine the following set closely, you will get a clear idea of where we're heading next:

$$\{\begin{pmatrix} x \\ y \end{pmatrix} \in \mathbb{R}^{n+m} \mid x \in \mathcal{P}, y = Ax\} \triangleq \mathcal{P}' \tag{1.1}$$

First, let's convince ourselves that the set in (1.1) is polyhedral. If $\mathcal{P}$ was to be described by the inequalities $\{x \mid Cx \leq d\}$, we can rewrite $\mathcal{P}'$ as

$$\mathcal{P}' = \{\begin{pmatrix} x \\ y \end{pmatrix} \in \mathbb{R}^{n+m} \mid \begin{bmatrix} C & \underline{0}_{m\times m} \end{bmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \leq d,\ \begin{bmatrix} A & -I_{m\times m} \end{bmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = 0\}$$

Now, appealing to our inductive argument, we can readily eliminate the first n variables in (1.1), giving us precisely the set $\{Ax \mid x \in \mathcal{P}\}$. $\square$

A numerical example in $\mathbb{R}^2$ Consider a very simple polyhedral constraint, shown below:

$$\mathcal{P} = \begin{cases} x_1 + x_2 & \geq 3 \\ 2x_1 - x_2 & \leq 5 \\ -x_1 + 2x_2 & \leq 3 \end{cases}$$

The polyhedron and its projection to the variable x_1 are shown in 1. Let's see what we can

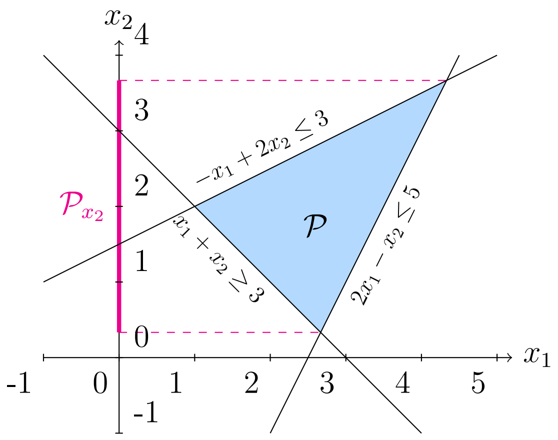


Figure 1: A polyhedron $\mathcal{P} \subseteq \mathbb{R}^2$ and its projection to x_2, $\mathcal{P}_{x_2}$

eliminate by hand: if we seek to bring x_1 to the LHS of the constraints shown above, we obtain

$$\begin{cases} x_1 \geq 3 - x_2, \\ 2x_1 \leq 5 + x_2, \\ -x_1 \leq 3 - 2x_2 \end{cases}$$

Dividing the second inequality by 2 and multiplying the last one by -1, gives us

$$\begin{cases} x_1 \geq 3 - x_2, \\ x_1 \leq \frac{5}{2} + \frac{x_2}{2}, \\ x_1 \geq 2x_2 - 3 \end{cases}$$

Now, combine the second inequality with the first and third ones to eliminate x_1 and obtain

$$\frac{1}{3} \leq x_2 \leq \frac{11}{3},$$

which is exactly the projection shown in 1.

An example in $\mathbb{R}^3$ Let's do the same for a polyhedral set in 3 dimensions that is easy to visualize. Consider the following polyhedron:

$$\mathcal{P} := \{x \in \mathbb{R}^3 \mid x \geq 0,\ x_1 + x_2 + x_3 \leq 1\} \tag{1.2}$$

Suppose we want to find the set $C = \{Px \mid x \in \mathcal{P}\}$, where P is the projection matrix that eliminates the variable x_1. If we follow the same strategy to eliminate x_1, we get

$$x_1 \leq 1 - x_2 - x_3,\ x_1 \geq 0$$

which gives us

$$1 - x_2 - x_3 \geq 0 \Rightarrow x_2 + x_3 \leq 1.$$

It is easy to verify visually that the resulting set is exactly what we would expect to obtain if we projected $\mathcal{P}$ to its last 2 coordinates, as shown in (2)

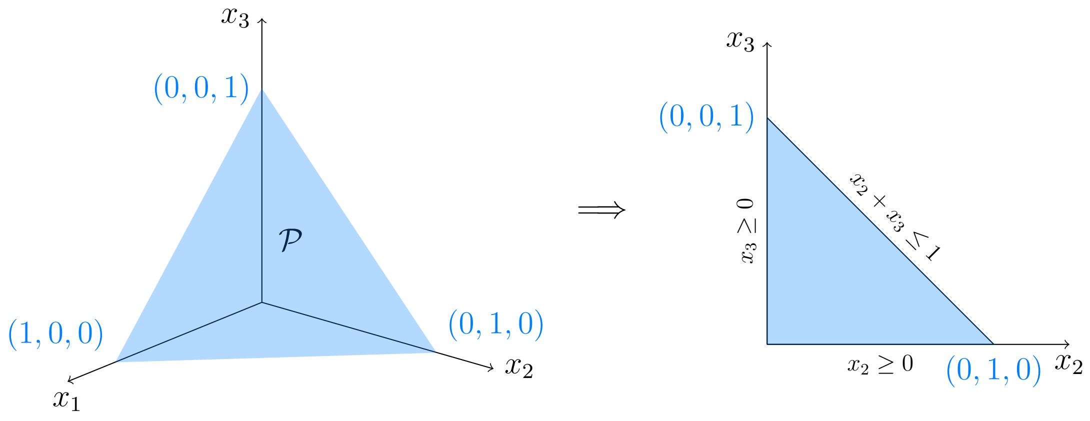


Figure 2: Eliminating x_1 from (1.2)

Proof of Proposition 1.1. Equipped with Claim 1, let us see how to prove the desired result for the case where we want to eliminate x_n. Like before, assume $\mathcal{P}$ is described by linear inequalities only. Additionally, let us index those inequalities, like below:

$$\begin{cases} a_1^T x \leq b_1, \\ a_2^T x \leq b_2, \\ \quad\vdots \\ a_m^T x \leq b_m \end{cases}.$$

Denote $I_0, I_+, I_- \subseteq [m]$ the index sets where the coefficient of x_n is 0, positive and negative, respectively. We maintain the the inequalities in I_0 as they were, since x_n is not present there. For the remaining constraints, multiply the left and right hand sides so as to make the coefficient of x_n equal to 1. Denote by c_i the scaled vectors of coefficients for the first $n-1$ variables, d_i for the scaled constants, and:

$$\bar{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_{n-1} \end{pmatrix}.$$

Then, we can write

$$\begin{cases} c_i^T \bar{x} \leq d_i, & i \in I_0 \\ c_i^T \bar{x} + x_n \leq d_i, & i \in I_+ \\ c_i^T \bar{x} + x_n \geq d_i, & i \in I_- \end{cases} \Leftrightarrow \begin{cases} c_i^T \bar{x} \leq d_i, & i \in I_0 \\ x_n \leq d_i - c_i^T \bar{x}, & i \in I_+ \\ x_n \geq d_i - c_i^T \bar{x}, & i \in I_- \end{cases}$$

For now, assume that both sets I_+, I_- are nonempty. Since $x_n \leq d_i - c_i^T \bar{x}$ for all $i \in I_+$ and also $x_n \geq d_i - c_i^T \bar{x}$ for all $i \in I_-$, we can combine the inequalities to obtain

$$\begin{cases} 0 \leq d_i - c_i^T \bar{x}, & i \in I_0 \\ d_j - c_j^T \bar{x} \leq x_n \leq d_i - c_i^T \bar{x}, & \forall (i,j) \in I_+ \times I_- \end{cases}$$

A point $\bar{x}$ is in the projection if it satisfies the above inequalities for some number x_n. This number exists if and only if

$$d_j - c_j^T \bar{x} \leq d_i - c_i^T \bar{x},\ \forall (i,j) \in I_+ \times I_-$$

which is a set of linear inequalities which do not involve x_n, hence a polyhedron in $\mathbb{R}^{n-1}$.

What if one of the two sets I_+, I_- is empty? In that case, the inequalities in the nonempty set are redundant. To see why, assume without loss of generality that the set I_- is empty. Then, the system becomes

$$\begin{cases} 0 \leq d_i - c_i^T \bar{x}, & i \in I_0 \\ d_j - c_j^T \bar{x} \leq x_n, & i \in I_+ \end{cases}$$

Letting x_n approach $+\infty$, we end up with a vector $\bar{x}$ that, additionally to the constraints in I_0, trivially satisfies all of the constraints in I_+. This completes the proof. $\square$

Remarks on the complexity of describing a polyhedron So far we have shown that sets $\{Ax \mid x \in \mathcal{P}\}$ are polyhedral. However, it may not be more efficient to describe this set by a collection of inequalities.

To be precise, suppose $\mathcal{P}$ can be described using n inequalities. Then our proof of Proposition 1.1 shows the polyhedron given by projecting one coordinate away can be described using at most $(n/2)^2$ inequalities (when $|I_-| = |I_+| = n/2$). Repeating this argument to remove d coordinates may result in needing an exponential number of inequalities. Hence even though $\{Ax \mid x \in \mathcal{P}\}$ is a polyhedron, it may not always be to your benefit to put it in that form.

A concrete example of this complexity difference appears in the first homework assignment for the ℓ_1-ball.

2 Recitation 2: Sufficient Optimality Conditions for Convex Optimization

Today's recitation provides two helpful lemmas for convex optimization.

Frechet gradients of convex functions. Recall the Frechet gradient provides the unique approximation of a function f up to first order. Namely,

$$f(y) = f(x) + \langle \nabla f(x), y - x\rangle + o_x(y) \quad \text{where} \quad \lim_{y\to x} \frac{o_x(y)}{\|y-x\|} = 0.$$

Here the *little-o* function captures all the ways that the function deviates from the first-order approximation $f(x) + \langle \nabla f(x), y - x\rangle$. For convex functions, we can say something stronger. Not only does the gradient provide an accurate approximation of f close to x, but it also provides a lower bound on f everywhere. This is formalized in the following lemma.

Lemma 2.1. *Consider any convex $f\colon \mathbb{R}^d \to \mathbb{R}$ that is (Frechet) differentiable at some point $x \in \mathrm{dom} f$. Then all $y \in \mathbb{R}^d$ satisfy*

$$f(y) \geq f(x) + \langle \nabla f(x), y - x\rangle.$$

Proof. Lets examine points on the line segment

$$[(x, f(x)), (y, f(y))] = \{\lambda(y, f(y)) + (1-\lambda)(x, f(x)) \mid 0 \leq \lambda \leq 1\}.$$

The definition of the (Frechet) gradient at x ensures for any $0 \leq \lambda \leq 1$

$$f(\lambda y + (1-\lambda)x) = f(x) + \langle \nabla f(x), \lambda y + (1-\lambda)x - x\rangle + o_x(\lambda y + (1-\lambda)x).$$

Simplifying this yields

$$f(\lambda y + (1-\lambda)x) = f(x) + \lambda\langle \nabla f(x), y - x\rangle + o_x(\lambda y + (1-\lambda)x).$$

Since f and consequently $\mathrm{epi} f$ are convex, $\lambda(y, f(y)) + (1-\lambda)(x, f(x)) \in \mathrm{epi} f$. Hence

$$\lambda f(y) + (1-\lambda) f(x) \geq f(x) + \lambda\langle \nabla f(x), y - x\rangle + o_x(\lambda y + (1-\lambda)x),$$

or equivalently,

$$f(y) - f(x) \geq \langle \nabla f(x), y - x\rangle + o_x(\lambda y + (1-\lambda)x)/\lambda.$$

Taking the limit as $\lambda \to 0$ (and so $\lambda y + (1-\lambda)x \to x$) gives the claimed inequality. □

This result can be viewed in terms of normal cones of the epigraph of f. Considering any $(y, t) \in \mathrm{epi} f$, we have $t \geq f(y) \geq f(x) + \langle \nabla f(x), y - x\rangle$. This can be restated as saying all $(y, t) \in \mathrm{epi} f$ satisfy

$$\langle (\nabla f(x), -1), (y, t) - (x, f(x))\rangle \leq 0.$$

Hence the gradient defines a normal vector to the epigraph of f at $(x, f(x))$ as

$$(\nabla f(x), -1) \in \mathcal{N}_{\mathrm{epi} f}((x, f(x))).$$

Global optimality conditions. In lecture, a necessary condition for first-order optimality (Theorem 3.3) was derived. In particular, any local minimizer $\bar{x}$ of a function $f\colon \mathbb{R}^d \to \mathbb{R}$ on a closed convex set $\mathcal{X} \subseteq \mathbb{R}^d$ must satisfy

$$-\nabla f(\bar{x}) \in \mathcal{N}_{\mathcal{X}}(\bar{x})$$

if f is (Frechet) differentiable at $\bar{x}$.

Alas this condition is not sufficient to determine whether a point $\bar{x}$ is a local minimizer of f. For example, when $\mathcal{X} = \mathbb{R}^d$, we have $\mathcal{N}_{\mathcal{X}}(\cdot) = \{0\}$ everywhere. Then this condition reduces to $\nabla f(x) = 0$. However simple nonconvex functions like $f(x, y) = x^2 - y^2$ can be given that have a saddle point at $(0, 0)$ despite having $\nabla f(0, 0) = 0$.

If we restrict the types of functions considered, there is hope. One possible approach is to additionally assume that f is twice differentiable and has a positive definite Hessian at $\bar{x}$. In this case, one can show having $-\nabla f(\bar{x}) \in \mathcal{N}_{\mathcal{X}}(\bar{x})$ does indeed imply $\bar{x}$ is a local minimizer. This would exclude bad cases like $f(x, y) = x^2 - y^2$, which has an indefinite Hessian $\nabla^2 f(0, 0) = \begin{bmatrix} 2 & 0 \\ 0 & -2 \end{bmatrix}$. However, we don't pursue proving this here.

Instead we show $-\nabla f(\bar{x}) \in \mathcal{N}_{\mathcal{X}}(\bar{x})$ is a sufficient condition if f is convex. Lemma 2.1 showed that for convex functions, the gradient provides a global bound on the function. Utilizing this, we find that the condition $-\nabla f(\bar{x}) \in \mathcal{N}_{\mathcal{X}}(\bar{x})$ not only ensures $\bar{x}$ is a local minimizer but ensures it is a global minimizer of f over $\mathcal{X}$.

Lemma 2.2. *Consider any closed convex set $\mathcal{X} \subseteq \mathbb{R}^d$ and convex $f\colon \mathbb{R}^d \to \mathbb{R}$ that is (Frechet) differentiable at some point $\bar{x} \in \mathcal{X}$. If*

$$-\nabla f(\bar{x}) \in \mathcal{N}_{\mathcal{X}}(\bar{x}),$$

then $\bar{x}$ is a global minimizer of f over $\mathcal{X}$.

Proof. By the definition of the normal cone, $-\nabla f(\bar{x}) \in \mathcal{N}_{\mathcal{X}}(\bar{x})$ ensures

$$\langle -\nabla f(\bar{x}), y - \bar{x} \rangle \leq 0$$

for all $y \in \mathcal{X}$. Then Lemma 2.1 implies

$$f(y) - f(\bar{x}) \geq 0$$

for all $y \in \mathcal{X}$, which is exactly the statement that $\bar{x}$ globally minimizes f over $\mathcal{X}$. □

3 Recitation 3: Linear and SOC Program Modeling

Today's recitation focuses on how we can model of different problems using standard types of conic programs (specifically, we will consider linear and second-order cone programs).

Definitions. Recall the following two types of conic programs:
Linear Programs are of the form

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & Ax = b \\ & x \in \mathbb{R}^n_+ \end{array}$$

where the conic constraint is defined with the nonnegative orthant $\mathbb{R}^n_+ = \{x \in \mathbb{R}^n \mid x_i \geq 0 \; \forall i = 1, ..., n\}$.
Second-Order Cone Programs are of the form

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & Ax = b \\ & (D_i x + e_i, f_i^T x + g_i) \in SOC^{n_i+1} \; i = 1, \ldots, m \end{array}$$

where the conic constraint is defined as an affine transformation of x must lie in the second-order cone $SOC^{n+1}_+ = \{(\bar{x}, x_{n+1}) \in \mathbb{R}^n \mid \|\bar{x}\|_2 \leq x_{n+1}\}$. Each of these second-order cone constraints can be written as the inequality

$$\|D_i x + e_i\|_2 \leq f_i^T x + g_i.$$

Regularized Optimization At its simplest, second-order cone constraints allow things like $\|x\|_2 \leq D$, which can be viewed as regularizing the problem. That is, this constraint sets a scale for the problem and prevents undesirable outcomes like the solution x running off to infinity. As an aside, constraints of this form are common throughout the machine learning literature as a tool to prevent overfitting to training data.

Robust Linear Programming. Suppose we want to solve a linear program, but have uncertainty in the exact values of a_i, b_i, or c. We may then want to expand our formulation to be robust to small variations in these values. To make this concrete, we will focus on uncertainty in the constraint vectors a_i and suppose b_i and c are fixed. We can reasonably model the possible variations in a_i by asserting that it lies in a given ellipsoid

$$a_i \in \mathcal{E}_i = \{\bar{a}_i + P_i u \mid \|u\|_2 \leq 1\}.$$

Here $\bar{a}_i$ controls the center of the ellipsoid and P_i controls its shape. Taking $P_i = I$ simply gives a ball around $\bar{a}_i$. If P_i is singular, we have a 'flat ellipsoid' with no width in directions from its null space. Hence we can model certainty in some aspects of a_i and uncertainty in others.

From this, we formulate our *Robust Linear Program* as

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & a_i^T x \leq b_i, \text{ for all } a_i \in \mathcal{E}_i, \; i = 1, \ldots, m \end{array}$$

We remark that this is a generalization of standard linear programs since setting $P_i = 0$ makes $\mathcal{E}_i = \{\bar{a}_i\}$. Hence we can still have linear inequalities (and consequently, linear equalities by adding $a_i^T x \leq b_i$ and $-a_i^T x \leq -b_i$).

Alas this problem has infinitely many constraints and so this formulation does not seem very tractable. To resolve this, consider the robust linear constraint $a_i^T x \leq b_i,$ for all $a_i \in \mathcal{E}_i$. We can express this as

$$\sup\{a_i^T x \mid a_i \in \mathcal{E}_i\} \leq b_i.$$

Notice that the lefthand side of this can be expressed as

$$\begin{aligned}\sup\{a_i^T x \mid a_i \in \mathcal{E}_i\} &= \bar{a}_i^T x + \sup\{u^T P_i x \mid \|u\|_2 \leq 1\} \\ &= \bar{a}_i^T x + \|P_i x\|_2.\end{aligned}$$

Thus the robust linear constraint can be written as $\bar{a}_i^T x + \|P_i x\|_2 \leq b_i$, which is indeed a second-order cone constraint. Thus we can write our robust linear program as the following second-order cone program

$$\begin{array}{ll}\text{minimize} & c^T x \\ \text{subject to} & \bar{a}_i^T x + \|P_i x\|_2 \leq b_i,\ i = 1, \ldots, m\end{array}$$

which only has finitely many constraints.

Linear Programming with Random Constraints. A reasonable critique of the above model is that we are required to satisfy very possible outcome for the uncertain a_i. Instead, one may want a statistical approach that requires our solution is feasible 95% of the time. To give such a formulation, we need to specify a statistical model generating the vectors a_i. One reasonable approach (which we will pursue) is to assert that each a_i comes from an independent Gaussian distribution with mean $\bar{a}_i$ and covariance Σ_i. Then probabilistic constraint can be written as

$$\textbf{prob}(a_i^T x \leq b_i) \geq \eta$$

for some fixed $\eta \geq 1/2$ representing the level of confidence we want to impose.

Notice that this type of problem is a generalization of standard linear programming. Setting the covariance of some a_i to be $\Sigma_i = 0$ will make a_i always take its mean value $\bar{a}_i$. Then $\textbf{prob}(a_i^T x \leq b_i) \geq \eta$ happens if and only if $\bar{a}_i^T x \leq b_i$. As before, we can then model linear equality constraints by having a pair of deterministic inequality constraints $a_i^T x \leq b_i$ and $-a_i^T x \leq -b_i$.

We will show that this type of probabilistic constraint can be written as a second-order cone constraint. Define the random variable $u = a_i^T x$ and denote its mean by $\bar{u}$ and variance by σ^2. Noting that u must also have a Gaussian distribution, we can write our probabilistic constraint as

$$\textbf{prob}\left(\frac{u - \bar{u}}{\sigma} \leq \frac{b_i - \bar{u}}{\sigma}\right) \geq \eta.$$

Since $(u - \bar{u})/\sigma$ is a zero mean, unit variance Gaussian variable, the probability above is simply $\Phi((b_i - \bar{u})/\sigma)$ where

$$\Phi(z) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{z} e^{-t^2/2} dt$$

is the cumulative distribution function of a standard Gaussian random variable. Since this function monotonically increases, it is invertible. Hence, we can rewrite our constraint as

$$\frac{b_i - \bar{u}}{\sigma} \geq \Phi^{-1}(\eta)$$

or equivalently,

$$\bar{u} + \Phi^{-1}(\eta)\sigma \leq b_i.$$

Our definition of $u = a_i^T x$ ensures that $\bar{u} = \bar{a}_i^T x$ and $\sigma^2 = x^T \Sigma_i x$. Hence, we obtain the equivalent constraint

$$\bar{a}_i^T x + \Phi^{-1}(\eta)\|\Sigma_i^{1/2} x\|_2 \leq b_i.$$

Our assumption that $\eta \geq 1/2$ guarantees that the coefficient $\Phi^{-1}(\eta)$ of $\|\Sigma_i^{1/2} x\|$ is nonnegative. Thus this is a second-order cone constraint.

In summary, we have found that a *Linear Program with Random Constraints*

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & \mathbf{prob}(a_i^T x \leq b_i) \geq \eta,\ i = 1, \ldots, m \end{array}$$

can be expressed as the second order cone program

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & \bar{a}_i^T x + \Phi^{-1}(\eta)\|\Sigma_i^{1/2} x\|_2 \leq b_i,\ i = 1, \ldots, m. \end{array}$$

4 Recitation 4: Polyhedral Geometry

Today's recitation focuses on characterizing the *extreme points* of polyhedrons both geometrically and algebraically. This perspective will form the *basis* for our discussion in the following recitations of the simplex algorithm for solving linear programs.

Corner Points of Polyhedrons. Consider a polyhedron $P = \{x \in \mathbb{R}^d \mid Ax \leq b\}$ where $A \in \mathbb{R}^{m \times d}$ and $b \in \mathbb{R}^m$. We begin by offering three different definitions for "corner points" of a polyhedron P.

First, we say some $x \in P$ is an **Extreme Point** if no two points $y, z \in P$ exist different from x such that $x = \lambda y + (1 - \lambda)z$ for some $\lambda \in [0, 1]$. That is, an extreme point cannot be written as a convex combination of any other two points in the polyhedron. Notice that this definition is purely geometric and does not depend on the way the constraints A are presented.

Second, we say some $x \in P$ is an **Vertex** if there exists some $c \in \mathbb{R}^d$ such that $c^T x < c^T y$ for all $y \in P \setminus \{x\}$. That is, a vertex is the unique minimizer of some linear function over the polyhedron. Geometrically, this means there is a hyperplane with P on one side of it, only touching at x.

Third and lastly, we say some $x \in P$ is an **Basic Feasible Solution (BFS)** if there exist d linearly independent constraint vectors a_i with $a_i^T x = b_i$. If we relax the requirement that $x \in P$, we say that x is a simply a **Basic Solution**. Notice that unlike the previous two definitions, this one is algebraic in nature depending on how the constraints defining P are given.

Equivalence of Corner Point Definitions. Despite these definitions all being different in their motivation, they are all equivalent. We will now prove this.

Theorem 4.1. *Let P be a nonempty polyhedron and $x^* \in P$. Then x^* is an extreme point if and only if x^* is a vertex if and only if x^* is a basic feasible solution.*

Proof. Without loss of generality, P is defined by constraints $a_i^T x \leq b_i$. We prove this by showing a cycle of implications among these three definitions.

Vertex $\implies$ Extreme Point. Suppose x^* is a vertex and let $c \in \mathbb{R}^d$ be the vector such that $c^T x^* < c^T y$ for all $y \in P \setminus \{x^*\}$. Then for any $y, z \in P \setminus \{x^*\}$, we have $c^T x^* < c^T y$ and $c^T x^* < c^T z$. Hence for any $\lambda \in [0, 1]$, $c^T x^* < c^T(\lambda y + (1 - \lambda)z)$. From this, we can conclude that $x^* \neq \lambda y + (1 - \lambda)z$ for any y, z, λ and thus is an extreme point.

Extreme Point $\implies$ Basic Feasible Solution. Suppose x^* is not a basic feasible solution and we will show x^* is not an extreme point. Let $I \subseteq [m]$ be the set of indices of constraints with $a_I^T x = b_i$. By assumption $\{a_i \mid i \in I\}$ does not contain d linearly independent vectors. Thus some nonzero vector $d \in \mathbb{R}^d$ must exist with $a_I^T d = 0$ for all $i \in I$. Then notice that for any $\epsilon > 0$, the points $x + \epsilon d$ and $x - \epsilon d$ still satisfy all the constraints in I with equality. Moreover, any constraint not in I is only met with slack at x^* (that is, $a_j^T x^* < b_i$). Thus for sufficiently small ϵ, both $x + \epsilon d$ and $x - \epsilon d$ still satisfy each constraint not in I. Hence we can write x^* as the average of two feasible solutions $x + \epsilon d$ and $x - \epsilon d$, and so x^* is not an extreme point.

Basic Feasible Solution $\implies$ Vertex. Consider a basic feasible solution x^* and let $I \subseteq [m]$ be the set of indices of constraints with $a_I^T x = b_i$. Consider minimizing over the polyhedron in the direction $c = -\sum_{i \in I} a_i$. Noting all feasible points x have $a_i^T x \leq b_i$, we know that

$$c^T x = -\sum_{i \in I} a_i^T x \geq -\sum_{i \in I} b_i = c^T x^*.$$

Hence x^* is a minimizer in the direction c. Moreover, the above reasoning shows that the optimal objective value can only be attained if all constraints in I are met with equality. However since I has d linearly independent constraints, there is a unique solution to the system $a_i^T x = b_i$ for $i \in I$, namely x^*. Hence x^* is the unique minimizer over P and thus a vertex. □

Finally we note that there can only be a finite number of extreme points in a polyhedron P. Since these are equivalent to basic feasible solutions, we can associate each extreme point with a set of d linearly independent constraints defining P. There are only m choose d ways this can be done, and so there are at most m choose d extreme points in P.

Standard Form Polyhedrons. Without loss of generality, we can assume that polyhedrons are defined by $P = \{x \in \mathbb{R}^d \mid Ax = b, g \geq 0\}$ where $A \in \mathbb{R}^{m \times d}$ and $b \in \mathbb{R}^m$. We will assume from here onward that the rows of A are linearly independent (since we can otherwise delete any linearly dependent rows as they correspond to redundant constraints).

Basic feasible solutions of standard form polyhedrons can be understood in a particularly nice way. As previously discussed, a BFS is fully determined by selecting d linearly independent constraints to be tight (and then solving the system of linear equations corresponding to them). This has particularly nice interpretation for standard form polyhedrons.

The m equality constraints must be met, so $d-m$ of the nonnegativity constraints $x_i \geq 0$ then need to be selected to determine a basic feasible solution. To algrebatically describe this BFS, we need some definitions: Let $B = \{B(1), B(2), \dots, B(m)\} \subseteq [d]$ (called the basis) denote the set of indices not chosen to fix $x_i = 0$. Let $x_B = [x_{B(1)}, x_{B(2)}, \dots, x_{B(m)}]$ denote x restricted to the indices B and similarly, its complement is $x_{\bar{B}} = [x_{\bar{B}(1)}, x_{\bar{B}(2)}, \dots, x_{\bar{B}(d-m)}]$.

Then the unique point x^* satisfying $Ax = b$ and $x_i = 0$ for $i \in \bar{B}$ is given by

$$x_{\bar{B}} = 0,$$

$$Ax = A_B x_B + A_{\bar{B}} x_{\bar{B}} = b$$

or equivalently,

$$x_{\bar{B}} = 0,$$
$$x_B = A_B^{-1} b$$

where $A_B = [A_{B(1)}, A_{B(2)}, \dots, A_{B(m)}]$ is defined like x_B to be the columns of A in B. Thus we see that basic feasible solutions have at most m nonzero elements which are determined by solving an $m \times m$ linear system.

One nicety about writing polyhedrons in standard form is that they always have at least one basic feasible solution (provided the polyhedron is nonempty). Proving this is left as an exercise.

Exercise 4.1. *Give an example of a nonempty polyhedron with no extreme points.*

Exercise 4.2. *Any nonempty polyhedron in standard form $P = \{x \in \mathbb{R}^d \mid Ax = b, x \geq 0\}$ has at least one extreme point.*

Degeneracy. Although we can think about basic feasible solutions x as being defined by a basis B, this may not uniquely define x. Namely, there could be two distinct bases B and B' that both generate x. This happens precisely when

$$x_{\bar{B}} = 0,$$

$$A_B x_B = b$$

as well as

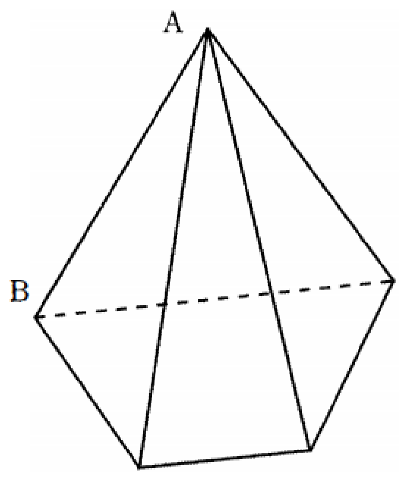
$$x_{\bar{B}'} = 0,$$

$$A_{B'} x_{B'} = b.$$

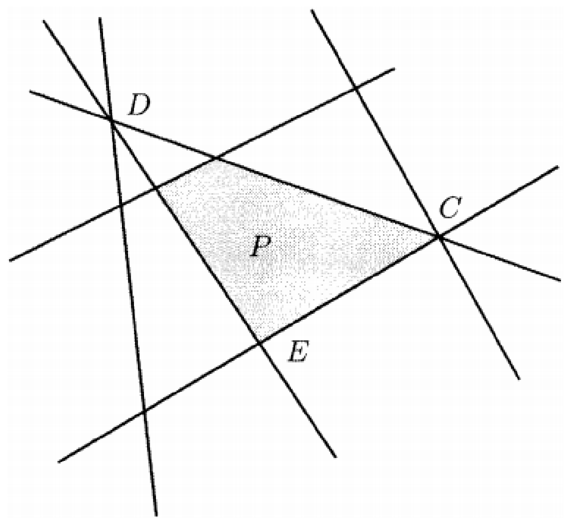
Hence any x_i with $i \in B \cup B'$ must have $x_i = 0$. This means that some constraint $x_i \geq 0$ for $i \in B$ that was not required to equal zero by our choice B was still equal to zero.

This means there are multiple equivalent ways to describe a basic feasible solution if any of the basis coordinates happen to take value 0. When this happens, we say that x is a **degenerate BFS**. Conversely, if all $i \in B$ have $x_i > 0$, we say x is a **nondegenerate BFS**.

Degeneracy occurs when there is ambiguity in which d constraints need to be tight to determine the point x. Consider the following pair of examples:



(a)



(b)

In (a) above, we see that the point A is at the intersection of four different planes. Hence selecting any three of these planes will give a system of equations that determines A is a basic feasible solution. Thus it is degenerate. Conversely, the point B is the intersection of three planes and so there is only one selection of constraints that shows B is a basic feasible solution. Thus it is nondegenerate.

In (b) above, we see another example where the degeneracy does not correspond to the shape of the underlying polyhedron. The point C lies at the intersection of three lines and so it is degenerate. However, one of these lines comes from a redundant constraint and so the representation of P is what makes this degenerate.

Optimality of Extreme Points. So far, we have seen that the corner points of a polyhedron have particularly nice structure (geometrically, in terms of optimization, and algebraically). For standard form polyhedrons, we have a simple form for these points in terms of selecting a basis of m (potentially) nonzero entries. Moreover, standard form polyhedrons always have at least one extreme point (Exercise 4.2). To connect these results to optimization, we will see next time that when minimizing a linear function over a polyhedron, some extreme point is optimal (provided one exists).

Theorem 4.2. *For any polyhedron $P \subset \mathbb{R}^d$ and $c \in \mathbb{R}^d$, suppose that P has an extreme point and a minimizer in the direction c. Then some extreme point is a minimizer.*

From this, we have a (terribly slow) algorithm for solving linear programs in standard form. There are only finitely many basic feasible solutions given by enumerating all possible choices for the basis B. Iterating over all B (exponentially many!), we can find one corresponding to the smallest objective value $c_B^T x_B = C_B^T A_B^{-1} b$ that gives a feasible solution

$x_B = A_B^{-1}b \geq 0$. Next time, we will give a better algorithm for solving linear programs called the Simplex Method.

5 Recitation 5: The Simplex Method I

Today's recitation begins building the simplex method for solving linear programs:

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & Ax = b \\ & x \geq 0. \end{array}$$

Recap on Basic Feasible Solutions (BFS) Recall for any standard form polyhedron $P = \{x \in \mathbb{R}^d \mid Ax = b, x \geq 0\}$ with $A \in \mathbb{R}^{m \times d}$, the corners of P are identified by selecting m coordinates to be nonzero. Namely let B, called the basis, be such a choice of indices. Then we have the corresponding point

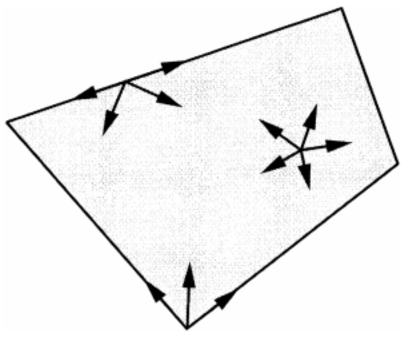
$$x_B = A_B^{-1}b$$

$$x_{\bar{B}} = 0,$$

which we call a **basic solution**. If this point is feasible $A_B^{-1}b \geq 0$ (that is, it lies in P), we say that it is a **basic feasible solution (BFS)**. Each x_i with $i \in B$ is called a basic variable while the other x_j with $j \notin B$ are called nonbasic variables.

Optimality Conditions The main goal today is to build an algorithm for solving linear programs by moving through a sequence of feasible solutions with decreasing objective value. Specifically, we will move from basic feasible solution to basic feasible solution, following a algorithm known as the simplex method.

The simplest thing we need to guarantee then is that we maintain feasibility. We say a vector $d \in \mathbb{R}^d$ is a **feasible direction** at $x \in P$, if there exists $\theta > 0$, such that $x + \theta d \in P$. For example, all of the arrows below denote feasible directions.



We are particularly interested in the feasible directions at a basic feasible solution. At a BFS given by B with corresponding solution $x = (x_B, x_{\bar{B}})$, its natural to consider moving in a direction d that increases one coordinate j that was previously set to zero. Lets suppose $d_j = 1$ for some $j \in \bar{B}$ and $d_i = 0$ for all other $i \in \bar{B}$.

Since we want to maintain feasibility moving in the direction d, we require $A(x+\theta d) = b$. Since x is feasible with $Ax = b$, we must have $Ad = 0$. Hence

$$0 = Ad = A_B d_B + A_j d_j = A_B d_B + A_j.$$

Therefore, we must have $d_B = -A_B^{-1}A_j$. Refer to this direction has the j**th basic direction**.

So far, we have guaranteed that moving in his direction does not violate the equality constraints $Ax = b$. Moreover it cannot violate the negativity constraints for any nonbasic $i \in \bar{B}$, since we have $d_i \geq 0$. However, this is still not enough to conclude that the jth basic direction is a feasible direction since we may violate a basic variables nonnegativity constraint. In particular, we consider this in two cases:

(a) Suppose that the current BFS x is nondegenerate. That is, $x_B > 0$ strictly holds. Then we know that for small enough θ, we must also have $x_B + \theta d_B > 0$ still holds. Hence the jth basic direction is a feasible direction.

(b) Supposing that the current BFS is degenerate, we cannot make a similar guarantee. If some basic variable $x_{B(i)} = 0$ while the direction has $d_{B(i)} < 0$, then moving any positive amount in the direction d_B will lead to an infeasible solution.

For today, we will carry on assuming that all the points visited are nondegenerate as a simplifying assumption. In the following recitation, we will return to this sticking point and fix out methodology to work regardless of whether degeneracy occurs.

So assuming the BFS x is nondegenerate, we find that the basic directions are all feasible. Then the next important question is how the objective value changes as we move in the jth basic direction d. For any $\theta > 0$, we find that the change in objective value is given by

$$c^T(x+\theta d) - c^T x = \theta c^T d = \theta(c_B^T d_b + c_j d_j) = \theta(c_j - c_B^T A_B^{-1} A_j).$$

Hence the rate of change is given by $c_j - c_B^T A_B^{-1} A_j$ where the c_j term corresponds to the improvement from increasing the jth coordinate to be nonzero and the $-c_B^T A_B^{-1} A_j$ component corresponds to the cost of compensating in the basic variables necessitated by the constraint $Ax = b$. This quantity will repeatedly occur, so we give it a name: the **reduced cost** of the jth basic direction is

$$\bar{c}_j = c_j - c_B^T A_B^{-1} A_j.$$

Collecting these values for all j, gives a vector of reduced costs $\bar{c} \in \mathbb{R}^d$.

Notice that each $i \in B$ must have a reduced cost of zero. This follows since $A_B^{-1}A_i$ must equal the ith unit vector e_i, giving

$$\bar{c}_i = c_i - c_B^T A_B^{-1} A_i = c_i - c_B^T e_i = c_i - c_i = 0.$$

In the following result, we see that the reduced cost vector gives us conditions for when a basic feasible solution is a minimizer of the given linear program. This characterization is exact when the given BFS is nondegenerate.

Theorem 5.1. *Consider a basic feasible solution x given by basis B with reduced costs $\bar{c}$.*
(a) If $\bar{c} \geq 0$, then x is optimal.
(b) If x is optimal and nondegenerate, then $\bar{c} \geq 0$.

Proof. We prove these one at a time.

(a) Suppose $\bar{c} \geq 0$ and consider any $y \in P = \{x \mid Ax = b, x \geq 0\}$. Consider the direction $d = y - x$. Since both x and y are feasible, we have $Ad = Ay - Ax = b - b = 0$. This equality can be written as

$$A_B d_B + \sum_{i \in \bar{B}} A_i d_i = 0$$

by considering the basic and nonbasic variables separately. Since A_B is invertible, we can rewrite this as

$$d_B = \sum_{i \in \bar{B}} -A_B^{-1} A_i d_i.$$

Hence we can write the objective change of moving in the direction d as

$$c^T d = c_B^T d_B + \sum_{i \in \bar{B}} c_i d_i = \sum_{i \in \bar{B}} (c_i - c_B^T A_B^{-1} A_i) d_i = \sum_{i \in \bar{B}} \bar{c}_i d_i.$$

Since we have $\bar{c} \geq 0$ and $d_i \geq 0$ for each $i \in \bar{B}$, this sum must be nonnegative. Thus $c^T(y - x) = c^T d \geq 0$. Thus x is optimal.

(b) Suppose x is a nondegenerate basic feasible solution and that some $\bar{c}_j < 0$. Then the jth basic direction d is a feasible direction (since x is nondegenerate) and moving in this direction decreases the objective value at a rate of $\bar{c}_j$. Hence $x - \theta d$ is a feasible point with lower objective than x, and consequently x is not optimal.

□

From this, we say a basis B is **optimal** if the following two conditions hold

$$A_B^{-1} b \geq 0$$

$$c^T - c_B^T A_B^{-1} A \geq 0$$

which correspond to feasibility and optimality of the related basic solution.

Moving To Adjacent BFS Suppose we are at a nondegenerate basic feasible solution x and want to move in the jth basic direction d. Nondegeneracy ensures that this direction is feasible, so some $\theta > 0$ has $x + \theta d \in P$. Lets consider the largest θ that allows us to maintain feasibility, denoted by θ^*, in two cases.

(a) Suppose all $d_i \geq 0$. Then no nonnegativity constraint will ever be violated by increasing the value of θ. Since d was defined to have $Ad = 0$, we can then conclude that the entire ray $\{x + \theta d \mid \theta > 0\}$ is feasible. Hence we have $\theta^* = \infty$.

(b) If some $d_i < 0$, then the nonnegativity constraint $x_i + \theta d_i \geq 0$ provides an upper bound on how large θ can be. Namely we have $\theta^* \leq -x_i/d_i$. Since the equality constraints are always satisfied, the largest possible value for θ^* is given by

$$\theta^* = \min_{\{i \mid d_i < 0\}} -x_i/d_i.$$

Notice that the nonbasic variables all have $d_i \geq 0$ by definition (in fact, they are all zero except for $d_j = 1$). Thus we only need to consider the basic variables in the above formula. Then we can conclude that $\theta^* > 0$ since we have assumed that all basic variables have $x_{B(i)} > 0$ through nondegeneracy.

Lets suppose we are in case (b) above and can only move a finite amount θ^* in the jth basic direction. Some $l \in B$ must attain the maximum defining θ^* and so that coordinate must have

$$(x + \theta^* d)_l = 0$$

while the jth coordinate which used to have value zero now has

$$(x + \theta^* d)_j = \theta_j^* > 0.$$

Thus the point $x + \theta^* d$ can be described by removing one coordinate l from the basis of x and adding in the coordinate j. Hence we have a new basis $B' = B \cup \{j\} \setminus \{l\}$. Below we verify that B' is indeed a basis and corresponds to $x + \theta^* d$.

Theorem 5.2. *The columns $A_{B(i)}$ for $i \neq l$ and A_j are linearly independent and therefore B' is a basis. Further, B' has associated basic feasible solution $x + \theta^* d$.*

Proof. First we observe that the columns $A_{B'}$ are linearly independent if and only if the columns o $A_B^{-1} A_{B'}$ are linearly independent (since we only applied an invertible map). For each $i \in B \setminus \{j\}$, we have

$$A_B^{-1} A_i = e_i$$

where e_i is the ith standard basis vector. Hence all of the columns $i \neq j$ have a zero in the lth component. However, $A_B^{-1} A_j$ is equal to $-d_B$ by definition and we know that d_l is nonzero by definition. Thus $A_B^{-1} A_j$ is independent of from all the other $A_B^{-1} A_i$. ☐

We say that a bases B and B' (like those just constructed) are adjacent if they differ by one element swap.

The Simplex Method The simplex method works iteratively by moving from basic feasible solution to an adjacent basic feasible solution with lower objective value. This is repeated until no adjacent bases offer any improvement, which happens when $\bar{c} \geq 0$. Thus we must be at an optimal basis. This is formalized below.

1. Let B denote the current basis with associated solution $x_B = A_B^{-1} b, x_{\bar{B}} = 0$.

2. Compute the reduced costs $\bar{c}_j = c_j - c_B^T A_B^{-1} A_j$ for each $j \in \bar{B}$.
   If $\bar{c} \geq 0$, then TERMINATE with x as an optimal solution.

3. Compute a basic direction $d = -A_B^{-1} A_j$ for some j with $\bar{c}_j < 0$.
   If $d \geq 0$, then TERMINATE with $x + \theta d$ as a ray verifying the objective is $-\infty$.

4. Compute $\theta^* = \min_{\{i | d_i < 0\}} -x_i/d_i$.

5. Let l be some index with $\theta^* = -x_l/d_l$. Form a new basis for the next iteration $B' = B \cup \{j\} \setminus \{l\}$ with associated solution $y = x + \theta^* d$.

The two critical assumptions to make the above procedure work are that we (i) have an initial basic feasible solution to start from and (ii) never encounter a degenerate basic feasible solution. These two difficulties will be addressed in the coming recitations to give a more robust algorithm that applies to any linear programming problem.

Correctness Assuming Nondegeneracy

Theorem 5.3. *Assuming that all basic feasible solutions of $P = \{x \mid Ax = b, x \geq 0\}$ are nondegenerate, the simplex method terminates in finite time with either*
(a) an optimal basis B and associated basic feasible solution x^*
(b) a vector d satisfying $Ad = 0$, $d \geq 0$, and $c^T d < 0$, and so the optimal cost is $-\infty$.

Proof. If either of the termination conditions are met, our previous discussion verifies that x is optimal (if we terminated in step 2) or the entire ray $x + \theta d$ is feasible with decreasing objective value (if we terminate in step 3). All that we need to show is that eventually one of these conditions is met.

At each iteration the algorithm moves a positive amount θ^* in the direction d with $c^T d < 0$. Therefore each iteration moves to a basic feasible solution with strictly lower objective value than any previous iteration. Since there are only finitely many basic feasible solutions and we can never visit a solution twice, the algorithm must terminate eventually. □

6 Recitation 6: The Simplex Method II

Today's recitation continues building the simplex method for solving linear programs:

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & Ax = b \\ & x \geq 0. \end{array}$$

Last time, we developed the following algorithm:

1. Let B denote the current basis with associated solution $x_B = A_B^{-1} b, x_{\bar{B}} = 0$.

2. Compute the reduced costs $\bar{c}_j = c_j - c_B^T A_B^{-1} A_j$ for each $j \in \bar{B}$.
   If $\bar{c} \geq 0$, then TERMINATE with x as an optimal solution.

3. Compute a basic direction $d = -A_B^{-1} A_j$ for some j with $\bar{c}_j < 0$.
   If $d \geq 0$, then TERMINATE with $x + \theta d$ as a ray verifying the objective is $-\infty$.

4. Compute $\theta^* = \min_{\{i \mid d_i < 0\}} -x_i/d_i$.

5. Let l be some index with $\theta^* = -x_l/d_l$. Form a new basis for the next iteration $B' = B \cup \{j\} \setminus \{l\}$ with associated solution $y = x + \theta^* d$.

We made two main assumptions in our motivation of this method, which we will remove this time: (i) We have an initial basic feasible solution and (ii) We never encounter a degenerate basic feasible solution.

Dual Solutions In The Simplex Method First, we remark that finding an optimal basis for a linear program (as done by the simplex method) provides us with a dual feasible solution as well. In particular, last time we saw that a basis B is optimal if it corresponds to a feasible point

$$x_B = A_B^{-1} b \geq 0$$
$$x_{\bar{B}} = 0$$

and has nonnegative reduced costs

$$\bar{c}^T = c^T - c_B^T A_B^{-1} A \geq 0.$$

For this, we can extract a dual feasible solution y satisfying $c - A^T y \geq 0$ by rewriting the reduced costs as

$$\bar{c} = c - A^T A_B^{-T} c_B \geq 0$$

and so $y = A_B^{-T} c_B$ is dual feasible. Recalling the optimality conditions for conic programs (from lecture), a pair of feasible primal dual solutions x and y are optimal if they satisfy complementary slackness $x^T(c - A^T y) = 0$. Indeed the points given by any basis B satisfy this by definition since each $i \in B$ has $\bar{c}_i = 0$ and each $i \in \bar{B}$ has $x_i = 0$.

From this perspective, the simplex method can be viewed as maintaining a primal feasible solution while pursuing a dual feasible solution $\bar{c} \geq 0$. In a future recitation, we will return the this idea and give a variant of the simplex method that works from the dual perspective. It would maintain a dual feasible solution while pursuing primal feasibility.

Computational Concerns As written the simplex method needs to solve three linear systems at each iteration. Namely, it solves

$$A_B x_B = b$$
$$A_B^T y = c_B$$
$$A_B d = -A_j$$

Using Gaussian elimination, each of these systems would require $O(m^3)$ operations to solve. We can do much better by using the structure we know about the basis A_B.

The most common presentation of the simplex method is done by using a tableau. This approach stores the quantities $A_B^{-1}A$, $A_B^{-1}b$, and $A_B^{-T}c_B$ in a table and then performs elementary row operations to update the table corresponding to each pivot adding some index j and removing some l from the basis B. This requires $O(md)$ memory and only $O(m^2)$ operations to update the tableau.

Rather than focusing on this tableau approach, we consider a method that only stores a representation of A_B^{-1}. This can be done in $O(m^2)$ memory and will still only require $O(m^2)$ operations to update.

Consider some iteration of the simplex method adding an index j and removing an index l. Let $d = -A_B^{-1}A_j$ and so we have $d_l < 0$. Then observe that

$$A_j \left(\frac{1}{-d_l}\right) + \sum_{i \in B \setminus \{l\}} A_i \left(\frac{d_i}{-d_l}\right) = A_l.$$

Hence $A_{B'}^{-1}A_l = \eta$ where $\eta_j = 1/-d_l$ and all other $\eta_i = d_i/d_l$.

Thus for $E_k = [e_1 \dots e_{j-1}, \eta, e_{j+1} \dots e_m]$, we have

$$A_{B'}E_k = A_B.$$

Inverting this equation yields

$$A_{B'}^{-1} = E_k A_B^{-1}.$$

Thus we can update the inverse of our basis by applying a simple matrix with only one column differing from the identity matrix. Applying this update only takes $O(m^2)$ operations and memory. Then all of the necessary operations for the simplex method can be accomplished using matrix vector products with this (that also take at most $O(m^2)$ operations).

Alternatively, the basis matrix after k iterations can have its inverse matrix written as

$$A_{B_k}^{-1} = E_k E_{k-1} \dots E_2 E_1 A_{B_1}^{-1}$$

Periodically, we can collapse all of the matrices E_k and compute the inverse matrix for the current B_k, and the begin building up a product form expression again.

Finding An Initial Feasible Solution One of the assumptions we have made so far in our discussion of the simplex method is that an initial basic feasible solution is known. However this does not hold in general. Indeed it may even be the case that no feasible solution (let alone basic feasible solution) exists and so the method needs to be adapted to handle this.

We accomplish this by using a two phase version of the simplex method. In the first phase, we will apply the simplex method to an auxiliary optimization problem that will either verify infeasibility or give us an initial basic feasible solution. In the second phase, we can then apply the simplex method to solve the original linear program.

Without loss of generality, we can suppose that $b \geq 0$ (by multiplying any equality constraint with a negative b_i by -1). Then we consider the following optimization problem:

$$\begin{array}{ll} \text{minimize} & z_1 + z_2 + \cdots + z_m \\ \text{subject to} & Ax + z = b \\ & x, z \geq 0 \end{array}$$

which has m new auxiliary variables z. Notice that $(x, z) = (0, b)$ is a feasible solution to this linear program and is in fact a basic feasible solution (with basis given by selecting all of the z variables). Hence we have the necessary initial point to apply the simplex method to this problem.

First, suppose that this auxiliary optimization problem has strictly positive minimum value. Then the polyhedron $P = \{x \mid Ax = b, x \geq 0\}$ must be nonempty since any $x \in P$ has $(x, 0)$ feasible for the auxiliary problem with objective value zero. Hence, if our first phase of computation finds positive objective value, we can certify that the original linear program is infeasible.

Now suppose that this auxiliary problem has non positive minimum value. Since we have the constraints that $z \geq 0$, any feasible solution must have nonnegative objective value. Hence the auxiliary problem minimizes with value zero. Moreover, we know that any minimizing (x, z) must have $z = 0$ and so $x \in P$ since $Ax + 0 = b$.

Thus applying the simplex method to solve this auxillary problem yields a feasible solution. In fact, it must yield a basic feasible solution of P since the simplex method always returns a basic feasible solution. That is, the returned (x, z) will have at most m nonzero x and z indices that must have linearly independent columns, and so x has at most m nonzero x entries with linearly independent columns.

Bring this all together gives us a method for solving generic linear programs (infeasible, unbounded, or finite-valued) without assuming any knowledge of feasible solutions a priori.

Cycling From Degenerate Basic Feasible Solutions Lastly we consider the problem of degenerate basic feasible solutions in the simplex method. Our proof of correctness relied on every vertex visited being nondegenerate. This allowed us to assume that θ^* was always positive and so a positive decrease in objective value was attained at each iteration.

However when a BFS is degenerate, the simplex method may find that $\theta^* = 0$ and so the subsequent basis $B \cup \{j\} \setminus \{l\}$ will correspond to the same BFS as the original basis B. This raises the concern that the simplex method may never terminate, instead cycling through a sequence of bases that all correspond to the same solution.

Sadly, this can happen. To construct such an example, we need to fully specify how the simplex method will handle the freedom it has in choosing an index j with $\bar{c}_j < 0$ and an index l with $\theta^* = -x_l/d_l$. These are known as pivoting rules.

Two common heuristics for choosing j are

- Select j with most negative $\bar{c}_j$.
- Select j with steepest rate of decrease $\bar{c}_j/\|d\|$ where $d = -A_B^{-1}A_j$.

Under either of these choices and any selection rule for l, infinite cycling can occur (and so the simplex method is not guaranteed to work). As an example see `https://www.maths.ed.ac.uk/hall/MS-96/MS96010.pdf`

However, Robert Bland (from Cornell ORIE) showed that if careful pivoting rules are used then we can guarantee no basis is ever visited twice. In particular, Bland's pivoting rules are the following:

- Select j with the smallest index number satisfying $\bar{c}_j < 0$.
- Select l with the smallest index number satisfying $\theta^* = -x_l/d_l$.

Theorem 6.1. *Under Bland's pivoting, each basis B is visited at most once by the simplex method. Hence the simplex method terminates in finite time.*

Proof. See `https://people.orie.cornell.edu/dpw/orie6300/Lectures/lec13.pdf`. □

Lastly, we remark on a conceptually different way to handle degeneracy than using pivoting rules. Notice that perturbing the vector b slightly will almost surely remove all degeneracy from the problem. Namely, consider a Gaussian random vector g $N(0, I)$ and some small ϵ. Then setting

$$\bar{b} = b + g\epsilon$$

we have that each basis corresponds to the basic solution

$$\bar{x}_B = A_B^{-1}(b + g\epsilon) = x_B + A_B^{-1}g\epsilon.$$

Since this last term is gaussian, we almost surely have no entry in $\bar{x}_B$ takes on value zero. Hence no basic feasible solution is degenerate.

Taking sufficiently small epsilon, one can solve a nearly equivalent problem without worrying about degeneracy. Ever better, one can consider ϵ symbolically to be an infinitesimal amount, smaller than any constant, and then use symbolic arithmetic throughout the simplex method. The final solution from such an approach would then also be a solution for the original problem.

7 Recitation 7: CVX Programming

This recitation was given by Lijun Ding. See `https://people.orie.cornell.edu/dsd95/orie6300.html` where the code used in class is provided.

8 Recitation 8: Duality in Linear Programming

In today's recitation, we are going to discuss two ways of viewing the dual of a linear program. Then informed by these methods we will give a dual simplex method, mirroring the primal simplex method discussed in previous recitations.

Geometry of Dual Linear Programs Consider the primal problem

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & a_i^T x \geq b_i \text{ for } i = 1 \dots m \end{array}$$

where $x \in \mathbb{R}^n$ and the vectors a_i span all of $\mathbb{R}^n$. Then the corresponding dual problem is

$$\begin{array}{ll} \text{maximize} & p^T x \\ \text{subject to} & \sum p_i a_i = c \\ & p \geq 0. \end{array}$$

A vertex of the primal feasible region is given by selecting n of the linearly independent constraints $I \subseteq [m]$ to be met with equality. Then the corresponding solution is the unique solution to the system $a_i^T x = b_i$ for $i \in I$. For the sake of gaining insight in to what dual solutions to this problem look like, we will assume the resulting x_I is nondegenerate (that is, all $i \notin I$ have $a_i^T x > b_i$ strictly).

For this primal solution x_I to be optimal, the strong duality of linear programs ensures that there exists a corresponding dual solution p. Namely, together these need to satisfy the optimality conditions

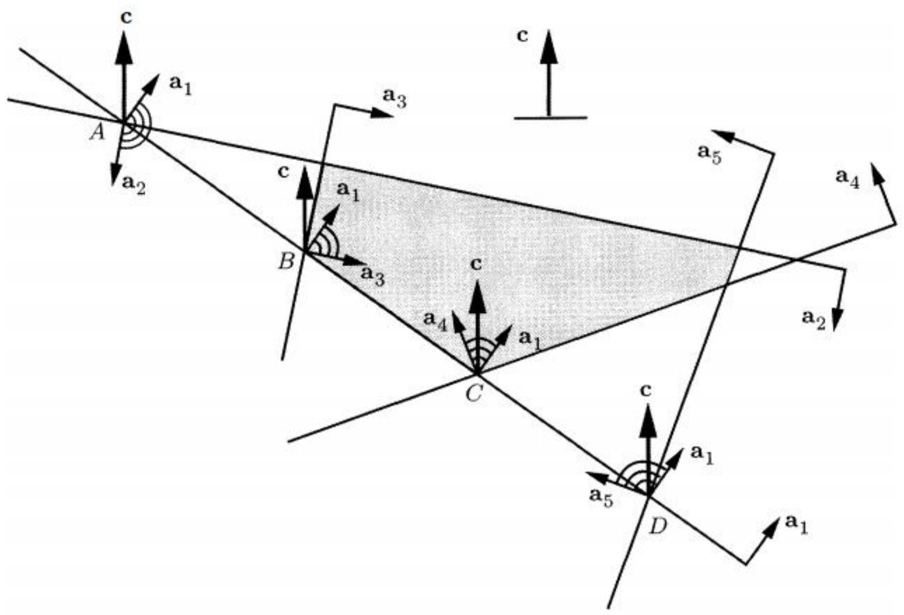
$$a_i^T x_I \geq b_i$$

$$p_i \text{ for all } i \notin I$$

$$\sum p_i a_i = c$$
$$p \geq 0$$

for primal feasibility, complementary slackness, and dual feasibility, respectively.

From these conditions, we can geometrically identify what it means to be dual feasible. The complementary slackness condition ensures that only the p_i corresponding to constraints in I play a role in our solution. Then the dual feasibility constraints say that c is a conic combination of the vectors a_i for $i \in I$. An example of this condition is given below, where each basic solution can easily have its primal and dual feasibility checked as whether the point lies in the feasible region and whether the vector c pointing up lies in the cone generated by a_i.



Shadow Prices from Dual Linear Programs A second way to view the dual solutions of a linear program comes in the form of shadow prices. Consider a standard form linear program

$$\begin{array}{ll} \text{minimize} & c^T x \\ \text{subject to} & Ax = b \\ & x \geq 0 \end{array}$$

Lets suppose that this problem has a nondegenerate optimal solution x corresponding to the basis B. Then we know that the primal and dual feasibility conditions hold:

$$A_B^{-1} b > 0$$

$$c - c_B^T A_B^{-1} A \geq 0.$$

Then consider the perturbed optimization problem given by replacing b by $b + d$ for some small d. Notice the objective value of this perturbed problem is given by $\texttt{val}(b+d)$. Then we observe that for sufficiently small d, we will not have violated the primal feasibility condition

$$A_B^{-1}(b + d) > 0.$$

Further, the dual feasibility conditions are independent of the right-hand side vector that we select. Hence dual feasibility still holds for $b + d$. Thus B is still an optimal basis for slightly perturbed problem instances. In particular, we then know that for small values of d

$$val(b + d) = p^T(b + d)$$

where $p = c_B^T A_B^{-1}$ are the dual values associated with the basis B. That is, the value function of the linear program is linear near b. Moreover, the coefficients of this linear function are given exactly by the dual variables.

From this, we can interpret the dual values as indicating the importance of each constraint in the original problem. For example, suppose that each b_i corresponded to a limit how much of material i a factory has to work with. Then the dual variables indicate that providing the factory with one additional unit of material i would improve the objective value by p_i. Hence, these values are often referred to as *shadow prices*, as they establish the marginal value of each constraint.

The Dual Simplex Method Lastly, we consider a variation of the simplex method that works on the dual problem rather than the primal. Recall the simplex method as previously developed was given by

1. Let B denote the current basis with associated solution $x_B = A_B^{-1}b, x_{\bar{B}} = 0$.

2. Compute the reduced costs $\bar{c}_j = c_j - c_B^T A_B^{-1} A_j$ for each $j \in \bar{B}$.
   If $\bar{c} \geq 0$, then TERMINATE with x as an optimal solution.

3. Compute a basic direction $d = -A_B^{-1} A_j$ for some j with $\bar{c}_j < 0$.
   If $d \geq 0$, then TERMINATE with $x + \theta d$ as a ray verifying the objective is $-\infty$.

4. Compute $\theta^* = \min_{\{i | d_i < 0\}} -x_i/d_i$.

5. Let l be some index with $\theta^* = -x_l/d_l$. Form a new basis for the next iteration $B' = B \cup \{j\} \setminus \{l\}$ with associated solution $y = x + \theta^* d$.

This procedure maintains a basis that is primal feasible $A_B^{-1}b$ and pursues a basis that is dual feasible $c - c_B^T A_B^{-1} A \geq 0$. This is done by repeatedly picking an index j with $c_j - c_B^T A_B^{-1} A_j < 0$ to add in to the basis, which forces some x_l to take value zero and thus leave the basis.

We could give a mirrored algorithm if we had an original basis that was dual feasible rather than primal feasible. Namely, we could maintain a basis with $c - c_B^T A_B^{-1} A \geq 0$ and pursue a basis that is primal feasible $A_B^{-1} b \geq 0$. So at each iteration, we would need to pick an index l failing to be satisfy the nonnegativity constraint $(A_B^{-1}b)_l < 0$. Then as we remove l from the basis, some j will have $c_j - c_B^T A_B^{-1} A_j$ take value zero and thus leaves the basis.

This inverted version of the simplex method is formalized as follows

1. Let B denote the current basis with associated solution $\bar{c} = c - c_B^T A_B^{-1} A$.

2. Compute the primal values $x_B = A_B^{-1} b$.
   If $x_B \geq 0$, then TERMINATE with this as an optimal solution.

3. Compute the direction $v = -e_l^T A_B^{-1} A$ for some l with $x_l < 0$.
   If $v \geq 0$, then TERMINATE with $\bar{c} + \theta v$ as a ray verifying the objective is ∞.

4. Compute $\theta^* = \min_{\{i | v_i < 0\}} -\bar{c}_i / v_i$.

5. Let j be some index with $\theta^* = -x_j/d_j$. Form a new basis for the next iteration $B' = B \cup \{l\} \setminus \{j\}$ with associated solution $\bar{c}' = \bar{c} + \theta^* v$.

This method is equivalent to reformulating the dual linear program into standard form and then applying the regular simplex method. In particular, consider the formulation

$$\begin{array}{ll} \text{minimize} & b^T p \\ \text{subject to} & c - A^T p = s \\ & s \geq 0, p \text{ free} \end{array}$$

Every basis for this problem comes from selecting $n - m$ of the slack variables s to allow to be nonzero. Zero slack means that the corresponding regular dual constraint $c - A^T p \geq 0$ is tight. Then by complementary slackness, this related primal x_i is allowed to be nonzero. Hence selecting $n - m$ slack variables for a dual basis, exactly defines a choice of m indices for a basis of the original linear program.

From this one-to-one mapping between primal and dual formulation bases, we can directly read the dual simplex method presented above as an the simplex method applied to this standard form version of the dual linear program. Hence, the dual simplex method still corresponds geometrically to stepping from corner to corner of a polyhedron (just using the dual polyhedron instead of the primal polyhedron).

Finally, we remark that the dual simplex method is a reasonable choice to use if a dual feasible solution is known, but not a primal feasible solution. This would allow you to skip solving an additional initialization problem as we discussed in Recitation 6. One example of when this would occur is if you are solving many perturbed versions of the same linear program. If you want to solve the original linear program for many alternative right-handside vectors $b_1, \ldots, b_k$, then each one would need its own primal initial solution for the normal simplex method. However, as observed in our discussion of shadow prices, modifying the vector b has no effect on dual feasibility. Thus we can use the optimal dual solution from the original linear program as a starting point for each of our perturbed problems. This has the additional benefit that if b_1 is near b, we expect this initialization to be near optimal and (hopefully) require very few pivots to reach optimality.

9 Recitation 9: Transportation Problems

In today's recitation and the following one, we are going to discuss applying the simplex method to solve network optimization problems. Today we will specifically consider the following transportation problem.

Suppose there is a set, S, of suppliers, each with supply s_i and a set, D, of customers each with demand d_j that must be met. Further, suppose that each unit shipped from supply node $i \in S$ to demand node $j \in D$ incurs a cost $c_i j$. The goal is then to find a minimum-cost shipping scheme that satisfies all the demand and supply restrictions.

Clearly, this problem is only feasible if $\sum s_i \geq \sum d_j$. Without loss of generality, suppose $\sum s_i = \sum d_j$ since otherwise we can set a dummy demand node k with $d_k = \sum s_i - \sum d_j$. Then we can formulate this as an linear program by

$$\begin{array}{ll} \text{minimize} & \sum_{i \in S, j \in D} c_{ij} x_{ij} \\ \text{subject to} & \sum_{j \in D} x_{ij} = s_i \;\text{ for all } i \in S \\ & \sum_{i \in S} x_{ij} = d_j \;\text{ for all } j \in D \\ & x_{ij} \geq 0 \end{array}$$

One may additionally want to constrain that the variables x_{ij} are integral, depending on the type of commodity being transported. For our purposes, we not consider this extra complexity.

Observe that the constrains of this linear program are linearly dependent as written. This is because adding all of the rows of the first constraint set gives the same result as adding all of the rows of the second constraint set:

$$\sum s_i = \sum_{i,j} x_{ij} = \sum d_j.$$

Hence, we can delete any one of the constraints without changing the feasible region. Therefore, letting $m = |S|$ and $n = |D|$, we really only have $n + m - 1$ constraints.

This problem can be represented graphically where G is a bipartite graph with vertex set $S \cup D$ and edges $E = S \times D$. Note then that the decision variables correspond to each edge in the graph. Hence basic solutions to this linear program correspond to choosing $n+m-1$ edges to give non-zero value, inducing a subgraph.

Lemma 9.1. *The subgraph corresponding to a basic solution of the linear program does not contain any cycles.*

Proof. Consider any cycle C among in the graph defining our problem. Notice that since the graph is bipartite, the cycle C must have even length. Then define a vector h with values $+1$ and -1 alternating along each edge in the cycle. This vector must satisfy

$$A_C h = 0$$

since each node has one incoming edge and one out going edge with values -1 and 1. Hence the edges of any cycle have linearly dependent columns, and so no basis could include a cycle. □

From this we conclude that the subgraph corresponding to each basis must be a tree (that is, it is connected and acyclic). Below we formalize why this is the case.

Lemma 9.2. *(a) A graph G is a tree if and only if*
(b) it is acyclic with $n + m - 1$ edges if and only if
(c) it is connected with $n + m - 1$ edges.

Proof. We prove this inductively. Supposing G has $|V| = 1$ nodes, then trivially all three conditions holds. Now consider a graph with $|V| > 1$ and suppose this equivalence holds for all smaller graphs.

(a)$\Rightarrow$(b) Every tree has at least one leaf node, that is a node with a single edge incident to it (*check this yourself*). So we can construct a smaller tree G' by deleting a leaf node and its incident edge from G. Our inductive hypothesis ensures G' is acyclic with $n+m-2$ edges. Then adding this left back in cannot create a cycle, so G is acyclic with $n+m-1$ edges.

(b)$\Rightarrow$(c) Suppose to the contrary that G is not connected. Then we can consider the connected components of G, given by $G_1 \dots G_k$. Each of these components is then a tree (since they are acyclic and connected). Applying our inductive hypothesis shows each of them has $|V_k|-1$ edges. However, this implies that all together they have $\sum |V_k| - 1 = n+m-k$ edges, which contracts our assumption that the graph has $n+m-1$ edges.

(c)$\Rightarrow$(a) Suppose to the contrary that the graph G has a cycle C. Then we can delete any edge from C without causing the graph to become disconnected. Repeating this process will eventually produce a subgraph of G that is acyclic and connected (and thus has $n+m-1$ edges). However, the original graph had $n+m-1$ edges. Hence it must not have had any cycles. □

Thus each basis can be viewed as a spanning tree on the set of nodes $S \cup D$ for our transportation problem. The reduced costs corresponding to such a basis B are given by considering the dual linear program:

$$\begin{array}{ll} \text{maximize} & s^T u + d^T v \\ \text{subject to} & u_i + v_j \le c_{ij} \text{ for all } i \in S, j \in D. \end{array}$$

For a given basis B, that is, spanning tree of the problem's bipartite graph, we can easily compute the related dual problem solution as follows:

Recall that the primal constraints are linearly dependent. Hence the dual variables above are linearly dependent, and so we can set one of them arbitrarily. Then we can compute the dual variable for each node adjacent to that one (in the spanning tree B) by the corresponding edge equation $u_i + v_j = c_{ij}$. Note this equation must be met with equality by complementary slackness for the basis B. Repeating this process iteratively across the spanning tree will determine the dual solution corresponding to B without needing to compute any matrix inverse.

Hence we can compute reduced costs for the simplex method more efficiently than the method usually does. Putting this all together gives a more efficient version of the simplex method that only needs to store a spanning tree, rather than an inverse matrix. Formally, we have:

1. Let B denote the current spanning tree (basis).
2. Choose a dual variable to set arbitrarily (for example, $u_1 = 7$). Solve for the other dual variables to maintain complementary slackness: if x_{ij} is in the basis, then $u_i + v_j = c_{ij}$.
3. For every nonbasic edge, compute the reduced costs $\bar{c}_{ij} = c_{ij} - u_i - v_j$.
4. Pick any ij with $\bar{c}_{ij} < 0$. Let C be the cycle in $B \cup \{ij\}$.

5. Going around C, increase the flow on x_{ij}, then decrease on the next edge, then increase, and so on. Let $\delta = \min\{f_k \mid k \text{ is a decreasing edge}\}$. Increase all forward edges by δ and decrease the decreasing edges by δ. Add ij to the spanning tree and remove an edge that achieves the minimum.

10 Recitation 10: Network Flows

In today's recitation, we will continue discussing applying the simplex method to solve network optimization problems. Today we consider the following network flow problem (generalizing the transportation problem from last week).

Suppose we have a directed graph $G = (V, E)$ with n nodes and m arcs. Additionally, there is a vector b of supplies available at each node (where $b_i < 0$ corresponds to a demand). Finally, there is a vector of costs c such that for each edge $e \in E$, c_e is the cost of sending one unit of flow along edge e. Assume the graph is connected and

$$\sum b_i = 0,$$

i.e., the total supply equals the total demand.

Note that the transportation problem discussed last week can be considered a special case of problems under this framework. (Set of supply nodes, $S = i \in V : b_i > 0$, demand nodes $D = V\ S, E = (i,j) : i \in S, j \in D$.) A vector of flows, $f \in \mathbb{R}^m$ is feasible for this problem if $f \geq 0$ and the flow constraints for each vertex i are satisfied, i.e.:

$$\sum_{ij \in E} f_{ij} - \sum_{ji \in E} f_{ji} = b_i$$

If we define our matrix A by

$$A_{ik} = \begin{cases} 1 & \text{if } i \text{ is the start node of edge } k \\ -1 & \text{if } i \text{ is the end node of edge } k \\ 0 & \text{otherwise,} \end{cases}$$

then we can formulate this as an linear program by

$$\begin{array}{ll} \text{minimize} & c^T f \\ \text{subject to} & Af = b \\ & f \geq 0. \end{array}$$

Just like with the transportation problem, it is easy to verify that this matrix is not full rank, but by deleting one constraint row arbitrarily, it will be full rank assuming the graph is connected.

Cycles and Basic Feasible Solutions. Much like we saw last week for the special case of the transportation problem subproblem, edges that form a cycle have linearly dependent columns.

Lemma 10.1. *For any undirected cycle C in G, the columns corresponding to the arcs of C are linearly dependent.*

Proof. Define a vector h with value $+1$ on each forward arc in the cycle and -1 on each backwards arc in the cycle. This vector must satisfy

$$A_C h = 0$$

since each node has one incoming edge and one out going edge with values -1 and 1. Hence the edges of any cycle have linearly dependent columns. $\square$

From this we conclude that the undirected subgraph corresponding to each basis must be a tree (that is, it is connected and acyclic). This follows from our lemma from last week that showed trees are exactly acyclic graphs with $n-1$ edges. Thus we have a combinatorial view of the basis being used at each iteration of the simplex method applied to this network problem.

Reduced Costs and Network Simplex. The reduced costs corresponding to such a basis (spanning tree) B are given by considering the dual linear program:

$$\begin{array}{ll} \text{maximize} & b^T w \\ \text{subject to} & w_i - w_j \leq c_{ij} \text{ for all } ij \in E. \end{array}$$

For a given basis B, we can compute the complementary dual solution as follows: Recall that the primal constraints are linearly dependent. Hence the dual variables above are linearly dependent, and so we can set one of them arbitrarily. Then we can compute the dual variable for each node adjacent to that one (in the spanning tree B) by the corresponding edge equation $u_i + v_j = c_{ij}$. Note this equation must be met with equality by complementary slackness for the basis B. Repeating this process iteratively across the spanning tree will determine the dual solution corresponding to B without needing to compute A_B^{-1}, which will not be sparse like A_B.

Hence we can compute reduced costs for the simplex method more efficiently than the method usually does. Putting this all together gives a more efficient version of the simplex method that only needs to store a spanning tree, rather than an inverse matrix. Formally, we have the following generalization of the network simplex method from last week:

1. Let B denote the current spanning tree (basis).
2. Choose a dual variable to set arbitrarily (for example, $u_1 = 7$). Solve for the other dual variables to maintain complementary slackness: if x_{ij} is in the basis, then $u_i + v_j = c_{ij}$.
3. For every nonbasic edge, compute the reduced costs $\bar{c}_{ij} = c_{ij} - u_i - v_j$.
4. Pick any ij with $\bar{c}_{ij} < 0$. Let C be the cycle in $B \cup \{ij\}$.
5. Going around C, increase the flow each forward edge and decrease it on each backwards arc. Let $\delta = \min\{f_k \mid k \text{ is a backwards edge}\}$. Increase all forward edges by δ and decrease the decreasing edges by δ. Add ij to the spanning tree and remove an edge that achieves the minimum.