18.415/6.854 Advanced Algorithms October 1994

Linear Programming

Lecturer: Michel X. Goemans

1 An Introduction to Linear Programming

Linear programming is a very important class of problems, both algorithmically and combinatorially. Linear programming has many applications. From an algorithmic point-of-view, the simplex was proposed in the forties (soon after the war, and was motivated by military applications) and, although it has performed very well in practice, is known to run in exponential time in the worst-case. On the other hand, since the early seventies when the classes P and NP were defined, it was observed that linear programming is in NP$\cap$ co-NP although no polynomial-time algorithm was known at that time. The first polynomial-time algorithm, the ellipsoid algorithm, was only discovered at the end of the seventies. Karmarkar's algorithm in the mid-eighties lead to very active research in the area of interior-point methods for linear programming. We shall present one of the numerous variations of interior-point methods in class. From a combinatorial perspective, systems of linear inequalities were already studied at the end of the last century by Farkas and Minkovsky. Linear programming, and especially the notion of duality, is very important as a proof technique. We shall illustrate its power when discussing approximation algorithms. We shall also talk about network flow algorithms where linear programming plays a crucial role both algorithmically and combinatorially. For a more in-depth coverage of linear programming, we refer the reader to [1, 4, 7, 8, 5].

A linear program is the problem of optimizing a linear objective function in the decision variables, $x_1 \dots x_n$, subject to linear equality or inequality constraints on the x_i's. In standard form, it is expressed as:

$$\begin{array}{llll} \text{Min} & \sum_{j=1}^{n} c_j x_j & & \text{(objective function)} \\ \text{subject to:} & & & \\ & \sum_{j=1}^{n} a_{ij} x_j = b_i, & i = 1 \dots m & \text{(constraints)} \\ & x_j \geq 0, & j = 1 \dots n & \text{(non-negativity constraints)} \end{array}$$

where $\{a_{ij}, b_i, c_j\}$ are given.

A linear program is expressed more conveniently using matrices:

$$\min c^T x \;\; \text{subject to} \;\; \left\{ \begin{array}{rcl} Ax & = & b \\ x & \geq & 0 \end{array} \right.$$

where

$$
\begin{aligned}
x &= \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} && \in \mathbb{R}^{n\times 1} \\
b &= \begin{pmatrix} b_1 \\ \vdots \\ b_m \end{pmatrix} && \in \mathbb{R}^{m\times 1} \\
c &= \begin{pmatrix} c_1 \\ \vdots \\ c_n \end{pmatrix} && \in \mathbb{R}^{n\times 1} \\
A &= \begin{pmatrix} a_{11} & & \\ & \ddots & \\ & & a_{mn} \end{pmatrix} && \in \mathbb{R}^{m\times n}
\end{aligned}
$$

2 Basic Terminology

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

Definition 2 *A linear program (LP) is feasible if there exists a feasible solution, otherwise it is said to be* **infeasible**.

Definition 3 *An* **optimal solution** *x^* is a feasible solution s.t. $c^T x^* = \min\{c^T x : Ax = b, x \geq 0\}$.*

Definition 4 *LP is unbounded (from below) if $\forall \lambda \in \mathbb{R}$, $\exists$ a feasible x^* s.t. $c^T x^* \leq \lambda$.*

3 Equivalent Forms

A linear program can take on several forms. We might be maximizing instead of minimizing. We might have a combination of equality and inequality contraints. Some variables may be restricted to be non-positive instead of non-negative, or be unrestricted in sign. Two forms are said to be equivalent if they have the same set of optimal solutions or are both infeasible or both unbounded.

1. A maximization problem can be expressed as a minimization problem.

$$\max c^T x \Leftrightarrow \min -c^T x$$

2. An equality can be represented as a pair of inequalities.

$$
a_i^T x = b_i \quad
\begin{aligned}
&\Leftrightarrow \left\{ \begin{array}{rcr} a_i^T x & \leq & b_i \\ a_i^T x & \geq & b_i \end{array} \right. \\
&\Leftrightarrow \left\{ \begin{array}{rcr} a_i^T x & \leq & b_i \\ -a_i^T x & \leq & -b_i \end{array} \right.
\end{aligned}
$$

3. By adding a slack variable, an inequality can be represented as a combination of equality and non-negativity constraints.

$$a_i^T x \le b_i \Leftrightarrow a_i^T x + s_i = b_i,\ s_i \ge 0.$$

4. Non-positivity constraints can be expressed as non-negativity constraints.

   To express $x_j \le 0$, replace x_j everywhere with $-y_j$ and impose the condition $y_j \ge 0$.

5. x may be unrestricted in sign.

   If x is unrestricted in sign, i.e. non-positive or non-negative, everywhre replace x_j by $x_j^+ - x_j^-$, adding the constraints $x_j^+,\ x_j^- \ge 0$.

In general, an inequality can be represented using a combination of equality and non-negativity constraints, and *vice versa.*

Using these rules, $\min\left\{c^T x \text{ s.t. } Ax \ge b\right\}$ can be transformed into $\min\left\{c^T x^+ - c^T x^- \text{ s.t. } Ax^+ - Ax^- - Is = b,\ x^+, x^-, s \ge 0\right\}$. The former LP is said to be in **canonical** form, the latter in **standard** form.

Conversely, an LP in standard form may be written in canonical form. $\min\left\{c^T x \text{ s.t. } Ax = b,\ x \ge 0\right\}$ is equivalent to $\min\left\{c^T x \text{ s.t. } Ax \ge b,\ -Ax \ge -b,\ Ix \ge 0\right\}$. This may be rewritten as $A'x \ge b'$, where $A' = \begin{pmatrix} \text{A} \\ \text{-A} \\ \text{I} \end{pmatrix}$ and $b' = \begin{pmatrix} \text{b} \\ \text{-b} \\ \text{0} \end{pmatrix}$.

4 Example

Consider the following linear program:

$$\min x_2 \text{ subject to } \left\{ \begin{array}{rcrcl} x_1 & & & \ge & 2 \\ 3x_1 & - & x_2 & \ge & 0 \\ x_1 & + & x_2 & \ge & 6 \\ -x_1 & + & 2x_2 & \ge & 0 \end{array} \right.$$

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. We shall formalize this notion in the next section.

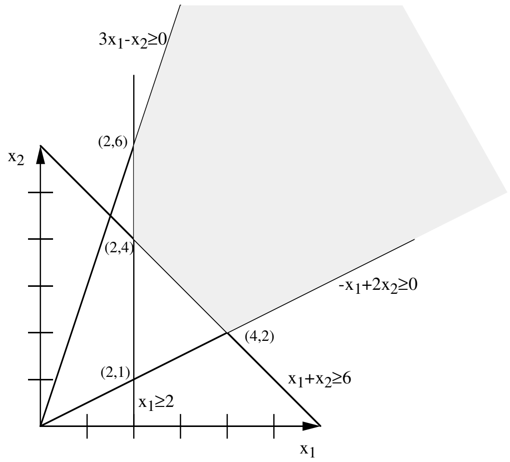

Figure 1: Graph representing primal in example.

An example of an infeasible linear program can be obtained by reversing some of the inequalities of the above LP:

$$\begin{array}{rcrcl} x_1 & & & \leq & 2 \\ 3x_1 & - & x_2 & \geq & 0 \\ x_1 & + & x_2 & \geq & 6 \\ -x_1 & + & 2x_2 & \leq & 0. \end{array}$$

5 The Geometry of LP

Let $P = \{x : Ax = b,\ x \geq 0\} \subseteq \mathbb{R}^n$.

Definition 5 *x is a* **vertex** *of P if $\nexists y \neq 0$ s.t. $x + y,\ x - y \in P$.*

Theorem 1 *Assume* $\min\{c^T x : x \in P\}$ *is finite, then $\forall x \in P, \exists$ a vertex x' such that $c^T x' \leq c^T x$.*

Proof:

If x is a vertex, then take $x' = x$.

If x is not a vertex, then, by definition, $\exists y \neq 0$ s.t. $x + y,\ x - y \in P$. Since $A(x + y) = b$ and $A(x - y) = b$, $Ay = 0$.

WLOG, assume $c^T y \leq 0$ (take either y or $-y$). If $c^T y = 0$, choose y such that $\exists j$ s.t. $y_j < 0$. Since $y \neq 0$ and $c^T y = c^T(-y) = 0$, this must be true for either y or $-y$.

Consider $x + \lambda y,\ \lambda > 0$. $c^T(x + \lambda y) = c^T x + \lambda c^T y \leq c^T x$, since $c^T y$ is assumed non-positive.

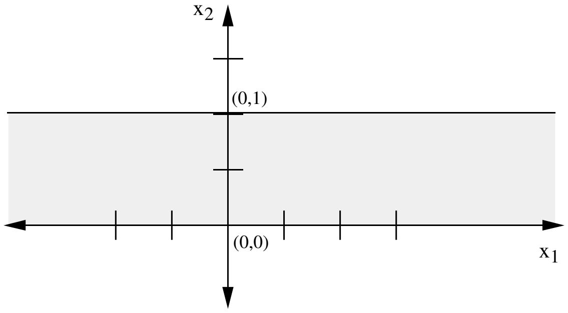

Figure 2: A polyhedron with no vertex.

Case 1 $\exists j$ such that $y_j < 0$

As λ increases, component j decreases until $x + \lambda y$ is no longer feasible.

Choose $\lambda = \min_{\{j: y_j < 0\}} \{x_j / - y_j\} = x_k / - y_k$. This is the largest λ such that $x + \lambda y \geq 0$. Since $Ay = 0$, $A(x + \lambda y) = Ax + \lambda Ay = Ax = b$. So $x + \lambda y \in P$, and moreover $x + \lambda y$ has one more zero component, $(x + \lambda y)_k$, than x.

Replace x by $x + \lambda y$.

Case 2 $y_j \geq 0 \; \forall j$

By assumption, $c^T y < 0$ and $x + \lambda y$ is feasible for all $\lambda \geq 0$, since $A(x + \lambda y) = Ax + \lambda Ay = Ax = b$, and $x + \lambda y \geq x \geq 0$. But $c^T(x + \lambda y) = c^T x + \lambda c^T y \to -\infty$ as $\lambda \to \infty$, implying LP is unbounded, a contradiction.

Case 1 can happen at most n times, since x has n components. By induction on the number of non-zero components of x, we obtain a vertex x'.

□

Remark: The theorem was described in terms of the polyhedral set $P = \{x : Ax = b : x \geq 0\}$. Strictly speaking, the theorem is not true for $P = \{x : Ax \geq b\}$. Indeed, such a set P might not have any vertex. For example, consider $P = \{(x_1, x_2) : 0 \leq x_2 \leq 1\}$ (see Figure 2). This polyhedron has no vertex, since for any $x \in P$, we have $x + y$, $x - y \in P$, where $y = (1, 0)$. It can be shown that P has a vertex iff $Rank(A) = n$. Note that, if we transform a program in canonical form into standard form, the non-negativity constraints imply that the resulting matrix A has full column rank, since

$$Rank \begin{bmatrix} A \\ -A \\ I \end{bmatrix} = n.$$

Corollary 2 *If* $\min\{c^T x : Ax = b, x \geq 0\}$ *is finite, There exists an optimal solution,* x^**, which is a vertex.*

Proof:

Suppose not. Take an optimal solution. By Theorem 1 there exists a vertex costing no more and this vertex must be optimal as well. $\square$

Corollary 3 *If* $P = \{x : Ax = b, x \geq 0\} \neq \emptyset$*, then* P *has a vertex.*

Theorem 4 *Let* $P = \{x : Ax = b, x \geq 0\}$*. For* $x \in P$*, let* A_x *be a submatrix of* A *corresponding to* j *s.t.* $x_j > 0$*. Then* x *is a vertex iff* A_x *has linearly independent columns. (i.e.* A_x *has full column rank.)*

Example $A = \begin{bmatrix} 2 & 1 & 3 & 0 \\ 7 & 3 & 2 & 1 \\ 0 & 0 & 0 & 5 \end{bmatrix}$ $x = \begin{bmatrix} 2 \\ 0 \\ 1 \\ 0 \end{bmatrix}$ $A_x = \begin{bmatrix} 2 & 3 \\ 7 & 2 \\ 0 & 0 \end{bmatrix}$, and x is a vertex.

Proof:

Show $\neg i \rightarrow \neg ii$.

Assume x is not a vertex. Then, by definition, $\exists y \neq 0$ s.t. $x + y,\ x - y \in P$. Let A_y be submatrix corresponding to non-zero components of y.

As in the proof of Theorem 1,

$$\left.\begin{array}{ccccc} Ax & + & Ay & = & b \\ Ax & - & Ay & = & b \end{array}\right\} \Rightarrow Ay = 0.$$

Therefore, A_y has dependent columns since $y \neq 0$.

Moreover,

$$\left.\begin{array}{ccccc} x & + & y & \geq & 0 \\ x & - & y & \geq & 0 \end{array}\right\} \Rightarrow y_j = 0 \text{ whenever } x_j = 0.$$

Therefore A_y is a submatrix of A_x. Since A_y is a submatrix of A_x, A_x has linearly dependent columns.

Show $\neg ii \rightarrow \neg i$.

Suppose A_x has linearly dependent columns. Then $\exists y$ s.t. $A_x y = 0,\ y \neq 0$. Extend y to $\mathbb{R}^n$ by adding 0 components. Then $\exists y \in \mathbb{R}^n$ s.t. $Ay = 0,\ y \neq 0$ and $y_j = 0$ wherever $x_j = 0$.

Consider $y' = \lambda y$ for small $\lambda \geq 0$. Claim that $x + y',\ x - y' \in P$, by argument analogous to that in Case 1 of the proof of Theorem 1, above. Hence, x is not a vertex.

$\square$

6 Bases

Let x be a vertex of $P = \{x : Ax = b, x \geq 0\}$. Suppose first that $|\{j : x_j > 0\}| = m$ (where A is $m \times n$). In this case we denote $B = \{j : x_j > 0\}$. Also let $A_B = A_x$; we use this notation not only for A and B, but also for x and for other sets of indices. Then A_B is a square matrix whose columns are linearly independent (by Theorem 4), so it is non-singular. Therefore we can express x as $x_j = 0$ if $j \notin B$, and since $A_B x_B = b$, it follows that $x_B = A_B^{-1} b$. The variables corresponding to B will be called *basic*. The others will be referred to as *nonbasic*. The set of indices corresponding to nonbasic variables is denoted by $N = \{1, \ldots, n\} - B$. Thus, we can write the above as $x_B = A_B^{-1} b$ and $x_N = 0$.

Without loss of generality we will assume that A has full row rank, $\text{rank}(A) = m$. Otherwise either there is a redundant constraint in the system $Ax = b$ (and we can remove it), or the system has no solution at all.

If $|\{j : x_j > 0\}| < m$, we can augment A_x with additional linearly independent columns, until it is an $m \times m$ submatrix of A of full rank, which we will denote A_B. In other words, although there may be less than m positive components in x, it is convenient to always have a *basis* B such that $|B| = m$ and A_B is non-singular. This enables us to always express x as we did before, $x_N = 0$, $x_B = A_B^{-1} b$.

Summary x is a vertex of P iff there is $B \subseteq \{1, \ldots, n\}$ such that $|B| = m$ and

1. $x_N = 0$ for $N = \{1, \ldots, n\} - B$
2. A_B is non-singular
3. $x_B = A_B^{-1} b \geq 0$

In this case we say that x is a *basic feasible solution*. Note that a vertex can have several basic feasible solution corresponding to it (by augmenting $\{j : x_j > 0\}$ in different ways). A basis might not lead to any basic feasible solution since $A_B^{-1} b$ is not necessarily nonnegative.

Example:

$$\begin{aligned} x_1 + x_2 + x_3 &= 5 \\ 2x_1 - x_2 + 2x_3 &= 1 \\ x_1, x_2, x_3 &\geq 0 \end{aligned}$$

We can select as a basis $B = \{1, 2\}$. Thus, $N = \{3\}$ and

$$\begin{aligned} A_B &= \begin{pmatrix} 1 & 1 \\ 2 & -1 \end{pmatrix} \\ A_B^{-1} &= \begin{pmatrix} \frac{1}{3} & \frac{1}{3} \\ \frac{2}{3} & \frac{-1}{3} \end{pmatrix} \\ A_B^{-1} b &= \begin{pmatrix} 2 \\ 3 \end{pmatrix} \\ x &= (2, 3, 0) \end{aligned}$$

Remark. A crude upper bound on the number of vertices of P is $\binom{n}{m}$. This number is exponential (it is upper bounded by n^m). We can come up with a tighter approximation of $\binom{n-\frac{m}{2}}{\frac{m}{2}}$, though this is still exponential. The reason why the number is much smaller is that most basic solutions to the system $Ax = b$ (which we counted) are not feasible, that is, they do not satisfy $x \geq 0$.

7 The Simplex Method

The Simplex algorithm [Dantzig,1947] [2] solves linear programming problems by focusing on basic feasible solutions. The basic idea is to start from some vertex v and look at the adjacent vertices. If an improvement in cost is possible by moving to one of the adjacent vertices, then we do so. Thus, we will start with a bfs corresponding to a basis B and, at each iteration, try to improve the cost of the solution by removing one variable from the basis and replacing it by another.

We begin the Simplex algorithm by first rewriting our LP in the form:

$$\begin{array}{ll} \min & c_B x_B + c_N x_N \\ \text{s.t.} & A_B x_B + A_N x_N = b \\ & x_B, x_N \geq 0 \end{array}$$

Here B is the basis corresponding to the bfs we are starting from. Note that, for any solution x, $x_B = A_B^{-1} b - A_B^{-1} A_N x_N$ and that its total cost, $c^T x$ can be specified as follows:

$$\begin{aligned} c^T x &= c_B x_B + c_N x_N \\ &= c_B (A_B^{-1} b - A_B^{-1} A_N x_N) + c_N x_N \\ &= c_B A_B^{-1} b + (c_N - c_B A_B^{-1} A_N) x_N \end{aligned}$$

We denote the *reduced* cost of the non-basic variables by $\tilde{c}_N$, $\tilde{c}_N = c_N - c_B A_B^{-1} A_N$, i.e. the quantity which is the coefficient of x_N above. If there is a $j \in N$ such that

$\tilde{c}_j < 0$, then by increasing x_j (up from zero) we will decrease the cost (the value of the objective function). Of course x_B depends on x_N, and we can increase x_j only as long as all the components of x_B remain positive.

So in a step of the Simplex method, we find a $j \in N$ such that $\tilde{c}_j < 0$, and increase it as much as possible while keeping $x_B \geq 0$. It is not possible any more to increase x_j, when one of the components of x_B is zero. What happened is that a non-basic variable is now positive and we include it in the basis, and one variable which was basic is now zero, so we remove it from the basis.

If, on the other hand, there is no $j \in N$ such that $\tilde{c}_j < 0$, then we stop, and the current basic feasible solution is an optimal solution. This follows from the new expression for $c^T x$ since x_N is nonnegative.

Remarks:

1. Note that some of the basic variables may be zero to begin with, and in this case it is possible that we cannot increase x_j at all. In this case we can replace say j by k in the basis, but without moving from the vertex corresponding to the basis. In the next step we might replace k by j, and be stuck in a loop. Thus, we need to specify a "pivoting rule" to determine which index should enter the basis, and which index should be removed from the basis.

2. While many pivoting rules (including those that are used in practice) can lead to infinite loops, there is a pivoting rule which will not (known as the minimal index rule - choose the minimal j and k possible [Bland, 1977]). This fact was discovered by Bland in 1977. There are other methods of "breaking ties" which eliminate infinite loops.

3. There is no known pivoting rule for which the number of pivots in the worst case is better than exponential.

4. The question of the complexity of the Simplex algorithm and the last remark leads to the question of what is the length of the shortest path between two vertices of a convex polyhedron, where the path is along edges, and the length of the path in measured in terms of the number of vertices visited.

   Hirsch Conjecture: For m hyperplanes in d dimensions the length of the shortest path between any two vertices of the arrangement is at most $m - d$.

   This is a very open question — there is not even a polynomial bound proven on this length.

   On the other hand, one should note that even if the Hirsch Conjecture is true, it doesn't say much about the Simplex Algorithm, because Simplex generates paths which are monotone with respect to the objective function, whereas the shortest path need not be monotone.

Recently, Kalai (and others) has considered a randomized pivoting rule. The idea is to randomly permute the index columns of A and to apply the Simplex method, always choosing the smallest j possible. In this way, it is possible to show a subexponential bound on the expected number of pivots. This leads to a subexponential bound for the diameter of any convex polytope defined by m hyperplanes in a d dimension space.

The question of the existence of a polynomial pivoting scheme is still open though. We will see later a completely different algorithm which *is* polynomial, although not strongly polynomial (the existence of a strongly polynomial algorithm for linear programming is also open). That algorithm will not move from one vertex of the feasible domain to another like the Simplex, but will confine its interest to points in the interior of the feasible domain.

A visualization of the geometry of the Simplex algorithm can be obtained from considering the algorithm in 3 dimensions (see Figure 3). For a problem in the form $\min\{c^T x : Ax \leq b\}$ the feasible domain is a polyhedron in $\mathbb{R}^3$, and the algorithm moves from vertex to vertex in each step (or does not move at all).

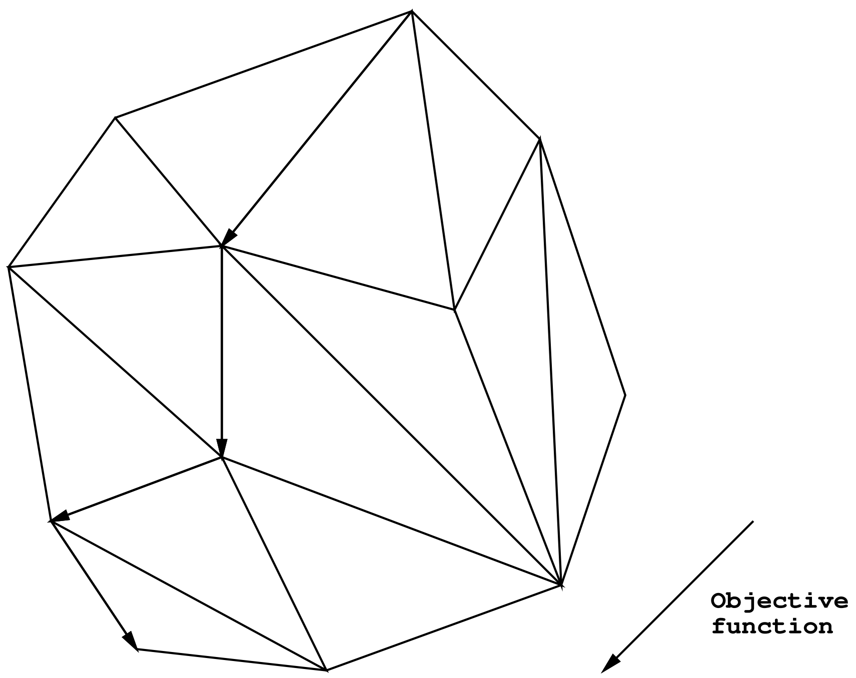

Figure 3: Traversing the vertices of a convex body (here a polyhedron in $\mathbb{R}^3$).

8 When is a Linear Program Feasible ?

We now turn to another question which will lead us to important properties of linear programming. Let us begin with some examples.

We consider linear programs of the form $Ax = b$, $x \geq 0$. As the objective function has no effect on the feasibility of the program, we ignore it.

We first restrict our attention to systems of equations (i.e. we neglect the non-negativity constraints).

Example: Consider the system of equations:

$$\begin{array}{rcrcrcl} x_1 & + & x_2 & + & x_3 & = & 6 \\ 2x_1 & + & 3x_2 & + & x_3 & = & 8 \\ 2x_1 & + & x_2 & + & 3x_3 & = & 0 \end{array}$$

and the linear combination

$$\begin{array}{rcrcrcrcl} -4 & \times & x_1 & + & x_2 & + & x_3 & = & 6 \\ 1 & \times & 2x_1 & + & 3x_2 & + & x_3 & = & 8 \\ 1 & \times & 2x_1 & + & x_2 & + & 3x_3 & = & 0 \end{array}$$

The linear combination results in the equation

$$0x_1 + 0x_2 + 0x_3 = -16$$

which means of course that the system of equations has no feasible solution.

In fact, an elementary theorem of linear algebra says that if a system has no solution, there is always a vector y such as in our example ($y = (-4, 1, 1)$) which proves that the system has no solution.

Theorem 5 *Exactly one of the following is true for the system $Ax = b$:*

1. *There is x such that $Ax = b$.*
2. *There is y such that $A^T y = 0$ but $y^T b = 1$.*

This is not quite enough for our purposes, because a system can be feasible, but still have no non-negative solutions $x \geq 0$. Fortunately, the following lemma establishes the equivalent results for our system $Ax = b, x \geq 0$.

Theorem 6 (Farkas' Lemma) *Exactly one of the following is true for the system $Ax = b$, $x \geq 0$:*

1. *There is x such that $Ax = b$, $x \geq 0$.*
2. *There is y such that $A^T y \geq 0$ but $b^T y < 0$.*

Proof:

We will first show that the two conditions cannot happen together, and then than at least one of them must happen.

Suppose we do have both x and y as in the statement of the theorem.

$$Ax = b \Longrightarrow y^T Ax = y^T b \Longrightarrow x^T A^T y = y^T b$$

but this is a contradiction, because $y^T b < 0$, and since $x \geq 0$ and $A^T y \geq 0$, so $x^T A^T y \geq 0$.

The other direction is less trivial, and usually shown using properties of the Simplex algorithm, mainly duality. We will use another tool, and later use Farkas' Lemma to prove properties about duality in linear programming. The tool we shall use is the Projection theorem, which we state without proof:

Theorem 7 (Projection Theorem) *Let K be a closed convex (see Figure 4) non-empty set in $\mathbb{R}^n$, and let b be any point in $\mathbb{R}^n$. The projection of b onto K is a point $p \in K$ that minimizes the Euclidean distance $\|b - p\|$. Then p has the property that for all $z \in K$, $(z - p)^T (b - p) \leq 0$ (see Figure 5) non-empty set.*

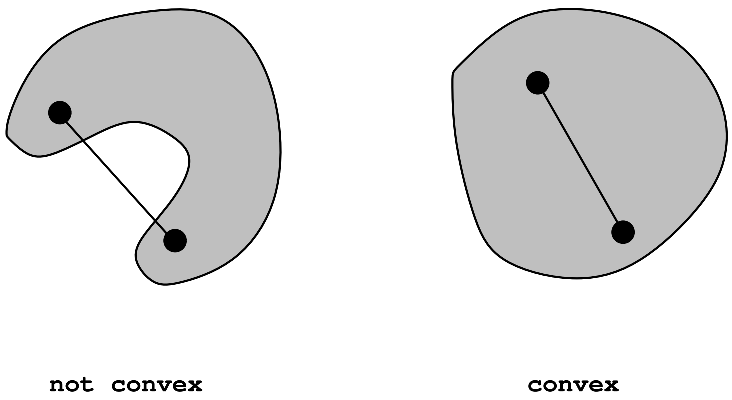

Figure 4: Convex and non-convex sets in $\mathbb{R}^2$.

We are now ready to prove the other direction of Farkas' Lemma. Assume that there is no x such that $Ax = b$, $x \geq 0$; we will show that there is y such that $A^T y \geq 0$ but $y^T b < 0$.

Let $K = \{Ax : x \geq 0\} \subseteq \mathbb{R}^m$ (A is an $m \times n$ matrix). K is a cone in $\mathbb{R}^m$ and it is convex, non-empty and closed. According to our assumption, $Ax = b$, $x \geq 0$ has no solution, so b does not belong to K. Let p be the projection of b onto K.

Since $p \in K$, there is a $w \geq 0$ such that $Aw = p$. According to the Projection Theorem, for all $z \in K$, $(z - p)^T (b - p) \leq 0$ That is, for all $x \geq 0$ $(Ax - p)^T (b - p) \leq 0$

We define $y = p - b$, which implies $(Ax - p)^T y \geq 0$. Since $Aw = p$, $(Ax - Aw)^T y \geq 0$. $(x - w)^T (A^T y) \geq 0$ for all $x \geq 0$ (remember that w was fixed by choosing b).

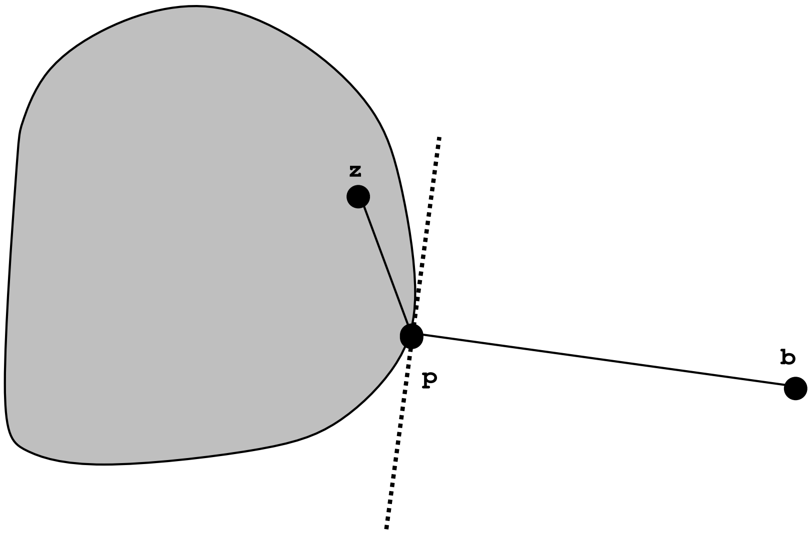

Figure 5: The Projection Theorem.

Set $x = w + \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{pmatrix}$ (w plus a unit vector with a 1 in the i-th row). Note that x is non-negative, because $w \geq 0$.

This will extract the i-th column of A, so we conclude that the i-th component of $A^T y$ is non-negative $(A^T y)_i \geq 0$, and since this is true for all i, $A^T y \geq 0$.

Now it only remains to show that $y^T b < 0$.

$y^t b = (p - y)^T y = p^T y - y^T y$ Since $(Ax - p)^T y \geq 0$ for all $x \geq 0$, taking x to be zero shows that $p^T y \leq 0$. Since $b \notin K$, $y = p - b \neq 0$, so $y^T y > 0$. So $y^T b = p^T y - y^T y < 0$.

□

Using a very similar proof one can show the same for the canonical form:

Theorem 8 *Exactly one of the following is true for the system $Ax \leq b$:*

1. *There is x such that $Ax \leq b$.*

2. *There is $y \geq 0$ such that $A^T y = 0$ but $y^T b < 0$.*

The intuition behind the precise form for 2. in the previous theorem lies in the proof that both cannot happen. The contradiction $0 = 0x = (y^T A)x = y^T(Ax) = y^T b < 0$ is obtained if $A^T y = 0$ and $y^T b < 0$.

9 Duality

Duality is the most important concept in linear programming. Duality allows to provide a *proof* of optimality. This is not only important algorithmically but also it leads to beautiful combinatorial statements. For example, consider the statement

> In a graph, the smallest number of edges in a path between two specified vertices s and t is equal to the maximum number of $s-t$ cuts (i.e. subsets of edges whose removal disconnects s and t).

This result is a direct consequence of duality for linear programming.

Duality can be motivated by the problem of trying to find lower bounds on the value of the optimal solution to a linear programming problem (if the problem is a maximization problem, then we would like to find upper bounds). We consider problems in standard form:

$$\begin{array}{ll} \min & c^T x \\ \text{s.t.} & Ax = b \\ & x \geq 0 \end{array}$$

Suppose we wanted to obtain the best possible upper bound on the cost function. By multiplying each equation $A_m x = b_m$ by some number y_m and summing up the resulting equations, we obtain that $y^T A x = b^T y$. if we impose that the coefficient of x_j in the resulting inequality is less or equal to c_j then $b^T y$ must be a lower bound on the optimal value since x_j is constrained to be nonnegative. To get the best possible lower bound, we want to solve the following problem:

$$\begin{array}{ll} \max & b^T y \\ \text{s.t.} & A^T y \leq c \end{array}$$

This is another linear program. We call this one the *dual* of the original one, called the *primal*. As we just argued, solving this dual LP will give us a lower bound on the optimum value of the primal problem. *Weak duality* says precisely this: if we denote the optimum value of the primal by z, $z = \min c^T x$, and the optimum value of the dual by w, then $w \leq z$. We will use Farkas' lemma to prove *strong duality* which says that these quantities are in fact equal. We will also see that, in general, the dual of the dual is the problem.

Example:

$$\begin{array}{rcccccl} z = \min & x_1 & + & 2x_2 & + & 4x_3 & \\ & x_1 & + & x_2 & + & 2x_3 & = 5 \\ & 2x_1 & + & x_2 & + & 3x_3 & = 8 \end{array}$$

The first equality gives a lower bound of 5 on the optimum value z, since $x_1 + 2x_2 + 4x_3 \geq x_1 + x_2 + 2x_3 = 5$ because of nonnegativity of the x_i. We can get an even better lower bound by taking 3 times the first equality minus the second one. This gives $x_1 + 2x_2 + 3x_3 = 7 \leq x_1 + 2x_2 + 4x_3$, implying a lower bound of 7 on z. For $x = \begin{pmatrix} 3 \\ 2 \\ 0 \end{pmatrix}$, the objective function is precisely 7, implying optimality. The mechanism of generating lower bounds is formalized by the dual linear program:

$$\begin{array}{rcrcl} \max & 5y_1 & + & 8y_2 & \\ y_1 & + & 2y_2 & \leq & 1 \\ y_1 & + & y_2 & \leq & 2 \\ 2y_1 & + & 3y_2 & \leq & 4 \end{array}$$

y_1 represents the multiplier for the first constraint and y_2 the multiplier for the second constraint, This LP's objective function also achieves a maximum value of 7 at $y = \begin{pmatrix} 3 \\ -1 \end{pmatrix}$.

We now formalize the notion of duality. Let P and D be the following pair of dual linear programs:

$$\begin{array}{ll} (P) & z = \min\{c^T x : Ax = b, x \geq 0\} \\ (D) & w = \max\{b^T y : A^T y \leq c\}. \end{array}$$

(P) is called the *primal* linear program and (D) the *dual* linear program.

In the proof below, we show that the dual of the dual is the primal. In other words, if one formulates (D) as a linear program in standard form (i.e. in the same form as (P)), its dual $D(D)$ can be seen to be equivalent to the original primal (P). In any statement, we may thus replace the roles of primal and dual without affecting the statement.

Proof:

The dual problem D is equivalent to $\min\{-b^T y : A^T y + Is = c, s \geq 0\}$. Changing forms we get $\min\{-b^T y^+ + b^T y^- : A^T y^+ - A^T y^- + Is = c, \text{ and } y^+, y^-, s \geq 0\}$. Taking the dual of this we obtain: $\max\{-c^T x : A(-x) \leq -b, -A(-x) \leq b, I(-x) \leq 0\}$. But this is the same as $\min\{c^T x : Ax = b, x \geq 0\}$ and we are done. $\square$

We have the following results relating w and z.

Lemma 9 *(Weak Duality)* $z \geq w$.

Proof:

Suppose x is primal feasible and y is dual feasible. Then, $c^T x \geq y^T Ax = y^T b$, thus $z = \min\{c^T x : Ax = b, x \geq 0\} \geq \max\{b^T y : A^T y \leq c\} = w$. $\square$

From the preceding lemma we conclude that the following cases are *not* possible (these are dual statements):

1. P is feasible and unbounded and D feasible.

2. P is feasible and D is feasible and unbounded.

We should point out however that both the primal and the dual might be infeasible.

To prove a stronger version of the *weak duality lemma,* let's recall the following corollary of *Farkas' Lemma* (Theorem 8):

Corollary 10 *Exactly one of the following is true:*

1. $\exists x' : A'x' \leq b'$.
2. $\exists y' \geq 0 : (A')^T y' = 0$ *and* $(b')^T y' < 0$.

Theorem 11 *(Strong Duality) If P or D is feasible then $z = w$.*

Proof:

We only need to show that $z \leq w$. Assume without loss of generality (by duality) that P is feasible. If P is unbounded, then by *Weak Duality*, we have that $z = w = -\infty$. Suppose P is bounded, and let x^* be an optimal solution, *i.e.* $Ax^* = b$, $x^* \geq 0$ and $c^T x^* = z$. We claim that $\exists y$ s.t. $A^T y \leq c$ and $b^T y \geq z$. If so we are done. Suppose no such y exists. Then, by the preceding corollary, with $A' = \begin{pmatrix} A^T \\ -b^T \end{pmatrix}$, $b' = \begin{pmatrix} c \\ -z \end{pmatrix}$, $x' = y$, $y' = \begin{pmatrix} x \\ \lambda \end{pmatrix}$, $\exists x \geq 0$, $\lambda \geq 0$ such that

$$\begin{aligned} & Ax = \lambda b \\ \text{and} \quad & c^T x < \lambda z. \end{aligned}$$

We have two cases

- **Case 1:** $\lambda \neq 0$. Since we can normalize by λ we can assume that $\lambda = 1$. This means that $\exists x \geq 0$ such that $Ax = b$ and $c^T x < z$. But this is a contradiction with the optimality of x^*.
- **Case 2:** $\lambda = 0$. This means that $\exists x \geq 0$ such that $Ax = 0$ and $c^T x < 0$. If this is the case then $\forall \mu \geq 0$, $x^* + \mu x$ is feasible for P and its cost is $c^T(x^* + \mu x) = c^T x^* + \mu(c^T x) < z$, which is a contradiction.

$\square$

9.1 Rules for Taking Dual Problems

If P is a minimization problem then D is a maximization problem. If P is a maximization problem then D is a minimization problem. In general, using the rules for transforming a linear program into standard form, we have that the dual of (P):

$z = \min\ c_1^T x_1 + c_2^T x_2 + c_3^T x_3$

s.t.

$$\begin{aligned} A_{11}x_1 + A_{12}x_2 + A_{13}x_3 &= b_1 \\ A_{21}x_1 + A_{22}x_2 + A_{23}x_3 &\geq b_2 \\ A_{31}x_1 + A_{32}x_2 + A_{33}x_3 &\leq b_3 \\ x_1 \geq 0\ ,\ x_2 \leq 0\ ,\ x_3 \text{ UIS} & \end{aligned}$$

(where UIS means "unrestricted in sign" to emphasize that no constraint is on the variable) is (D)

$w = \max\ b_1^T y_1 + b_2^T y_2 + b_3^T y_3$

s.t.

$$\begin{aligned} A_{11}^T y_1 + A_{21}^T y_2 + A_{31}^T y_3 &\leq c_1 \\ A_{12}^T y_1 + A_{22}^T y_2 + A_{32}^T y_3 &\geq c_2 \\ A_{13}^T y_1 + A_{23}^T y_2 + A_{33}^T y_3 &= c_3 \\ y_1 \text{ UIS}\ ,\ y_2 \geq 0\ ,\ y_3 \leq 0 & \end{aligned}$$

10 Complementary Slackness

Let P and D be

$$\begin{aligned} &(P) \quad z = \min\{c^T x : Ax = b, x \geq 0\} \\ &(D) \quad w = \max\{b^T y : A^T y \leq c\}, \end{aligned}$$

and let x be feasible in P, and y be fesible in D. Then, by weak duality, we know that $c^T x \geq b^T y$. We call the difference $c^T x - b^T y$ the *duality gap*. Then we have that the duality gap is zero iff x is optimal in P, and y is optimal in D. That is, the duality gap can serve as a good measure of how close a feasible x and y are to the optimal solutions for P and D. The duality gap will be used in the description of the interior point method to monitor the progress towards optimality.

It is convenient to write the dual of a linear program as

$$(D) \qquad w = \max\{b^T y : A^T y + s = c \ \text{ for some } s \geq 0\}$$

Then we can write the duality gap as follows:

$$\begin{aligned} c^T x - b^T y &= c^T x - x^T A^T y \\ &= x^T (c - A^T y) \\ &= x^T s \end{aligned}$$

since $A^T y + s = c$.

The following theorem allows to check optimality of a primal and/or a dual solution.

Theorem 12 *(Complementary Slackness)*

Let x^*, (y^*, s^*) be feasible for (P), (D) respectively. The following are equivalent:

1. *x^* is an optimal solution to (P) and (y^*, s^*) is an optimal solution to (D).*
2. *$(s^*)^T x^* = 0$.*
3. *$x_j^* s_j^* = 0$, $\forall\, j = 1, \ldots, n$.*
4. *If $s_j^* > 0$ then $x_j^* = 0$.*

Proof:

Suppose (1) holds, then, by strong duality, $c^T x^* = b^T y^*$. Since $c = A^T y^* + s^*$ and $Ax^* = b$, we get that $(y^*)^T A x^* + (s^*)^T x^* = (x^*)^T A^T y^*$, and thus, $(s^*)^T x^* = 0$ (i.e (2) holds). It follows, since x_j^*, $s_j^* \geq 0$, that $x_j^* s_j^* = 0$, $\forall\, j = 1, \ldots, n$ (i.e. (3) holds). Hence, if $s_j^* > 0$ then $x_j^* = 0$, $\forall\, j = 1, \ldots, n$ (i.e. (4) holds). The converse also holds, and thus the proof is complete. □

In the example of section 9, the complementary slackness equations corresponding to the primal solution $x = (3, 2, 0)^T$ would be:

$$\begin{aligned} y_1 + 2y_2 &= 1 \\ y_1 + y_2 &= 2 \end{aligned}$$

Note that this implies that $y_1 = 3$ and $y_2 = -1$. Since this solution satisfies the other constraint of the dual, y is dual feasible, proving that x is an optimum solution to the primal (and therefore y is an optimum solution to the dual).

11 Size of a Linear Program

11.1 Size of the Input

If we want to solve a Linear Program in polynomial time, we need to know what would that mean, i.e. what would the size of the input be. To this end we introduce two notions of the size of the input with respect to which the algorithm we present will run in polynomial time. The first measure of the input size will be the *size* of a LP, but we will introduce a new measure L of a LP that will be easier to work with. Moreover, we have that $L \leq \text{size}(LP)$, so that any algorithm running in time polynomial in L will also run in time polynomial in *size(LP)*.

Let's consider the linear program of the form:

$$\begin{aligned} &\min c^T x \\ s.t.& \\ &Ax = b \\ &x \geq 0 \end{aligned}$$

where we are given as inputs the coefficients of A (an $m \times n$ matrix), b (an $m \times 1$ vector), and c (an $n \times 1$ vector), whith rational entries.

We can further assume, without loss of generality, that the given coefficients are all integers, since any LP with rational coefficients can be easily transformed into an equivalent one with integer coefficients (just multiply everything by l.c.d.). In the rest of these notes, we assume that A, b, c have integer coefficients.

For any integer n, we define its size as follows:

$$size(n) \triangleq 1 + \lceil \log_2(|n| + 1) \rceil$$

where the first 1 stands for the fact that we need one bit to store the sign of n, $size(n)$ represents the number of bits needed to encode n in binary. Analogously, we define the size of a $p \times 1$ vector d, and of a $p \times l$ matrix M as follows:

$$size(v) \triangleq \sum_{i=1}^{p} size(v_i)$$
$$size(M) \triangleq \sum_{i=1}^{p} \sum_{j=1}^{l} size(m_{ij})$$

We are then ready to talk about the size of a *LP*.

Definition 6 (Size of a linear program)

$$\text{size(LP)} \triangleq \text{size}(A) + \text{size}(b) + \text{size}(c).$$

A more convenient definition of the size of a linear program is given next.

Definition 7

$$L \triangleq \text{size}(det_{max}) + \text{size}(b_{max}) + \text{size}(c_{max}) + m + n$$

where

$$\begin{aligned} det_{max} &\triangleq \max_{A'}(|\det(A')|) \\ b_{max} &\triangleq \max_{i}(|b_i|) \\ c_{max} &\triangleq \max_{j}(|c_j|) \end{aligned}$$

and A' is any square submatrix of A.

Proposition 13 $L < \text{size(LP)}, \ \forall A, b, c.$

Before proving this result, we first need the following lemma:

Lemma 14 1. *If $n \in \mathbb{Z}$ then $|n| \leq 2^{size(n)-1} - 1$.*

2. If $v \in \mathbb{Z}^n$ then $\|v\| \leq \|v\|_1 \leq 2^{size(v)-n} - 1$.

3. If $A \in \mathbb{Z}^{n \times n}$ then $|det(A)| \leq 2^{size(A)-n^2} - 1$.

Proof:

1. By definition.

2. $1 + \|v\| \leq 1 + \|v\|_1 = 1 + \sum_{i=1}^{n} |v_i| \leq \prod_{i=1}^{n}(1 + |v_i|) \leq \prod_{i=1}^{n} 2^{size(v_i)-1} = 2^{size(v)-n}$ where we have used 1.

3. Let $a_1, \ldots, a_n$ be the columns of A. Since $|det(A)|$ represents the volume of the parallelepiped spanned by $a_1, \ldots, a_n$, we have

$$|det(A)| \leq \prod_{i=1}^{n} \|a_i\|.$$

   Hence, by 2,

$$1 + |det(A)| \leq 1 + \prod_{i=1}^{n} \|a_i\| \leq \prod_{i=1}^{n}(1 + \|a_i\|) \leq \prod_{i=1}^{n} 2^{size(a_i)-n} = 2^{size(A)-n^2}.$$

$\square$

We now prove Proposition 13.

Proof:

If B is a square submatrix of A then, by definition, $size(B) \leq size(A)$. Moreover, by lemma 14, $1 + |det(B)| \leq 2^{size(B)-1}$. Hence,

$$\lceil \log(1 + |det(B)|) \rceil \leq size(B) - 1 < size(B) \leq size(A). \tag{1}$$

Let $v \in \mathbb{Z}^p$. Then $size(v) \geq size(\max_j |v_j|) + p - 1 = \lceil \log(1 + \max_j |v_j|) \rceil + p$. Hence,

$$size(b) + size(c) \geq \lceil \log(1 + \max_j |c_j|) \rceil + \lceil \log(1 + \max_i |b_i|) \rceil + m + n. \tag{2}$$

Combining equations (1) and (2), we obtain the desired result. $\square$

Remark 1 *$det_{max} * b_{max} * c_{max} * 2^{m+n} < 2^L$, since for any integer n, $2^{\text{size}(n)} > |n|$.*

In what follows we will work with L as the size of the input to our algorithm.

11.2 Size of the Output

In order to even hope to solve a linear program in polynomial time, we better make sure that the solution is representable in size polynomial in L. We know already that if the LP is feasible, there is at least one vertex which is an optimal solution. Thus, when finding an optimal solution to the LP, it makes sense to restrict our attention to vertices only. The following theorem makes sure that vertices have a compact representation.

Theorem 15 *Let x be a vertex of the polyhedron defined by $Ax = b, x \geq 0$. Then,*

$$x^T = \left(\frac{p_1}{q} \; \frac{p_2}{q} \ldots \frac{p_n}{q} \right),$$
$$\text{where } p_i \; (i = 1, \ldots, n), \; q \in \mathbb{N},$$

and

$$0 \leq p_i < 2^L$$
$$1 \leq q < 2^L.$$

Proof:

Since x is a basic feasible solution, $\exists$ a basis B such that $x_B = A_B^{-1} b$ and $x_N = 0$. Thus, we can set $p_j = 0$, $\forall \; j \in N$, and focus our attention on the x_j's such that $j \in B$. We know by linear algebra that

$$x_B = A_B^{-1} b = \frac{1}{det(A_B)} cof(A_B) b$$

where $cof(A_B)$ is the cofactor matrix of A_B. Every entry of A_B consists of a determinant of some submatrix of A. Let $q = |det(A_B)|$, then q is an integer since A_B has integer components, $q \geq 1$ since A_B is invertible, and $q \leq det_{\max} < 2^L$. Finally, note that $p_B = q x_B = |cof(A_B) b|$, thus $p_i \leq \sum_{j=1}^{m} |cof(A_B)_{ij}||b_j| \leq m \; det_{\max} \; b_{\max} < 2^L$. $\square$

12 Complexity of linear programming

In this section, we show that linear programming is in NP$\cap$ co-NP. This will follow from duality and the estimates on the size of any vertex given in the previous section. Let us define the following decision problem:

Definition 8 ($\mathcal{LP}$)

Input: Integral A, b, c, and a rational number λ,
Question: Is $\min\{c^T x : Ax = b, \; x \geq 0\} \leq \lambda$?

Theorem 16 $\mathcal{LP} \in \mathrm{NP} \cap \text{co-NP}$

Proof:

First, we prove that $\mathcal{LP} \in \mathrm{NP}$.

If the linear program is feasible and bounded, the "certificate" for verification of instances for which $\min\{c^T x : Ax = b, x \geq 0\} \leq \lambda$ is a vertex x' of $\{Ax = b, x \geq 0\}$ s.t. $c^T x' \leq \lambda$. This vertex x' always exists since by assumption the minimum is finite. Given x', it is easy to check in polynomial time whether $Ax' = b$ and $x' \geq 0$. We also need to show that the size of such a certificate is polynomially bounded by the size of the input. This was shown in section 11.2.

If the linear program is feasible and unbounded, then, by strong duality, the dual is infeasible. Using Farkas' lemma on the dual, we obtain the existence of $\tilde{x}$: $A\tilde{x} = 0$, $\tilde{x} \geq 0$ and $c^T\tilde{x} = -1 < 0$. Our certificate in this case consists of both a vertex of $\{Ax = b,\ x \geq 0\}$ (to show feasiblity) and a vertex of $\{Ax = 0,\ x \geq 0,\ c^T x = -1\}$ (to show unboundedness if feasible). By choosing a vertex x' of $\{Ax = 0,\ x \geq 0,\ c^T x = -1\}$, we insure that x' has polynomial size (again, see Section 11.2).

This proves that $\mathcal{LP} \in \mathrm{NP}$. (Notice that when the linear program is infeasible, the answer to $\mathcal{LP}$ is "no", but we are not responsible to offer such an answer in order to show $\mathcal{LP} \in \mathrm{NP}$).

Secondly, we show that $\mathcal{LP} \in \text{co-NP}$, i.e. $\overline{\mathcal{LP}} \in \mathrm{NP}$, where $\overline{\mathcal{LP}}$ is defined as:

Input: A, b, c, and a rational number λ,

Question: Is $\min\{c^T x : Ax = b,\ x \geq 0\} > \lambda$?

If $\{x : Ax = b,\ x \geq 0\}$ is nonempty, we can use strong duality to show that $\overline{\mathcal{LP}}$ is indeed equivalent to:

Input: A, b, c, and a rational number λ,

Question: Is $\max\{b^T y : A^T y \leq c\} > \lambda$?

which is also in NP, for the same reason as $\mathcal{LP}$ is.

If the primal is infeasible, by Farkas' lemma we know the existence of a y s.t. $A^T y \geq 0$ and $b^T y = -1 < 0$. This completes the proof of the theorem. $\square$

13 Solving a Liner Program in Polynomial Time

The first polynomial-time algorithm for linear programming is the so-called *ellipsoid algorithm* which was proposed by Khachian in 1979 [6]. The ellipsoid algorithm was in fact first developed for convex programming (of which linear programming is a special case) in a series of papers by the russian mathematicians A.Ju. Levin and, D.B. Judin and A.S. Nemirovskii, and is related to work of N.Z. Shor. Though of polynomial running time, the algorithm is impractical for linear programming. Nevertheless it has extensive theoretical applications in combinatorial optimization. For example, the stable set problem on the so-called perfect graphs can be solved in polynomial time using the ellipsoid algorithm. This is however a non-trivial non-combinatorial algorithm.

In 1984, Karmarkar presented another polynomial-time algorithm for linear programming. His algorithm avoids the combinatorial complexity (inherent in the simplex algorithm) of the vertices, edges and faces of the polyhedron by staying well inside the polyhedron (see Figure 13). His algorithm lead to many other algorithms for linear programming based on similar ideas. These algorithms are known as *interior point methods*.

Figure 6: Exploring the interior of a convex body.

It still remains an open question whether there exists a strongly polynomial algorithm for linear programming, i.e. an algorithm whose running time depends on m and n and not on the size of any of the entries of A, b or c.

In the rest of these notes, we discuss an interior-point method for linear programming and show its polynomiality.

High-level description of an interior-point algorithm:

1. If x (current solution) is close to the boundary, then map the polyhedron onto another one s.t. x is well in the interior of the new polyhedron (see Figure 7).

2. Make a step in the transformed space.

3. Repeat (a) and(b) until we are close enough to an optimal solution.

Before we give description of the algorithm we give a theorem, the corollary of which will be a key tool used in determinig when we have reached an optimal solution.

Theorem 17 *Let x_1, x_2 be vertices of* $\begin{array}{l} Ax = b, \\ x \geq 0. \end{array}$

If $c^T x_1 \neq c^T x_2$ then $|c^T x_1 - c^T x_2| > 2^{-2L}$.

Proof:

By Theorem 15, $\exists\, q_i$, q_2, such that $1 \leq q_1, q_2 < 2^L$, and $q_1 x_1, q_2 x_2 \in \mathbb{N}^n$. Furthermore,

$$\begin{aligned}
|c^T x_1 - c^T x_2| &= \left| \frac{q_1 c^T x_1}{q_1} - \frac{q_2 c^T x_2}{q_2} \right| \\
&= \left| \frac{q_1 q_2 (c^T x_1 - c^T x_2)}{q_1 q_2} \right| \\
&\geq \frac{1}{q_1 q_2} \qquad \text{since } c^T x_1 - c^T x_2 \neq 0,\ q_1, q_2 \geq 1 \\
&> \frac{1}{2^L 2^L} = 2^{-2L} \qquad \text{since } q_1, q_2 < 2^L.
\end{aligned}$$

□

Corollary 18 *Assume* $z = \min\{c^T x : \underbrace{Ax = b, x \geq 0}_{\textit{polyhedron } P}\}$.

Assume x is feasible to P, and such that $c^T x \leq z + 2^{-2L}$.

Then, any vertex x' such that $c^T x' \leq c^T x$ is an optimal solution of the LP.

Proof:

Suppose x' is not optimal. Then, $\exists x^*$, an optimal vertex, such that $c^T x^* = z$. Since x' is not optimal, $c^T x' \neq c^T x^*$, and by Theorem 17

$$\begin{aligned}
\Rightarrow \quad c^T x' - c^T x^* &> 2^{-2L} \\
\Rightarrow \quad c^T x' &> c^T x^* + 2^{-2L} \\
&= Z + 2^{-2L} \\
&\geq c^T x \qquad \text{by definition of } x \\
&\geq c^T x' \qquad \text{by definition of } x' \\
\Rightarrow \quad c^T x' &> c^T x',
\end{aligned}$$

a contradiction. □

What this corollary tells us is that we do not need to be very precise when choosing an optimal vertex. More precisely we only need to compute the objective function with error less than 2^{-2L}. If we find a vertex that is within that margin of error, then it will be optimal.

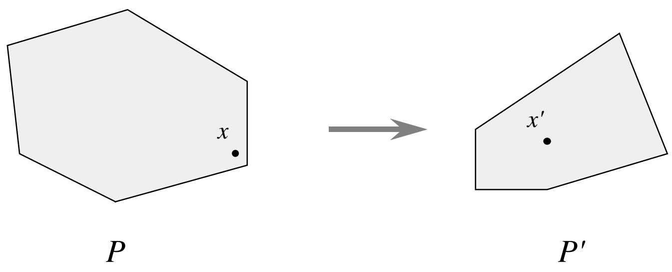

Figure 7: A centering mapping. If x is close to the boundary, we map the polyhedron P onto another one P', s.t. the image x' of x is closer to the center of P'.

13.1 Ye's Interior Point Algorithm

In the rest of these notes we present Ye's [9] interior point algorithm for linear programming. Ye's algorithm (among several others) achieves the best known asymptotic running time in the literature, and our presentation incorporates some simplifications made by Freund [3].

We are going to consider the following linear programming problem:

$$(P)\begin{cases} \text{minimize} & Z = c^T x \\ \text{subject to} & Ax = b, \\ & x \geq 0 \end{cases}$$

and its dual

$$(D)\begin{cases} \text{maximize} & W = b^T y \\ \text{subject to} & A^T y + s = c, \\ & s \geq 0. \end{cases}$$

The algorithm is primal-dual, meaning that it simultaneously solves both the primal and dual problems. It keeps track of a primal solution $\overline{x}$ and a vector of dual slacks $\overline{s}$ (i.e. $\exists\ \overline{y} : A^T\overline{y} = c - \overline{s}$) such that $\overline{x} > 0$ and $\overline{s} > 0$. The basic idea of this algorithm is to stay away from the boundaries of the polyhedron (the hyperplanes $x_j \geq 0$ and $s_j \geq 0$, $j = 1, 2, \ldots, n$) while approaching optimality. In other words, we want to make the duality gap

$$c^T\overline{x} - b^T\overline{y} = \overline{x}^T\overline{s} > 0$$

very small but stay away from the boundaries. Two tools will be used to achieve this goal in polynomial time.

Tool 1: Scaling (see Figure 7)

Scaling is a crucial ingredient in interior point methods. The two types of scaling commonly used are *projective scaling* (the one used by Karmarkar) and *affine scaling* (the one we are going to use).

Suppose the current iterate is $\overline{x} > 0$ and $\overline{s} > 0$, where $\overline{x} = (\overline{x}_1, \overline{x}_2, \ldots, \overline{x}_n)^T$, then the affine scaling maps x to x' as follows.

$$x = \begin{pmatrix} x_1 \\ x_2 \\ \cdot \\ \cdot \\ \cdot \\ x_n \end{pmatrix} \longrightarrow x' = \begin{pmatrix} \frac{x_1}{\overline{x}_1} \\ \frac{x_2}{\overline{x}_2} \\ \cdot \\ \cdot \\ \cdot \\ \frac{x_n}{\overline{x}_n} \end{pmatrix}.$$

Notice this transformation maps $\overline{x}$ to $e = (1, \ldots, 1)^T$.

We can express the scaling transformation in matrix form as $x' = \overline{X}^{-1} x$ or $x = \overline{X} x'$, where

$$\overline{X} = \begin{pmatrix} \overline{x}_1 & 0 & 0 & \ldots & 0 \\ 0 & \overline{x}_2 & 0 & \ldots & 0 \\ \vdots & & \vdots & & \vdots \\ 0 & 0 & \ldots & \overline{x}_{n-1} & 0 \\ 0 & 0 & \ldots & 0 & \overline{x}_n \end{pmatrix}.$$

Using matrix notation we can rewrite the linear program (P) in terms of the transformed variables as:

$$\begin{aligned} \text{minimize} \quad & Z = c^T \overline{X} x' \\ \text{subject to} \quad & A\overline{X} x' = b, \\ & x' \geq 0. \end{aligned}$$

If we define $\overline{c} = \overline{X} c$ (note that $\overline{X} = \overline{X}^T$) and $\overline{A} = A\overline{X}$ we can get a linear program in the original form as follows.

$$\begin{aligned} \text{minimize} \quad & Z = \overline{c}^T x' \\ \text{subject to} \quad & \overline{A} x' = b, \\ & x' \geq 0. \end{aligned}$$

We can also write the dual problem (D) as:

$$\begin{aligned} \text{maximize} \quad & W = b^T y \\ \text{subject to} \quad & (A\overline{X})^T y + \overline{X} s = \overline{c}, \\ & \overline{X} s \geq 0 \end{aligned}$$

or, equivalently,

$$\begin{aligned} \text{maximize} \quad & W = b^T y \\ \text{subject to} \quad & \overline{A}^T y + s' = \overline{c}, \\ & s' \geq 0 \end{aligned}$$

where $s' = \overline{X}s$, i.e.

$$s' = \begin{pmatrix} s_1\overline{x}_1 \\ s_2\overline{x}_2 \\ \cdot \\ \cdot \\ s_n\overline{x}_n \end{pmatrix}.$$

One can easily see that

$$x_j s_j = x'_j s'_j \qquad \forall j \in \{1, \ldots, n\} \tag{3}$$

and, therefore, the duality gap $x^T s = \sum_j x_j s_j$ remains unchanged under affine scaling. As a consequence, we will see later that one can always work equivalently in the transformed space.

Tool 2: Potential Function

Our potential function is designed to measure how small the duality gap is and how far the current iterate is away from the boundaries. In fact we are going to use the following "logarithmic barrier function".

Definition 9 (Potential Function, $G(x,s)$)

$$G(x,s) \triangleq q\ln(x^T s) - \sum_{j=1}^{n} \ln(x_j s_j), \qquad \text{for some q,}$$

where q is a parameter that must be chosen appropriately.

Note that the first term goes to $-\infty$ as the duality gap tends to 0, and the second term goes to $+\infty$ as $x_i \to 0$ or $s_i \to 0$ for some i. Two questions arise immediately concerning this potential function.

Question 1: How do we choose q?

Lemma 19 *Let $x, s > 0$ be vectors in $\mathbb{R}^{n\times 1}$. Then*

$$n \ln x^T s - \sum_{j=1}^{n} \ln x_j s_j \geq n \ln n.$$

Proof:

Given any n positive numbers $t_1, \ldots, t_n$, we know that their geometric mean does not exceed their arithmetic mean, i.e.

$$\left(\prod_{j=1}^{n} t_j\right)^{1/n} \leq \frac{1}{n}\left(\sum_{j=1}^{n} t_j\right).$$

Taking the logarithms of both sides we have

$$\frac{1}{n}\left(\sum_{j=1}^{n}\ln t_j\right) \leq \ln\left(\sum_{j=1}^{n} t_j\right) - \ln n.$$

Rearranging this inequality we get

$$n\ln\left(\sum_{j=1}^{n} t_j\right) - \left(\sum_{j=1}^{n}\ln t_j\right) \geq n\ln n.$$

(In fact the last inequality can be derived directly from the concavity of the logarithmic function). The lemma follows if we set $t_j = x_j s_j$. □

Since our objective is that $G \to -\infty$ as $x^T s \to 0$ (since our primary goal is to get close to optimality), according to Lemma 19, we should choose some $q > n$ (notice that $\ln x^T s \to -\infty$ as $x^T s \to 0$) . In particular, if we choose $q = n+1$, the algorithm will terminate after $O(nL)$ iterations. In fact we are going to set $q = n + \sqrt{n}$, which gives us the smallest number — $O(\sqrt{n}L)$ — of iterations by this method.

Question 2: When can we stop?

Suppose that $x^T s \leq 2^{-2L}$, then $c^T x - Z \leq c^T x - b^T y = x^T s \leq 2^{-2L}$, where Z is the optimum value to the primal problem. From Corollary 18, the following claim follows immediately.

Claim 20 *If $x^T s \leq 2^{-2L}$, then any vertex x^* satisfying $c^T x^* \leq c^T x$ is optimal.*

In order to find x^* from x, two methods can be used. One is based on purely algebraic techniques (but is a bit cumbersome to describe), while the other (the cleanest one in literature) is based upon basis reduction for lattices. We shall not elaborate on this topic, although we'll get back to this issue when discussing basis reduction in lattices.

Lemma 21 *Let x, s be feasible primal-dual vectors such that $G(x,s) \leq -k\sqrt{n}L$ for some constant k. Then*

$$x^T s < e^{-kL}.$$

Proof:

By the definition of $G(x,s)$ and the previous theorem we have:

$$\begin{aligned} -k\sqrt{n}L &\geq G(x,s) \\ &= (n+\sqrt{n})\ln x^T s - \sum_{j=1}^{n}\ln x_j s_j \\ &\geq \sqrt{n}\ln x^T s + n\ln n. \end{aligned}$$

Rearranging we obtain

$$\begin{aligned}\ln x^T s &\leq -kL - \sqrt{n}\ln n \\ &< -kL.\end{aligned}$$

Therefore

$$x^T s < e^{-kL}. \qquad \Box$$

The previous lemma and claim tell us that we can stop whenever $G(x, s) \leq -2\sqrt{n}L$. In practice, the algorithm can terminate even earlier, so it is a good idea to check from time to time if we can get the optimal solution right away.

Please notice that according to Equation (3) the affine transformation does not change the value of the potential function. Hence we can work either in the original space or in the transformed space when we talk about the potential function.

14 Description of Ye's Interior Point Algorithm

Initialization:

Set $i = 0$.

Choose $x^0 > 0$, $s^0 > 0$, and y^0 such that $Ax^0 = b$, $A^T y^0 + s^0 = c$ and $G(x^0, s^0) = O(\sqrt{n}L)$. (Details are not covered in class but can be found in the appendix. The general idea is as follows. By augmenting the linear program with additional variables, it is easy to obtain a feasible solution. Moreover, by carefully choosing the augmented linear program, it is possible to have feasible primal and dual solutions x and s such that all x_j's and s_j's are large (say 2^L). This can be seen to result in a potential of $O(\sqrt{n}L)$.)

Iteration:

$$\textbf{while } G(x^i, s^i) > -2\sqrt{n}L$$
$$\textbf{do}\left\{\begin{array}{l}\left.\begin{array}{l}\text{either a primal step (changing } \mathrm{x^i} \text{ only)}\\ \text{or a dual step (changing } \mathrm{s^i} \text{ only)}\end{array}\right\}\text{ to get } (x^{i+1}, s^{i+1})\\ i := i+1\end{array}\right.$$

The iterative step is as follows. Affine scaling maps (x^i, s^i) to (e, s'). In this transformed space, the point is far away from the boundaries. Either a dual or primal step occurs, giving $(\tilde{x}, \tilde{s})$ and reducing the potential function. The point is then mapped back to the original space, resulting in (x^{i+1}, s^{i+1}).

Next, we are going to describe precisely how the primal or dual step is made such that

$$G(x^{i+1}, s^{i+1}) - G(x^i, s^i) \leq -\frac{7}{120} < 0$$

holds for either a primal or dual step, yielding an $O(\sqrt{n}L)$ total number of iterations.

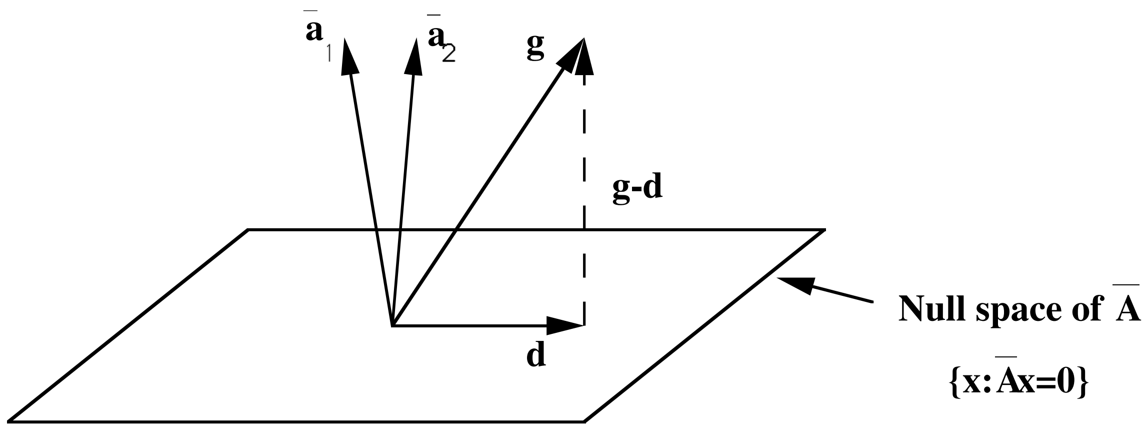

Figure 8: Null space of $\overline{A}$ and gradient direction g.

In order to find the new point $(\tilde{x}, \tilde{s})$ given the current iterate (e, s') (remember we are working in the transformed space), we compute the gradient of the potential function. This is the direction along which the value of the potential function changes at the highest rate. Let g denote the gradient. Recall that (e, s') is the map of the current iterate, we obtain

$$\begin{aligned} g &= \nabla_x G(x,s)|_{(e,s')} \\ &= \frac{q}{x^T s} s - \left.\begin{pmatrix} 1/x_1 \\ \vdots \\ 1/x_n \end{pmatrix}\right|_{(e,s')} \\ &= \frac{q}{e^T s'} s' - e \qquad (4) \end{aligned}$$

We would like to maximize the change in G, so we would like to move in the direction of $-g$. However, we must insure the new point is still feasible (i.e. $\overline{A}\tilde{x} = b$). Let d be the projection of g onto the null space $\{x : \overline{A}x = 0\}$ of $\overline{A}$. Thus, we will move in the direction of $-d$.

Claim 22 $d = (I - \overline{A}(\overline{A}\,\overline{A}^T)^{-1}\overline{A})g$.

Proof:

Since $g - d$ is orthogonal to the null space of $\overline{A}$, it must be the combination of some row vectors of $\overline{A}$. Hence we have

$$\begin{cases} \overline{A}d = 0 \\ \exists w, \text{ s.t. } \overline{A}^T w = g - d. \end{cases}$$

This implies

$$\begin{cases} \overline{A}^T w = g - d \\ (\overline{A}\,\overline{A}^T)w = \overline{A}g \end{cases} \qquad \text{(normal equations).}$$

Solving the normal equations, we get

$$w = (\overline{A}\,\overline{A}^T)^{-1}\overline{A}g$$

and

$$d = g - \overline{A}^T(\overline{A}\,\overline{A}^T)^{-1}\overline{A}g = (I - \overline{A}^T(\overline{A}\,\overline{A}^T)^{-1}\overline{A})g.$$

□

A potential problem arises if g is nearly perpendicular to the null space of $\overline{A}$. In this case, $||d||$ will be very small, and each primal step will not reduce the potential greatly. Instead, we will perform a dual step.

In particular, if $||d|| = ||d||_2 = \sqrt{d^T d} \geq 0.4$, we make a primal step as follows.

$$\begin{aligned} \tilde{x} &= e - \frac{1}{4||d||}d \\ \tilde{s} &= s'. \end{aligned}$$

Claim 23 $\tilde{x} > 0$.

Proof:
$\tilde{x}_j = 1 - \frac{1}{4}\frac{d_j}{||d||} \geq \frac{3}{4} > 0$. □

This claim insures that the new iterate is still an interior point. For the similar reason, we will see that $\tilde{s} > 0$ when we make a dual step.

Proposition 24 *When a primal step is made,* $G(\tilde{x}, \tilde{s}) - G(e, s') \leq -\frac{7}{120}$.

If $||d|| < 0.4$, we make a dual step. Again, we calculate the gradient

$$\begin{aligned} h &= \nabla_s G(x, s)|_{(e,s')} \\ &= \frac{q}{e^T s'}e - \begin{pmatrix} 1/s'_1 \\ \vdots \\ 1/s'_n \end{pmatrix} \end{aligned} \tag{5}$$

Notice that $h_j = g_j/s_j$, thus h and g can be seen to be approximately in the same direction.

Suppose the current dual feasible solution is y', s' such that

$$\overline{A}^T y' + s' = \overline{c}.$$

Again, we restrict the solution to be feasible, so

$$\begin{aligned} \overline{A}^T y + \tilde{s} &= \overline{c} \\ \tilde{s} - s' &= \overline{A}^T (y' - y) \end{aligned}$$

Thus, in the dual space, we move perpendicular to the null space and in the direction of $-(g-d)$.

Thus, we have

$$\tilde{s} = s' - (g-d)\mu$$

For any μ, $\exists y \;\; \overline{A}^T y + \tilde{s} = c$

So, we can choose $\mu = \frac{e^T s'}{q}$ and get $\overline{A}^T(y' + \mu w) + \tilde{s} = c$.

Therefore,

$$\begin{aligned} \tilde{s} &= s' - \frac{e^T s'}{q}(g-d) \\ &= s' - \frac{e^T s'}{q}(q\frac{s'}{e^T s'} - e - d) \\ &= \frac{e^T s'}{q}(d+e) \\ \tilde{x} &= x' = e. \end{aligned}$$

One can show that $\tilde{s} > 0$ as we did in Claim 23. So such move is legal.

Proposition 25 *When a dual step is made,* $G(\tilde{x}, \tilde{s}) - G(e, s') \leq -\frac{1}{6}$

According to these two propositions, the potential function decreases by a constant amount at each step. So if we start from an initial interior point (x^0, s^0) with $G(x^0, s^0) = O(\sqrt{n}L)$, then after $O(\sqrt{n}L)$ iterations we will obtain another interior point (x^j, s^j) with $G(x^j, s^j) \leq -k\sqrt{n}L$. From Lemma 21, we know that the duality gap $(x^j)^T s^j$ satisfies

$$(x^j)^T s^j \leq 2^{-kL},$$

and the algorithm terminates by that time. Moreover, each iteration requires $O(n^3)$ operations. Indeed, in each iteration, the only non-trivial task is the computation of the projected gradient d. This can be done by solving the linear system $(\bar{A}\bar{A}^T)w = \bar{A}g$ in $O(n^3)$ time using Gaussian elimination. Therefore, the overall time complexity of this algorithm is $O(n^{3.5}L)$. By using approximate solutions to the linear systems, we can obtain $O(n^{2.5})$ time per iteration, and total time $O(n^3 L)$.

15 Analysis of the Potential Function

In this section, we prove the two propositions of the previous section, which concludes the analysis of Ye's algorithm.

Proof of Proposition 24:

$$
\begin{aligned}
G(\tilde{x},\tilde{s}) - G(e,s') &= G(e - \frac{1}{4||d||}d, \tilde{s}) - G(e,s') \\
&= q\ln\left(e^T s' - \frac{d^T s'}{4||d||}\right) - \sum_{j=1}^{n}\ln\left(1 - \frac{d_j}{4||d||}\right) - \sum_{j=1}^{n}\ln s'_j - \\
&\quad -q\ln\left(e^T s'\right) + \sum_{j=1}^{n}\ln 1 + \sum_{j=1}^{n}\ln s'_j \\
&= q\ln\left(1 - \frac{d^T s'}{4||d||e^T s'}\right) - \sum_{j=1}^{n}\ln\left(1 - \frac{d_j}{4||d||}\right).
\end{aligned}
$$

Using the relation

$$
-x - \frac{x^2}{2(1-a)} \leq \ln(1-x) \leq -x \tag{6}
$$

which holds for $|x| \leq a < 1$, we get:

$$
\begin{aligned}
G(\tilde{x},\tilde{s}) - G(e,s') &\leq -\frac{q\, d^T s'}{4||d||e^T s'} + \sum_{j=1}^{n}\frac{d_j}{4||d||} + \sum_{j=1}^{n}\frac{d_j^2}{16||d||^2\, 2(3/4)} \qquad \text{for } a = 1/4 \\
&= -\frac{q\, d^T s'}{4||d||e^T s'} + \frac{e^T d}{4||d||} + \frac{1}{24} \\
&= \frac{1}{4||d||}(e - \frac{q}{e^T s'}s')^T d + \frac{1}{24} \\
&= \frac{1}{4||d||}(-g)^T d + \frac{1}{24} \\
&= -\frac{||d||^2}{4||d||} + \frac{1}{24} \\
&= -\frac{||d||}{4} + \frac{1}{24} \\
&\leq -\frac{1}{10} + \frac{1}{24} \\
&= -\frac{7}{120}.
\end{aligned}
$$

Note that $g^T d = ||d||^2$, since d is the projection of g. (This is where we use the fact that d is the projected gradient!) $\square$

Before proving Proposition 25, we need the following lemma.

Lemma 26

$$\sum_{j=1}^{n} \ln(\tilde{s}_j) - n\ln(\frac{e^T\tilde{s}}{n}) \geq \frac{-2}{15}.$$

Proof:

Using the equality $\tilde{s} = \frac{\Delta}{q}(e+d)$ and Equation 6, which holds for $|x| \leq a < 1$, we see that

$$\begin{aligned}
\textstyle\sum_{j=1}^{n} \ln(\tilde{s}_j) - n\ln(\frac{e^T\tilde{s}}{n}) &= \textstyle\sum_{j=1}^{n} \ln(\frac{\Delta}{q}(1+d_j)) - n\ln(\frac{\Delta}{q}(1+\frac{e^Td}{n})) \\
&\geq \textstyle\sum_{j=1}^{n}(d_j - \frac{{d_j}^2}{2(3/5)}) - n\frac{e^Td}{n} \\
&\geq -\frac{||d||^2}{6/5} \\
&\geq \frac{-2}{15}
\end{aligned}$$

□

Proof of Proposition 25:

Using Lemma 26 and the inequality

$$\sum_{j=1}^{n} \ln(s_j) \leq n\ln(\frac{e^Ts}{n}),$$

which follows from the concavity of the logarithm function, we have

$$\begin{aligned}
G(e,\tilde{s}) - G(e,s') &= q\ln(\tfrac{e^T\tilde{s}}{e^Ts'}) - \textstyle\sum_{j=1}^{n}\ln(\tilde{s}_j) + \sum_{j=1}^{n}\ln(s'_j) \\
&\leq q\ln(\tfrac{e^T\tilde{s}}{e^Ts'}) + \tfrac{2}{15} - n\ln(\tfrac{e^T\tilde{s}}{n}) + n\ln(\tfrac{e^Ts'}{n}) \\
&= \tfrac{2}{15} + \sqrt{n}\ln(\tfrac{e^T\tilde{s}}{e^Ts'})
\end{aligned}$$

On the other hand,

$$e^T\tilde{s} = \frac{\Delta}{q}(n + e^Td)$$

and recall that $\Delta = e^Ts'$,

$$\frac{e^T\tilde{s}}{e^Ts'} = \frac{1}{q}(n+e^Td) \leq \frac{1}{n+\sqrt{n}}(n + 0.4\sqrt{n}),$$

since, by Cauchy-Schwartz inequality, $|e^Td| \leq ||e||\,||d|| = \sqrt{n}||d||$. Combining the above inequalities yields

$$\begin{aligned}
G(e,\tilde{s}) - G(e,s') &\leq \tfrac{2}{15} + \sqrt{n}\ln(1 - \tfrac{0.6\sqrt{n}}{n+\sqrt{n}}) \\
&\leq \tfrac{2}{15} - \tfrac{0.6n}{n+\sqrt{n}} \\
&\leq \tfrac{2}{15} - \tfrac{3}{10} = -\tfrac{1}{6}
\end{aligned}$$

since $n + \sqrt{n} \le 2n$. $\square$

This completes the analysis of Ye's algorithm.

16 Bit Complexity

Throughout the presentation of the algorithm, we assumed that all operations can be performed exactly. This is a fairly unrealistic assumption. For example, notice that $\|d\|$ might be irrational since it involves a square root. However, none of the thresholds we set were crucial. We could for example test whether $\|d\| \ge 0.4$ or $\|d\| \le 0.399$. To test this, we need to compute only a few bits of $\|d\|$. Also, if we perform a primal step (i.e. $\|d\| \ge 0.4$) and compute the first few bits of $\|d\|$ so that the resulting approximation $\|d\|_{ap}$ satisfies $(4/5)\|d\| \le \|d\|_{ap} \le \|d\|$ then if we go through the analysis of the primal step performed in Proposition 1, we obtain that the reduction in the potential function is at least $19/352$ instead of the previous $7/120$. Hence, by rounding $\|d\|$ we can still maintain a constant decrease in the potential function.

Another potential problem is when using Gaussian elimination to compute the projected gradient. We mentioned that Gaussian elimination requires $O(n^3)$ arithmetic operations but we need to show that, during the computation, the numbers involved have polynomial size. For that purpose, consider the use of Gaussian elimination to solve a system $Ax = b$ where

$$A = A^{(1)} = \begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \dots & a_{1n}^{(1)} \\ a_{21}^{(1)} & a_{22}^{(1)} & \dots & a_{2n}^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}^{(1)} & a_{m2}^{(1)} & \dots & a_{mn}^{(1)} \end{pmatrix}.$$

Assume that $a_{11} \neq 0$ (otherwise, we can permute rows or columns). In the first iteration, we substract $a_{i1}^{(1)}/a_{11}^{(1)}$ times the first row from row i where $i = 2, \dots, m$, resulting in the following matrix:

$$A^{(2)} = \begin{pmatrix} a_{11}^{(2)} & a_{12}^{(2)} & \dots & a_{1n}^{(2)} \\ 0 & a_{22}^{(2)} & \dots & a_{2n}^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & a_{m2}^{(2)} & \dots & a_{mn}^{(2)} \end{pmatrix}.$$

In general, $A^{(i+1)}$ is obtained by subtracting $a_{ji}^{(i)}/a_{ii}^{(i)}$ times row i from row j of $A^{(i)}$ for $j = i+1, \dots, m$.

Theorem 27 *For all $i \le j, k$, $a_{jk}^{(i)}$ can be written in the form $\det(B)/\det(C)$ where B and C are some submatrices of A.*

Proof:

Let B_i denote the $i \times i$ submatrix of $A^{(i)}$ consisting of the first i entries of the first i rows. Let $B_{jk}^{(i)}$ denote the $i \times i$ submatrix of $A^{(i)}$ consisting of the first $i-1$ rows and row j, and the first $i-1$ columns and column k. Since B_i and $B_{jk}^{(i)}$ are upper triangular matrices, their determinants are the products of the entries along the main diagonal and, as a result, we have:

$$a_{ii}^{(i)} = \frac{\det(B_i)}{\det(B_{i-1})}$$

and

$$a_{jk}^{(i)} = \frac{\det(B_{jk}^{(i)})}{\det(B_{i-1})}.$$

Moreover, remember that row operations do not affect the determinants and, hence, the determinants of $B_{jk}^{(i)}$ and B_{i-1} are also determinants of submatrices of the original matrix A. $\square$

Using the fact that the size of the determinant of any submatrix of A is at most the size of the matrix A, we obtain that all numbers occuring during Gaussian elimination require only $O(L)$ bits.

Finally, we need to round the current iterates x, y and s to $O(L)$ bits. Otherwise, these vectors would require a constantly increasing number of bits as we iterate. By rounding up x and s, we insure that these vectors are still strictly positive. It is fairly easy to check that this rounding does not change the potential function by a significant amount and so the analysis of the algorithm is still valid. Notice that now the primal and dual constraints might be slightly violated but this can be taken care of in the rounding step.

A Transformation for the Interior Point Algorithm

In this appendix, we show how a pair of dual linear programs

$$(P) \quad \begin{array}{ll} Min & c^T x \\ s.t. & Ax = b \\ & x \geq 0 \end{array} \qquad\qquad (D) \quad \begin{array}{lcccc} Max & b^T y & & & \\ s.t. & A^T y & + & s & = & c \\ & & & s & \geq & 0 \end{array}$$

can be transformed so that we know a strictly feasible primal solution x_0 and a strictly feasible vector of dual slacks s_0 such that $G(x_0; s_0) = O(\sqrt{n}L)$ where

$$G(x;s) = q\ln(x^T s) - \sum_{j=1}^{n} \ln(x_j s_j)$$

and $q = n + \sqrt{n}$.

Consider the pair of dual linear programs:

$$
\begin{array}{llrcrcrcl}
 & Min & c^T x & + & k_c x_{n+1} & & & & \\
(P') & s.t. & Ax & + & (b - 2^{2L}Ae)x_{n+1} & & & = & b \\
 & & (2^{4L}e - c)^T x & & & + & 2^{4L}x_{n+2} & = & k_b \\
 & & x \geq 0 & & x_{n+1} \geq 0 & & x_{n+2} \geq 0 & &
\end{array}
$$

and

$$
\begin{array}{llrcrcccccl}
 & Min & b^T y & + & k_b y_{m+1} & & & & & & \\
(D') & s.t. & A^T y & + & (2^{4L}e - c)y_{m+1} & + & s & & & = & c \\
 & & (b - 2^{2L}Ae)^T y & & & & & + \; s_{n+1} & & = & k_c \\
 & & & & 2^{4L}y_{m+1} & & & & + \; s_{n+2} & = & 0 \\
 & & & & & & s, & s_{n+1}, & s_{n+2} & \geq & 0
\end{array}
$$

where $k_b = 2^{6L}(n+1) - 2^{2L}c^T e$ is chosen in such a way that $x' = (x, x_{n+1}, x_{n+2}) = (2^{2L}e, 1, 2^{2L})$ is a (strict) feasible solution to (P') and $k_c = 2^{6L}$. Notice that $(y', s') = (y, y_{m+1}, s, s_{n+1}, s_{n+2}) = (0, -1, 2^{4L}e, k_c, 2^{4L})$ is a feasible solution to (D') with $s' > 0$. x' and (y', s') serve as our initial feasible solutions.

We have to show:

1. $G(x'; s') = O(\sqrt{n'}L)$ where $n' = n + 2$,
2. the pair $(P') - (D')$ is equivalent to $(P) - (D)$,
3. the input size L' for (P') as defined in the lecture notes does not increase too much.

The proofs of these statements are simple but heavily use the definition of L and the fact that vertices have all components bounded by 2^L.

We first show 1. Notice first that $x'_j s'_j = 2^{6L}$ for all j, implying that

$$
\begin{aligned}
G(x'; s') &= (n' + \sqrt{n'}) \ln(x'^T s') - \sum_{j=1}^{n'} \ln(x'_j s'_j) \\
&= (n' + \sqrt{n'}) \ln(2^{6L} n') - n' \ln(2^{6L}) \\
&= \sqrt{n'} \ln(2^{6L}) + (n' + \sqrt{n'}) \ln(n') \\
&= O(\sqrt{n'}L)
\end{aligned}
$$

In order to show that $(P') - (D')$ are equivalent to $(P) - (D)$, we consider an optimal solution x^* to (P) and an optimal solution (y^*, s^*) to (D) (the case where (P) or (D) is infeasible is considered in the problem set). Without loss of generality, we can assume that x^* and (y^*, s^*) are vertices of the corresponding polyhedra. In particular, this means that $x^*_j, |y^*_j|, s^*_j < 2^L$.

Proposition 28 *Let* $x' = (x^*, 0, (k_b - (2^{4L}e - c)^T x^*)/2^{4L})$ *and let* $(y', s') = (y^*, 0, , s^*, k_c - (b - 2^{2L}Ae)^T y^*, 0)$. *Then*

1. x' *is a feasible solution to* (P') *with* $x'_{n+2} > 0$,
2. (y', s') *is a feasible solution to* (D') *with* $s'_{n+1} > 0$,
3. x' *and* (y', s') *satisfy complementary slackness, i.e. they constitute a pair of optimal solutions for* $(P') - (D')$.

Proof:

To show that x' is a feasible solution to (P') with $x'_{n+2} > 0$, we only need to show that $k_b - (2^{4L}e - c)^T x^* > 0$ (the reader can easily verify that x' satisfy all the equalities defining the feasible region of (P')). This follows from the fact that

$$(2^{4L}e - c)^T x^* \leq n(2^{4L} + 2^L)2^L = n(2^{5L} + 2^{2L}) < n2^{6L}$$

and

$$k_b = 2^{6L}(n+1) - 2^{2L}c^T e \geq 2^{6L}(n+1) - 2^{2L}n \max_j |c_j| \geq 2^{6L}n + 2^{6L} - 2^{3L} > n2^{6L}$$

where we have used the definition of L and the fact that vertices have all their entries bounded by 2^L.

To show that (y', s') is a feasible solution to (D') with $s'_{n+1} > 0$, we only need to show that $k_c - (b - 2^{2L}Ae)^T y^* > 0$. This is true since

$$\begin{aligned}
(b - 2^{2L}Ae)^T y^* &\leq b^T y^* - 2^{2L}e^T A^T y^* \\
&\leq m \max_i |b_i| 2^L + 2^{2L} nm \max_{i,j} |a_{ij}| 2^L \\
&= 2^{2L} + 2^{4L} < 2^{6L} = k_c.
\end{aligned}$$

x' and (y', s') satisfy complementary slackness since

- $x^{*T} s^* = 0$ by optimality of x^* and (y^*, s^*) for (P) and (D)
- $x'_{n+1} s'_{n+1} = 0$ and
- $x'_{n+2} s'_{n+2} = 0$.

□

This proposition shows that, from an optimal solution to $(P) - (D)$, we can easily construct an optimal solution to $(P') - (D')$ of the same cost. Since this solution has $s'_{n+1} > 0$, any optimal solution $\hat{x}$ to (P') must have $\hat{x}_{n+1} = 0$. Moreover, since $x'_{n+2} > 0$, any optimal solution $(\hat{y}, \hat{s})$ to (D') must satisfy $\hat{s}_{n+2} = 0$ and, as a result, $\hat{y}_{m+1} = 0$. Hence, from any optimal solution to $(P') - (D')$, we can easily deduce an optimal solution to $(P) - (D)$. This shows the equivalence between $(P) - (D)$ and $(P') - (D')$.

By some tedious but straightforward calculations, it is possible to show that L' (corresponding to $(P') - (D')$) is at most $24L$. In other words, $(P) - (D)$ and $(P') - (D')$ have equivalent sizes.

References

[1] V. Chvatal. *Linear Programming.* W.H. Freeman and Company, 1983.

[2] G. Dantzig. Maximization of a linear function of variables subject to linear inequalities. In T. Koopmans, editor, *Activity Analysis of Production and Allocation*, pages 339–347. John Wiley & Sons, Inc., 1951.

[3] R. M. Freund. Polynomial-time algorithms for linear programming based only on primal scaling and project gradients of a potential function. *Mathematical Programming*, 51:203–222, 1991.

[4] D. Goldfarb and M. Todd. Linear programming. In *Handbook in Operations Research and Management Science*, volume 1, pages 73–170. Elsevier Science Publishers B.V., 1989.

[5] C. C. Gonzaga. Path-following methods for linear programming. *SIAM Review*, 34:167–224, 1992.

[6] L. Khachian. A polynomial algorithm for linear programming. *Doklady Akad. Nauk USSR*, 244(5):1093–1096, 1979.

[7] K. Murty. *Linear Programming.* John Wiley & Sons, 1983.

[8] A. Schrijver. *Theory of Linear and Integer Programming.* John Wiley & Sons, 1986.

[9] Y. Ye. An $O(n^3L)$ potential reduction algorithm for linear programming. *Mathematical Programming*, 50:239–258, 1991.