Lecture 5

In which we introduce linear programming.

1 Linear Programming

A *linear program* is an optimization problem in which we have a collection of variables, which can take real values, and we want to find an assignment of values to the variables that satisfies a given collection of linear inequalities and that maximizes or minimizes a given linear function.

(The term *programming* in *linear programming*, is not used as in *computer programming*, but as in, e.g., *tv programming*, to mean *planning.*)

For example, the following is a linear program.

$$\begin{array}{ll} \text{maximize} & x_1 + x_2 \\ \text{subject to} & \\ & x_1 + 2x_2 \leq 1 \\ & 2x_1 + x_2 \leq 1 \\ & x_1 \geq 0 \\ & x_2 \geq 0 \end{array} \tag{1}$$

The linear function that we want to optimize ($x_1 + x_2$ in the above example) is called the *objective function.* A *feasible solution* is an assignment of values to the variables that satisfies the inequalities. The value that the objective function gives to an assignment is called the *cost* of the assignment. For example, $x_1 := \frac{1}{3}$ and $x_2 := \frac{1}{3}$ is a feasible solution, of cost $\frac{2}{3}$. Note that if x_1, x_2 are values that satisfy the inequalities, then, by summing the first two inequalities, we see that

$$3x_1 + 3x_2 \leq 2$$

that is,

$$x_1 + x_2 \leq \frac{2}{3}$$

and so no feasible solution has cost higher than $\frac{2}{3}$, so the solution $x_1 := \frac{1}{3}$, $x_2 := \frac{1}{3}$ is optimal. As we will see in the next lecture, this trick of summing inequalities to verify the optimality of a solution is part of the very general theory of *duality* of linear programming.

Linear programming is a rather different optimization problem from the ones we have studied so far. Optimization problems such as Vertex Cover, Set Cover, Steiner Tree and TSP are such that, for a given input, there is only a finite number of possible solutions, so it is always trivial to solve the problem in finite time. The number of solutions, however, is typically exponentially big in the size of the input and so, in order to be able to solve the problem on reasonably large inputs, we look for polynomial-time algorithms. In linear programming, however, each variable can take an infinite number of possible values, so it is not even clear that the problem is solvable in finite time.

As we will see, it is indeed possible to solve linear programming problems in finite time, and there are in fact, polynomial time algorithms and efficient algorithms that solve linear programs optimally.

There are at least two reasons why we are going to study linear programming in a course devoted to combinatorial optimization:

- Efficient linear programming solvers are often used as part of the toolkit to design exact or approximate algorithms for combinatorial problems.
- The powerful theory of *duality* of linear programming, that we will describe in the next lecture, is a very useful mathematical theory to reason about algorithms, including purely combinatorial algorithms for combinatorial problems that seemingly have no connection with continuous optimization.

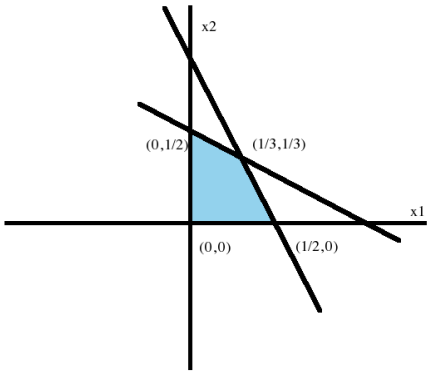
2 A Geometric Interpretation

2.1 A 2-Dimensional Example

Consider again the linear program (1). Since it has two variables, we can think of any possible assignment of values to the variables as a point (x_1, x_2) in the plane. With this interpretation, every inequality, for example $x_1 + 2x_2 \leq 1$, divides the plane into two regions: the set of points (x_1, x_2) such that $x_1 + 2x_2 > 1$, which are definitely not feasible solutions, and the points such that $x+1+2x_2 \leq 1$, which satisfy the inequality

and which could be feasible provided that they also satisfy the other inequalities. The line with equation $x_1 + 2x_2 = 1$ is the boundary between the two regions.

The set of feasible solutions to (1) is the set of points which satisfy all four inequalities, shown in blue below:

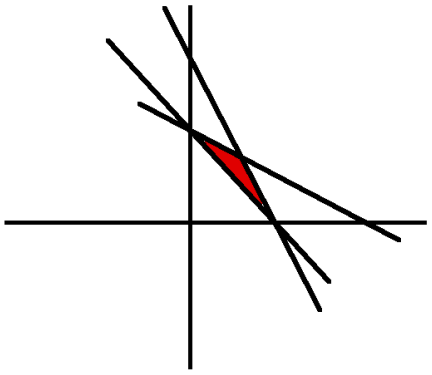


The feasible region is a polygon with four edges, one for each inequality. This is not entirely a coincidence: for each inequality, for example $x_1 + 2x_2 \leq 1$, we can look at the line which is the boundary between the region of points that satisfy the inequality and the region of points that do not, that is, the line $x_1 + 2x_1 = 1$ in this example. The points on the line that satisfy the other constraints form a segment (in the example, the segment of the line $x_1 + 2x_2 = 1$ such that $0 \leq x_1 \leq 1/3$), and that segment is one of the edges of the polygon of feasible solutions. Although it does not happen in our example, it could also be that if we take one of the inequalities, consider the line which is the boundary of the set of points that satisfy the inequality, and look at which points on the line are feasible for the linear program, we end up with the empty set (for example, suppose that in the above example we also had the inequality $x_1 + x_2 \geq -1$); in this case the inequality does not give rise to an edge of the polygon of feasible solutions. Another possibility is that the line intersects the feasible region only at one point (for example suppose we also had the inequality $x_1 + x_2 \geq 0$). Yet another possibility is that our polygon is unbounded, in which case one of its edges is not a segment but a half-line (for example, suppose we did not have the inequality $x_1 \geq 0$, then the half-line of points such that $x_2 = 0$ and $x_1 \leq 1$ would have been an edge).

To look for the best feasible solution, we can start from an arbitrary point, for example the vertex $(0, 0)$. We can then divide the plane into two regions: the set of points whose cost is greater than or equal to the cost of $(0, 0)$, that is the set of points such that $x_1 + x_2 \geq 0$, and the set of points of cost lower than the cost of $(0, 0)$, that is,

the set of points such that $x_1 + x_2 < 0$. Clearly we only need to continue our search in the first region, although we see that actually the entire set of feasible points is in the first region, so the point $(0, 0)$ is actually the *worst* solution.

So we continue our search by trying another vertex, for example $(1/2, 0)$. Again we can divide the plane into the set of points of cost greater than or equal to the cost of $(1/2, 0)$, that is the points such that $x_1 + x_2 \geq 1/2$, and the set of points of cost lower than the cost of $(1, 0)$. We again want to restrict ourselves to the first region, and we see that we have now narrowed down our search quite a bit: the feasible points in the first region are shown in red:



So we try another vertex in the red region, for example $\left(\frac{1}{3}, \frac{1}{3}\right)$, which has cost $\frac{2}{3}$. If we consider the region of points of cost greater than or equal to the cost of the point $(1/3, 1/3)$, that is, the region $x_1 + x_2 \geq 2/3$, we see that the point $(1/3, 1/3)$ is the only feasible point in the region, and so there is no other feasible point of higher cost, and we have found our optimal solution.

2.2 A 3-Dimensional Example

Consider now a linear program with three variables, for example

$$\begin{array}{ll} \text{maximize} & x_1 + 2x_2 - x_3 \\ \text{subject to} & \\ & x_1 + x_2 \leq 1 \\ & x_2 + x_3 \leq 1 \\ & x_1 \geq 0 \\ & x_2 \geq 0 \\ & x_3 \geq 0 \end{array} \tag{2}$$

In this case we can think of every assignment to the variables as a point (x_1, x_2, x_3) in three-dimensional space. Each constraint divides the space into two regions as before; for example the constraint $x_1 + x_2 \leq 1$ divides the space into the region of points (x_1, x_2, x_3) such that $x_1 + x_2 \leq 1$, which satisfy the equation, and points such that $x_1 + x_2 > 1$, which do not. The boundary between the regions is the *plane* of points such that $x_1 + x_2 = 1$. The region of points that satisfy an inequality is called a *half-space*.

The set of feasible points is a *polyhedron* (plural: polyhedra). A polyhedron is bounded by *faces*, which are themselves polygons. For example, a cube has six faces, and each face is a square. In general, if we take an inequality, for example $x_3 \geq 0$, and consider the plane $x_3 = 0$ which is the boundary of the half-space of points that satisfy the inequality, and we consider the set of feasible points in that plane, the resulting polygon (if it's not the empty set) is one of the faces of our polyhedron. For example, the set of feasible points in the place $x_3 = 0$ is the triangle given by the inequalities $x_1 \geq 0$, $x_2 \geq 0$, $x_1 + x_2 \leq 1$, with vertices $(0, 0, 0)$, $(0, 1, 0)$ and $(1, 0, 0)$. So we see that a 2-dimensional *face* is obtained by taking our inequalities and changing one of them to equality, provided that the resulting feasible region is two-dimensional; a 1-dimensional *edge* is obtained by changing two inequalities to equality, again provided that the resulting constraints define a 1-dimensional region; and a vertex is obtained by changing three inequalities to equality, provided that the resulting point is feasible for the other inequalities.

As before, we can start from a feasible point, for example the vertex $(0, 0, 0)$, of cost zero, obtained by changing the last three inequalities to equality. We need to check if there are feasible points, other than $(0, 0, 0)$, such that $x_1 + 2x_2 - x_3 \geq 0$. That is, we are interested in the set of points such that

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

which is again a polyhedron, of which $(0, 0, 0)$ is a vertex. To find another vertex, if any, we try to start from the three inequalities that we changed to equality to find $(0, 0, 0)$, and remove one to see if we get an edge of non-zero length or just the point $(0, 0, 0)$ again.

For example, if we keep $x_1 = 0$, $x_2 = 0$, we see that only feasible value of x_3 is zero, so there is no edge; if we keep $x_1 = 0, x_3 = 0$, we have that the region $0 \leq x_2 \leq 1$ is feasible, and so it is an edge of the above polyhedron. The other vertex of that edge is $(0, 1, 0)$, which is the next solution that we shall consider. It is a solution of cost

2, so, in order to look for a better solution, we want to consider the polyhedron

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

In order to see if this polyhedron has any edge of non-zero length, we again keep only two of the three equations that defined our vertex $(0, 1, 0)$, that is only two of the equations $x_1 = 0$, $x_3 = 0$, $x_1 + x_2 = 1$. If we keep the first two, $(0, 1, 0)$ is the only feasible point. If we keep $x_3 = 0$ and $x_1 + x_2 = 1$, then $(0, 1, 0)$ is again the only feasible point. If we keep $x_1 = 0$ and $x + 1 + x_2 = 1$, that is $x_1 = 0$ and $x_2 = 1$, we see again that the only feasible point is $(0, 1, 0)$. These are the only three edges that could have had $(0, 1, 0)$ as an endpoint, and since $(0, 1, 0)$ is a vertex of the above polytope, we have to conclude that the polytope has no edge, and so it is made of the single point $(0, 1, 0)$.

This means that $(0, 1, 0)$ is the optimal solution of (2)

2.3 The General Case

In general, if we have linear program with n variables $x_1, \ldots, x_n$, we can think of every assignment to the variables as an n-dimensional point $(x_1, \ldots, x_n)$ in $\mathbb{R}^n$.

Every inequality $a_1x_1 + \ldots a_nx_n \leq b$ divides the space $\mathbb{R}^n$ into the region that satisfies the inequality and the region that does not satisfy the inequality, with the *hyperplane* $a_1x_1 + \cdots a_nx_n$ being the boundary between the two regions. The two regions are called *half-spaces*.

The set of feasible points is an intersection of half-spaces and is called a *polytope*.

Generalizing the approach that we have used in the previous two examples, the following is the outline of an algorithm to find an optimal solution to a given maximization linear program:

1. Start from a vertex $(a_1, \ldots, a_n)$ in the feasible region, by changing n of the inequalities to equality in such a way that: (i) the resulting n equations are linearly independent, and (ii) the unique solution is feasible for the remaining inequalities;

    - If there is no such vertex, output "linear program is infeasible."

2. Consider the n possible edges of the polytope of feasible solutions that have $(a_1, \ldots, a_n)$ as an endpoint. That is, for each of the n equations that identified $(a_1, \ldots, a_n)$, set back that equation to an inequality, and consider the set of solutions that are feasible for the other $n-1$ equations and for the inequalities (this set of points can be a line, a half-line, a segment, or just the point $(a_1, \ldots, a_n)$).

   - If there is an edge that contains points of arbitrarily large cost, then output "optimum is unbounded"
   - Else, if there are edges that contain points of cost larger than $(a_1, \ldots, a_n)$, then let $(b_1, \ldots, b_n)$ be the second endpoint of one of such edges
     - $(a_1, \ldots, a_n) := (b_1, \ldots, b_n)$;
     - go to 2
   - Else, output "$(a_1, \ldots, a_n)$ is an optimal solution"

This is the outline of an algorithm called the *Simplex algorithm*. It is not a complete description because:

- We haven't discussed how to find the initial vertex. This is done by constructing a new polytope such that finding an optimal solution in the new polytope either gives us a feasible vertex for the original linear program, or a "certificate" that the original problem is infeasible. We then apply the Simplex algorithm to the new polytope. Now, this looks like the beginning of an infinite loop, but the new polytope is constructed in such a way that it is easy to find an initial feasible vertex.

- We haven't discussed certain special cases; for example it is possible for a polytope to not have any vertex, for example if the number of inequalities is less than the number of variables. (In such a case we can either add inequalities or eliminate variables in a way that do not change the optimum but that creates vertices.) Also there can be cases in which a vertex of a polytope in $\mathbb{R}^n$ is the endpoint of more than n edges (consider a pyramid with a square base in $\mathbb{R}^3$: the top vertex has is an endpoint of four edges), while the algorithm, as described above, considers at most n edges for every vertex.

The simplex algorithm shows that a linear program can always be solved in finite time, and in fact in time that is at most exponential in the number of variables. This is because each iteration takes polynomial time and moves to a new vertex, and if there are m inequalities and n variables there can be at most $\binom{m}{n} \leq m^n$ vertices.

Unfortunately, for all the known variants of the simplex method (which differ in the way they choose the vertex to move to, when there is more than possible choice) there

are examples of linear programs on which the algorithm takes exponential time. In practice, however, the simplex algorithm is usually very fast, even on linear programs with tens or hundreds of thousands of variables and constraints.

2.4 Polynomial Time Algorithms for LInear Programming

Two (families of) polynomial time algorithms for linear programming are known.

One, called the ellipsoid algorithm, starts by finding an ellipsoid (the high-dimensional analog of an ellipse, a "squashed disk") that contains the optimal solution, and then, at every step, it constructs another ellipsoid whose volume is a smaller than the previous one, while still being guaranteed to contain the optimal solution. After several iterations, the algorithm identifies a tiny region that contains the optimal solution. It is known that if a linear program has a finite optimum, the values of the x_i in the optimal solution are rational numbers in which both the denominator and numerator have a polynomial number of digits in the size of the input (assuming all coefficients in the objective function and in the inequalities are also rational numbers and that we count the number of digits in their fractional representation when we compute the size of the input), and so if the final ellipsoid is small enough there is only one point with such rational coordinates in the ellipsoid.

The other algorithm, which is actually a family of algorithms, uses the *interior point* method, in which the algorithm computes a sequence of points in the interior of the polytope (in contrast to the simplex algorithm, which finds a sequence of vertices on the exterior of the polytope), where each point is obtained from the previous one by optimizing a properly chosen function that favors points of higher cost for the objective function, and disfavors points that are too close to the boundary of the polytope. Eventually, the algorithm finds a point that is arbitrary close to the optimal vertex, and the actual optimal vertex can be found, like in the ellipsoid method, once it is the unique point with bounded rational coordinates that is close to the current point.

2.5 Summary

Here are the important points to remember about what we discussed so far:

- In a linear program we are given a set of variables $x_1, \ldots, x_n$ and we want to find an assignment to the variables that satisfies a given set of linear equalities, and that maximizes or minimizes a given linear *objective function* of the variables. An assignment that satisfies the inequalities is called a *feasible solution*;
- If we think of every possible assignment to the variables as a point in $\mathbb{R}^n$, then the set of feasible solutions forms a *polytope*;

- It is possible that there is no feasible solution to the given inequalities, in which case we call the linear program *infeasible*.
- If the linear program is feasible, it is possible that it is of maximization type and there are solutions of arbitrarily large cost, or that it is of minimization type and there are solutions of arbitrarily small cost. In this case we say that the linear program is *unbounded*. Sometimes we will say that the optimum of an unbounded maximization linear program is $+\infty$, and that the optimum of an unbounded minimization linear program is $-\infty$, even though this is not entirely correct because there is no feasible solution of cost $+\infty$ or $-\infty$, but rather a sequence of solutions such the limit of their cost is $+\infty$ or $-\infty$.
- If the linear program is feasible and not unbounded then it has a finite optimum, and we are interested in finding a feasible solution of optimum cost.
- The simplex algorithm, the ellipsoid algorithm, and the interior point algorithms are able, given a linear program, to determine if it is feasible or not, if feasible they can determine if it is bounded or unbounded, and if feasible and bounded they can find a solution of optimum cost. All three run in finite time; in the worst case, the simplex algorithm runs in exponential time, while the other algorithms run in time polynomial in the size of the input.
- When we refer to the "size of the input" we assume that all coefficients are rational numbers, and the size of the input is the total number of bits necessary to represent the coefficients of the objective function and of the inequalities as ratios of integers. For example, the rational number a/b requires $\log_2 a + \log_2 b + O(1)$ bits to be represented, if a and b have no common factor.

3 Standard Form for Linear Programs

We say that a maximization linear program with n variables is in *standard form* if for every variable x_i we have the inequality $x_i \geq 0$ and all other m inequalities are of $\leq$ type. A linear program in standard form can be written as

$$\begin{array}{ll} \text{maximize} & \mathbf{c}^T\mathbf{x} \\ \text{subject to} & \\ & A\mathbf{x} \leq \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0} \end{array} \tag{3}$$

Let us unpack the above notation.

The vector $\mathbf{c} = \begin{pmatrix} c_1 \\ \vdots \\ c_n \end{pmatrix} \in \mathbb{R}^n$ is the column vector of coefficients of the objective function, $\mathbf{x} = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$ is the column vector of variables, $\mathbf{c}^T = (c_1, \ldots, c_n)$ is the transpose of $\mathbf{c}$, a row vector, and $\mathbf{c}^T\mathbf{x}$ is the matrix product of the $1 \times n$ "matrix" $\mathbf{c}^T$ times the $n \times 1$ "matrix" $\mathbf{x}$, which is the value

$$c_1x_1 + \cdots c_nx_n$$

that is, the objective function.

The matrix A is the $n \times m$ matrix of coefficients of the left-hand sides of the inequalities, and b is the m-dimensional vector of right-hand sides of the inequalities. When we write $\mathbf{a} \leq \mathbf{b}$, for two vectors $\mathbf{a} = \begin{pmatrix} a_1 \\ \vdots \\ a_m \end{pmatrix}$ and $\mathbf{b} = \begin{pmatrix} b_1 \\ \vdots \\ b_m \end{pmatrix}$ we mean the m inequalities $a_1 \leq b_1$, $\ldots$, $a_m \leq b_m$, so the inequality $A\mathbf{x} \leq \mathbf{b}$ means the collection of inequalities

$$a_{1,1}x_1 + \cdots + a_{1,n}x_n \leq b_1$$

$$\cdots$$

$$a_{m,1}x_1 + \cdots + a_{m,n}x_n \leq b_m$$

Putting a linear program in standard form is a useful first step for linear programming algorithms, and it is also useful to develop the theory of *duality* as we will do in the next lecture.

It is easy to see that given an arbitrary linear program we can find an equivalent linear program in standard form.

- If we have a linear program in which a variable x is not required to be ≥ 0, we can introduce two new variables x' and x'', apply the substitution $x \leftarrow x' - x''$ in every inequality in which x appears, and add the inequalities $x' \geq 0$, $x'' \geq 0$. Any feasible solution for the new linear program is feasible for the original one, after assigning $x := x' - x''$, with the same cost; also, any solution that was feasible for the original program can be converted into a solution for the new linear program, with the same cost, by assigning $x' := x$, $x'' := 0$ if $x > 0$ and $x' := 0$, $x'' = -x$ if $x \leq 0$.

- If we have an inequality of $\geq$ type, other than the inequalities $x_i \geq 0$, we can change sign to the left-hand side and the right-hand side, and change the direction of the inequality.
- Some definitions of linear programming allow equations as constraints. If we have an equation $\mathbf{a}^T\mathbf{x} = b$, we rewrite it as the two inequalities $\mathbf{a}^T\mathbf{x} \leq b$ and $-\mathbf{a}^T\mathbf{x} \leq -b$.

The standard form for a minimization problem is

$$\begin{array}{ll} \text{minimize} & \mathbf{c}^T\mathbf{x} \\ \text{subject to} & \\ & A\mathbf{x} \geq \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0} \end{array} \tag{4}$$

As we did for maximization problems, every minimization problem can be put into normal form by changing the sign of inequalities and doing the substitution $x \to x' - x''$ for every variable x that does not have a non-negativity constraint $x \geq 0$.