Notes 17 for CS 170

1 Linear Programming

It turns out that a great many problems can be formulated as *linear programs*, i.e., maximizing (or minimizing) a linear function of some variables, subject to *constraints* on the variables; these constraints are either *linear equations* or *linear inequalities*, i.e., linear functions of the variables either set equal to a constant, or $\leq$ a constant, or $\geq$ a constant. Most of this lecture will concentrate on recognizing how to reformulate (or *reduce*) a given problem to a linear program, even though it is not originally given this way. The advantage of this is that there are several good algorithms for solving linear programs that are available. We will only say a few words about these algorithms, and instead concentrate on formulating problems as linear programs.

2 Introductory example in 2D

Suppose that a company produces two products, and wishes to decide the level of production of each product so as to maximize profits. Let x_1 be the amount of Product 1 produced in a month, and x_2 that of Product 2. Each unit of Product 1 brings to the company a profit of 120, and each unit of Product 2 a profit of 500. At this point it seems that the company should only produce Product 2, but there are some constraints on x_1 and x_2 that the company must satisfy (besides the obvious one, $x_1, x_2 \geq 0$). First, x_1 cannot be more than 200, and x_2 more than 300—because of raw material limitations, say. Also, the sum of x_1 and x_2 must be at most 400, because of labor constraints. What are the best levels of production to maximize profits?

We represent the situation by a *linear program*, as follows (where we have numbered the constraints for later reference):

$$\max 120x_1 + 500x_2$$

$$\begin{array}{ll} (1) & x_1 \leq 200 \\ (2) & x_2 \leq 300 \\ (3) & x_1 + x_2 \leq 400 \\ (4) & x_1 \geq 0 \\ (5) & x_2 \geq 0 \end{array}$$

The set of all *feasible* solutions of this linear program (that is, all vectors (x_1, x_2) in 2D space that satisfy all constraints) is precisely the (black) polygon shown in Figure 1 below, with vertices numbered 1 through 5.

The vertices are given in the following table, and labelled in Figure 1 (we explain the meaning of "active constraint" below):

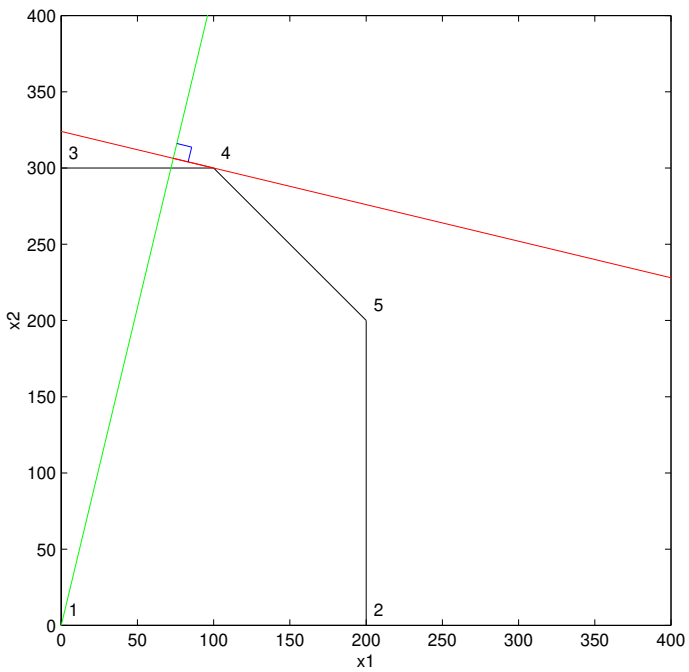

Figure 1: The feasible region (polygon), solution vertex (#4), and line of constant profit

Vertex	x1-coord	x2-coord	Active constraints
1	0	0	4,5
2	200	0	1,5
3	0	300	2,4
4	100	300	2,3
5	200	200	1,3

The reason all these constraints yield the polygon shown is as follows. Recall that a linear equation like $ax_1 + bx_2 = p$ defines a line in the plane. The *inequality* $ax_1 + bx_2 \leq p$ defines all points on one side of that line, i.e., a *half-plane*, which you can think of as an (infinite) polygon with just one side. If we have two such constraints, the points have to lie in the intersection of two half-planes, i.e., a polygon with 2 sides. Each constraint adds (at most) one more side to the polygon. For example, the 5 constraints above yield 5 sides in the polyhedron: constraint (1), $x_2 \leq 200$, yields the side with vertices #2 and #5, constraint (2), $x_3 \leq 300$, yields the side with vertices #3 and #4, constraint (3), $x_1 + x_2 \leq 400$, yields the side with vertices #4 and #5, constraint (4), $x_1 \geq 0$, yields the side with vertices #1 and #3, and constraint (5), $x_2 \geq 0$, yields the side with vertices #1 and #2. We also say that constraint (1) is *active* at vertices #2 and #5 since it is just barely satisfied at those vertices (at other vertices x_2 is strictly less than 200).

We wish to maximize the linear function profit=$120x_1 + 500x_2$ over all points of this polygon. We think of this geometrically as follows. The set of all points satisfying $p = 120x_1 + 500x_2$ for a fixed p is a line. As we vary p, we get different lines, all parallel to one another, and all perpendicular to the vector $(120, 500)$. (We review this basic geometrical fact below).

Geometrically, we want to increase p so that the line is just barely touching the polygon at one point, and increasing p would make the plane miss the polygon entirely. It should be clear geometrically that this point will usually be a *vertex* of the polygon. This point is the *optimal solution* of the linear program. This is shown in the figure above, where the green line (going from the origin to the top of the graph) is parallel to the vector $(120, 500)$, the red line (going all the way from left to right across the graph) is perpendicular to the green line and connects to the solution vertex #4 $(100, 300)$, which occurs for $p = 120 * 100 + 500 * 300 = 162000$ in profit. (The blue "L" connecting the green and red lines indicates that they are perpendicular.)

(Now we review why the equation $y_1 \cdot x_1 + y_2 \cdot x_2 = p$ defines a line perpendicular to the vector $y = (y_1, y_2)$. You may skip this if this is familiar material. Write the equation as a dot product of y and $x = (x_1, x_2)$: $y * x = p$. First consider the case $p = 0$, so $y * x = 0$. Recall that the if the dot product of two vectors is 0, then the vectors are perpendicular. So when $p = 0$, $y * x = 0$ defines the set of all vectors (points) x perpendicular to y, which is a line through the origin. When $p \neq 0$, we argue as follows. Note that $y * y = y_1^2 + y_2^2$. Then define the vector $\bar{y} = (p/(y * y))y$, a multiple of y. Then we can easily confirm that $\bar{y}$ satisfies the equation because $y * \bar{y} = (p/(y * y))(y * y) = p$. Now think of every point x as the sum of two vectors $x = \bar{x} + \bar{y}$. Substituting in the equation for x we get $p = y * x = y * (\bar{x} + \bar{y}) = y * \bar{x} + y * \bar{y} = y * \bar{x} + p$, or $y * \bar{x} = 0$. In other words, the points $\bar{x}$ lie in a plane through the origin perpendicular to y, and the points $x = \bar{x} + \bar{y}$ are gotten just by adding the vector $\bar{y}$ to each vector in this plane. This just shifts the plane in the direction $\bar{y}$, but leaves it perpendicular to y.)

There are three other geometric possibilities that could occur:

- If the planes for each p are parallel to an edge or face touching the solution vertex, then all points in that edge or face will also be solutions. This just means that the solution is not unique, but we can still solve the linear program. This would occur in the above example if we changed the profits from $(120, 500)$ to $(100, 100)$; we would get equally large profits of $p = 40000$ either at vertex #5 $(200, 200)$, vertex #4 $(100, 300)$, or anywhere on the edge between them.

- It may be that the polygon is *infinite*, and that p can be made arbitrarily large. For example, removing the constraints $x_1 + x_2 \leq 400$ and $x_1 \leq 200$ means that x_1 could become arbitrarily large. Thus $(x_1, 0)$ is in the polygon for all $x_1 > 0$, yielding an arbitrarily large profit $120x_1$. If this happens, it probably means you forgot a constraint and so formulated your linear program incorrectly.

- It may be that the polygon is *empty*, which is also called *infeasible*. This means that *no* points (x_1, x_2) satisfy the constraints. This would be the case if we added the constraint, say, that $x_1 + 2x_2 \geq 800$; since the largest value of $x_1 + 2x_2$ occurs at vertex #4, with $x_1 + 2x_2 = 100 + 2 * 300 = 700$, this extra constraint cannot be satisfied. When this happens it means that your problem is overconstrained, and you have to weaken or eliminate one or more constraints.

3 Introductory Example in 3D

Now we take the same company as in the last section, add Product 3 to its product line, along with some constraints, and ask how the problem changes. Each unit of Product 3 brings a profit of 200, and the sum of x_2 and three times x_3 must be at most 600, because Products 2 and 3 share the same piece of equipment ($x_2 + 3x_3 \leq 600$).

This changes the linear program to

$$\max 120x_1 + 500x_2 + 200x_3$$

$$\begin{aligned}
&(1) && x_1 \leq 200 \\
&(2) && x_2 \leq 300 \\
&(3) && x_1 + x_2 \leq 400 \\
&(4) && x_1 \geq 0 \\
&(5) && x_2 \geq 0 \\
&(6) && x_3 \geq 0 \\
&(7) && x_2 + 3x_3 \leq 600
\end{aligned}$$

Each constraint correspond to being on one side of a plane in (x_1, x_2, x_3) space, a *half-space*. The 7 constraints result in a 7-sided polyhedron shown in Figure 2. The polyhedron has vertices and active constraints show here:

Vertex	x1-coord	x2-coord	x3-coord	Active constraints
1	0	0	0	4,5,6
2	200	0	0	1,5,6
3	0	300	0	2,4,6
4	100	300	0	2,3,6
5	200	200	0	1,3,6
6	0	0	200	4,5,7
7	100	300	100	2,3,7
8	200	0	200	1,5,7

Note that a vertex now has 3 active constraints, because it takes the intersection of at least 3 planes to make a corner in 3D, whereas it only took the intersection of 2 lines to make a corner in 2D.

Again the (green) line is in the direction (120,500,200) of increasing profit, the maximum of which occurs at vertex #7. There is a (red) line connecting vertex #7 to the green line, to which it is perpendicular.

In general m constraints on n variables can yield an m-sided polyhedron in n-dimensional space. Such a polyhedron can be seen to have as many as $\begin{pmatrix} m \\ n \end{pmatrix}$ vertices, since n constraints are active at a corner, and there are $\begin{pmatrix} m \\ n \end{pmatrix}$ ways to choose n constraints. Each of these very many vertices is a candidate solution. So when m and n are large, we must rely on a systematic algorithm rather than geometric intuition in order to find the solution.

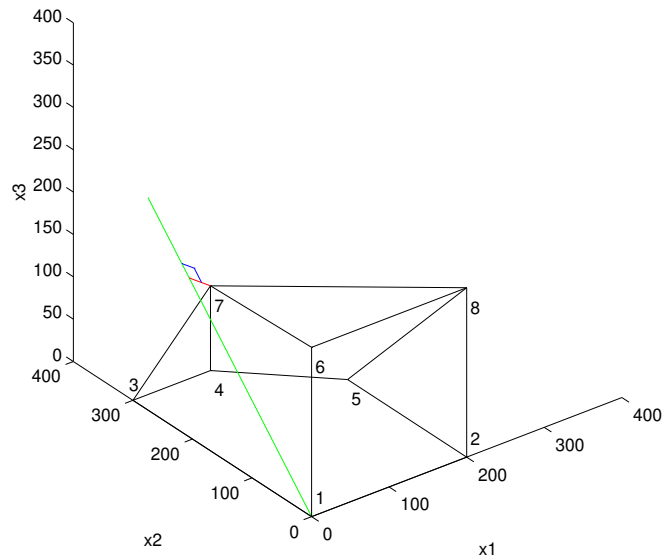

Figure 2: The feasible region (polyhedron).

4 Algorithms for Linear Programming

Linear programming was first solved by the *simplex method* devised by George Dantzig in 1947.

Consider the following linear program:

$$\max 120x_1 + 500x_2 + 200x_3$$

$$\begin{aligned}
&(1) && x_1 \leq 200 \\
&(2) && x_2 \leq 300 \\
&(3) && x_1 + x_2 \leq 400 \\
&(4) && x_1 \geq 0 \\
&(5) && x_2 \geq 0 \\
&(6) && x_3 \geq 0 \\
&(7) && x_2 + 3x_3 \leq 600
\end{aligned}$$

where the polyhedron of feasible solutions is shown in Figure 3 and its set of vertices is

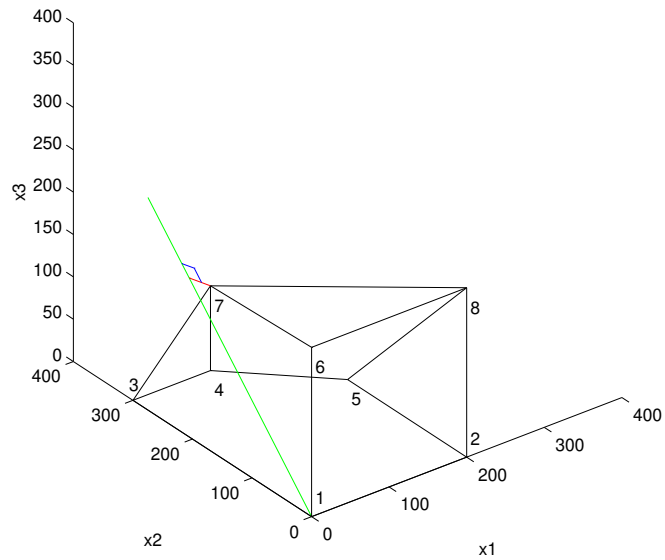

Figure 3: The feasible region (polyhedron).

Vertex	x1-coord	x2-coord	x3-coord	Active constraints
1	0	0	0	4,5,6
2	200	0	0	1,5,6
3	0	300	0	2,4,6
4	100	300	0	2,3,6
5	200	200	0	1,3,6
6	0	0	200	4,5,7
7	100	300	100	2,3,7
8	200	0	200	1,5,7

The simplex method starts from a vertex (in this case the vertex $(0, 0, 0)$) and repeatedly looks for a vertex that is adjacent, and has better objective value. That is, it is a kind of *hill-climbing* in the vertices of the polytope. When a vertex is found that has no better neighbor, simplex stops and declares this vertex to be the optimum. For example, in Figure 2, if we start at vertex #1 $(0, 0, 0)$, then the adjacent vertices are #2, #3, and #4 with profits 24000, 150000 and 40000, respectively. If the algorithm chooses to go to #3, it then examines vertices #6 and #7, and discovers the optimum #7. There are now implementations of simplex that solve routinely linear programs with *many* thousands of variables and constraints.

The simplex algorithm will also discover and report the other two possibilities: that the solution is infinite, or that the polyhedron is empty. In the worst case, the simplex algorithm takes exponential time in n, but this is very rare, so simplex is widely used in practice. There are other algorithms (by Khachian in 1979 and Karmarkar in 1984) that

are guaranteed to run in polynomial time, and are sometimes faster in practice.

5 Different Ways to Formulate a Linear Programming Problem

There are a number of equivalent ways to write down the constraints in a linear programming problem. Some formulations of the simplex method use one and some use another, so it is important to see how to transform among them.

One standard formulation of the simplex method is with a matrix A of constraint coefficients, a vector b of constraints, and a vector f defining the linear function $f * x = \sum_i f_i \cdot x_i$ (the dot product of f and x) to be maximized. The constraints are written as the single inequality $A \cdot x \leq b$, which means that every component $(A \cdot x)_i$ of the vector $A \cdot x$ is less than or equal to the corresponding component b_i: $(A \cdot x)_i \leq b_i$. Thus, A has as many rows as there are constraints, and as many columns as there are variables.

In the example above, $f = [120, 500, 200]$,

$$A = \begin{matrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 3 \end{matrix} \quad \text{and} \quad b = \begin{matrix} 200 \\ 300 \\ 400 \\ 0 \\ 0 \\ 0 \\ 600 \end{matrix}$$

Note that the constraints 4, 5 and 6 are $-x_1 \leq 0$, $-x_2 \leq 0$ and $-x_3 \leq 0$, or $x_1 \geq 0$, $x_2 \geq 0$ and $x_3 \geq 0$, respectively. In other words, constraints with $\geq$ can be changed into $\leq$ just by multiplying by -1.

Note that by changing f to $-f$, and maximizing $-f * x$, we are actually *minimizing* $f * x$. So linear programming handles both maximization and minimization equally easily.

Matlab 5.3, which is available on UNIX machines across campus, has a function linprog$(-f, A, b)$ for solving linear programs in this format. (This implementation *minimizes* the linear function instead of maximizing it, but since minimizing $-f * x$ is the same as maximizing $f * x$, we only have to negate the f input argument to get Matlab to maximize $f * x$). In earlier Matlab versions this program is called LP.

Now suppose that in addition to inequalities, we have equalities, such as $x_1 + x_3 = 10$. How do we express this in terms of inequalities? This is simple: write each equality as *two* inequalities: $x_1 + x_3 \leq 10$ and $x_1 + x_3 \geq 10$ (or $-x_1 - x_2 \leq -10$).

Similarly, one can turn any linear program into one just with equalities, and all inequalities of the form $x_i \geq 0$; some versions of simplex require this form. To turn an inequality like $x_1 + x_2 \leq 400$ into an equation, we introduce a new variable s (the *slack variable* for this inequality), and rewrite this inequality as $x_1 + x_2 + s = 400, s \geq 0$. Similarly, any inequality like $x_1 + x_3 \geq 20$ is rewritten as $x_1 + x_3 - s = 20, s \geq 0$; s is now called a *surplus* variable.

We handle an unrestricted variable x as follows: We introduce two nonnegative variables, x^+ and x^-, and replace x by $x^+ - x^-$. This way, x can take on any value.

6 A Production Scheduling Example

We have the demand estimates for our product for all months of 1997, $d_i : i = 1, \ldots, 12$, and they are very uneven, ranging from 440 to 920. We currently have 60 employees, each of which produce 20 units of the product each month at a salary of 2,000; we have no stock of the product. How can we handle such fluctuations in demand? Three ways:

- overtime—but this is expensive since it costs 80% more than regular production, and has limitations, as workers can only work 30% overtime.
- hire and fire workers—but hiring costs 320, and firing costs 400.
- store the surplus production—but this costs 8 per item per month

This rather involved problem can be formulated and solved as a linear program. As in all such reductions, a crucial first step is defining the variables:

- Let w_i be the number of workers we have in the ith month—we start with $w_0 = 60$.
- Let x_i be the production for month i.
- o_i is the number of items produced by overtime in month i.
- h_i and f_i are the numbers of workers hired/fired in the beginning of month i.
- s_i is the amount of product stored after the end of month i.

We now must write the constraints:

- $x_i = 20w_i + o_i$—the amount produced is the one produced by regular production, plus overtime.
- $w_i = w_{i-1} + h_i - f_i, w_i \geq 0$—the changing number of workers.
- $s_i = s_{i-1} + x_i - d_i \geq 0$—the amount stored in the end of this month is what we started with, plus the production, minus the demand.
- $o_i \leq 6w_i$—only 30% overtime.

Finally, what is the objective function? It is

$$\min 2000 \sum w_i + 400 \sum f_i + 320 \sum h_i + 8 \sum s_i + 180 \sum o_i.$$

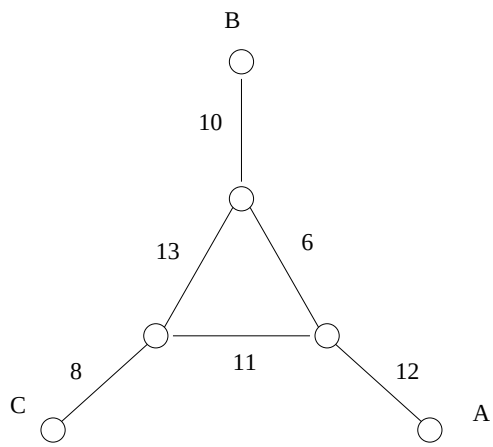

Figure 4: A communication network

7 A Communication Network Problem

We have a network whose lines have the bandwidth shown in Figure 4. We wish to establish three calls: One between A and B (call 1), one between B and C (call 2), and one between A and C (call 3). We must give each call at least 2 units of bandwidth, but possibly more. The link from A to B pays 3 per unit of bandwidth, from B to C pays 2, and from A to C pays 4. Notice that each call can be routed in two ways (the long and the short path), or by a combination (for example, two units of bandwidth via the short route, and three via the long route). How do we route these calls to maximize the network's income?

This is also a linear program. We have variables for each call and each path (long or short); for example x_1 is the short path for call 1, and x_2' the long path for call 2. We demand that (1) no edge bandwidth is exceeded, and (2) each call gets a bandwidth of 2.

$$
\begin{array}{rcl}
\max 3x_1 + 3x_1' + 2x_2 + 2x_2' + 4x_3 + 4x_3' & & \\
x_1 + x_1' + x_2 + x_2' & \leq & 10 \\
x_1 + x_1' + x_3 + x_3' & \leq & 12 \\
x_2 + x_2' + x_3 + x_3' & \leq & 8 \\
x_1 + x_2' + x_3' & \leq & 6 \\
x_1' + x_2 + x_3' & \leq & 13 \\
x_1' + x_2' + x_3 & \leq & 11 \\
x_1 + x_1' & \geq & 2 \\
x_2 + x_2' & \geq & 2 \\
x_3 + x_3' & \geq & 2 \\
x_1, x_1' \ldots, x_3' & \geq & 0
\end{array}
$$

The solution, obtained via simplex in a few milliseconds, is the following: $x_1 = 0, x_1' = 7, x_2 = x_2' = 1.5, x_3 = .5, x_3' = 4.5$.