Notes for Lecture 18

1 Algorithms for Linear Programming

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

We describe the algorithm with reference to last lecture's example. The linear program was

$$
\begin{array}{lll}
\max 120x_1 + 500x_2 + 200x_3 & & \\
& (1) & x_1 \leq 200 \\
& (2) & x_2 \leq 300 \\
& (3) & x_1 + x_2 \leq 400 \\
& (4) & x_1 \geq 0 \\
& (5) & x_2 \geq 0 \\
& (6) & x_3 \geq 0 \\
& (7) & x_2 + 3x_3 \leq 600
\end{array}
$$

and the polyhedron of feasible solutions is shown in Figure 1 and its set of vertices is

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.

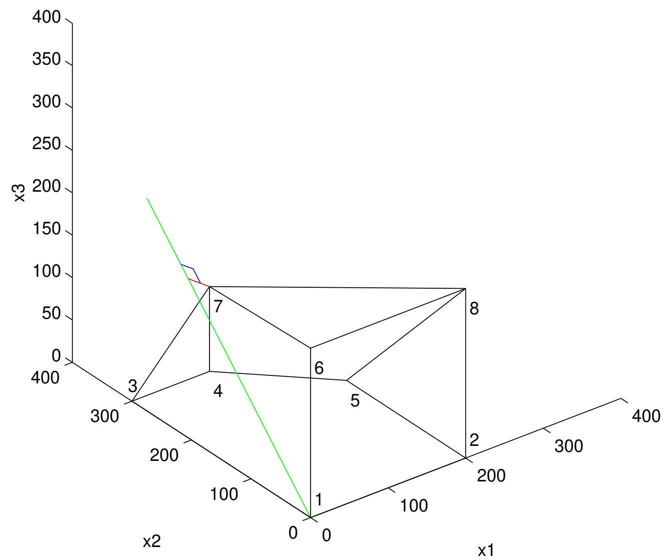

Figure 1: The feasible region (polyhedron).

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.

2 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.

A Java applet with a nice GUI for solving linear programming problems is available at URL riot.ieor.berkeley.edu/riot (click on "Linear Program Solver with Simplex").

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.

3 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

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

4 A Communication Network Problem

We have a network whose lines have the bandwidth shown in Figure 2. 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.

$$\begin{array}{rcl} \max 3x_1 + 3x_1' + 2x_2 + 2x_2' + 4x_3 + 4x_3' & & \\ x_1 + x_1' + x_2 + x_2' & \leq & 10 \\ x_1 + x_1' + x_3 + x_3' & \leq & 12 \\ x_2 + x_2' + x_3 + x_3' & \leq & 8 \\ x_1 + x_2' + x_3' & \leq & 6 \end{array}$$

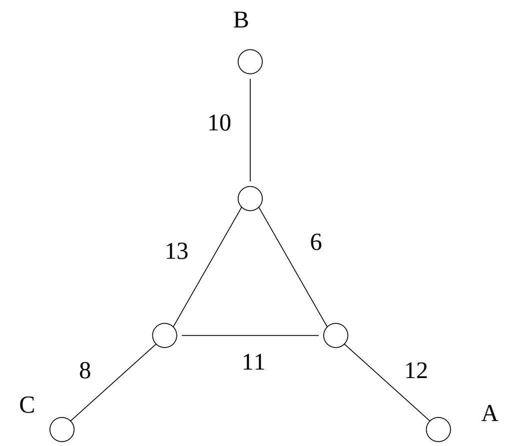

Figure 2: A communication network

$$
\begin{aligned}
x'_1 + x_2 + x'_3 &\leq 13 \\
x'_1 + x'_2 + x_3 &\leq 11 \\
x_1 + x'_1 &\geq 2 \\
x_2 + x'_2 &\geq 2 \\
x_3 + x'_3 &\geq 2 \\
x_1, x'_1 \ldots, x'_3 &\geq 0
\end{aligned}
$$

The solution, obtained via simplex in a few milliseconds, is the following: $x_1 = 0, x'_1 = 7, x_2 = x'_2 = 1.5, x_3 = .5, x'_3 = 4.5$.

Question: Suppose that we removed the constraints stating that each call should receive at least two units. Would the optimum change?