CS125	Lecture 9

An introductory example

Suppose that a company that produces three products wishes to decide the level of production of each so as to maximize profits. Let x_1 be the amount of Product 1 produced in a month, x_2 that of Product 2, and x_3 that of Product 3. Each unit of Product 1 yields a profit of 100, each unit of Product 2 a profit of 600, and each unit of Product 3 a profit of 1400. There are limitations on x_1, x_2, and x_3 (besides the obvious one, that $x_1, x_2, x_3 \geq 0$). First, x_1 cannot be more than 200, and x_2 cannot be more than 300, presumably because of supply limitations. Also, the sum of the three must be, because of labor constraints, at most 400. Finally, it turns out that Products 2 and 3 use the same piece of equipment, with Product 3 using three times as much, and hence we have another constraint $x_2 + 3x_3 \leq 600$. What are the best levels of production?

We represent the situation by a *linear program*, as follows:

$$\begin{aligned} \max\ & 100x_1 + 600x_2 + 1400x_3 \\ x_1 &\leq 200 \\ x_2 &\leq 300 \\ x_1 + x_2 + x_3 &\leq 400 \\ x_2 + 3x_3 &\leq 600 \\ x_1, x_2, x_3 &\geq 0 \end{aligned}$$

The set of all *feasible* solutions of this linear program (that is, all vectors in 3-d space that satisfy all constraints) is precisely the polyhedron shown in Figure 9.1.

We wish to maximize the linear function $100x_1 + 600x_2 + 1400x_3$ over all points of this polyhedron. Geometrically, the linear equation $100x_1 + 600x_2 + 1400x_3 = c$ can be represented by a plane parallel to the one determined by the equation $100x_1 + 600x_2 + 1400x_3 = 0$. This means that we want to find the plane of this type that touches the polyhedron and is as far towards the positive orthant as possible. Obviously, the optimum solution will be a vertex (or the optimum solution will not be unique, but a vertex will do). Of course, two other possibilities with linear programming are that (a) the optimum solution may be infinity, or (b) that there may be no feasible solution at all.

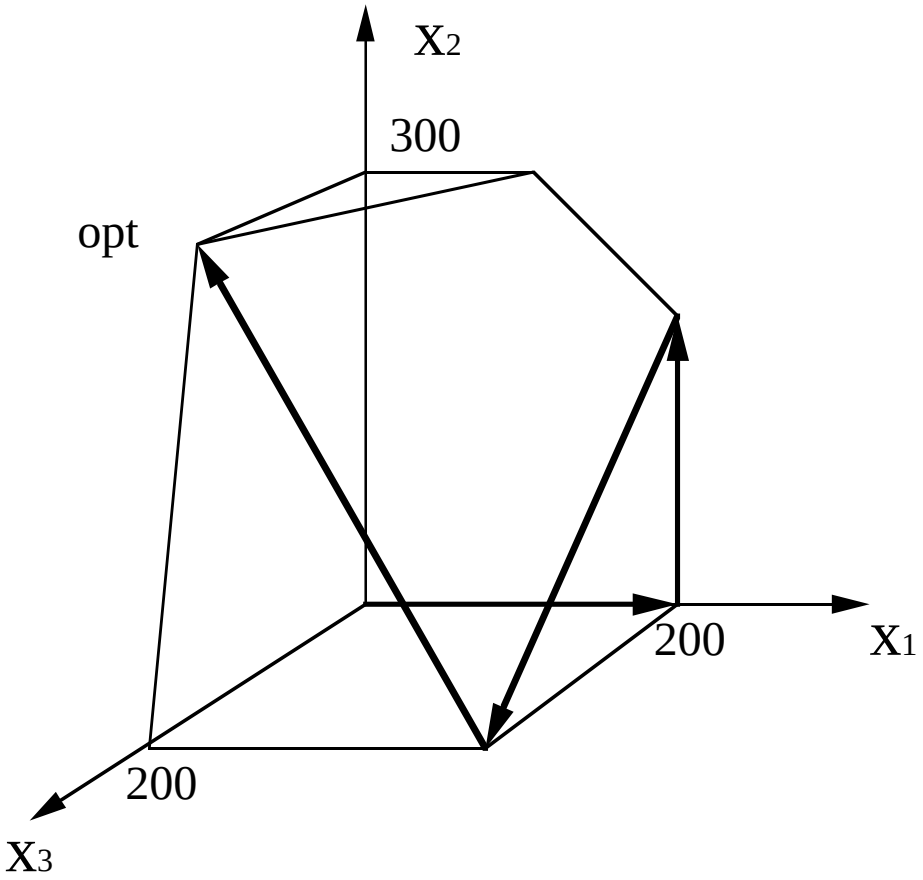

Figure 9.1: The feasible region

For this problem, an optimal solution exists, and moreover we shall show that it is easy to find.

Linear programs

Linear programs, in general, have the following form: there is an *objective function* that one seeks to optimize, along with *constraints* on the variables. The objective function and the constraints are all *linear* in the variables; that is, all equations have no powers of the variables, nor are the variables multiplied together. As we shall see, many problems can be represented by linear programs, and for many problems it is an extremely convenient representation. So once we explain how to solve linear programs, the question then becomes how to reduce other problems to linear programming (LP).

There are polynomial time algorithms for solving linear programs. In practice, however, such problems are solved by the *simplex method* devised by George Dantzig in 1947. 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 the figure one of the possible paths followed by simplex is shown. No known variant of the simplex algorithm has been proven to take polynomial time, and most of the variations used in practice have been shown to take exponential time on some examples. Fortunately, in practice, bad cases rarely arise, and the simplex algorithm runs extremely quickly. There are now implementations of simplex that solve routinely linear programs with *many thousands* of variables and constraints.

Of course, given a linear program, it is possible either that (a) the optimum solution may be infinity, or (b) that there may be no feasible solution at all. If this is the case, the simplex algorithm will discover it.

Reductions between versions of simplex

A general linear programming problem may involve constraints that are equalities or inequalities in either direction. Its variables may be nonnegative, or could be unrestricted in sign. And we may be either minimizing or maximizing a linear function. It turns out that we can easily translate any such version to any other. One such translation that is particularly useful is from the general form to the one required by simplex: *minimization, nonnegative variables, and equality constraints.*

To turn an inequality $\sum a_i x_i \leq b$ into an equality constraint, we introduce a new variable s (the *slack variable* for this inequality), and rewrite this inequality as $\sum a_i x_i + s = b, s \geq 0$. Similarly, any inequality $\sum a_i x_i \geq b$ is rewritten as $\sum a_i x_i - s = b, s \geq 0$; 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 $x^+ - x^-$ everywhere. The idea is that we let $x = x^+ - x^-$, where we may restrict both x^+ and x^- to be nonnegative. This way, x can take on any value, but there are only nonnegative variables.

Finally, to turn a maximization problem into a minimization one, we just multiply the objective function by -1.

Approximate Separation

An interesting application: suppose that we have two sets of points in the plane, the *black points* $(x_i, y_i) : i = 1, \ldots, m$ and the *white points* $(x_i, y_i) : i = m+1, \ldots, m+n$. We wish to separate them by a straight line $ax + by = c$, so that for all black points $ax + by \leq c$, and for all white points $ax + by \geq c$. In general, this would be impossible. Still, we may want to separate them by a line that minimizes the sum of the "displacement errors" (distance from the boundary) over all misclassified points. Here is the LP that achieves this:

$$\begin{array}{ll} \min e_1 & +e_2+\ldots+e_m+e_{m+1}+\ldots+e_{m+n} \\ e_1 \geq & ax_1+by_1-c \\ e_2 \geq & ax_2+by_2-c \\ \vdots & \\ e_m \geq & ax_m+by_m-c \\ e_{m+1} \geq & c-ax_{m+1}-by_{m+1} \\ \vdots & \\ e_{m+n} \geq & c-ax_{m+n}-by_{m+n} \\ & e_i \geq 0 \end{array}$$

Games

We can represent various situations of conflict in life in terms of *zero-sum matrix games*. For example, the game shown below is the *rock-paper-scissors* game. The Row player chooses a row strategy, the Column player chooses a column strategy, and then Column pays to Row the value at the intersection (if it is negative, Row ends up paying Column). That is why the games are called zero-sum; the "total payoff" for the two players is zero, or equivalently one player pays the other some amount.

$$\begin{array}{c} \\ r \\ p \\ s \end{array} \begin{array}{c} \begin{array}{ccc} r & p & s \end{array} \\ \begin{pmatrix} 0 & -1 & 1 \\ 1 & 0 & -1 \\ -1 & 1 & 0 \end{pmatrix} \end{array}$$

Zero-sum games do not necessarily have to be symmetric (that is, Row and Column have the same strategies, or, in terms of matrices, $A = -A^T$). For example, in the following fictitious *Clinton-Dole* game the strategies may be the issues on which a candidate for office may focus (the initials stand for "economy," "society," "morality," and "tax-cut") and the entries are the number of voters lost by Column.

$$\begin{array}{c} \\ e \\ s \end{array} \begin{array}{c} \begin{array}{cc} m & t \end{array} \\ \begin{pmatrix} 3 & -1 \\ -2 & 1 \end{pmatrix} \end{array}$$

We want to explore how the two players may play "optimally" in zero-sum games. It is not clear what this means. For example, in the first game there is no such thing as an optimal "pure" strategy (it very much depends on what your opponent does; similarly in the second game). But suppose that you play this game repeatedly. Then it makes sense to *randomize*. That is, consider a game given by an $m \times n$ matrix G_{ij}; define a *mixed strategy* for the row player to be a vector $(x_1,\ldots,x_m)$, such that $x_i \geq 0$, and $\sum_{i=1}^m x_i = 1$. Intuitively, x_i is the probability with which Row plays strategy i. Similarly, a mixed strategy for Column is a vector $(y_1,\ldots,y_n)$, such that $y_j \geq 0$, and $\sum_{j=1}^n y_j = 1$.

Suppose that, in the Clinton-Dole game, Row decides to play the mixed strategy $(.5,.5)$. What should Column do? The answer is easy: If the x_i's are given, there is a *pure strategy* (that is, a mixed strategy with all y_j's zero except for one) that is optimal. It is found by comparing the n numbers $\sum_{i=1}^{m} G_{ij}x_i$, for $j = 1,\ldots,n$ (in the Clinton-Dole game, Column would compare .5 with 0, and of course choose the smallest —remember, the entries denote what Column pays). That is, *if* Column knew Row's mixed strategy, s/he would end up paying the smallest among the n outcomes $\sum_{i=1}^{m} G_{ij}x_i$, for $j = 1,\ldots,n$. On the other hand, Row will seek the mixed strategy that *maximizes this minimum;* that is,

$$\max_x \min_j \sum_{i=1}^{m} G_{ij}x_i.$$

This maximum would be the best possible *guarantee* about an expected outcome that Row can have by choosing a mixed strategy. Let us call this guarantee z; what Row is trying to do is solve the following LP:

$$\begin{array}{lrrl} \max z & & & \\ z & -3x_1 & +2x_2 & \leq 0 \\ z & +x_1 & -x_2 & \leq 0 \\ & x_1 & +x_2 & = 1 \end{array}$$

Symmetrically, it is easy to see that Column would solve the following LP:

$$\begin{array}{lrrl} \min w & & & \\ w & -3y_1 & +y_2 & \geq 0 \\ w & +2y_1 & -y_2 & \geq 0 \\ & y_1 & +y_2 & = 1 \end{array}$$

For these two-player games with only two options per player, it is pretty simple to solve the linear programs. We do not need to use the simplex algorithm, or an LP solver; we can compute the optimal strategy for each player by hand.

Let us go back to our game and summarize: By solving an LP, Row can guarantee an expected income of at least V, and it turns out that Column can guarantee an expected loss of at most the same value. It follows that this is the uniquely defined optimal play (it was not *a priori* certain that such a play exists). V is called *the value of the game*. In this case, the optimum mixed strategy for Row is $(3/7,4/7)$, and for Column $(2/7,5/7)$, with a value of $1/7$ for the Row player.

The existence of mixed strategies that are optimal for both players and achieve the same value is a fundamental result in Game Theory called *the min-max theorem*. It can be written in equations as follows:

$$\max_x \min_y \sum x_i y_j G_{ij} = \min_y \max_x \sum x_i y_j G_{ij}.$$

It is surprising, because the left-hand side, in which Column optimizes last, and therefore has presumably an advantage, should be intuitively smaller than the right-hand side, in which Column decides first.

There is a fundamental reason that the two LPs have the same value; it has to do with what is called *duality*. The crucial observation now is that *these LP's are dual to each other*, and hence have the same optimum, call it V. That is, we have to introduce the notion of duality of LPs.

Duality

Basically, duality means that for each LP maximization problem there is a corresponding minimization problem with the property that any feasible solution of the min problem is greater than or equal any feasible solution of the max problem. Furthermore, and more importantly, when these LPs have finite solutions, *they have the same optimum*.

A generic form of a primal and dual linear program, when all the constraints are inequalities and the variable are constrained to be positive, is the following: The primal has the form:

$$\begin{aligned} \max c_1x_1 + \ldots + c_nx_n & & \\ a_{i1}x_1 + a_{i2}x_2 + \ldots a_{in}x_n & \leq & b_i \text{ for } i = 1,\ldots,m \\ x_j & \geq & 0 \text{ for } i = 1,\ldots,n \end{aligned}$$

The dual has the form:

$$\begin{aligned} \min b_1y_1 + \ldots + b_my_m & & \\ a_{1j}y_1 + a_{12}y_2 + \ldots a_{mj}y_m & \geq & c_j \text{ for } j = 1,\ldots,n \\ y_i & \geq & 0 \text{ for } i = 1,\ldots,m \end{aligned}$$

To see where the idea of duality comes from, let's start with the primal problem. Let us consider the constraint equations

$$a_{i1}x_1 + a_{i2}x_2 + \ldots a_{in}x_n \leq b_i \text{ for } i = 1,\ldots,m$$

and try to take a linear combination of them so that a few things hold. The linear combination should sum up so the coefficients are the c_i, or (since the x_i are non-negative) they can even be greater than the c_i. Then the linear combination witll give us a bound on the primal function we are trying to optimize. That is, we have

$$a_{i1}x_1 + a_{i2}x_2 + \ldots a_{in}x_n \leq b_i.$$

Let us choose multipliers $y_1, y_2, \ldots, y_m$. These multiplier will need to be positive (so the sign of our inequality doesn't change). Multiplying the ith multiplier by the ith equations gives

$$a_{i1}y_ix_1 + a_{i2}y_ix_2 + \ldots a_{in}y_ix_n \leq b_iy_i.$$

Now let's sum up the m equations above; we get

$$\sum_i \sum_j a_{ij}y_ix_j = \sum_j \sum_i a_{ij}y_ix_j \leq \sum_i b_iy_i.$$

Now suppose we choose the y_i carefully so that

$$\sum_i a_{ij}y_i \geq c_j.$$

Notice that these are the constraints of the dual problem. Then (since the x_j are all non-negative)

$$\sum_i b_iy_i \geq \sum_j c_jx_j.$$

That is, the solution to the primal problem (with the given constraints), the minimum possible value for $\sum_i b_iy_i$ necessarily gives an *upper bound* on the solution to the dual problem.

In fact, we can make a stronger statement. The *duality theorem* is the following:

Theorem 9.1 *if a linear program has a bounded optimum, then so does its dual, and the two optimum values are the same.*

Not only do we get an upper bound, but we get a matching bound (as we saw in the 2-player game example).

Here'a an example. Consider our original problem:

$$\begin{aligned} \max 100x_1 + 600x_2 + 1400x_3 & \\ x_1 &\leq 200 \\ x_2 &\leq 300 \\ x_1 + x_2 + x_3 &\leq 400 \\ x_2 + 3x_3 &\leq 600 \\ x_1, x_2, x_3 &\geq 0 \end{aligned}$$

If we take 200 times the third equation and 400 times the fourth equation we get:

$$200(x_1 + x_2 + x_3) + 400(x_2 + 3x_3) \leq 200 \cdot 400 + 400 \cdot 600,$$

or

$$200x_1 + 600x_2 + 1400x_3 \leq 320000.$$

Hence 320000 is an upper bound, and we saw that it is in fact achievable (by $x_2 = 300, x_1 = 100$); these were the "right" coefficients in terms of the dual problem.

It is mechanical, given an LP, to form its dual. Suppose we start with a maximization problem. Change all inequality constraints into $\leq$ constraints, negating both sides of an equation if necessary. Then

- transpose the coefficient matrix
- invert maximization to minimization
- interchange the roles of the right-hand side and the objective function
- introduce a nonnegative variable for each inequality, and an unrestricted one for each equality
- for each nonnegative variable introduce a $\geq$ constraint, and for each unrestricted variable introduce an equality constraint.

If we start with a minimization problem, we instead begin by turning all inequality constraints into $\geq$ constraints, we make the dual a maximization, and we change the last step so that each nonnegative variable corresponds to a $\leq$ constraint. Note that it is easy to show from this description that the dual of the dual is the original primal problem!

Network Flows

Suppose that we are given the network in top of Figure 9.2, where the numbers indicate capacities, that is, the amount of flow that can go through the edge. We wish to find the maximum amount of flow that can go through this network, from S to T.

This problem can also be reduced to linear programming. We have a nonnegative variable for each edge, representing the flow through this edge. These variables are denoted $f_{SA}, f_{SB}, \ldots$ We have two kinds of constraints: capacity constraints such as $f_{SA} \leq 5$ (a total of 9 such constraints, one for each edge), and flow conservation constraints (one for each node except S and T), such as $f_{AD} + f_{BD} = f_{DC} + f_{DT}$ (a total of 4 such constraints). We wish to maximize $f_{SA} + f_{SB}$, the amount of flow that leaves S, subject to these constraints. It is easy to see that this linear program is equivalent to the max-flow problem. The simplex method would correctly solve it. We discuss this further next time...

Convex Programming/Optimization

We have focused on linear programming/optimization, but more generally there are techniques for what is called *convex optimization*. In convex optimization, we seek to minimize some convex function $f(x)$ (here x should be thought of as a multi-dimensional variable over the reals) over a closed convex domain, given constraints of the form $g_i(x) \leq 0$ for convex functions g_i. Convex optimization generalizes linear programming, and can be used for more complex optimization problems or generalizations of linear programming optimization problems. For more on convex optimization, you might start with `http://en.wikipedia.org/wiki/Convex_optimization`.

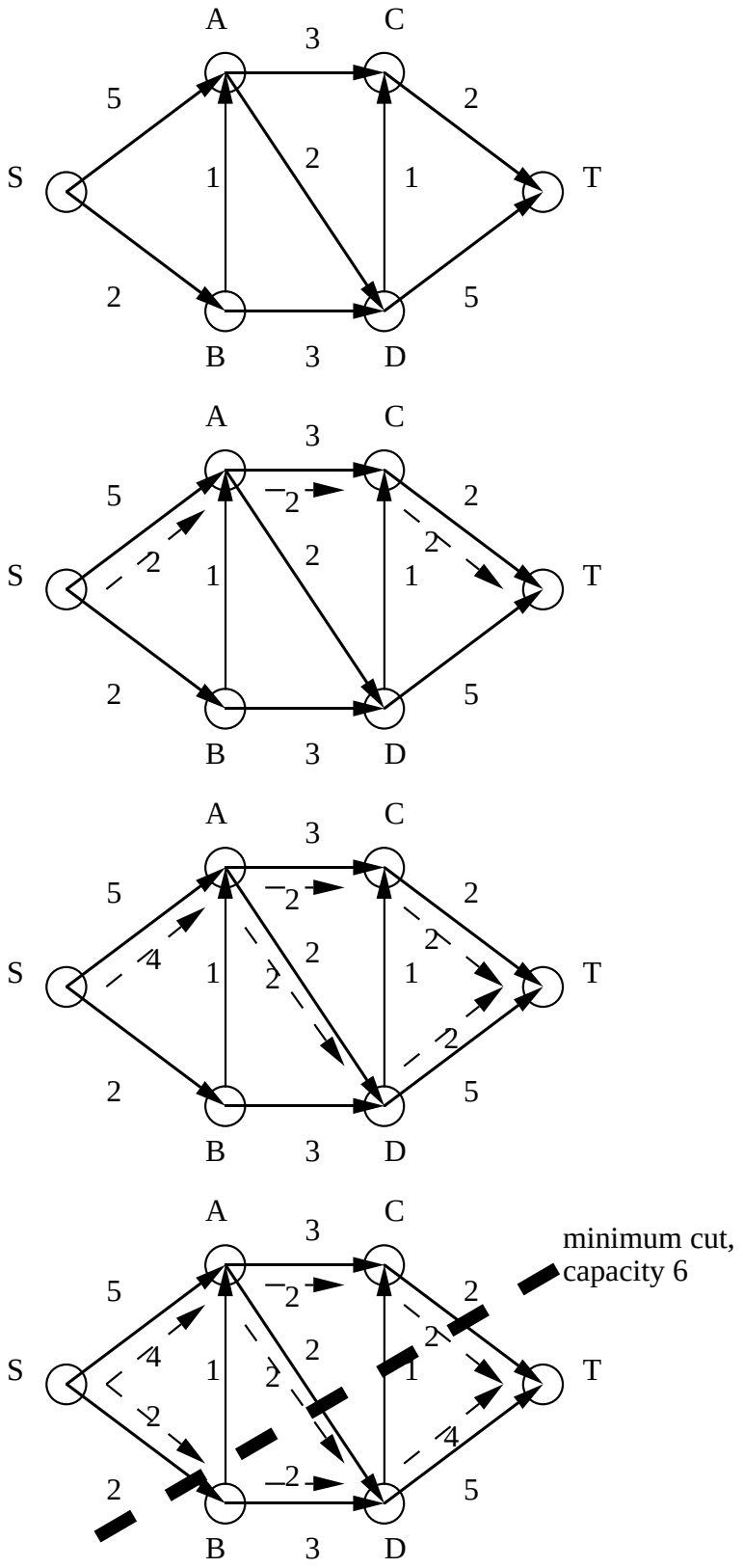

Figure 9.2: Max flow