ORIE 6300 Mathematical Programming I October 28, 2008

Lecture 16

Lecturer: David P. Williamson *Scribe: Zachary Rayfield*

1 Sensitivity Analysis (continued)

Last time, we looked at what happens to optimal solutions when we make small changes to b. We continue by examining changes in A and c.

1.1 Changes in c

Suppose we change $c_j \to c_j + \delta$. Then x stays feasible. There are now two cases:

1. $j \in N$. The objective function stays the same, since $x_j = 0$. c_B is unchanged, so $y = (A_B^T)^{-1} c_B$ stays the same, and therefore complementary slackness still holds. Is y feasible? Yes, if $A_j^T y \leq c_j + \delta$. If so, then x, y are still optimal.

2. $j \in B$. Then $c_B \to c_B + \delta e_j$ and then $y = (A_B^T)^{-1} c_B \to \tilde{y} = (A_B^T)^{-1} c_B + \delta e_j)$. This new y is feasible if $A_k^T \tilde{y} \leq c_k$ for all $k \in \mathbb{N}$, i.e. if $A_k^T y + \delta {A^T}_k (A_B^T)^{-1} e_j \leq c_k$. So there are bounds on δ such that y stays feasible. Note that the objective function changes by δx_j.

1.2 Changes in A

Now suppose we change a single entry of the constraint matrix A; suppose $a_{ij} \to a_{ij} + \delta$. Once again there are two cases:

1. $j \in N$. In this case, x stays feasible (since $x_j = 0$). Then $y = (A_B^T)^{-1} c_B$ stays the same, so that complementary slackness still holds. y is feasible if $A_j^T y + \delta y_i \leq c_j$.

2. $j \in B$. In this case, both $x_B = {A_B}^{-1} b$ and $y = (A_B^T)^{-1} c_B$ change. We need to check whether $A_B + \delta e_i e_j^T$ remains singular! If so, we can solve for both x and y and check whether they remain feasible; if so, they will be optimal because they both still obey complementary slackness.

2 The Cutting Stock Problem

This is a problem from the paper industry. Paper is produced in W inch long rolls called *raws* in which W is very large. These raws are cut into smaller lengths called *finals* for sale according to demand. Suppose there is demand for b_i rolls of length $s_i \leq W$, where $i = 1, \ldots, m$. The goal is to find a way to cut raws into finals such that we use the minimum possible number of raws. A clear upper-bound for this problem is $\sum_{i=1}^m b_i = N$ (i.e., using one complete roll of paper to produce each one of the rolls needed).

How can we use LP to solve this problem? We start out by formulating this as an *integer* program. Here is one possible formulation (which is not very good, as we will see). Let w_j denote the length of the jth final to be produced (i.e. it is s_i for some i). We set up decision variables y_i

and x_{ij} in which we use $y_i \in \{0,1\}$, $i = 1, 2, ..., N$ to denote whether or not the i-th roll is used in a solution and $x_{ij} \in \{0,1\}$, $j = 1, 2, ..., N$ to denote whether or not the j-th desired roll is obtained by cutting that length from the i-th complete raw. Thus we may formulate the integer programming problem as:

$$\begin{array}{ll} \min & \sum_{i=1}^{N} y_i \\ \text{s.t.} & (1): \sum_{j=1}^{N} x_{ij} w_j \leq W \text{ for } i = 1, \ldots, N \text{ (maximum available per roll)} \\ & (2): \sum_{i=1}^{N} x_{ij} = 1 \text{ for } \ j = 1, \ldots, N \text{ (all demands must be satisfied)} \\ & (3): 0 \leq x_{ij} \leq y_i \ \forall \ i, j \text{ (must cut from a complete roll which is used)} \\ & (4): x_{ij}, y_i \text{ integers.} \end{array}$$

It turns out this is a bad integer programming formulation, but to understand why, we need to know a little about how integer programs are solved using linear programs as a subroutine. First, we solve the LP relaxation of the IP: Relax $x_{ij} \in \{0,1\} \rightarrow 0 \leq x_{ij} \leq 1, y_i \in \{0,1\} \rightarrow 0 \leq y_i \leq 1$. The value of the LP relaxation is at most the value of the IP, since the optimal IP solution is feasible for the LP relaxation. If the optimal LP solution is integral (i.e. it has $x_{ij} \in \{0,1\}, y_i \in \{0,1\}$), then the optimal LP solution is the optimal IP solution. If the LP solution is not integer, then we can use a branch-and-bound algorithm. In branch-and-bound, we take some fraction variable (say y_i), and solve two subproblems, one in which $y_i = 0$ and the other in which $y_i = 1$. Clearly, the cheaper of the two subproblems should be the solution to our overall problem. We solve the subproblem as we started out solving the general problem; i.e., solving a linear programming relaxation, and hoping to obtain an integral solution, and "branching" on a fractional variable if needed.

For this process to work well, it desirable to have at least two things:

1. The value of the LP relaxation is close to the IP optimal solution, so that we don't need to enumerate many subproblems before we find an integer solution.

2. The two possible branches partition the space of solutions, so we aren't considering the same possibilities under both branches (this will slow things down otherwise).

Neither is true of the above IP formulation of the problem. First, a trivial solution to the LP relaxation is $x_{ij} = y_i = 1/N \ \forall i, j$, which has LP value of 1. So no matter what the true IP solution is, our LP solution can be quite far away from it. Second, there are lots of different ways of representing the same integer solution.

2.1 An alternate formulation

We switch our attention to one raw of length W. We use a column vector A_j to represent a feasible cutting *pattern* or *configuration* p_j. The i-th component of A_j (a_{ij}) corresponds to the number of pieces of length s_i cut in one roll of configuration p_j. For p_j to be a feasible cutting pattern, the elements of A_j must all be non-negative integers and the sum of them must be at most W ($\sum_{i=1}^{m} a_{ij} \leq W$).

Taking $W = 10$, $s_1 = 5$, $s_2 = 3$ and $s_3 = 2$ as an example, it is easy to verify that the set of all maximal feasible cutting patterns (expressed in rows) is
$\{(2,0,0), (1,1,1), (0,3,0), (1,0,2), (0,0,5), (0,1,3), (0,2,2), (1,0,2)\} = A$ union with all possible subsets of each of the elements of A. (These patterns are maximal in the sense that we can't cut any more finals from the raw than are given by the pattern; we can easily derive all the non-maximal patterns from these, since they are any vector $a \leq A_j$ for some j). Let n be the total number of distinct feasible cutting patterns (that is, the number of vectors A_j satisfying the constraints) and

let x_j denote the number of rolls cut according to the j-th pattern. We may now formulate the 1-dimensional cutting stock problem as the following IP:

$$\begin{array}{rl} \min & \sum_{j=1}^{n} x_j \\ \text{s.t.} & \sum_{j=1}^{n} a_{ij} x_j \geq b_i \text{ for } i = 1, \ldots, m \text{ (demand constraint)} \\ & x_j \geq 0 \text{ integer-valued} \end{array}$$

Solving the LP-relaxation will give us another lower bound for the IP. Recall that since there are only m constraints, there are at most m non-zero variables. Given the LP-solution, we may round up decision variables with fractional values to the next integer and conclude that **IP-optimal** $\leq$ **LP-optimal + m**.

However, this new formulation has the drawback that there are potentially a lot of variables! Next time we will see how we can solve this linear program.