Introduction to Revised Simplex

- Modern simplex does NOT use tableaus
  - Would require **n** x (**m+1**) storage - most of which would be 0's
  - The tableau updates **all** the columns with each pivot; do we need them all?
  - Researchers in the early 1950's realized that tableaus were inefficient
- To introduce you to how simplex really works, it is necessary to show simplex in a matrix format
- In this section (and in duality), I'll use Winston's notation, but not his general approach

Simplex In Matrix Form

- Using notation in Winston (6.2):
  - ***bv*** subscript - basic variables
  - ***nbv*** subscript - nonbasic variables
  - ***c*** = vector of objective function coefficients
  - ***A*** = matrix of constraint coefficients
  - ***B*** = submatrix of A; contains columns associated with basics
  - ***N*** = submatrix of A; contains columns associated with nonbasics
  - ***b*** = vector containing the RHS of the constraints
- So, the basic problem in standard form is:

$$\max z = cx$$

$$\text{subject to}$$

$$Ax = b, x \geq 0$$

The Problem at Any Particular Stage

- Assume we have a BFS, $\boldsymbol{x_{bv}}$. Then the problem can be written as:

$$
\begin{aligned}
&\max z, \text{subject to} \\
&z - c_{bv}x_{bv} - c_{nbv}x_{nbv} = 0 \\
&Ax = Bx_{bv} + Nx_{nbv} = b \\
&x \geq 0
\end{aligned}
$$

- First, how do we determine the value of $\boldsymbol{x_{bv}}$ and $\boldsymbol{z}$?

$$
\begin{aligned}
&Bx_{bv} + Nx_{nbv} = b \\
&Bx_{bv} + 0 = b \qquad \text{why?} \\
&x_{bv} = B^{-1}b; \quad z = c_{bv}B^{-1}b
\end{aligned}
$$

- Note all we needed to know was which variables were in the BFS, and the original problem data

Computing Reduced Costs

- Compute the reduced costs by writing the objective function in terms of the nonbasics:

$$Bx_{bv} + Nx_{nbv} = b$$

$$x_{bv} = B^{-1}b - B^{-1}Nx_{nbv}$$

substitute:

$$z - c_{bv}x_{bv} - c_{nbv}x_{nbv} = 0$$

$$z - c_{bv}\left(B^{-1}b - B^{-1}Nx_{nbv}\right) - c_{nbv}x_{nbv} = 0$$

$$z - c_{bv}B^{-1}b - \left(c_{nbv} - c_{bv}B^{-1}N\right)x_{nbv} = 0$$

$$\frac{dz}{dx_{nbv}} = -c_{nbv} + c_{bv}B^{-1}N$$

-(original profit/unit - cost/unit to produce) = -reduced cost

Computing the Column; Ratio Test

- Suppose $\boldsymbol{x_k}$ has the best reduced cost. How do we generate its current column ($\boldsymbol{y_k}$) for the ratio test?

$$Bx_{bv} + Nx_{nbv} = b$$

$$x_{bv} + B^{-1}Nx_{nbv} = B^{-1}b$$

$$\text{now, } N \text{ is just } \left[a_{(1)} \middle| \quad \ldots \middle| \quad a_k \middle| \quad \cdots\right], \text{so}$$

$$y_k = B^{-1}a_k$$

- The current right hand side is $\boldsymbol{B^{-1}b}$, so we have everything we need; the pivot row, $\boldsymbol{r}$, is

$$\min_r \frac{\left[B^{-1}b\right]_r}{y_{rk}} : y_{rk} \geq 0$$

- So, the basic variable in row $\boldsymbol{r}$ leaves, and $\boldsymbol{x_k}$ enters. Again, all we needed was $\boldsymbol{B^{-1}}$

Summary: the Revised Simplex Algorithm

1. Put problem in standard form
2. Find initial BFS
3. Compute reduced costs:

$$-c_{nbv} + c_{bv} B^{-1} N$$

4. If all reduced costs nonnegative, STOP; LP is optimal. Otherwise, choose $\boldsymbol{x_k}$, a variable with a negative reduced cost, to enter
5. Compute the column:

$$y_k = B^{-1} a_k$$

6. If $\boldsymbol{y_k} \le \mathbf{0}$, STOP: LP is unbounded. Otherwise, find r, the pivot row, via the ratio test:

$$\min_r \frac{\left[B^{-1} b\right]_r}{y_{rk}} : y_{rk} \ge 0$$

7. Update $\boldsymbol{B}$, $\boldsymbol{B^{-1}}$, and $\boldsymbol{B^{-1} b}$. Go to 3.

Relationship to Tableau

- You say, "this is new, foreign, and disturbing. It doesn't look like tableau simplex at all."
- But, take a look at an initial tableau for the problem: max ***cx***, st ***Ax <= b***, ***x >= 0***, with slack vector ***s***:

z	c	0	0
s	A	I	b

- I claim: here's what's in there after a few pivots:

z	$cB^{-1}A - c$	$c_{bv}B^{-1}I$	$c_{bv}B^{-1}b$
x_{bv}	$B^{-1}A$	B^{-1}	$B^{-1}b$

Further Insights

- If we shuffled the columns of the tableau into basics and nonbasics, it would look like this:

z	$c_{bv}B^{-1}N - c_{nbv}$	0	$c_{bv}B^{-1}b$
x_{bv}	$B^{-1}N$	I	$B^{-1}b$

- And this, in expanded form, is just revised simplex

Efficiency & Product Form of the Inverse

- So revised simplex is simple, right?
  - Had terrible computational performance in early codes
  - "One could have started an iteration, gone to lunch, and returned before [the iteration] finished" (William Orchard-Hays)
  - What's the problem?
- Consider the issue of updating the RHS
  - At any iteration, the values of the basics are given by $\boldsymbol{B}^{-1}\boldsymbol{b}$
  - But, suppose $\boldsymbol{B}$ is a 10,000 x 10,000 matrix
  - How much work is it to compute the inverse?
- On the other hand, what does it take to update it in the tableau? We're only substituting one column; why is this so tough?

An Example of RHS Updating

- Suppose the pivot column and current RHS are as below, and the pivot is in the 3rd row:

$$\begin{bmatrix} 2 \\ -1 \\ \textcircled{2} \end{bmatrix} \cdots \begin{bmatrix} 9 \\ 2 \\ 4 \end{bmatrix}$$

- The row operations are to add 1/2 of row 3 to row 2, subtract row 3 from row 1, and divide row 3 by 2 :

$$\begin{bmatrix} 2 \\ -1 \\ 2 \end{bmatrix} \cdots \begin{bmatrix} 9 \\ 2 \\ 4 \end{bmatrix} \Rightarrow \begin{bmatrix} 2 \\ 0 \\ 2 \end{bmatrix} \cdots \begin{bmatrix} 9 \\ 4 \\ 4 \end{bmatrix} \Rightarrow \begin{bmatrix} 0 \\ 0 \\ 2 \end{bmatrix} \cdots \begin{bmatrix} 5 \\ 4 \\ 4 \end{bmatrix} \Rightarrow \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \cdots \begin{bmatrix} 5 \\ 4 \\ 2 \end{bmatrix}$$

Extension to Matrix Multiplication

- The following matrix operation does the same thing:

$$\begin{bmatrix} 1 & 0 & -\frac{2}{2} \\ 0 & 1 & -\frac{-1}{2} \\ 0 & 0 & \frac{1}{2} \end{bmatrix} * \begin{bmatrix} 9 \\ 2 \\ 4 \end{bmatrix} = \begin{bmatrix} 5 \\ 4 \\ 2 \end{bmatrix}$$

- In general, the row ops for a pivot can be expressed as:

$$\begin{bmatrix} y_1 \\ \vdots \\ y_{rk} \\ \vdots \\ y_m \end{bmatrix} \Rightarrow \begin{bmatrix} -\frac{y_1}{y_{rk}} \\ \vdots \\ \frac{1}{y_{rk}} \\ \vdots \\ -\frac{y_m}{y_{rk}} \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & \cdots & -\frac{y_1}{y_{rk}} & \cdots & 0 \\ \vdots & \cdots & \vdots & \cdots & \vdots \\ 0 & \cdots & \frac{1}{y_{rk}} & \cdots & 0 \\ \vdots & \cdots & \vdots & \cdots & \vdots \\ 0 & \cdots & -\frac{y_m}{y_{rk}} & \cdots & 1 \end{bmatrix}$$

multipliers for a pivot in <u>row</u> *r* go in <u>column</u> *r* of this matrix

Elementary Matrices; Product Form of the Inverse

- These matrices are called elementary matrices
  - We can store them economically for each pivot
  - Just need the nonzero multipliers and the pivot row
- If E_j is the elementary matrix for the jth pivot, then:

$$B_j^{-1} = E_{j-1} E_{j-2} \cdots B_0^{-1}$$

- So, we don't recompute $\boldsymbol{B^{-1}}$ at every step; we use the sequence of pivots to generate any column we need!
- The exploitation of this "product form" of the inverse (due to Alex Orden in 1953) was probably the most crucial part of making simplex computable

Revised Simplex with Product Form Inverse

1. Put problem in standard form
2. Find initial BFS and initial $\boldsymbol{B^{-1}}$ (will be $\boldsymbol{I}$ in many cases)
3. Compute reduced costs for iteration $\boldsymbol{j}$:

$$w = c_{bv} E_{j-1} E_{j-2} \cdots E_1 B_0^{-1}; \quad \text{reduced costs} = -c_{nbv} + wN$$

4. If all reduced costs nonnegative, STOP; LP is optimal. Otherwise, choose $\boldsymbol{x_k}$, a variable with a negative reduced cost, to enter
5. Compute the column:

$$y_k = E_{j-1} E_{j-2} \cdots E_1 B_0^{-1} a_k$$

6. If $\boldsymbol{y_k}$ <= **0**, STOP: LP is unbounded. Otherwise, find r, the pivot row, via the ratio test:

$$\min_r \frac{\bar{b}_r}{y_{rk}} : y_{rk} \geq 0$$

7. Store $\boldsymbol{E_j}$ and update RHS: $\bar{b} := E_j \bar{b}$ Go to 3.

Example

$$\begin{aligned}&\max \quad x_1 + 2x_2 - x_3\\&\text{subject to}\\&x_1 + x_2 + x_3 \le 4\\&-x_1 + 2x_2 - 2x_3 \le 6\\&2x_1 + x_2 \le 5\\&x_1, x_2, x_3 \ge 0\end{aligned}
\quad\Longrightarrow\quad
\begin{aligned}&\max \quad x_1 + 2x_2 - x_3\\&\text{subject to}\\&x_1 + x_2 + x_3 + s_1 = 4\\&-x_1 + 2x_2 - 2x_3 + s_2 = 6\\&2x_1 + x_2 + s_3 = 5\\&x_1, x_2, x_3 \ge 0\end{aligned}$$

Iteration 1:

$$x_{bv} = \{s_1, s_2, s_3\},\ x_{nbv} = \{x_1, x_2, x_3\}$$

$$z = 0,\ B = B^{-1} = I$$

$$\bar{b} = \begin{bmatrix}4\\5\\6\end{bmatrix}, c_{bv} = [0,0,0]$$

$\Longrightarrow$

$$w = c_{bv}B^{-1} = 0$$

$$-c_{nbv} + wN =$$

$$-[1,2,-1] + [0,0,0] = [-1,-2,1]$$

$$x_2 \text{ enters; } y_2 = a_2 = \begin{bmatrix}1\\2\\1\end{bmatrix}$$

Example (cont'd)

$$\min \begin{bmatrix} 4/1 \\ 6/2 \\ 5/1 \end{bmatrix} = 3 \Rightarrow s_2 \text{ exits}; E_1 = \begin{bmatrix} 1 & -1/2 & 0 \\ 0 & 1/2 & 0 \\ 0 & -1/2 & 1 \end{bmatrix}; \bar{b} := E_1\bar{b} = \begin{bmatrix} 1 \\ 3 \\ 2 \end{bmatrix}$$

Iteration 2:

$$x_{bv} = \{s_1, x_2, s_3\},\ x_{nbv} = \{x_1, s_2, x_3\}$$

$$c_{bv} = [0,2,0],\ c_{nbv} = [1,0,-1]$$

$$z = c_{bv}\bar{b} = 6$$

$$w = c_{bv}E_1 = [0,1,0]$$

$$-c_{nbv} + wN = -[1,0,-1] + [0,1,0] * \begin{bmatrix} 1 & 0 & 1 \\ -1 & 1 & -2 \\ 2 & 0 & 0 \end{bmatrix}$$

$$= [-2,3,-1] \Rightarrow x_1 \text{ enters}$$

Example (cont'd)

$$y_1 = E_1 a_1 = E_1 \begin{bmatrix} 1 \\ -1 \\ 2 \end{bmatrix} = \begin{bmatrix} 3/2 \\ -1/2 \\ 5/2 \end{bmatrix}; \bar{b} = \begin{bmatrix} 1 \\ 3 \\ 2 \end{bmatrix}; \text{min ratio is } 2/3; s_1 \text{ exits}$$

$$\mathrm{E}_2 = \begin{bmatrix} 2/3 & 0 & 0 \\ 1/3 & 1 & 0 \\ -5/3 & 0 & 1 \end{bmatrix}; \bar{b} := E_2 \bar{b} = \begin{bmatrix} 2/3 \\ 10/3 \\ 1/3 \end{bmatrix}$$

Iteration 3:

$$x_{bv} = \{x_1, x_2, s_3\},\ x_{nbv} = \{s_1, s_2, x_3\}, c_{bv} = [1,2,0],\ c_{nbv} = [0,0,-1]$$

$$z = c_{bv}\bar{b} = 22/3$$

$$w = c_{bv} E_2 E_1 = [4/3, 1/3, 0]$$

$$-c_{nbv} + wN = -[0,0,-1] + [4/3, 1/3, 0] * \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & -2 \\ 0 & 0 & 0 \end{bmatrix}$$

$$= [4/3, 1/3, 5/3] \Rightarrow \text{no favorable reduced cost; solution is optimal}$$

What Happens in Modern LP Codes

- You may notice that, after many iterations, we start maintaining *lots* of elementary matrices
- To solve this, simplex codes do periodic “reinversions” to build a new $\boldsymbol{B}^{-1}$
- Then, they start all over again
- Other details:
  - Most LP codes use a different factorization (LU) to store the pivots (won’t cover this here, but it will be in your next LP course)
  - Basis reinversion also helps control roundoff errors
  - LP codes also pay a lot of attention to the order of rows and columns in $\boldsymbol{B}^{-1}$; goal is to keep the stored matrices and vectors sparse

Final Tricks with Elementary Matrices

- Premultiplication:
  - Suppose ***E*** is an elementary matrix with a "nonidentity" column ***g*** in the rth position, and ***c*** is a row vector. Then:

$$\mathbf{cE} = \left[c_1, c_2, \cdots, c_{r-1}, \mathbf{cg}, c_{r1}, \cdots, c_m\right]$$

  - The result is equal to ***c***, except the rth element is ***cg*** (dot product)
- Postmultiplication:
  - Same as before, but now ***a*** is a column vector. Then:

$$\mathbf{Ea} = \begin{bmatrix} a_1 \\ \vdots \\ a_{r-1} \\ 0 \\ a_{r+1} \\ \vdots \\ a_m \end{bmatrix} + a_r \mathbf{g}$$

Duality

- Our standard problem (call it *P*) is:

$$
\begin{aligned}
P: \quad & \max\ z = cx \\
& \text{subject to} \\
& Ax \le b \\
& x \ge 0
\end{aligned}
$$

- Suppose we use the same A, b, c data and "transpose" the problem:

$$
\begin{aligned}
D: \quad & \min\ y = wb \\
& \text{subject to} \\
& \mathrm{w}A \ge c \\
& w \ge 0
\end{aligned}
$$

- The related problem *D* is called the "dual" of the "primal" problem *P*

Functional Relationship between Primal, Dual

- These problems share parameters, but use them differently
- One interpretation:
  - Primal: determine mix of products (***x***'s) to maximize profit (***c***) for given availability of resources (***b***)
  - Dual: determine prices (w's) to minimize the total paid for resources (***b***) with a particular profit potential (***c***)
- Economic theory would assert that these two problems should have some sort of equilibrium solution
- So what are the relationships?

Weak Duality

- Suppose $\boldsymbol{x_f}$ is a feasible solution for *P*, and $\boldsymbol{w_f}$ is a feasible solution for *D*. Then:

$$\begin{pmatrix} Ax_f \leq b \\ w_f A \geq c \end{pmatrix} \Rightarrow \begin{pmatrix} w_f Ax_f \leq w_f b \\ w_f Ax_f \geq cx_f \end{pmatrix} \Rightarrow cx_f \leq w_f Ax_f \leq w_f b$$

$$\Rightarrow cx_f \leq w_f b$$

- So, any feasible solution for *P* has an objective function value <= any feasible solution for *D*
- This property is called ***weak duality*** (and we just proved it)

Strong Duality

- If there's a weak case, is there a strong one? Suppose ***x***** is optimal for *P*. Then:

$$z^* = c_{bv} x^* = c_{bv} B^{-1} b$$
$$c_{bv} B^{-1} N - c_{nbv} \geq 0$$

- Assume that *D* can reach this value. If so:

$$z^* = c_{bv} x^* = c_{bv} B^{-1} b = y^*$$
$$\Rightarrow w^* = c_{bv} B^{-1}$$

- Is ***w***** feasible for *D*? Check:

$$w^* A \geq c \quad ?$$
$$w^* [B \quad N] \geq [c_{bv} \quad c_{nbv}] \quad ?$$
$$w^* = c_{bv} B^{-1}, \text{so}$$
$$c_{bv} B^{-1} [B \quad N] \geq [c_{bv} \quad c_{nbv}] \ ?$$
$$[c_{bv} \quad c_{bv} B^{-1} N] \geq [c_{bv} \quad c_{nbv}] \ ?$$

Answer is yes; last equation is primal optimality condition

Implications

- Weak duality says for any set of feasible solutions for *P* and *D*, the objective function of P <= the objective function of *D*
- Strong duality says that at optimality, the objective function values are ***equal*** (provided both *P* and *D* are feasible)
- Furthermore, there is a strong relationship between resource use and prices (more on that in a moment)
- Consequently, it is worth studying the solution of the dual to learn more about the solution of the primal

Writing the Dual of a General LP

- Here's the rule for writing the dual of an LP with variables and constraints in various forms:

$$
\begin{aligned}
P:\quad & \max\ z = cx \\
& \text{subject to} \\
& \quad A_1 x \le b_1 \qquad (w_1) \\
& \quad A_2 x \ge b_2 \qquad (w_2) \\
& \quad A_3 x = b_3 \qquad (w_3) \\
& \quad x \ge 0
\end{aligned}
\quad \Longleftrightarrow \quad
\begin{aligned}
D:\quad & \min\ y = w_1 b_1 + w_2 b_2 + w_3 b_3 \\
& \text{subject to} \\
& \quad w_1 A_1 + w_2 A_2 + w_3 A_3 \ge c \qquad (x) \\
& \quad w_1 \ge 0 \\
& \quad w_2 \le 0 \\
& \quad w_3 \text{ unrestricted}
\end{aligned}
$$

- Note the correspondences between types of constraints and bounds of variables
- Good habit: write names of dual variables next to constraints

Example Dual Formulations

- Have to think hard to write duals of “real” problems
- Remember - a constraint in the primal is a variable in the dual, and vice versa
- Example: product blending
  - Indices
    - ***p*** = products {1,2}
    - ***f*** = factories {1,2,3}
  - Data
    - $\boldsymbol{PROFIT_p}$ = $ profit per unit of p sold
    - $\boldsymbol{CAP_{pf}}$ = capacity required per unit of p built at f
    - $\boldsymbol{TOTCAP_f}$ = total capacity available at f
  - Variables
    - $\boldsymbol{num_p}$ = units of p to produce
    - ***totprofit*** = total profit

Dual of Product Mix Problem

***P*:** $$\max \quad totprofit = \sum_p PROFIT_p * num_p$$

subject to

$$\sum_p CAP_{pf} * num_p \leq TOT_f \text{ for all } f \quad (price_f)$$

$$num_p \geq 0 \text{ for all } p$$

***D*:** $$\min \quad totcost = \sum_f TOT_f * price_f$$

subject to

$$\sum_f CAP_{pf} * price_f \geq PROFIT_p \text{ for all } p \quad (num_p)$$

$$price_f \geq 0 \text{ for all } f$$

A Harder Example: Product Blending, p. 93, #14

- Indicies
  - **g** = gasolines {r,p}
  - **i** = inputs {ref, fcg, iso, pos, mtb, but}
- Data
  - $\mathbf{AVAIL_i}$ = daily availability of input i in liters
  - $\mathbf{RON_i}$ = octane of input i
  - $\mathbf{RVP_i}$ = RVP rating of input i
  - $\mathbf{A70_i}$ = ASTM volatility of i at 70C
  - $\mathbf{A130_i}$ = ASTM volatility of i at 130C
  - $\mathbf{RONRQ_g}$ = required octane of gas g
  - $\mathbf{RVPRQ_g}$ = required RVP rating of gas g
  - $\mathbf{A70RQ_g}$ = ASTM volatility of g at 70C required
  - $\mathbf{A130RQ_g}$ = ASTM volatility of g at 130C required
  - $\mathbf{DEMAND_g}$ = daily minimum demand for gas g
  - $\mathbf{PRICE_g}$ = selling price/liter of gas g
  - **FCGLIM** = limit on proportion of FCG in each gas g

Blending Dual (cont'd)

- Variables
  - ***inp_{gi}*** = liters of input i used to make gas g (all >=0)
  - ***totgross*** = total gross from gas sales

***P*:** $\max \quad totgross = \sum_{g,i} PRICE_g * inp_{gi}$

subject to

$$\sum_g inp_{gi} \le AVAIL_i \quad \text{for all } i$$

$$\sum_i inp_{gi} \ge DEMAND_g \quad \text{for all } g$$

$$inp_{g,\text{"}fcg\text{"}} \le FGCLIM * \sum_i inp_{gi} \quad \text{for all } g$$

$$\sum_i RON_i * inp_{gi} \ge RONRQ_g * \sum_i inp_{gi} \quad \text{for all } g$$

$$\sum_i RVP_i * inp_{gi} = RVPRQ_g * \sum_i inp_{gi} \quad \text{for all } g$$

$$\sum_i A70_i * inp_{gi} \ge A70RQ_g * \sum_i inp_{gi} \quad \text{for all } g$$

$$\sum_i A130_i * inp_{gi} \ge A130RQ_g * \sum_i inp_{gi} \quad \text{for all } g$$

NOW WHAT?

Disentangling the Dual

- 1st step: rewrite the constraints in *P* in standard form for a min problem

$$-\sum_g inp_{gi} \geq -AVAIL_i \text{ for all } i$$

$$\sum_i inp_{gi} \geq DEMAND_g \text{ for all } g$$

$$FGCLIM * \sum_i inp_{gi} \text{ -} inp_{g,\text{"}fcg\text{"}} \geq 0 \text{ for all } g$$

$$\sum_i \left(RON_i - RONRQ_g\right) * inp_{gi} \geq 0 \text{ for all } g$$

$$\sum_i \left(RVP_i - RVPRQ_g\right) * inp_{gi} = 0 \text{ for all } g$$

$$\sum_i \left(A70_i - A70RQ_g\right) * inp_{gi} \geq 0 \text{ for all } g$$

$$\sum_i \left(A130_i - A130RQ_g\right) * inp_{gi} \geq 0 \text{ for all } g$$

$$inp_{gi} \geq 0 \text{ for all } g, i$$

Disentangling the Dual (cont'd)

- Second step: assign dual variable names for each constraint, and determine their bounds

$$-\sum_g inp_{gi} \geq -AVAIL_i \text{ for all } i \; \left(w1_i \geq 0\right)$$

$$\sum_i inp_{gi} \geq DEMAND_g \text{ for all } g \; \left(w2_g \geq 0\right)$$

$$FGCLIM * \sum_i inp_{gi} \text{ -} inp_{g,\text{"}fcg\text{"}} \geq 0 \text{ for all } g \quad \left(w3_g \geq 0\right)$$

$$\sum_i \left(RON_i - RONRQ_g\right) * inp_{gi} \geq 0 \text{ for all } g \quad \left(w4_g \geq 0\right)$$

$$\sum_i \left(RVP_i - RVPRQ_g\right) * inp_{gi} = 0 \text{ for all } g \quad \left(w5_g \text{ unrestricted}\right)$$

$$\sum_i \left(A70_i - A70RQ_g\right) * inp_{gi} \geq 0 \text{ for all } g \quad \left(w6_g \geq 0\right)$$

$$\sum_i \left(A130_i - A130RQ_g\right) * inp_{gi} \geq 0 \text{ for all } g \quad \left(w7_g \geq 0\right)$$

$$inp_{gi} \geq 0 \text{ for all } g, i$$

Disentangling the Dual (cont'd)

- Third step: write the objective function of *D* using the dual variables and RHS of *P*

***D*:** $$\min\ y = \sum_i \left(-AVAIL_i * w1_i\right) + \sum_g \left(DEMAND_g * w2_g\right)$$

- Note that the RHS's of all the other constraints are 0; the associated dual variables DO NOT appear in the objective

Disentangling the Dual (cont'd)

- Fourth step: write a constraint for every variable in the objective function of ***P***
  - ***D*** will have ***g*** X ***i*** constraints, each with a RHS of $\boldsymbol{PRICE_g}$
  - What do these constraints look like?
- Hint: transpose the coefficients from each *column* in ***P*** to a constraint *row* in ***D***

$$
\begin{aligned}
-1 * w1_i &+ \\
1 * w2_g &+ \\
FGCLIM * w3_g &+ \\
\left(RON_i - RONRQ_g\right) * w4_g &+ \\
\left(RVP_i - RVPRQ_g\right) * w5_g &+ \\
\left(A70_i - A70RQ_g\right) * w6_g &+ \\
\left(A130_i - A130RQ_g\right) * w7_g &\le PRICE_g \quad \text{for all g, i <> "fcg"}
\end{aligned}
$$

Handling the Exception

- We need different dual constraints when i = “fcg” because the coefficients in the FGC constraint are different:

$$
\begin{aligned}
-1 * w1_i &+ \\
1 * w2_g &+ \\
(FGCLIM - 1) * w3_g &+ \\
(RON_i - RONRQ_g) * w4_g &+ \\
(RVP_i - RVPRQ_g) * w5_g &+ \\
(A70_i - A70RQ_g) * w6_g &+ \\
(A130_i - A130RQ_g) * w7_g &\le PRICE_g \quad \text{for all g, i = "fcg"}
\end{aligned}
$$

Complementary Slackness

- Go back to the "standard" primal and dual problems:

$$P: \quad \max\ z = cx \quad \text{subject to} \quad Ax \le b, \quad x \ge 0$$

$$D: \quad \min\ y = wb \quad \text{subject to} \quad wA \ge c, \quad w \ge 0$$

- Strong duality says the following:

$$z^* = cx^* = w^*b = y^*$$

- But, feasibility in ***P*** and ***D*** stipulates the following:

$$\begin{pmatrix} Ax^* \le b \\ w^* \ge 0 \end{pmatrix} \Rightarrow w^*Ax^* \le w^*b = y^*$$

$$\begin{pmatrix} w^*A \ge c \\ x^* \ge 0 \end{pmatrix} \Rightarrow w^*Ax^* \ge cx^* = z^*$$

Complementary Slackness Theorem

- The only way to get the strong duality result (equality) is:
  - For each of the n constraints in ***P***, either

$$\left(Ax^*\right)_i = b_i \quad \text{OR} \quad w_i^* = 0$$

  - For each of m constraints in ***D***, either

$$\left(w^* A\right)_j = c_j \quad \text{OR} \quad x_j^* = 0$$

- This result is called "complementary slackness," and has a simple economic interpretation
  - If you don't use all of the ***i***th resource, how much would you pay for more? **0**!
  - If you do use all of the ***i***th resource, how much would you pay for one more unit? w_i!

Shadow Prices

- This is why we care about the dual solution
  - The optimal dual values give sensitivity information about the primal constraints
  - Similarly, the optimal primal variables give sensitivity information about the dual constraints
- Some asides on shadow prices
  - Note from the text that the reduced cost for a slack (surplus) variable *does* give the value (negative value) of the dual variable; why does this make sense?
  - Winston has all sorts of discussion about tricky ways to find shadow prices; just compute them via $\boldsymbol{w} = \boldsymbol{c}_{bv}\boldsymbol{B}^{-1}$!

Dual Variable Values

$$y = c_{bv} B^{-1} b = z$$

Primal Variable Values

Warnings on Shadow Prices

- These are estimates of objective function changes *at a point*
- These estimates only apply to changes in a *single* right-hand-side; they are *not* additive across multiple changes
- They *are* good indications of the relative importance of resources, and are good indicators for further analysis
- Degeneracy makes shadow prices meaningless
  - If a slack variable is 0 and basic, the shadow price of the associated constraint can be 0 or large
  - The situation is ambiguous, and cannot be resolved unless you change some parameters and run the LP again

Objective Function and RHS Ranging

- Most LP solvers give "range" information on objective function and RHS coefficients
- **Objective function range**
  - For each $\boldsymbol{c^i}$, gives range $c_l \le c^i \le c_h$ for which the basic variables do not change (either the basics or their values)
  - Get new objective function value by multiplying the change in the cost coefficient by the value of the variable (which is 0 if nonbasic)
- **RHS range**
  - For each b^j, gives range $b_l \le b^j \le b_h$ for which the optimal solution will not change
  - Have to compute $\boldsymbol{x} = \boldsymbol{B}^{-1}\boldsymbol{b}$ to get new $\boldsymbol{x}$'s; however, can get new objective function quickly using shadow prices

Example: Stochastic Cop Problem

- Here's some of the MPL/CPLEX output:

```
VARIABLE cop[t] :

   t             Activity     Reduced Cost
  -----------------------------------------
   a12             4.0000           0.0000
   a6              0.0000           0.0000
   p12             7.0000           0.0000
   p6              8.0000           0.0000
  -----------------------------------------
VARIABLE cop[t] :

   t          Coefficient      Lower Range      Upper Range
  ----------------------------------------------------------
   a12            48.0000          45.0000          48.0000
   a6             48.0000          48.0000           1E+020
   p12            48.0000          45.0000          48.0000
   p6             48.0000          48.0000          51.0000
  ----------------------------------------------------------
```

- Changing *p6* to 51 increases objective by 8*(51-48) = 24
- How about changing these coefficients: *a12* = 47, *a6* = 49, *p12* = 47, *p6* = 51? (should give z = 1185 - 4 - 7 + 24 = 1198)

Moral: Only Valid for One Change at a Time

- Note changes in variable values; objective change NOT as predicted (z = 1190)

VARIABLE cop[t] :

t	Activity	Reduced Cost
a12	8.0000**(4)**	0.0000
a6	0.0000**(0)**	6.0000
p12	11.0000**(7)**	0.0000
p6	4.0000**(8)**	0.0000

VARIABLE cop[t] :

t	Coefficient	Lower Range	Upper Range
a12	47.0000	31.0000	49.0000
a6	49.0000	43.0000	1E+020
p12	47.0000	45.0000	49.0000
p6	51.0000	49.0000	53.0000

Problems with Sensitivity Analysis

- Most of this theory was developed when it was time-consuming and expensive to rerun an LP
- *This is no longer the case*
- LP sensitivity analysis only applies to changes in a *single* parameter
  - Again, ranges given in solution outputs are NOT additive
  - There is no way to assess interactions among parameter changes
- The sensitivities, particularly in large problems, are only valid over a *uselessly small* region
- If you want sensitivity analysis, run the #@%^&!! LP again!

Other Uses for Dual Values

- These are the foundation for so-called “decomposition methods”
  - Column generation
  - Dantzig-Wolfe and Benders’ decomposition
- Duality theory is also crucial in nonlinear optimization
  - Theory also applies to linear problems
  - We will talk more about this in the nonlinear optimization part of the course

Final Notes on Primal-Dual Relationships

- Suppose you have an optimal solution
- Change the cost parameters (***c***)
  - Can this affect primal feasibility? NO
  - Can this affect dual feasibility? YES
- Change the RHS (***b***)
  - Can this affect primal feasibility? YES
  - Can this affect dual feasibility? NO
- "Screw-up" relationships
  - Primal infeasible = dual unbounded or infeasible
  - Primal unbounded = dual infeasible
  - Moral: if one is screwed up, so is the other