Max – Min Problems

- These types of problems have become common recently in military and security circles
- General characteristics
  - Two opposed sides
  - One side is attempting to use a system
  - The other side is trying to thwart use of that system
  - System user has to commit to a course of action, then opposition reacts
  - Each side has limited resources
  - Each side knows the other's capabilities
- This is a Stackelberg (leader-follower) game
- Literature refers to these as attacker-defender models

Example: Roadblock Delay Model

- Consider the modified shortest path model below:

$$\max_{y}\min_{x} \quad z = \sum_{i,j \in ARCS(i,j)} \left(C_{ij} + D_{ij} y_{ij}\right) * x_{ij}$$

subject to

$$\left[\sum_{j \in ARCS(i,j)} x_{ij}\right] - \left[\sum_{j \in ARCS(j,i)} x_{ji}\right] = \begin{Bmatrix} 1, i = s \\ 0, i \neq s \text{ and } i \neq d \\ -1, i = d \end{Bmatrix}$$

$$0 \leq x_{ij} \leq 1 \text{ for all } ARCS(i,j)$$

$$\sum_{i,j \in ARCS(i,j)} y_{ij} \leq MAXCHECKPOINTS$$

$$y_{ij} \in \{0,1\} \text{ for all } ARCS(i,j)$$

- What situation is being modeled? What are the D_{ij}*'s*?

Problems (and the Solution)

- We don't know how to maximize one set of variables while minimizing another
- The objective function is nonlinear
- What to do?
  - Note that if we *fixed* the *y's*, we'd have a typical shortest path formulation
  - So, let's do that, and write the *dual* of the problem:

$$\max\ z = u_d - u_s$$

$$\text{subject to}$$

$$u_j - u_i \le C_i + D_{ij} y_{ij} \quad \text{for all}\ i, j \in ARCS(i, j)$$

$$u_i \ \text{unrestricted for all}\ i$$

Now We Have Something We Can Solve

- The dual problem is *linear* in the *y's*
- The dual problem is also a maximization
- So, we can solve a *single* optimization:

$$\max\ z = u_d - u_s$$

subject to

$$u_j - u_i \le C_i + D_{ij} y_{ij} \quad \text{for all } i, j \in ARCS(i, j)$$

$$u_i \text{ unrestricted for all } i$$

$$\sum_{i, j \in ARCS(i, j)} y_{ij} \le MAXCHECKPOINTS$$

$$y_{ij} \in \{0,1\} \text{ for all } i, j \in ARCS(i, j)$$

Application: Changi Naval Base, Singapore

Intelligence service gets word of a possible terrorist attack against a ship in port at Changi

Where should the checkpoints be set?

Ship needs 2 hours to get away to safe distance

From "How to Attack a Linear Program," Jerry Brown, Matt Carlyle, Terry Harrison, Javier Salmeron, and Kevin Wood, Naval Postgraduate School, copyright 2003

When Does This Trick Work?

- You need the following structure in your model:
  - Variables associated with both sides only appear in the objective function
  - These variables appear in a multiplicative form
  - One side can be modeled using continuous variables (so you can form its dual)
- Methodology is available to iterate between optimizations
  - Necessary when both sides require integral variables
  - Mechanics of passing solutions between optimizations (and avoiding cycling) can be complicated
- You can also exploit network structure …

Project Delay Model

- One side has a project, another wants to delay it
- Try augmenting the easy (dual) CPM formulation:

$$\max_{y} \min_{u} z = u_d - u_s$$

subject to

$$u_j - u_i \geq C_i + D_i y_i \quad \text{for all } i, j \in ARCS(i, j)$$

$$\sum_i DC_i y_i \leq DT$$

$$u_i \text{ unrestricted for all } i$$

$$y_i \in \{0,1\} \text{ for all } i$$

D_i: delay if opposition attacks task *i*

DC_i: resource required to attack task *i*

DT: total resources available

- No go: mixes *u's* and *y's* in the constraints

Try the Primal Form of CPM …

- I'm maximizing both sides, but the objective is nonlinear

$$\max_{y} \max_{x} z = \sum_{i,j \in ARCS(i,j)} \left(C_i + D_i y_i \right) * x_{ij}$$

subject to

$$\left[\sum_{j \in ARCS(i,j)} x_{ij} \right] - \left[\sum_{j \in ARCS(j,i)} x_{ji} \right] = \left\{ \begin{matrix} 1, i = s \\ 0, i \neq s \text{ and } i \neq d \\ -1, i = d \end{matrix} \right\}$$

$$0 \leq x_{ij} \leq 1 \text{ for all } ARCS(i,j)$$

$$\sum_{i} DC_i y_i \leq DT$$

$$y_i \in \{0,1\} \text{ for all } i$$

- Am I stuck?

The Trick

- Create additional arcs, which the attacker controls

$$\max_{y} \max_{x} z = \sum_{i,j \in ARCS(i,j)} (C_i + D_i) x'_{ij} + \sum_{i,j \in ARCS(i,j)} C_i x_{ij}$$

subject to

$$\left[\sum_{j \in ARCS(i,j)} x_{ij} + x'_{ij} \right] - \left[\sum_{j \in ARCS(j,i)} x_{ji} + x'_{ji} \right] = \begin{Bmatrix} 1, i = s \\ 0, i \neq s \text{ and } i \neq d \\ -1, i = d \end{Bmatrix}$$

What do these constraints do?

$$x'_{ij} \leq y_i \text{ for all } ARCS(i,j)$$

$$x_{ij} \leq 1 - y_i \text{ for all } ARCS(i,j)$$

$$0 \leq x_{ij} \leq 1, 0 \leq x'_{ij} \leq 1 \text{ for all } ARCS(i,j)$$

$$\sum_{i} DC_i y_i \leq DT$$

$$0 \leq x_{ij} \leq 1, 0 \leq x'_{ij} \leq 1 \text{ for all } ARCS(i,j)$$

$$y_i \in \{0,1\} \text{ for all } i$$

Can We Allow the Other Side to Crash Jobs?

- Change the (one-sided) project crashing model a bit to minimize total project time, with a constraint on added resources

$$
\begin{aligned}
&\min\ z = u_d - u_s \\
&\text{subject to} \\
&u_j - u_i \geq C_i - cr_i \text{ for all } i, j \in ARCS(i, j) \quad (x_{ij}) \\
&\sum_i CC_i \cdot cr_i \leq CT \quad (v) \\
&u_i \text{ unrestricted for all } i \\
&0 \leq cr_i \leq C_i - MIN_i \text{ for all } i \quad (q_i)
\end{aligned}
$$

cr**$_i$: amount to expedite task *i***

CC**$_i$: resource per unit time to expedite task *i***

***CT*: total resources available**

- We need to form the dual of this model

Dual of Project Crashing Model

$$\max_{x,q,v} z = \sum_{i,j \in ARCS(i,j)} C_i x_{ij} + \sum_i \left(MIN_i - C_i\right) q_i - CT \cdot v$$

subject to

$$\left[\sum_{j \in ARCS(i,j)} x_{ij}\right] - \left[\sum_{j \in ARCS(j,i)} x_{ji}\right] = \begin{Bmatrix} 1, i = s \\ 0, i \neq s \text{ and } i \neq d \\ -1, i = d \end{Bmatrix} \quad \left(u_i\right)$$

$$\sum_{j \in ARCS(i,j)} x_{ij} - CC_i \cdot v - q_i \leq 0 \text{ for all } i \quad \left(cr_i\right)$$

$$0 \leq x_{ij} \leq 1 \text{ for all } ARCS(i,j)$$

$$q_i \geq 0 \text{ for all } i, v \geq 0$$

Now, Use the Same Trick …

$$\max z = \sum_{i,j \in ARCS(i,j)} (C_i + D_i) x'_{ij} + \sum_{i,j \in ARCS(i,j)} C_i x_{ij} + \sum_i (MIN_i - C_i) q_i - CT \cdot v$$

subject to

$$\left[\sum_{j \in ARCS(i,j)} x_{ij} + x'_{ij} \right] - \left[\sum_{j \in ARCS(j,i)} x_{ji} + x'_{ji} \right] = \begin{Bmatrix} 1, i = s \\ 0, i \neq s \text{ and } i \neq d \\ -1, i = d \end{Bmatrix}$$

$$\left[\sum_{j \in ARCS(i,j)} x_{ij} + x'_{ij} \right] - CC_i \cdot v - q_i \leq 0 \text{ for all } i$$

$$x'_{ij} \leq y_i \text{ for all } ARCS(i,j)$$

$$x_{ij} \leq 1 - y_i \text{ for all } ARCS(i,j)$$

$$0 \leq x_{ij} \leq 1, 0 \leq x'_{ij} \leq 1 \text{ for all } ARCS(i,j)$$

$$\sum_i DC_i y_i \leq DT$$

$$0 \leq x_{ij} \leq 1, 0 \leq x'_{ij} \leq 1 \text{ for all } ARCS(i,j)$$

$$q_i \geq 0 \text{ for all } i, v \geq 0$$

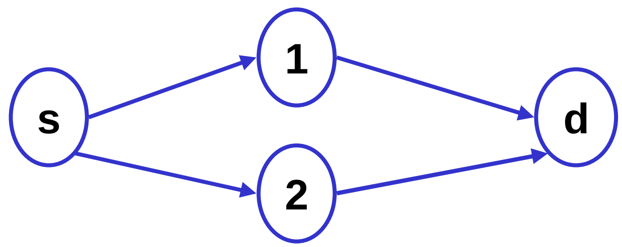
$$y_i \in \{0,1\} \text{ for all } i$$

Morals …

- We can solve a very important set of two-sided models using elementary LP theory
- A wide range of such models can be solved as a single optimization
- You have to be able to form the dual of one of the sides to do this
- You have to know which constraints in this dual correspond to the variables of that side (why?)

Review of Forming Duals

- Let's do a simple version of the project crashing problem



- Assume we can crash the jobs *s, 1,* and *2*
- Let's write down the problem in standard form …

$$
\begin{aligned}
&\min\ z = u_d - u_s \\
&\text{subject to} \\
&u_j - u_i + cr_i \geq C_i \text{ for all } i, j \in ARCS(i,j) \quad (x_{ij}) \\
&-\sum_i CC_i \cdot cr_i \geq -CT \quad (v) \\
&u_i \text{ unrestricted for all } i \\
&-cr_i \geq MIN_i - C_i \text{ for all } i \quad (q_i)
\end{aligned}
$$

Write the Problem in Tableau Form

- This is an exercise you can do in a spreadsheet
- Remember that each row will become a variable in the dual, and each column will become a constraint

	Z	U(s)	U(1)	U(2)	U(d)	cr(s)	cr(1)	cr(2)	
dual vars	1	-1	0	0	1	0	0	0	
x(s,1)	0	-1	1	0	0	1	0	0	>= C(s)
x(s,2)	0	-1	0	1	0	1	0	0	>= C(s)
x(1,d)	0	0	-1	0	1	0	1	0	>= C(1)
x(2,d)	0	0	0	-1	1	0	0	1	>= C(2)
q(s)	0	0	0	0	0	-1	0	0	>= MIN(s) - C(s)
q(1)	0	0	0	0	0	0	-1	0	>= MIN(1) - C(1)
q(2)	0	0	0	0	0	0	0	-1	>= MIN(2) - C(2)
v	0	0	0	0	0	-CC(s)	-CC(1)	-CC(2)	>= -CT

Write the Transpose to Form the Dual

- Again, you can cut and paste the transpose of the matrix in the spreadsheet

	Z	x(s,1)	x(s,2)	x(1,d)	x(2,d)	q(s)	q(1)	q(2)	v	
dual vars	1	C(s)	C(s)	C(1)	C(2)	MIN(s) - C(s)	MIN(1) - C(1)	MIN(2) - C(2)	-CT	
U(s)	0	-1	-1	0	0	0	0	0	0	= -1
U(1)	0	1	0	-1	0	0	0	0	0	= 0
U(2)	0	0	1	0	-1	0	0	0	0	= 0
U(d)	0	0	0	1	1	0	0	0	0	= 1
cr(s)	0	1	1	0	0	-1	0	0	-CC(s)	<= 0
cr(1)	0	0	0	1	0	0	-1	0	-CC(1)	<= 0
cr(2)	0	0	0	0	1	0	0	-1	-CC(2)	<= 0

- Does this match the dual formulation?
- If not, what doesn’t match?

Making the Formulation Match

- The network flow constraints are all *equalities*
- If you multiply them each by –1, you get the dual formulation back

	Z	x(s,1)	x(s,2)	x(1,d)	x(2,d)	q(s)	q(1)	q(2)	v	
dual vars	1	C(s)	C(s)	C(1)	C(2)	MIN(s) - C(s)	MIN(1) - C(1)	MIN(2) - C(2)	-CT	
U(s)	0	1	1	0	0	0	0	0	0	= 1
U(1)	0	-1	0	1	0	0	0	0	0	= 0
U(2)	0	0	-1	0	1	0	0	0	0	= 0
U(d)	0	0	0	-1	-1	0	0	0	0	= -1
cr(s)	0	1	1	0	0	-1	0	0	-CC(s)	<= 0
cr(1)	0	0	0	1	0	0	-1	0	-CC(1)	<= 0
cr(2)	0	0	0	0	1	0	0	-1	-CC(2)	<= 0

- This preserves the network convention that flow out is positive

Another Useful Model - Network Interdiction

- Two players:
  - The network *user* wants to maximize flow through a capacitated network
  - The network *interdictor* wants to reduce flow by interdicting arcs in the same network
- The interdictor can interdict a limited number of arcs
- Rules of the game
  - Both players know the network and the arc capacities
  - The interdictor chooses which arcs to interdict
  - All interdictions are completely successful
  - The user observes the interdictions and then maximizes flow on the remaining network

The Model (sparing the derivation)

$$\min\ z = \sum_{i,j \in ARCS(i,j)} U_{ij} b_{ij}$$

subject to

$$w_i - w_j + b_{ij} + y_{ij} \geq 0 \ \text{ for all } ARCS(i,j) - (d,s)$$

$$w_d - w_s + b_{ds} \geq 1$$

$$Ry \leq r$$

$$y_{ij}, b_{ij} \in \{0,1\} \ \text{ for all } i,j \in ARCS(i,j)$$

$$w_i \in \{0,1\} \ \text{ for all } i$$

***U_{ij}*: upper bound for flow on arc**

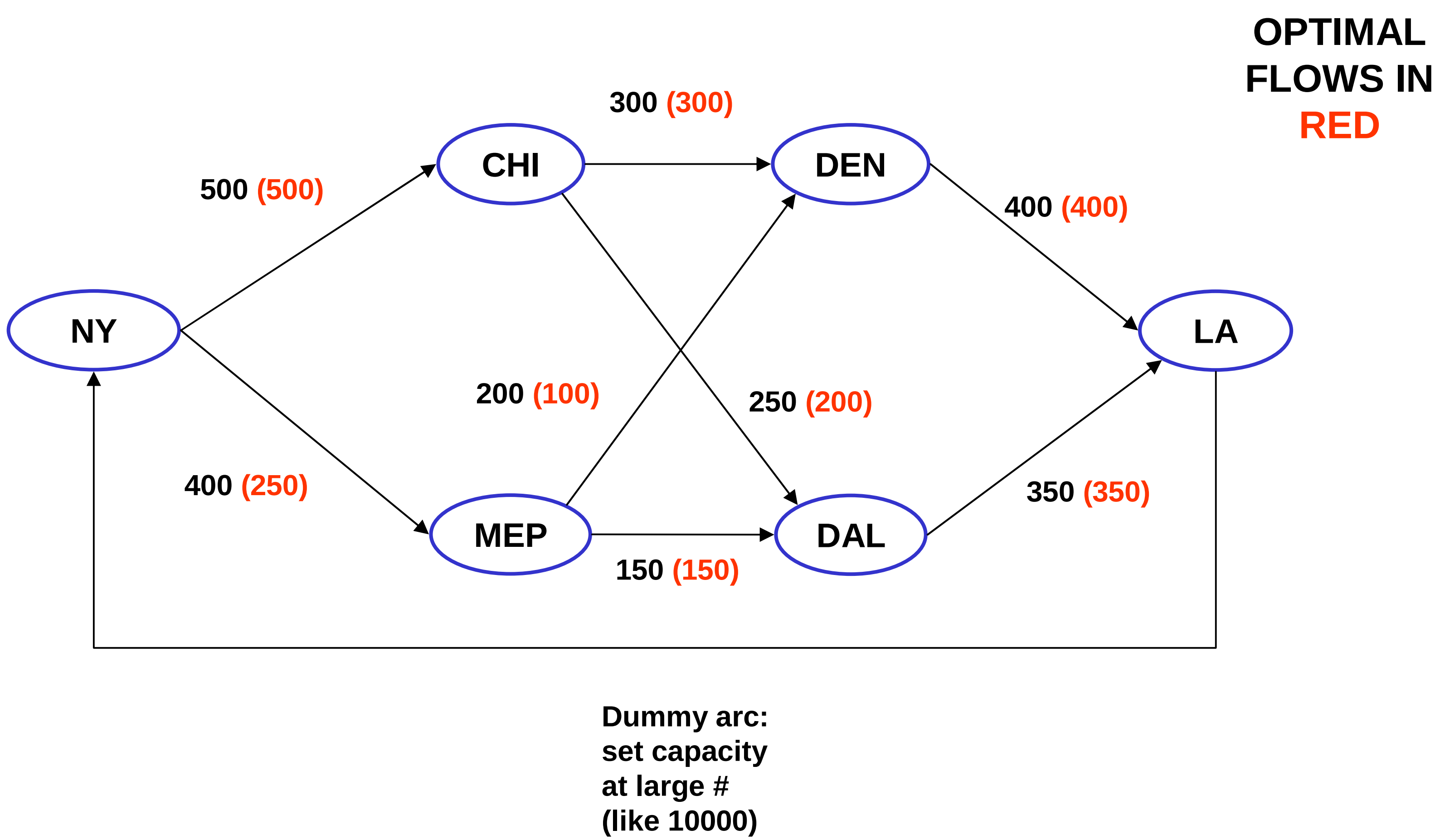
***d (s):* source (destination) node**

***R*: interdiction consumption parameters**

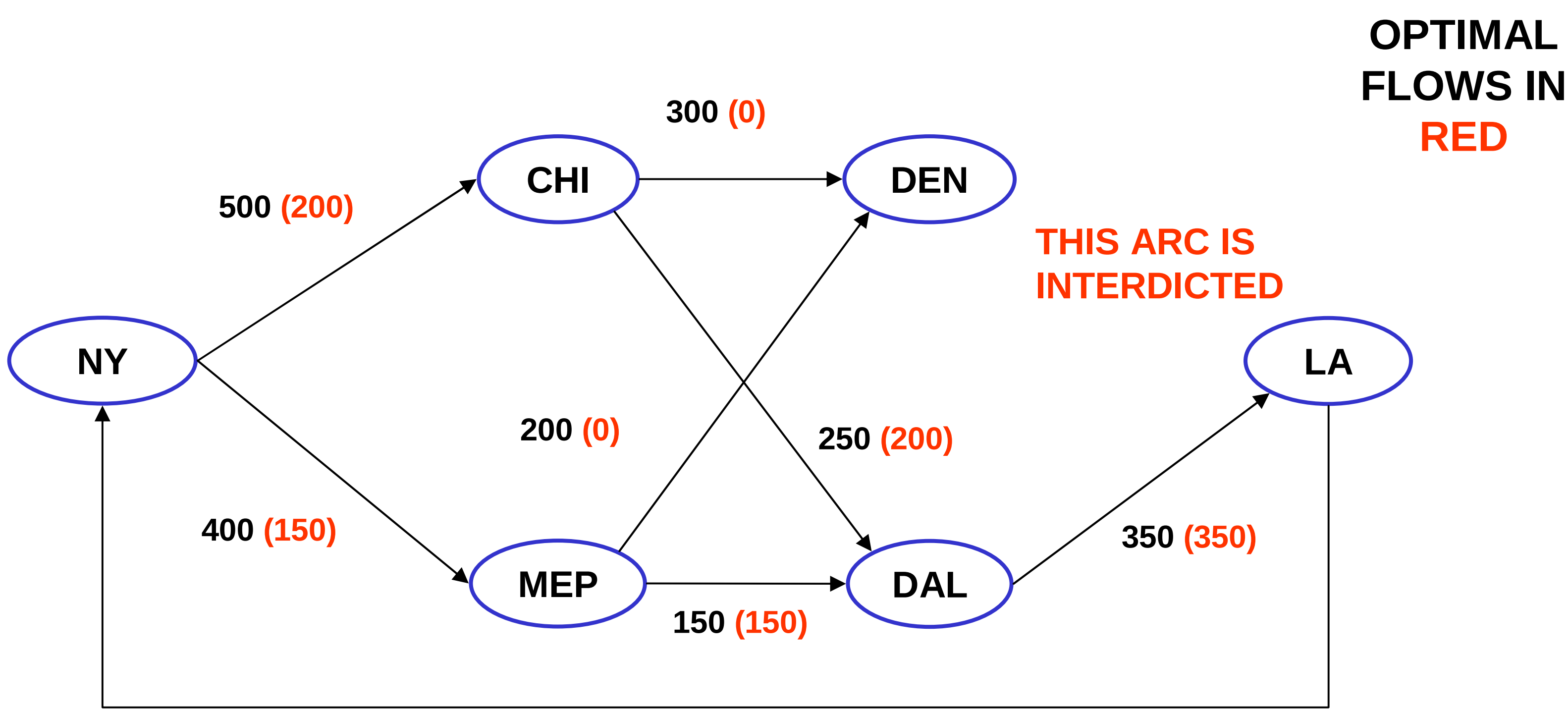
***r*: interdiction resources available**

***w, b*: dual variables for network user**

Example: Winston p. 472, #2



Solution with New Model, 1 Arc Interdicted



$$Ry \le r \;\Rightarrow\; \sum_{i,j \in ARCS(i,j)} y_{ij} \le 1$$

Other Notes About This Model

- Interpreting the solution
  - The *y's* that are 1 give the arcs that are interdicted
  - The objective function gives the resulting max flow
  - The *b's* that are 1 give the min-cut arcs in the remaining (uninterdicted) network
- Some interesting extensions
  - Suppose interdicting each arc has a an interdiction cost; then, you could let the interdiction be subject to a budget constraint
  - To make an arc "uninterdictable" just bound the corresponding *y* variable to be equal to 0

A General Attacker – Defender Model

- Suppose *y* represents the defender, and *x* the attacker
- The attacker can take away certain of the defender's resources, but he is limited by his own resources
- The resulting model (in matrix-vector notation) is:

$$\max_x \min_y z = cy$$

subject to

$$Ay \geq b$$

$$Fy \leq U(1 - x)$$

$$Cx \leq d$$

$$x \geq 0,\ y \geq 0$$

***c*: defender's cost vector**

***A*: consumption parameters for unattackable resources**

***b*: available unattackable resources**

***F:* consumption parameters for attackable resources**

***U:* available attackable resources = diag(*u*)**

***C*: attacker's consumption parameters**

***d*: attacker's available resources**

Switching Problem to "Cost Attack"

- As before, we can't handle *x's* and y's in the constraints
- Instead, we penalize *y's* use of resources that *x* attacks by attaching penalties in the objective

P: matrix containing penalties for using attacked resources

$$\max_x \min_y z = cy + x^T PFy$$

subject to

$$Ay \geq b \quad (v)$$

$$Fy \leq u \quad (w)$$

$$Cx \leq d$$

$$x \geq 0,\ y \geq 0$$

Proportion of y's attackable resources taken away by attacker

Converting to a Single Optimization

- As before, take the dual of the inner problem to form a single maximization:

$$\max_{x,v,w} = b^T v + uw$$

subject to

$$A^T v + F^T w \geq c + F^T P x$$

$$Cx \leq d$$

$$x \geq 0, v \geq 0, w \leq 0$$

Parting Notes

- If you build a model like this:
  - You have to choose which side will be represented by the dual
  - You must be careful about choosing penalties; should be as small as possible, otherwise results may be unreasonable
- This is growth area in optimization modeling
- Some useful articles:
  - Brown, Carlyle, Salmeron, and Wood, “Defending Critical Infrastructure”
  - Brown, G., Carlyle, M., Diehl, D., Kline, J. and Wood, K., 2005, “ A Two-Sided Optimization for Theater Ballistic Missile Defense,” Operations Research,53, pp. 263-275
  - Available at http://www.nps.navy.mil/orfacpag/resumePages/papers/browngpa.htm

Integer Programming

- Time to drop the divisibility assumption of LP
- Most obvious reason
  - Many resources or decisions restricted to integral values
  - Rounding an LP answer often infeasible or suboptimal
- Less obvious (but maybe more important) reason
  - Integer variables can implement *logical* conditions (if-then, one of many, etc.)
  - Allows the model to make complex decisions
- Another reason
  - Integer variables can be used to approximate nonlinear functions
  - Often employed for things like quantity discounts

These Capabilities Come at a Price

- Integer programming much more difficult
  - We're searching a "lattice" of points, not a continuous space
  - Many problems contain a combinatorially explosive number of possible solutions
- Example: "NOSWOT" problem from MIPLIB
  - 128 total variables - 75 binary {0,1}, 25 integer
  - CPLEX 6.0: did not solve after running for several *days*
  - CPLEX 6.5: solved in 6.2 hours, but required solving *26,521,191* LP's in a branch-and-cut tree
- And what became of NOSWOT?
  - CPLEX guys declared war, examined core problem
  - Added 8 additional constraints; problem now solves in 16 *seconds*

Morals of Integer Programming

- It is very, very difficult to beat an experienced human scheduler
- If you have an existing heuristic way to get a solution, you should start with *that*
- Problems that look innocuous can be very tough or impossible
- Add any constraints or exploit any problem structure you can
- *Read both Woolsey articles!*

Restricting Variables to Integral Values

- For variables restricted to integral values, just declare them as "integer"
- We'll deal with how this works later
- If your problem has no logical conditions, rounding often works
  - Such problems are said to have a feasible "interior"
  - Early IP literature filled with rounding schemes
  - Some still used on enormous problems

Using Binary Variables for Logical Conditions

- Suppose y1, y2, and y3 are binary {0,1} variables
- Let 1 represent true (or “on”), 0 be false (or “off”)
- The following table gives a logical expression and the appropriate constraint:

  - y3 = y1 and y2 ⇨
  $$y_3 \le y_1$$
  $$y_3 \le y_2$$
  $$y_3 \ge y_1 + y_2 - 1$$

  - y3 = y1 or y2 ⇨
  $$y_3 \ge y_1$$
  $$y_3 \ge y_2$$
  $$y_3 \le y_1 + y_2$$

  - if y1, then y2 ⇨
  $$y_1 \le y_2$$

Fixed Charge Formulations

- Typical situation: have to pay a fixed cost before producing or consuming something
  - Example: have to build a factory before making a car
  - If cars made = 0, you don't need the factory
  - If cars made > 0, you need the factory (but just one!)
- How to do this:
  - Assume ***x*** is the variable that depends on some fixed condition
  - Let ***y*** be a binary {0,1} variable, with 0 = off, 1 = on
  - Let ***U*** be the upper bound on ***x***
  - The following constraint forces ***x*** to 0 unless ***y*** is 1 (on)

$$Uy \geq x$$

Either-Or Conditions

- Used in situations where one of two constraints apply, depending on a decision variable
- Example: saving for your kid's future
  - ***y*** = 0 send kid to vocational school, at cost U_v
  - ***y*** = 1 send kid to Harvard, at cost U_h
  - $\boldsymbol{x_v}$ = amount saved for vocational school
  - $\boldsymbol{x_h}$ = amount saved for Harvard
  - The following enforces this condition:

$$U_v(1-y) \le x_v$$

$$U_h y \le x_h$$

Generalizing the Either-Or Conditions

- We may have situations where we want to choose among constraints
  - Example: ***y*** = 1 means overthrow despot of oil rich country; ***y*** = 0 means don't overthrow him
  - Constraints on military expenditures and oil availability may or may not apply, depending on the value of ***y***
- Choosing among two constraints:
  - y = 1 "turns off" constraint 1
  - y = 0 "turns off" constraint 2

$$\sum_i a_i^1 x_i \le b_1 \Rightarrow \sum_i a_i^1 x_i - b_1 \le M_1 y$$

or

$$\sum_i a_i^2 x_i \le b_2 \Rightarrow \sum_i a_i^2 x_i - b_2 \le M_2 (1 - y)$$

Choosing K out of N Constraints

- You can extend this to the "K out of N" case:
  - Define $\boldsymbol{y_1} \ldots \boldsymbol{y_N}$ as binary variables
  - The following ensures that only N-K constraints will hold:

$$\sum_i a_i^1 x_i \le b_1 \Rightarrow \sum_i a_i^1 x_i - b_1 \le M_1 y_1$$

$$\vdots$$

$$\sum_i a_i^N x_i \le b_N \Rightarrow \sum_i a_i^N x_i - b_N \le M_N y_N$$

$$\sum_i y_i \le K$$

$$y_i \in \{0,1\} \text{for all } i$$

Functions or Variables with N Possible Values

- Sometimes a function or variable can only take on a set of values
  - Example: raw materials available only in certain lot sizes
  - Only certain combinations of waist size and sleeve length available
- Let $\boldsymbol{D_1} \ldots \boldsymbol{D_N}$ be the values the function can take on; then:

$$\sum_j a_j x_j = \sum_i D_i y_i$$

$$\sum_i y_i = 1$$

$$y_i \in \{0,1\} \text{ for all } i$$

If-Then Conditions

- If the first condition applies, then so must the second
- Example from Winston:

$$f(x) > 0 \Rightarrow g(x) \geq 0$$

- This is logically equivalent to an either-or condition:

$$f(x) \leq 0 \text{ OR } g(x) \geq 0 \text{ OR BOTH}$$

- So, if $\boldsymbol{G_l}$ is a lower bound on ***g(x)***, $\boldsymbol{F_u}$ is an upper bound on ***f(x)***, and ***y*** is binary, the following constraints implement the condition:

$$g(x) \geq G_l * y$$
$$f(x) \leq F_u * (1 - y)$$
$$y \in \{0,1\}$$