Reading for This Lecture

• N&W Sections I.1.1-I.1.6

• Wolsey Chapter 1
Distinguishing “Formulations” and “Models”

• Recall our discussion in the last class about the modeling process.
  – The first step was to create a conceptual model in the real world.
  – The second step was to create a mathematical program whose solutions could be interpreted in terms of this real-world model.

• Before constructing the mathematical program, however, what we really do in practice is to first build a second “mathematical model”.
  – In the mathematical model, the variables can only take real (usually rational) values.
  – We initially describe what values of those variables we would like to allow in logical/conceptual terms.
  – Our constraints are then chosen to ensure that the feasible solutions to the resulting mathematical program are indeed the ones we described conceptually.

• Sometime, we will have to prove formally that the formulation does in fact correspond to the model.
Formal Definition

• Suppose $\mathcal{F} \subseteq \mathbb{Z}_+^p \times \mathbb{R}_+^{n-p}$ is a set describing the solutions to our mathematical model.

• Then

$$\mathcal{S} = \{(x, y) \in (\mathbb{Z}_+^p \times \mathbb{R}_+^{n-p}) \times (\mathbb{Z}_+^t \times \mathbb{R}_+^{r-t}) \mid Ax + Gy \leq b\}$$

is a valid (linear) formulation if $\mathcal{F} = (\text{proj})_x(\mathcal{S})$.

• Note that the formulation may have auxiliary variables that are not in the mathematical model.

• A typical mathematical model can have many valid formulations.

• In this class, we will focus on problems that have linear formulations (naturally, not every problem does).

• We will see that the specific formulation we choose can have a big impact on the efficiency of the solutions method.

• Finding a “good” formulation is critical to solving a given linear model efficiently and is a good deal of what this course is about.
Modeling with Integer Variables

• From a practical standpoint, why do we need integer variables?
Modeling with Integer Variables

• From a practical standpoint, why do we need integer variables?

• If the variable is associated with a physical entity that is indivisible, then it must be integer.
  – Product mix problem.
  – Cutting stock problem.

• We can use 0-1 (binary) variables for a variety of purposes.
  – Modeling yes/no decisions.
  – Enforcing disjunctions.
  – Enforcing logical conditions.
  – Modeling fixed costs.
  – Modeling piecewise linear functions.
Conjunction versus Disjunction

• A more general mathematical view that ties integer programming to logic is to think of integer variables as expressing disjunction.

• The constraints of a standard mathematical program are conjunctive.
  – All constraints must be satisfied.
  – In terms of logic, we have
    \[ g_1(x) \leq b_1 \ \text{AND} \ g_2(x) \leq b_2 \ \text{AND} \ \cdots \ \text{AND} \ g_m(x) \leq b_m \]  
      \hspace{1cm} (1)
  – This corresponds to intersection of the regions associated with each constraint.

• Integer variables introduce the possibility to model disjunction.
  – At least one constraint must be satisfied.
  – In terms of logic, we have
    \[ g_1(x) \leq b_1 \ \text{OR} \ g_2(x) \leq b_2 \ \text{OR} \ \cdots \ \text{OR} \ g_m(x) \leq b_m \]  
      \hspace{1cm} (2)
  – This corresponds to union of the regions associated with each constraint.
Modeling Binary Choice

- We use binary variables to model yes/no decisions.
- **Example**: Integer knapsack problem
  - We are given a set of items with associated values and weights.
  - We wish to select a subset of maximum value such that the total weight is less than a constant $K$.
  - We associate a 0-1 variable with each item indicating whether it is selected or not.

\[
\begin{align*}
\text{max} & \quad \sum_{j=1}^{m} c_j x_j \\
\text{s.t.} & \quad \sum_{j=1}^{m} w_j x_j \leq K \\
& \quad x \in \{0, 1\}^n
\end{align*}
\]
Modeling Dependent Decisions

• We can also use binary variables to enforce the condition that a certain action can only be taken if some other action is also taken.

• Suppose $x$ and $y$ are binary variables representing whether or not to take certain actions.

• The constraint $x \leq y$ says “only take action $x$ if action $y$ is also taken”.
Example: Facility Location Problem

- We are given \( n \) potential facility locations and \( m \) customers that must be serviced from those locations.
- There is a fixed cost \( c_j \) of opening facility \( j \).
- There is a cost \( d_{ij} \) associated with serving customer \( i \) from facility \( j \).
- We have two sets of binary variables.
  - \( y_j \) is 1 if facility \( j \) is opened, 0 otherwise.
  - \( x_{ij} \) is 1 if customer \( i \) is served by facility \( j \), 0 otherwise.

\[
\begin{align*}
\min & \quad \sum_{j=1}^{n} c_j y_j + \sum_{i=1}^{m} \sum_{j=1}^{n} d_{ij} x_{ij} \\
\text{s.t.} & \quad \sum_{j=1}^{n} x_{ij} = 1 \quad \forall i \\
& \quad x_{ij} \leq y_j \quad \forall i, j \\
& \quad x_{ij}, y_j \in \{0, 1\} \quad \forall i, j
\end{align*}
\]
Selecting from a Set

• We can use constraints of the form \( \sum_{j \in T} x_j \geq 1 \) to represent that at least one item should be chosen from a set \( T \).

• Similarly, we can also model that at most one or exactly one item should be chosen.

• **Example**: Set covering problem
  
  – A set covering problem is any problem of the form
    
    \[
    \begin{align*}
    \text{min} & \quad c^\top x \\
    \text{s.t.} & \quad Ax \geq 1 \\
    & \quad x_j \in \{0, 1\} \forall j
    \end{align*}
    \]
    
    where \( A \) is a **0-1** matrix.
  
  – Each **row** of \( A \) represents an item from a set \( S \).
  
  – Each **column** \( A_j \) represents a subset \( S_j \) of the items.
  
  – Each **variable** \( x_j \) represents selecting subset \( S_j \).
  
  – The **constraints** say that \( \bigcup_{\{j | x_j = 1\}} S_j = S \).
  
  – In other words, each item must appear in at least one selected subset.
Modeling Disjunctive Constraints

- We are given two constraints \( a^\top x \geq b \) and \( c^\top x \geq d \) with nonnegative coefficients.
- Instead of insisting both constraints be satisfied, we want at least one of the two constraints to be satisfied.
- To model this, we define a binary variable \( y \) and impose
  \[
  a^\top x \geq yb, \\
  c^\top x \geq (1 - y)d, \\
  y \in \{0, 1\}.
  \]
- More generally, we can impose that exactly \( k \) out of \( m \) constraints be satisfied with
  \[
  (a_i')^\top x \geq b_i y_i, \quad i \in [1..m] \\
  \sum_{i=1}^{m} y_i \geq k, \\
  y_i \in \{0, 1\}
  \]
Modeling a Restricted Set of Values

- We may want variable $x$ to only take on values in the set $\{a_1, \ldots, a_m\}$.
- We introduce $m$ binary variables $y_j, j = 1, \ldots, m$ and the constraints

$$x = \sum_{j=1}^{m} a_j y_j,$$

$$\sum_{j=1}^{m} y_j = 1,$$

$$y_j \in \{0, 1\}$$
**Piecewise Linear Cost Functions**

- We can use binary variables to model arbitrary piecewise linear cost functions.

- The function is specified by ordered pairs \((a_i, f(a_i))\) and we wish to evaluate it at a point \(x\).

- We have a binary variable \(y_i\), which indicates whether \(a_i \leq x \leq a_{i+1}\).

- To evaluate the function, we will take linear combinations \(\sum_{i=1}^{k} \lambda_i f(a_i)\) of the given functions values.

- This only works if the only two nonzero \(\lambda_i\)s are the ones corresponding to the endpoints of the interval in which \(x\) lies.
Minimizing Piecewise Linear Cost Functions

• The following formulation minimizes the function.

\[
\min \sum_{i=1}^{k} \lambda_i f(a_i)
\]

s.t. \[ \sum_{i=1}^{k} \lambda_i = 1, \]
\[ \lambda_1 \leq y_1, \]
\[ \lambda_i \leq y_{i-1} + y_i, \quad i \in [2..k-1], \]
\[ \lambda_k \leq y_{k-1}, \]
\[ \sum_{i=1}^{k-1} y_i = 1, \]
\[ \lambda_i \geq 0, \]
\[ y_i \in \{0, 1\}. \]

• The key is that if \( y_j = 1 \), then \( \lambda_i = 0, \forall i \neq j, j+1 \).
Modeling General Nonconvex Functions

- One way of dealing with general nonconvexity is by dividing the domain of a nonconvex function into regions over which it is convex (or concave).
- We can do this using integer variables to choose the region.
- This is precisely what is done in the case of the piecewise linear cost function above.
- Most methods of general global optimization use some form of this approach.
Fixed-charge Problems

• In many instances, there is a fixed cost and a variable cost associated with a particular decision.

• Example: Fixed-charge Network Flow Problem

  – We are given a directed graph $G = (N, A)$.
  – There is a fixed cost $c_{ij}$ associated with “opening” arc $(i, j)$ (think of this as the cost to “build” the link).
  – There is also a variable cost $d_{ij}$ associated with each unit of flow along arc $(i, j)$.
  – Consider an instance with a single supply node.
    * Minimizing the fixed cost by itself is a minimum spanning tree problem (easy).
    * Minimizing the variable cost by itself is a minimum cost network flow problem (easy).
    * We want to minimize the sum of these two costs (difficult).
Modeling the Fixed-charge Network Flow Problem

- To model the FCNFP, we associate two variables with each arc.
  - $x_{ij}$ (fixed-charge variable) indicates whether arc $(i, j)$ is open.
  - $f_{ij}$ (flow variable) represents the flow on arc $(i, j)$.
  - Note that we have to ensure that $f_{ij} > 0 \Rightarrow x_{ij} = 1$.

$$\min \sum_{(i,j) \in A} c_{ij}x_{ij} + d_{ij}f_{ij}$$

s.t. $$\sum_{j \in O(i)} f_{ij} - \sum_{j \in I(i)} f_{ji} = b_i \quad \forall i \in N$$

$$f_{ij} \leq C \cdot x_{ij} \quad \forall (i, j) \in A$$

$$f_{ij} \geq 0 \quad \forall (i, j) \in A$$

$$x_{ij} \in \{0, 1\} \quad \forall (i, j) \in A$$
Alternative Formulations

• Recall our earlier definition of a valid formulation.

• A key concept in the rest of the course will be that every mathematical model has many alternative formulations.

• Many of the key methodologies in integer programming are essentially automatic methods of reformulating a given model.

• The goal of the reformulation is to make the model easier to solve.
**Example: Facility Location Problem**

- We are given \( n \) potential facility locations and \( m \) customers.
- There is a fixed cost \( c_j \) of opening facility \( j \).
- There is a cost \( d_{ij} \) associated with serving customer \( i \) from facility \( j \).
- We have two sets of binary variables.
  - \( y_j \) is 1 if facility \( j \) is opened, 0 otherwise.
  - \( x_{ij} \) is 1 if customer \( i \) is served by facility \( j \), 0 otherwise.
- Here is one formulation:

\[
\text{min} \sum_{j=1}^{n} c_j y_j + \sum_{i=1}^{m} \sum_{j=1}^{n} d_{ij} x_{ij} \\
\text{s.t.} \sum_{j=1}^{n} x_{ij} = 1 \quad \forall i \\
\sum_{i=1}^{m} x_{ij} \leq m y_j \quad \forall j \\
x_{ij}, y_j \in \{0, 1\} \quad \forall i, j
\]
Example: Facility Location Problem

• Here is another formulation for the same problem:

\[
\begin{align*}
\text{min} & \quad \sum_{j=1}^{n} c_j y_j + \sum_{i=1}^{m} \sum_{j=1}^{n} d_{ij} x_{ij} \\
\text{s.t.} & \quad \sum_{j=1}^{n} x_{ij} = 1 \quad \forall i \\
& \quad x_{ij} \leq y_j \quad \forall i, j \\
& \quad x_{ij}, y_j \in \{0, 1\} \quad \forall i, j
\end{align*}
\]

• Notice that the set of integer solutions contained in each of the polyhedra is the same (why?).

• However, the second polyhedron is strictly included in the first one.

• Therefore, the second polyhedron will yield a better lower bound.

• The second polyhedron is a better approximation to the convex hull of integer solutions.
Formulation Strength and Ideal Formulations

• Consider two formulations $A$ and $B$ for the same ILP.

• Denote the feasible regions corresponding to their LP relaxations as $P_A$ and $P_B$.

• Formulation $A$ is said to be at least as strong as formulation $B$ if $P_A \subseteq P_B$.

• If the inclusion is strict, then $A$ is stronger than $B$.

• If $F$ is the set of all feasible integer solutions for the ILP, then we must have $\text{conv}(F) \subseteq P_A$ (why?).

• $A$ is ideal if $\text{conv}(F) = P_A$.

• If we know an ideal formulation, we can solve the IP (why?).
Strengthening Formulations

- Often, a given formulation can be strengthened with additional inequalities satisfied by all feasible integer solutions.

- **Example**: The Perfect Matching Problem
  - We are given a set of $n$ people that need to paired in teams of two.
  - Let $c_{ij}$ represent the “cost” of the team formed by person $i$ and person $j$.
  - We wish to maximize efficiency over all teams.
  - We can represent this problem on an undirected graph $G = (N, E)$.
  - The nodes represent the people and the edges represent pairings.
  - We have $x_e = 1$ if the endpoints of $e$ are matched, $x_e = 0$ otherwise.

\[
\begin{align*}
\text{min} & \quad \sum_{e=\{i,j\}\in E} c_e x_e \\
\text{s.t.} & \quad \sum_{\{j\}|\{i,j\}\in E} x_{ij} = 1, \quad \forall i \in N \\
& \quad x_e \in \{0, 1\}, \quad \forall e = \{i, j\} \in E.
\end{align*}
\]
Valid Inequalities for Matching

• Consider the graph on the left above.
• The optimal perfect matching has value $L + 2$.
• The optimal solution to the LP relaxation has value 3.
• This formulation can be extremely weak.
• Add the valid inequality $x_{24} + x_{35} \geq 1$.
• Every perfect matching satisfies this inequality.
The Odd Set Inequalities

- We can generalize the inequality from the last slide.
- Consider the cut $S$ corresponding to any odd set of nodes.
- The *cutset* corresponding to $S$ is

$$
\delta(S) = \{ \{i, j\} \in E | i \in s, j \notin S \}.
$$

- An *odd cutset* is any $\delta(S)$ for which $|S|$ is odd.
- Note that every perfect matching contains at least one edge from every odd cutset.
- Hence, each odd cutset induces a possible valid inequality.

$$
\sum_{e \in \delta(S)} x_e \geq 1, S \subset N, |S| \text{ odd}.
$$
Using the New Formulation

• If we add all of the odd set inequalities, the new formulation is ideal.
• Hence, we can solve this LP and get a solution to the IP.
• However, the number of inequalities is exponential in size, so this is not really practical.
• Recall that only a small number of these inequalities will be active at the optimal solution.
• Later, we will see how we can efficiently generate these inequalities on the fly to solve the IP.
Extended Formulations

- We have so far focused on strengthening formulations using additional constraints.
- However, changing the set of variables can also have a dramatic effect.
- **Example**: A Lot-sizing Problem
  - We want to minimize the costs of production, storage, and set-up.
  - Data for period $t = 1, \ldots, T$:
    * $d_t$: total demand,
    * $c_t$: production set-up cost,
    * $p_t$: unit production cost,
    * $h_t$: unit storage cost.
  - Variables for period $t = 1, \ldots, T$:
    * 
    * 
    * 
    *
Lot-sizing: The “natural” formulation

• Here is the formulation based on the “natural” set of variables:

\[
\begin{align*}
\min & \sum_{t=1}^{T} (p_t y_t + h_t s_t + c_t x_t) \\
\text{s.t.} & \quad y_1 = d_1 + s_1, \\
& \quad s_{t-1} + y_t = d_t + s_t, \quad \text{for } t = 2, \ldots, T, \\
& \quad y_t \leq \omega x_t, \quad \text{for } t = 1, \ldots, T, \\
& \quad s_T = 0, \\
& \quad s, y \in \mathbb{R}_+^T, \\
& \quad x \in \{0, 1\}^T.
\end{align*}
\]

• Here, \( \omega = \sum_{t=1}^{T} d_t \), an upper bound on \( y_t \).
Lot-sizing: The “extended” formulation

- Suppose we split the production lot in period $t$ into smaller pieces.
- Define the variables $q_{it}$ to be the production in period $i$ designated to satisfy demand in period $t \geq i$.
- Now, $y_i = \sum_{t=i}^{T} q_{it}$.
- With the new set of variables, we can impose the tighter constraint

$$q_{it} \leq d_t x_i \text{ for } i = 1, \ldots, T \text{ and } t = 1, \ldots, T.$$  

- The additional variables make the formulation ideal.
- If we project into the original space, we will get the convex hull of solutions to the first formulation.
- Again, this in contrary to conventional wisdom for formulating linear programs.
Contrast with Linear Programming

• In linear programming, the same problem can also have multiple formulations.

• In LP, however, conventional wisdom is that bigger formulations take longer to solve.

• In IP, this conventional wisdom does not hold.

• We have already seen two examples where it is not valid.

• Generally speaking, the size of the formulation does not determine how difficult the IP is.