Reading for This Lecture

- N&W Sections I.1.1-I.1.6
- Wolsey Chapter 1
- CCZ Chapter 2
Formulations and Models

- Our description in the last lecture boiled the modeling process down to two basic steps.
  1. Create a *conceptual model* of the real-world problem.
  2. Translate the conceptual model into a *formulation*.

- In the *conceptual model*, we initially describe what values of the variables we would like to allow in logical/conceptual terms (the feasible set).

- In the *formulation*, we specify constraints that ensure that the feasible solutions to the resulting mathematical optimization problem are indeed “feasible” in terms of the conceptual model.

- Integer (and other) variables that don’t appear in the conceptual model may be introduced to enforce logical conditions.

- We also try to account for “solvability.”

- We may have to prove formally that the resulting formulation does in fact correspond to the model (and eventually to the real-world problem).
Formal Definition

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

• Then

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

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

• Note that the formulation may have auxiliary variables that are not in the conceptual model (we will see an example later in the lecture).

• 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.
Proving Correctness

• There are two parts to proving a formulation is correct, although one of both of these may be “obvious”.
  – First, we have to prove that $\mathcal{F}$ is in fact the set of solutions to the original problem, which may have been described non-mathematically.
  – Second, we have to prove our formulation is correct.

• Proving correctness of a given formulation generally means proving $\mathcal{F} = \text{proj}_x(S)$.

• The most straightforward way of doing this involves proving
  – $x \in \mathcal{F} \Rightarrow x \in \text{proj}_x(S)$, and
  – $x \in \text{proj}_x(S) \Rightarrow x \in \mathcal{F}$. 
Problem Reduction

• Modeling involves transformation of a problem described in one formal (or informal) language into an equivalent problem described in another.

• Such transformations are formally known as *reductions* and we will study them in more detail later in the course.

• Informally, reducing problem A to problem B involves showing that there is
  
  – a mapping of each “instance” of problem A to an “instance” of problem B, and
  – a mapping of solutions to problem B to solutions of problem A
  
such that we can solve problem A correctly by

  1. Mapping the instance of problem A to an instance of problem B;
  2. Solving the instance of problem B; and then
  3. Mapping the solution we obtain back to a solution of problem A.
Problem Reduction and Modeling

• Modeling of a general optimization problem involves reducing that model to a mathematical optimization problem.

• Proving a formulation correct amounts to proving that the general optimization problem over feasible set $\mathcal{F}$ can be reduced to a mathematical optimization problem.

• We may also do reductions from one mathematical optimization problem to another in some cases.

• These reductions may involve problems defined over completely different sets of variables.
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?

• We have seen in the last lecture that integer variable essentially allow us to introduce *disjunctive logic*

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

• At its heart, integrality is a kind of disjunction constraint.

• *0-1 (binary) variables* are often used to model more abstract kinds of disjunctions (non-numerical).
  – Modeling yes/no decisions.
  – Enforcing logical conditions.
  – Modeling fixed costs.
  – Modeling piecewise linear functions.
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.

$$\begin{align*}
\text{max} & \sum_{j=1}^{m} c_j x_j \\
\text{s.t.} & \sum_{j=1}^{m} w_j x_j \leq K \\
& x \in \{0, 1\}^n
\end{align*}$$
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.
• 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:

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

  $$\min c^\top x$$

  $$\text{s.t. } Ax \geq 1$$

  $$x_j \in \{0, 1\} \forall j$$

  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 \( a^T x \geq b \) and \( c^T x \geq d \) 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

\[
\begin{align*}
  a^T x & \geq yb, \\
  c^T x & \geq (1 - y)d, \\
  y & \in \{0, 1\}.
\end{align*}
\]

- More generally, we can impose that exactly \( k \) out of \( m \) constraints be satisfied with

\[
\begin{align*}
  (a_i')^T x & \geq b_i y_i, \quad i \in [1..m] \\
  \sum_{i=1}^{m} y_i & \geq k, \\
  y_i & \in \{0, 1\}
\end{align*}
\]
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 $f_{ij} > 0 \Rightarrow x_{ij} = 1$.

$$
\begin{align*}
\text{min} & \quad \sum_{(i,j) \in A} c_{ij} x_{ij} + d_{ij} f_{ij} \\
\text{s.t.} & \quad \sum_{j \in O(i)} f_{ij} - \sum_{j \in I(i)} f_{ji} = b_i \quad \forall i \in N \\
& \quad f_{ij} \leq C x_{ij} \quad \forall (i, j) \in A \\
& \quad f_{ij} \geq 0 \quad \forall (i, j) \in A \\
& \quad x_{ij} \in \{0, 1\} \quad \forall (i, j) \in A
\end{align*}
$$
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.
Simple Example: Knapsack Problem

• We are given a set \( N = \{1, \ldots, n\} \) of items and a capacity \( W \).
• There is a profit \( p_i \) and a size \( w_i \) associated with each item \( i \in N \).
• We want to choose the set of items that maximizes profit subject to the constraint that their total size does not exceed the capacity.
• The most straightforward formulation is to introduce a binary variable \( x_i \) associated with each item.
• \( x_i \) takes value 1 if item \( i \) is chosen and 0 otherwise.
• Then the formulation is

\[
\begin{align*}
\min & \quad \sum_{j=1}^{n} p_j x_j \\
\text{s.t.} & \quad \sum_{j=1}^{n} w_j x_j \leq W \\
& \quad x_i \in \{0, 1\} \quad \forall i
\end{align*}
\]

• Is this formulation correct?
An Alternative Formulation

- Let us call a set \( C \subseteq N \) a *cover* if \( \sum_{i \in C} w_i > W \).
- Further, a cover \( C \) is *minimal* if \( \sum_{i \in C \setminus \{j\}} w_i > W \) for all \( j \in C \).
- Then we claim that the following is also a valid formulation of the original problem.

\[
\begin{align*}
\min \sum_{j=1}^{n} p_j x_j \\
\text{s.t. } \sum_{j \in C} x_j &\leq |C| - 1 \text{ for all minimal covers } C \\
x_i &\in \{0, 1\} \quad i \in N
\end{align*}
\]

- Which formulation is “better”?
Back to the Facility Location Problem

• Recall our earlier formulation of this problem.

• Here is another formulation for the same problem:

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

• 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 (how do we prove this?).

• 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 \textit{at least as strong as} formulation $B$ if $P_A \subseteq P_B$.
• If the inclusion is \textit{strict}, then $A$ is \textit{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 \textit{ideal} if $\text{conv}(F) = P_A$.
• If we know an ideal formulation, we can solve the IP (why?).
• How do our formulations of the knapsack problem compare by this measure?
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 be 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.

$$
\begin{align*}
\min & \quad \sum_{e=\{i,j\} \in E} c_e x_e \\
\text{s.t.} & \quad \sum_{\{j\} | \{i,j\} \in E} x_{ij} = 1, \forall i \in N \\
& \quad x_e \in \{0, 1\}, \quad \forall e = \{i,j\} \in E.
\end{align*}
$$
• 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
  \[ \delta(S) = \{ \{i, j\} \in E \mid i \in s, j \notin S \}. \]
- 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:

\[
\min \sum_{t=1}^{T} (p_t y_t + h_t s_t + c_t x_t)
\]

s.t. \( y_1 = d_1 + s_1, \)
\( s_{t-1} + y_t = d_t + s_t, \) for \( t = 2, \ldots, T, \)
\( y_t \leq \omega x_t, \) for \( t = 1, \ldots, T, \)
\( s_T = 0, \)
\( s, y \in \mathbb{R}^T_+, \)
\( x \in \{0, 1\}^T. \)

• 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 strengthen the formulation.
- Again, this is contrary to conventional wisdom for formulating linear programs.
Strength of Formulation for Lot-sizing

• Although the formulation from the previous slide is much stronger than our original, it is still not ideal.

• Consider the following sample data.

```python
# The demands for six periods
DEMAND = [6, 7, 4, 6, 3, 8]

# The production cost for six periods
PRODUCTION_COST = [3, 4, 3, 4, 4, 5]

# The storage cost for six periods
STORAGE_COST = [1, 1, 1, 1, 1, 1]

# The set up cost for six periods
SETUP_COST = [12, 15, 30, 23, 19, 45]

# Set of periods
PERIODS = range(len(DEMAND))
```
Strength of Formulation for Lot-sizing (cont’d)

Optimal Total Cost is: 171.42016761

Period 0: 13 units produced, 7 units stored, 6 units sold
0.38235294 is the value of the fixed charge variable

Period 1: 0 units produced, 0 units stored, 7 units sold
0.0 is the value of the fixed charge variable

Period 2: 4 units produced, 0 units stored, 4 units sold
0.19047619 is the value of the fixed charge variable

Period 3: 6 units produced, 0 units stored, 6 units sold
0.35294118 is the value of the fixed charge variable

Period 4: 11 units produced, 8 units stored, 3 units sold
1.0 is the value of the fixed charge variable

Period 5: 0 units produced, 0 units stored, 8 units sold
0.0 is the value of the fixed charge variable

What is happening here?
Strength of Formulation for Lot-sizing (cont’d)

Let’s take a more detailed look:

production in period 0 for period 0 : 2.2941176
production in period 0 for period 1 : 2.6764706
production in period 0 for period 2 : 1.5294118
production in period 0 for period 3 : 2.2941176
production in period 0 for period 4 : 1.1470588
production in period 0 for period 5 : 3.0588235

What is the problem?
An Ideal Formulation for Lot-sizing

• We can further strengthen the formulation by adding the constraint

\[
\sum_{i=1}^{t} q_{it} \geq d_t \text{ for } t = 1, \ldots, T
\]

• In fact, adding these additional constraints makes the formulation ideal.

• If we *project* into the original space, we will get the convex hull of solutions to the first formulation.

• How would we prove this?
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.