Reading for This Lecture

• Wolsey Sections 7.4-7.5
• Nemhauser and Wolsey Section II.4.2
• Linderoth and Savelsburgh, (1999)
• Martin (2001)
• Achterberg, Koch, Martin (2005)
• Karamanov and Cornuejols, Branching on General Disjunctions (2007)
• Achterberg, Conflict Analysis in Mixed Integer Programming (2007)
Branch and Bound Recap

• As usual, suppose $S$ is the feasible region of an MILP and we wish to solve $\max_{x \in S} c^\top x$.

• To apply branch and bound, we consider a partition of $S$ into subsets $S_1, \ldots, S_k$. Then

$$\max_{x \in S} c^\top x = \max_{1 \leq i \leq k} \max_{x \in S_i} c^\top x.$$  

• In other words, we can optimize over each subset separately.

• **Idea**: If we can’t solve the original problem directly, we might be able to solve the smaller subproblems recursively.

• Dividing the original problem into subproblems is called **branching**.

• Taken to the extreme, this scheme is equivalent to complete enumeration.
Branching

• We have now spent several lectures discussing methods for bounding.

• Obtaining tight bounds is the most important aspect of the branch-and-bound algorithm.

• Branching effectively is a very close second.

• Choosing an effective method of branching can make orders of magnitude difference in the size of the search tree and the solution time.
Disjunctions and Branching

• Recall that branching is generally achieved by selecting an admissible disjunction $\{X_i\}_{i=1}^k$ and using it to partition $S$, e.g., $S_i = S \cap X_i$.

• The way this disjunction is selected is called the branching method and is the topic we now examine.

• Generally speaking, we want $x^* \notin \bigcup_{1 \leq i \leq k} X_i$, where $x^*$ is the (infeasible) solution produced by solving the bounding problem associated with a given subproblem.
Split Disjunctions

- The most easily handled disjunctions are those based on dividing the feasible region using a given hyperplane.
- In such cases, each term of the disjunction can be imposed by addition of a single inequality.
- A hyperplane defined by a vector $\pi \in \mathbb{R}^n$ is said to be integer if $\pi_i \in \mathbb{Z}$ for $0 \leq i \leq p$ and $\pi_i = 0$ for $p + 1 \leq i \leq n$.
- Note that if $\pi$ is integer, then we have $\pi^\top x \in \mathbb{Z}$ whenever $x \in \mathbb{Z}^p \times \mathbb{R}^{n-p}$.
- Then the disjunction defined by

$$X_1 = \{x \in \mathbb{R}^n \mid \pi x \leq \pi_0\}, \quad X_2 = \{x \in \mathbb{R}^n \mid \pi x \geq \pi_0 + 1\}, \quad (1)$$

is valid when $\pi_0 \in \mathbb{Z}$.
- This is known as a split disjunction.
**Variable Disjunctions**

- The simplest split disjunction is to take $\pi = e_i$ for $0 \leq i \leq p$, where $e_i$ is the $i^{th}$ unit vector.
- If we branch using such a disjunction, we simply say we are *branching on* $x_j$.
- For such a branching disjunction to be admissible, we should have $\pi_0 < x_i^* < \pi_0 + 1$.
- In the special case of a 0-1 IP, this dichotomy reduces to
  
  $$x_j = 0 \text{ OR } x_j = 1$$

- In general IP, branching on a variable involves imposing *new bound constraints* in each one of the subproblems.
- This is easily handled implicitly in most cases.
- This is the most common method of branching.
- What are the benefits of such a scheme?
The Geometry of Branching

Figure 1: Feasible region of an MILP
The Geometry of Branching (Variable Disjunction)

Figure 2: Branching on disjunction $x \leq 2$ OR $x \geq 3$
Figure 3: Branching on disjunction \( y \leq 4 \) OR \( y \geq 5 \) in Subproblem 2
The Geometry of Branching (General Split Disjunction)

Figure 4: Branching on disjunction $x + 2y \leq 11$ OR $x + 2y \geq 12$
The Geometry of Branching (General Split Disjunction)

Figure 5: Branching on hyperplane $x \leq 2$ OR $x \geq 3$ in Subproblem 1
Other Disjunctions

• A *generalized upper bound* (GUB) is of the form:

\[ \sum_{j \in Q} x_j = 1, \quad x \in \{0, 1\}^Q \]

• Suppose \(|Q| = 10\) and we branch on the disjunction \(x_1 \leq 0 \text{ OR } x_1 \geq 1\).

• How many possible solutions to the above equation are there in each of the branches? Is this a “good” disjunction to branch on?

• Consider the disjunction \(\sum_{j=1}^{5} x_j = 0 \text{ OR } \sum_{j=6}^{10} x_j = 0\).

• Is this better?
Logical Disjunctions

- We can derive other types of branching based on logical considerations.

- **Example #1:**
  
  - $y_i$ binary variable and $y_i = 0 \Rightarrow \pi x \leq \pi_0$.
  
  - Possible branching:
    
    $y_i = 1$,
    
    $y_i = 0$ and $\pi x \leq \pi_0$.

- This avoids using the big $M$ method.

- **Example #2:** Solving the TSP with Lagrangian relaxation.
Choosing a Branching Disjunction

• What is the real goal of branching?
• This may depend on the goal of the search
  – Find the best feasible solution possible in a limited time.
  – Find the provably optimal solution as quickly as possible.
• It is difficult to know how our branching decision will impact these goals, but we may want to choose a branching that
  – Decreases the upper bound,
  – Increases the lower bound, or
  – Result in one or more (nearly) infeasible subproblem.
• Most of the time, we focus on decreasing the upper bound.
Choosing a Branching Disjunction (cont’d)

• There are many possible disjunctions to choose from.

• We generally choose the branching disjunction based on the predicted amount of progress it will produce towards our goal.

• If the goal is to minimize time to optimality, bound improvement is often used as a proxy.

• How do we efficiently predict the bound improvement that will result from the imposition of a given disjunction?
Strong Branching

• *Strong branching* provides the most accurate estimate, but is computationally very expensive.

• The idea is to compute the *actual* change in bound by solving the bounding problems resulting from imposing the disjunction.

• This can be very costly. How can we moderate this?
  – Do only a limited number of dual-simplex pivots for each candidate for each child.
  – Use this as an estimate.

• Empirically, this reduces number of nodes, but this must be traded against the computational expense.
Pseudocost Branching

• An alternative to strong branching is *pseudocost branching*.

• This is suitable primarily for branching on branching on variables.

• The pseudocost of a variable is an estimate derived by averaging observed changes resulting from branching on each of the variables.

• For each variable, we maintain an “up pseudocost” \( P_j^+ \) and a “down pseudocost” \( P_j^- \).

• Then the change in bound for each child can be estimated as:

\[
D_j^+ = P_j^+(1 - f_j) \\
D_j^- = P_j^- f_j,
\]

where \( f_j = x_j^* - \lfloor x_j^* \rfloor \).

• In other words, \( D_j^+ \) and \( D_j^- \) are estimates of the *change* in bound that will result from imposing \( x_j \geq \lfloor x_j^* \rfloor \) and \( x_j \geq \lceil x_j^* \rceil \), respectively.
Pseudocost Initialization

• Is it reasonable to assume that effect of branching on a particular variable is actually roughly the same in different parts of the tree?

• Empirical evidence shows that this is the case.

• Another important question is how to get initial estimates before any branching has occurred.

• This can be done initially using strong branching.

• After initialization, we switch to pseudocost branching, updating the pseudocost estimates after each bounding operation.

• A more systematic approach to doing this is to use what is called reliability branching.
Reliability Branching

- Strong branching is effective in reducing the number of nodes, but can be costly.
- Using pseudocosts is inexpensive, but requires good initialization.
- Reliability branching combines both.
  - Use strong branching in the early stages of the tree. Initialize/update pseudo-costs of variables using these bounds.
  - Once strong branching (or actual branching) has been carried out $\eta$ number of times on a variable, only use pseudo-costs after that.
  - $\eta$ is called reliability parameter.
  - What does $\eta = 0$ imply? What does $\eta = \infty$ imply?
  - Empirically $\eta = 4$ seems to be quite effective.
Comparing Branching Candidates

- So far we have seen, how to evaluate a candidate in several ways.
- Sometimes the choice of candidate is clear after this evaluation.

- Are we minimizing or maximizing?
- Which candidate would you choose?
Comparing Candidates

• However, choice of candidates is not always clear.

• Consider

\[
\begin{align*}
\begin{array}{ccc}
  x_1 & \leq & 0 \\
  x_1 & \geq & 1 \\
  x_2 & \leq & 0 \\
  x_2 & \geq & 1 \\
  x_3 & \leq & 0 \\
  x_3 & \geq & 1 \\
\end{array}
\end{align*}
\]

• Possible metrics (\(\tilde{z}_1, \tilde{z}_2, \ldots, \tilde{z}_r\) are the estimates for \(r\) children of a candidate):
  
  - \(\max \tilde{z}_i\)
  - \(\sum_i \tilde{z}_i / r\)
  - \(\max_i \tilde{z}_i - \min_i \tilde{z}_i\)
  - \(\alpha_1 \max_i \tilde{z}_i + \alpha_2 \min_i \tilde{z}_i\)
Comparing Candidates

- The number of fractional variables (after full strong branching) is another possible criterion.

- For more criteria based on structure of constraints, see *Active-Constraint Variable Ordering for Faster Feasibility of MILPs*, by Patel and Chinneck, 2006.
Local Branching

- Local branching is a branching scheme that emphasizes finding feasible solutions over improving the upper bound.

- Consider the solution \( x^* \) to an LP relaxation at a certain node in the tree of a binary program.

- Let \( S \) be the set: \( \{ j | x_j^* = 0 \} \).

- Consider the disjunction

\[
\sum_{j \in S} x_j \leq k \text{ OR } \sum_{j \in S} x_j \geq k + 1
\]

for small \( k \).

- Is this a valid rule?

- Which child is easier to solve?

- For full details, see *Local Branching* by Fischetti and Lodi.

- We will discuss this and other methods when we talk about *primal heuristics*. 
Valid Inequalities by Branching

• Note this one of the subproblems obtained by imposing a given binary disjunction is infeasible, the we obtain a valid inequality!

• This is in some sense what a valid inequality is.

• For the problem in Figure ??, branching on the valid disjunction $x_2 - x_1 \leq 1$ OR $x_2 - x_1 \geq 2$ immediately solves the problem.

• This may make it seem easy to find valid inequalities, but we will see later why this is not the case.