Integer Programming
ISE 418

Lecture 9

Dr. Ted Ralphs

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)

Branching and Disjunction

- Recall that branching is generally achieved by selecting an admissible disjunction $\{X_i\}_{i=1}^k$ and using it to partition $\mathcal{S}$, e.g., $\mathcal{S}_i = \mathcal{S} \cap X_i$.
- The overall strategy for selecting these disjunctions is called the *branching strategy* and is the topic we now examine.
- Generally speaking, we want $x^* \notin \cup_{1 \le i \le k} X_i$, where x^* is the (infeasible) solution produced by solving the *bounding problem*.
- It is typically easy to identify multiple such disjunctions, but it is not clear what our overall strategy should be in choosing among them.
- The end goal is for the algorithm to terminate as quickly as possible, but how do the individual branching decision contribute?
- This is still a difficult question to answer, but the first step is to identify a broad but enumerable class of disjunctions to choose from.

Split Disjunctions

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

$$X_1 = \{x \in \mathbb{R}^n \mid \alpha x \leq \beta\}, X_2 = \{x \in \mathbb{R}^n \mid \alpha x \geq \beta + 1\}, \tag{1}$$

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

Variable Disjunctions

- The simplest split disjunction is to take $\alpha = e_i$ 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 $\beta < x_i^* < \beta + 1$.
- 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

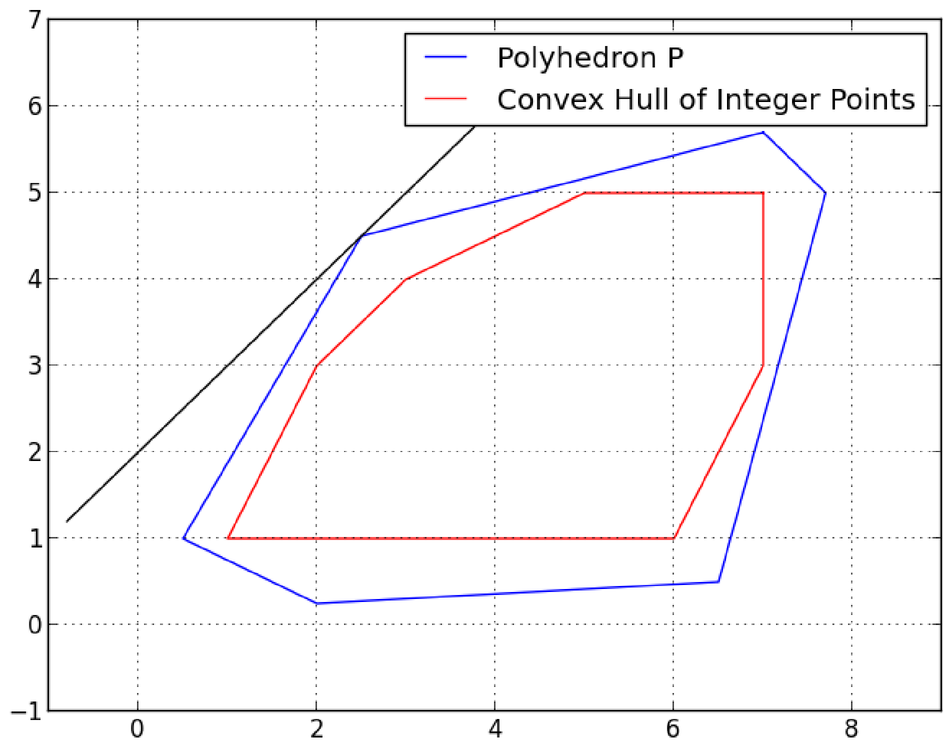


Figure 1: Feasible region of an MILP

The Geometry of Branching (Variable Disjunction)

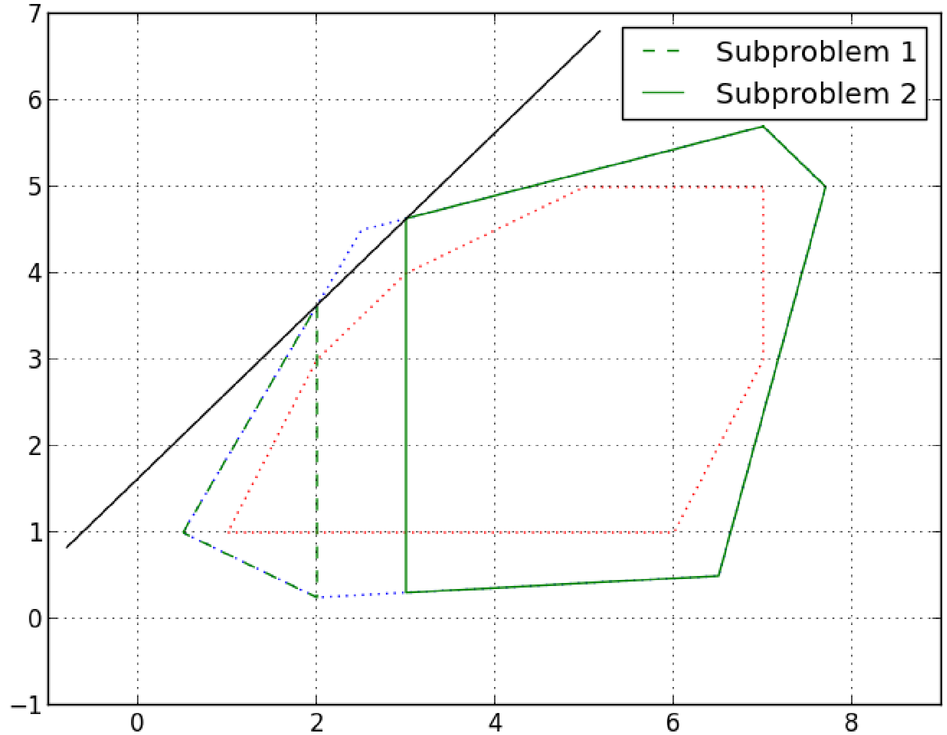


Figure 2: Branching on disjunction $x \leq 2$ OR $x \geq 3$

The Geometry of Branching (Variable Disjunction)

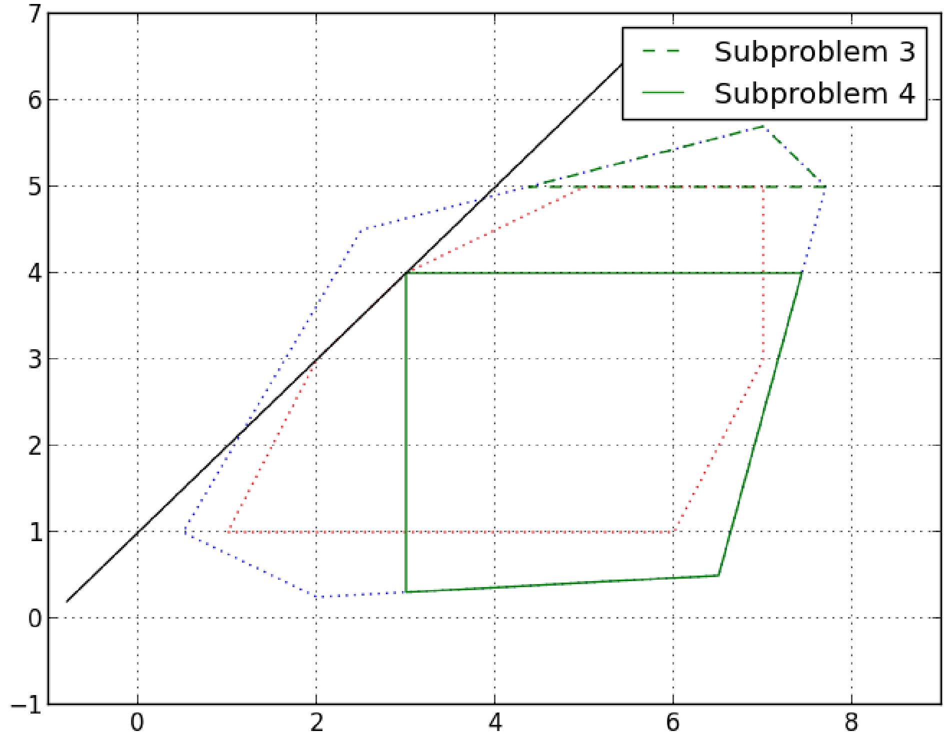


Figure 3: Branching on disjunction $y \leq 4$ OR $y \geq 5$ in Subproblem 2

The Geometry of Branching (General Split Disjunction)

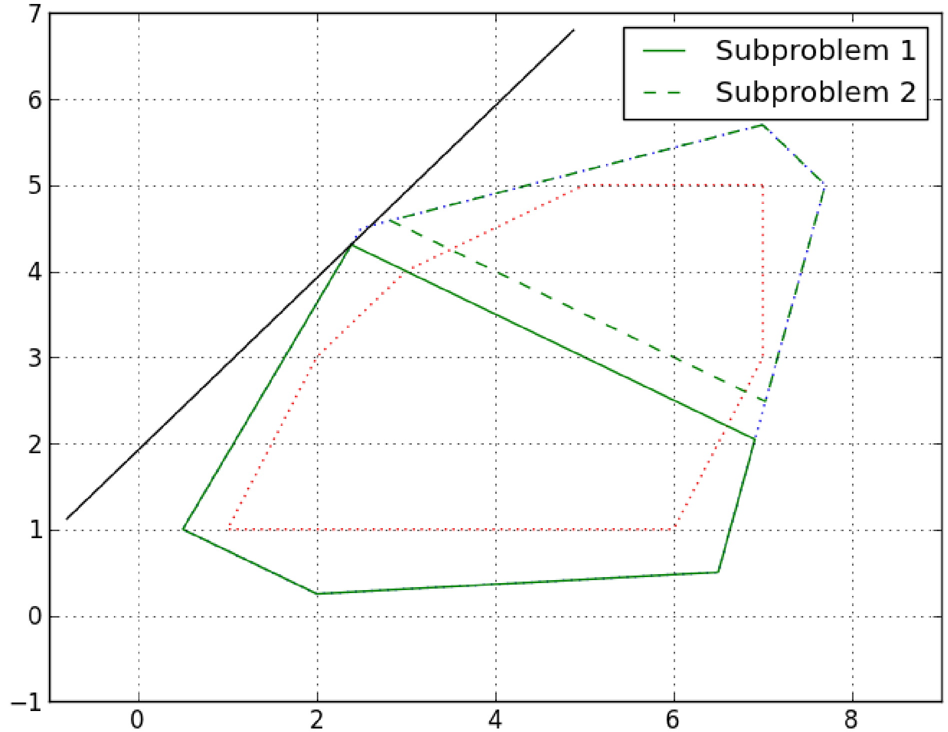


Figure 4: Branching on disjunction $x + 2y \leq 11$ OR $x + 2y \geq 12$

The Geometry of Branching (General Split Disjunction)

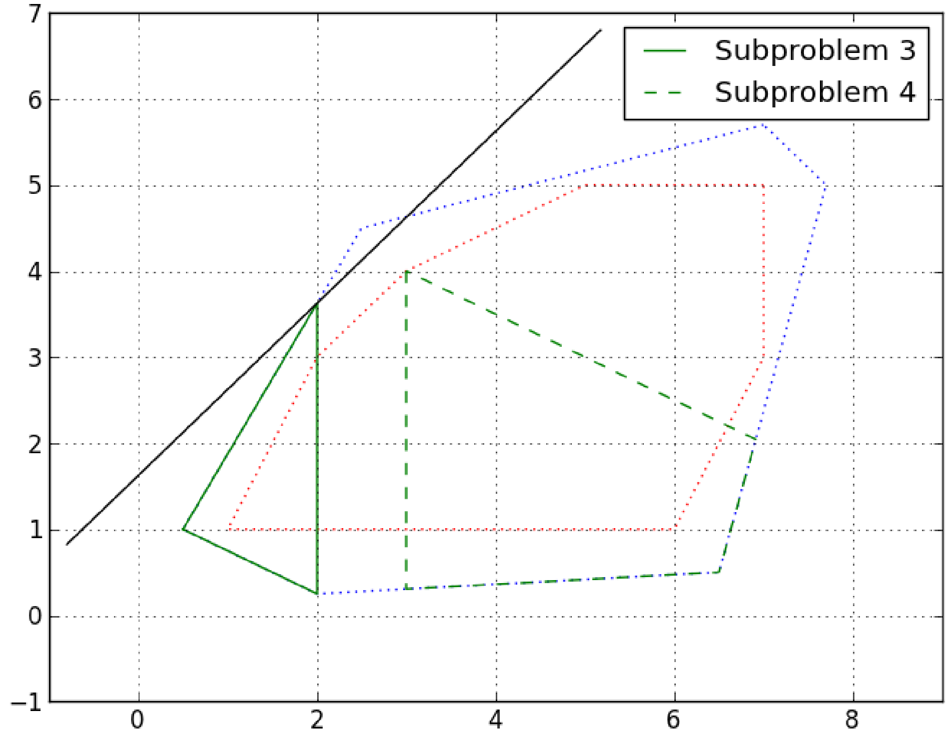


Figure 5: Branching on hyperplane $x \leq 2$ OR $x \geq 3$ in Subproblem 1

Other Disjunctions

- A *specially ordered set* (SOS) of Type I is a set Q of (binary) variables that must satisfy a constraint 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?
- There are also SOS Type II constraints in which two variables in a set may be nonzero.

Logical Disjunctions

- We can derive other types of branching based on logical considerations.
- Example:
  - y_i binary variable and $y_i = 0 \Rightarrow \pi x \leq \pi_0$.
  - Possible branching:

$$
\begin{aligned}
y_i &= 1, \\
y_i &= 0 \text{ and } \pi x \leq \pi_0.
\end{aligned}
$$

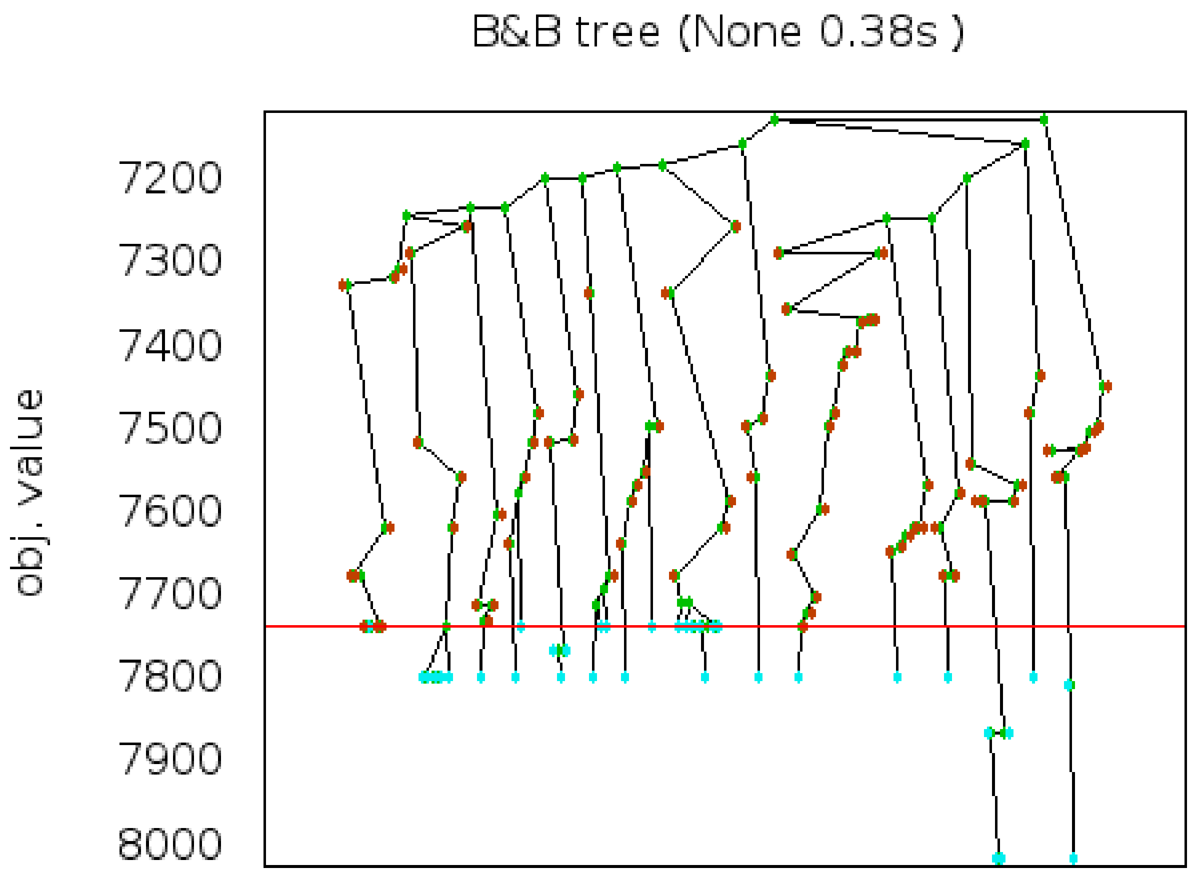
  - This avoids using the big M method.

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.

A Thousand Words

- Recall this picture.
- Think about what effect different branching has on closing the gap.



Recall that we are minimizing here!

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.
- Two questions
  - How do we efficiently **predict** the progress (bound improvement) that will result from the imposition of possible branching disjunctions?
  - How do we **choose** from among the alternative disjunctions based on the given predictions?
- We'll first address the question of choice, assuming we have a good method of prediction.

What Are We Predicting?

- Recall the nodal dual functions from the previous lecture, defined as follows (the optimization sense is now reverted to maximization).

$$\begin{aligned} \phi^t(\beta) = \min\ & \pi^t\beta + \underline{\pi}^t l^t + \bar{\pi}^t u^t \\ \text{s.t. } & \pi^t A + \underline{\pi}^t + \bar{\pi}^t \leq c^\top \\ & \pi, \bar{\pi} \geq 0, \underline{\pi} \leq 0 \end{aligned} \tag{BB.LP.D}$$

- Then the bound resulting from branching on variable x_i would be

$$\max\{\phi^1(b), \phi^2(b)\},$$

where ϕ^1 and ϕ^2 are the nodal dual functions of the child nodes resulting from the branching.

- It is the values $\phi^1(b)$ and $\phi^2(b)$ that we are estimating.

A Related Function

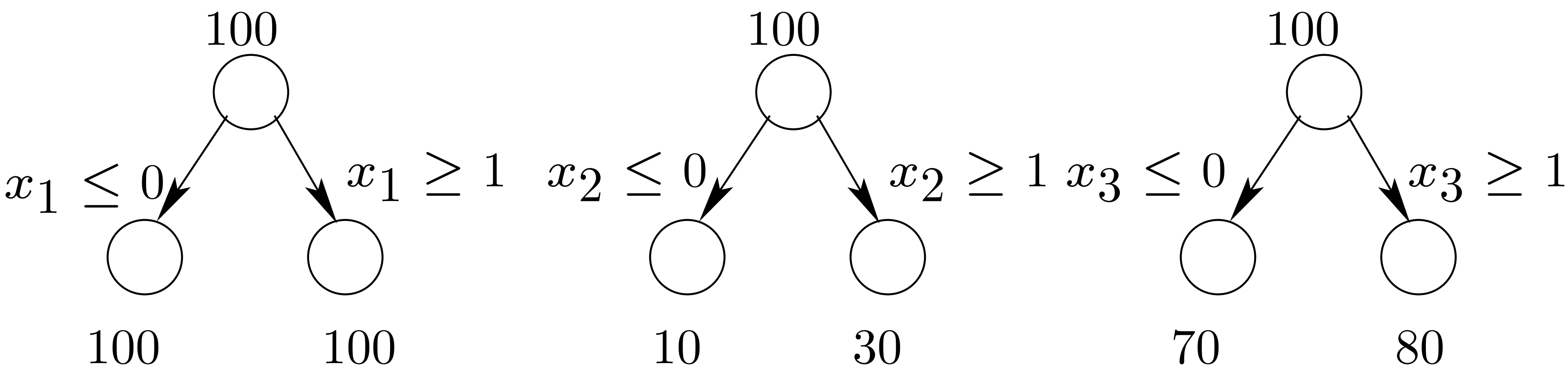
- In the previous slide, we have characterized the nodal dual function ϕ^t as a function of β with fixed vectors (l^t, u^t) of bounds on the variables.
- Note, however, that the nodal dual functions are themselves closely related.
- Let us define a new function in which the vectors of variable bounds are the argument and the right-hand side is fixed.

$$\begin{aligned} \phi_{\mathrm{LP}}(l,u) = \min\ & \pi b + \underline{\pi} l + \bar{\pi} u \\ \text{s.t. } & \pi A + \underline{\pi} + \bar{\pi} \le c^\top \\ & \underline{\pi} \ge 0, \bar{\pi} \le 0 \end{aligned}$$

- Then we have that $\phi^t(b) = \phi_{\mathrm{LP}}(l^t, u^t)$.
- The function ϕ_{LP} can be considered as a kind of value function.
- Branching methods essentially try to build a very crude esimate of this function, as we will see.
- A possible topic of research would be to consider building a better estimate in a more direct fashion.

Comparing Branching Candidates

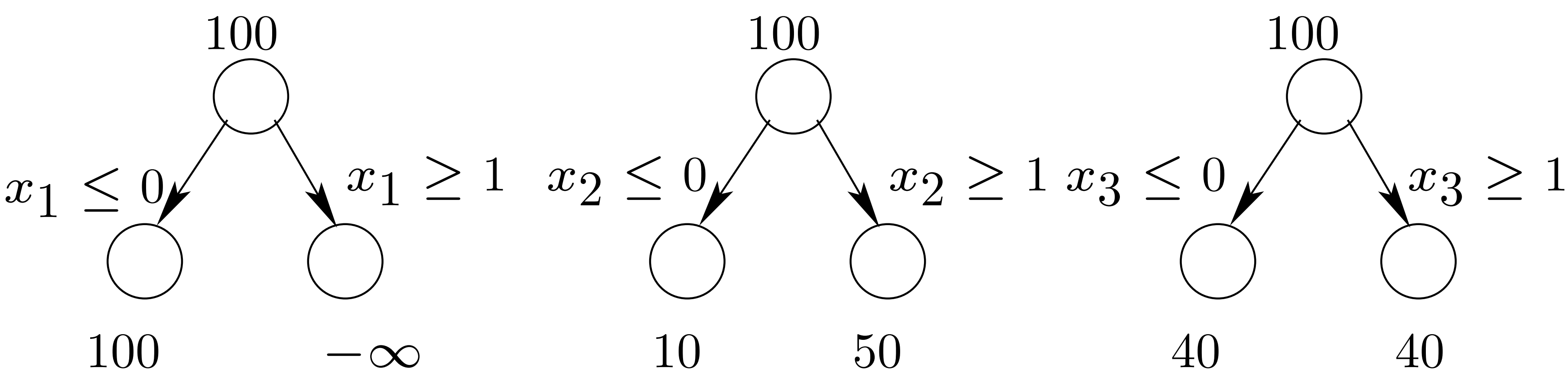
- So far we have seen, how to evaluate a candidate in several ways.
- Sometimes the choice of candidate is clear after this evaluation.



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

Comparing Candidates

- However, choice of candidates is not always clear.
- Consider



- 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.

Estimating: 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 (called *pre-solving*).
  - Use this as an estimate.
- Empirically, this reduces number of nodes, but this must be traded against the computational expense.

Estimating: 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:

$$
\begin{aligned}
D_j^+ &= P_j^+(1 - f_j) \\
D_j^- &= P_j^- f_j,
\end{aligned}
$$

  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.

Estimating: 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*.

Estimating: 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 η number of times on a variable, only use pseudo-costs after that.
  - η is called reliability parameter.
  - What does $\eta = 0$ imply? What does $\eta = \infty$ imply?
  - Empirically $\eta = 4$ seems to be quite effective.

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, then we obtain a valid inequality!
- This is in some sense what a valid inequality is.
- For the problem in Figure 1, 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.

Interpreting Branching in the Dual

- An alternative way of viewing branch and bound is simply as a method of iteratively refining a single overall disjunction (or dual function).
- Recall the dual function arising from the branch-and-bound tree (again, we are minimizing now).

$$\underline{\phi}^T_{\mathrm{LP}}(\beta) = \min_{t\in T} \underline{\phi}^t_{\mathrm{LP}}(\beta) = \min_{t\in T}\{\hat{\pi}^t\beta + \hat{\underline{\pi}}^t l^t + \hat{\bar{\pi}}^t u^t\} \qquad \text{(BB.D)}$$

where $(\hat{\pi}^t, \hat{\underline{\pi}}^t, \hat{\bar{\pi}}^t)$ is an optimal solution to the following dual at node t.

$$\begin{aligned} \phi^t(b) = \max\ & \pi^t b + \underline{\pi}^t l^t + \bar{\pi}^t u^t \\ \text{s.t. } & \pi^t A + \underline{\pi}^t + \bar{\pi}^t \leq c^\top \\ & \underline{\pi} \geq 0, \bar{\pi} \leq 0 \end{aligned} \qquad \text{(BB.LP.D)}$$

- When we branch, we remove one linear function from the above minimum and replace it with the minimum of two others.
- Depending on how we choose the disjunction, this will hopefully improve the bound yielded by the dual function.

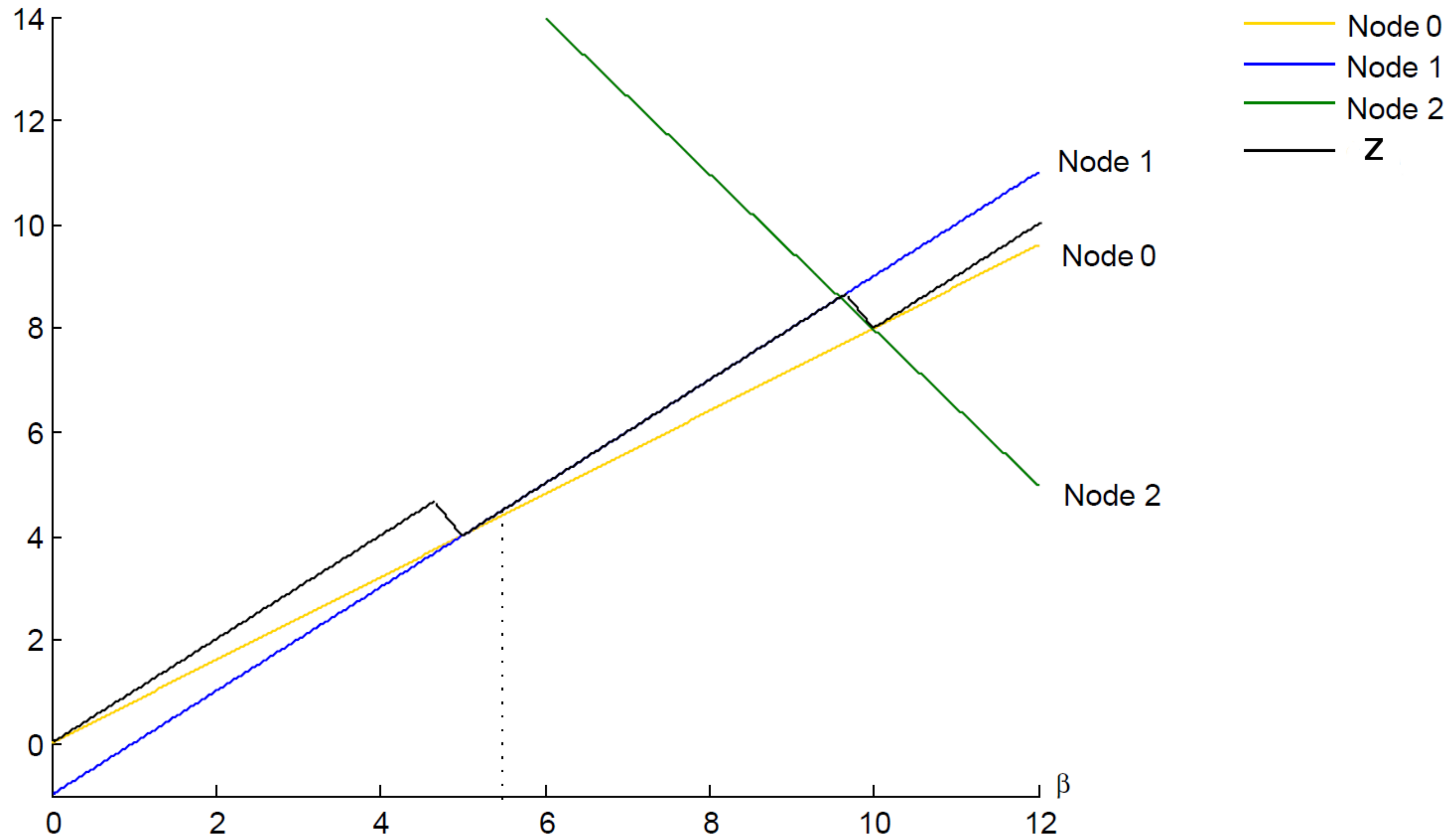
Example: Branching as Dual Improvement

- Recall again the following value function associated with an MILP from the last lecture.

$$\begin{aligned}\phi(\beta) = \min\ & 6x_1 + 4x_2 + 3x_3 + 4x_4 + 5x_5 + 7x_6 \\ \text{s.t. } & 2x_1 + 5x_2 - 2x_3 - 2x_4 + 5x_5 + 5x_6 = \beta \\ & x_1, x_2, x_3 \in \mathbb{Z}_+, x_4, x_5, x_6 \in \mathbb{R}_+.\end{aligned}$$

- Suppose we again evaluate $\phi(5.5)$ by solving the instance with fixed right-hand side by branch-and-bound.
- Solving the root LP relaxation, we obtain a solution in which $x_2 = 1.1$ and the optimal dual multipler for the single constraint is $c_2/a_2 = 4/5 = 0.8$.
- Branching on variable x_2 effectively replaces the single linear under-estimator with one that is the max of two affine functions.
- The branching can thus be seen as a kind of improvement to the solution to the MILP dual being constructed by the algorithm.

Example: Visualizing Branching as Dual Improvement



Ensuring Finite Convergence

- For LP-based branch and bound, ensuring convergence requires a convergent branching method.
- Roughly speaking, a convergent branching method is one which will
  - produce a violated admissible disjunction whenever the solution to the bounding problem is infeasible; and
  - if applied recursively, guarantees that at some finite depth, any resulting bounding problem will either
    - produce a feasible solution (to the original MILP); or
    - be proven infeasible; or
    - be pruned by bound.
- Typically, we achieve this by ensuring that at some finite depth, we have a complete description of the convex hull of solutions to the subproblem.
- This is always the case if we branch on variable disjunctions and $\mathcal{S}$ is bounded.
- We will also revisit this result more formally as we develop the supporting theory.