Computational Integer Programming
Universidad de los Andes

Lecture 4

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)

Branch and Bound Recap

- Suppose F is the feasible region for some MILP and we wish to solve $\max_{x \in F} c^\top x$.
- Consider a partition of F into subsets $F_1, \ldots F_k$. Then

$$\max_{x \in F} c^\top x = \max_{1 \le i \le k} \max_{x \in F_i} c^\top x.$$

- In other words, we optimize over each subset separately.
- Dividing the original problem into subproblems is called *branching*.

Branching

- We have now discussed several basic methods for bounding.
- Obtaining tight bounds is the most important aspect of the branch-and-bound algorithm.
- Branching effectively is a very close second.
- In fact, methods for branching and bounding are more closely related than it might initially appear.
- This will be a theme in the remaining lectures.
- Choosing an effective method of branching can make orders of magnitude difference in the size of the search tree and the solution time.

Some Definitions

- Let us consider a pure IP with feasible set $S \subseteq \mathbb{Z}^n$.
- A *branching* (used as a noun) is a division of the original feasible set S into subsets $S_1, \ldots, S_r$.
- A branching is *valid* if
  - $\cup_{i=1}^r S_i = S$, and
  - It is possible to describe and optimize over each set S_i.
- Note that there are branching schemes that are not valid by this definition, but that still lead to correct algorithms.
- A branching is called *partitive* if the sets S_i are disjoint.
- It is desirable for a branching to be partitive for obvious reasons.

Branching on Hyperplanes

- Branchings are derived from logical disjunctions.
- The most easily handled disjunctions are those derived from an integer vector $\pi \in \mathbb{Z}^n$.
- Given $\pi \in \mathbb{Z}^n$ and $\nu \in \mathbb{Z}$, the disjunction

$$\pi x \geq \nu \text{ OR } \pi x \leq \nu - 1, \tag{1}$$

  is always a valid disjunction.
- The vector π could be the left-hand side of a valid inequality or may be derived in some other fashion.
- This disjunction defines a valid branching in the obvious way.

Branching on Variables

- Note that the bound constraints are valid inequalities and can be used to derive disjunctions.
- If we branch on the bound constraint of variable x_j, we simply say we are *branching on* x_j.
- 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

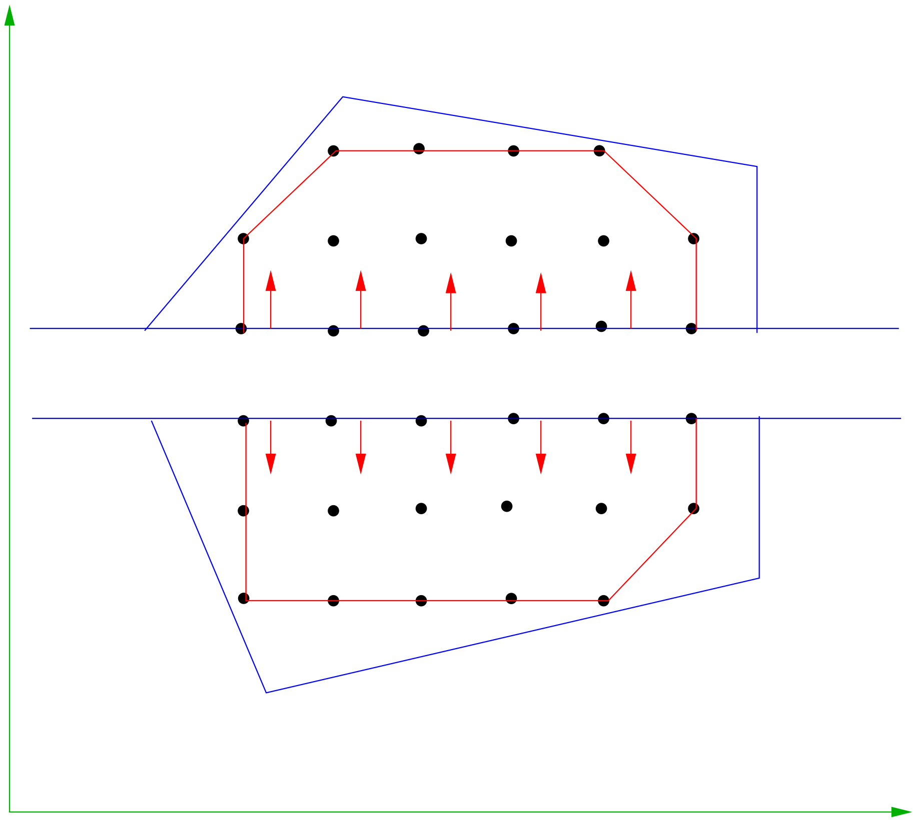


Figure 1: Branching on a variable

The Geometry of Branching

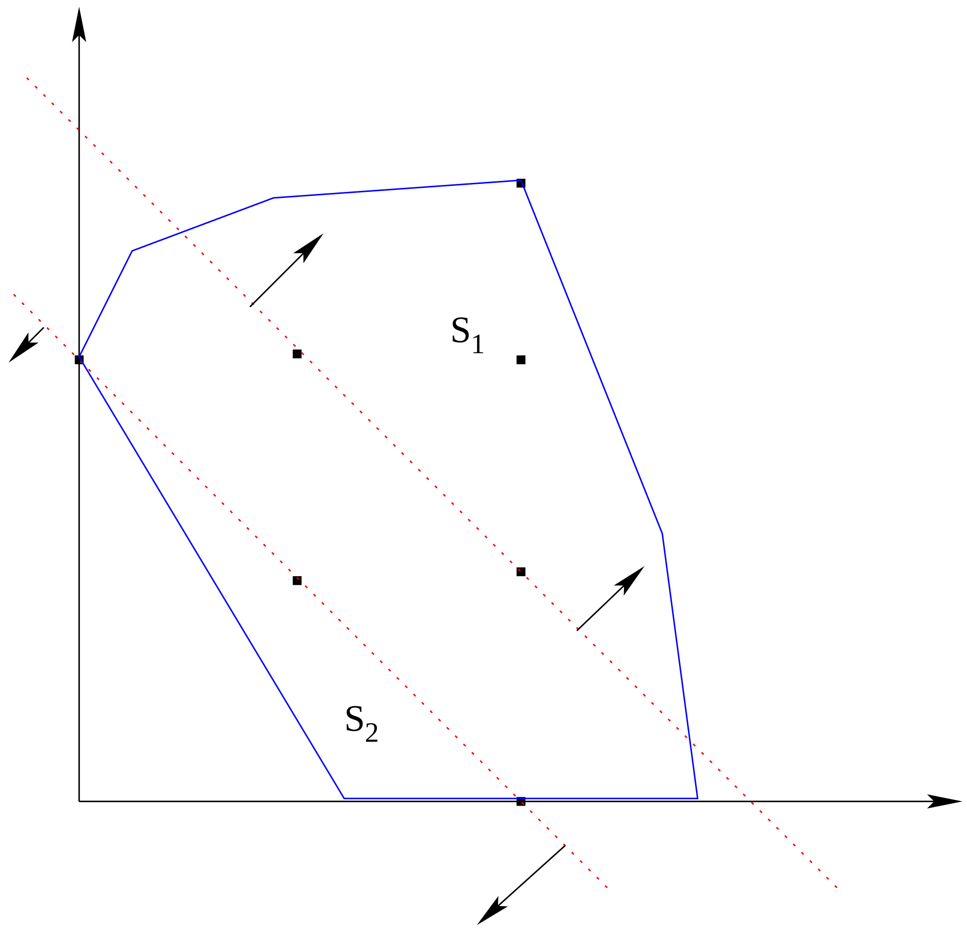


Figure 2: Branching on a hyperplane

Bad Example

$$
\begin{aligned}
&\min 2x_1 + 2x_2 + 2x_3 - 3x_4 \\
&\text{s.t.} \\
&2x_1 + x_2 + x_3 - 2x_4 - 2r_1 = 0.1 \\
&x_1 + 2x_2 + x_3 - 2x_4 - 2r_2 = 0.1 \\
&x_1 + x_2 + 2x_3 - 2x_4 - 2r_3 = 0.1 \\
&2x_1 + 2x_2 + 2x_3 \geq 0.1 \\
&0 \leq r_1 \leq 0.90 \\
&0 \leq r_2 \leq 0.90 \\
&0 \leq r_3 \leq 0.90 \\
&x_1, x_2, x_3, x_4 \in \mathbb{Z}_+.
\end{aligned} \tag{2}
$$

Branch and Bound Tree

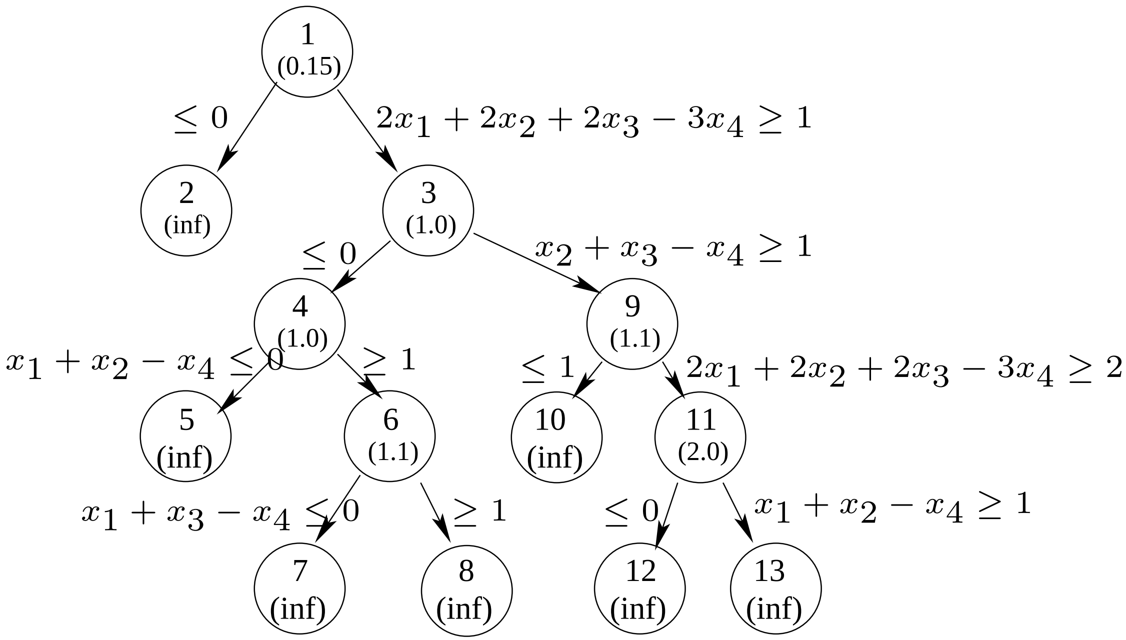


Figure 3: A branch-and-bound tree to prove the infeasibility of problem (2).

- Branching on variables produces a tree with many millions of nodes.
- Infeasibility is proved easily using general hyperplanes.
- In general, however, it is difficult to determine what are the "good hyperplanes" to branch on.
- This is the subject of active research.

Branching on Special Hyperplanes

- A *generalized upper bound* (GUB) constraint 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?

Multiple Disjunctions

- Multi-way branching is to create more than two children, such as when simultaneously branching on multiple valid inequalities.
- In this case, we need to consider all possible combinations.
- Example:
  - Suppose x_i and x_j are 0-1 variables.
  - If we branch on x_i and x_j simultaneously, we get four subproblems

$$x_i = 0,\ x_j = 0$$

$$x_i = 0,\ x_j = 1$$

$$x_i = 1,\ x_j = 0$$

$$x_i = 1,\ x_j = 1$$

  - Often, logical considerations can eliminate one or more of the subproblems.

Branching on Other Types of 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.

Model-based Branching

- In some cases, the structure of the model may make it clear that branching on some variables will be more effective than on others.
- There may also be branching rules that come from the structure of the problem.
- You may want to try to branch on variables for which fixing their values will have a large impact.
- Example: Location and routing problems
- In such a case, you can use *branching priorities* to reflect this or implement a custom branching rule.

Goal of Branching

- 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 overall, so we must use more myopic criteria.
  - Decrease the upper bound,
  - Increase the lower bound, or
  - Produce one or more (nearly) infeasible subproblems.
- Most of the time, we will want to focus on decreasing the upper bound. (why?)

Choosing a Disjunction

- There are many possible disjunctions to choose from.
- In fact, the choice of disjunction is itself an optimization problem that can be analyzed.
- In order to limit the choice to small set, most solvers branch on variables by default.
- This still leaves the question of what variable to choose.
- An intuitive choice is to branch on the most fractional variable ($\text{argmin}\{|0.5 - f_i|\}$)?
- This does not work well in practice.
- We need a measure for comparing different choices.

Pseudocost Branching

- For now, we focus on branching on variables.
- In order to decide which variable to branch on, we would like to estimate the effect of the branching.
- In other words, we would like to know how much the bound in each of the children will be decreased over that of their parent.
- The pseudocost of a variable is an estimate derived by averaging observed changes resulting from branching on each of the variables.
- For each variables, 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:

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

Pseudocost Initialization

- An important question is how to get initial estimates before any branching has occurred.
- This can be done using *strong branching*.
- After initialization, we update pseudocost after each bounding operation.
- Empirical evidence shows that these pseudocosts are roughly constant throughout the branch-and-bound tree.

Strong Branching

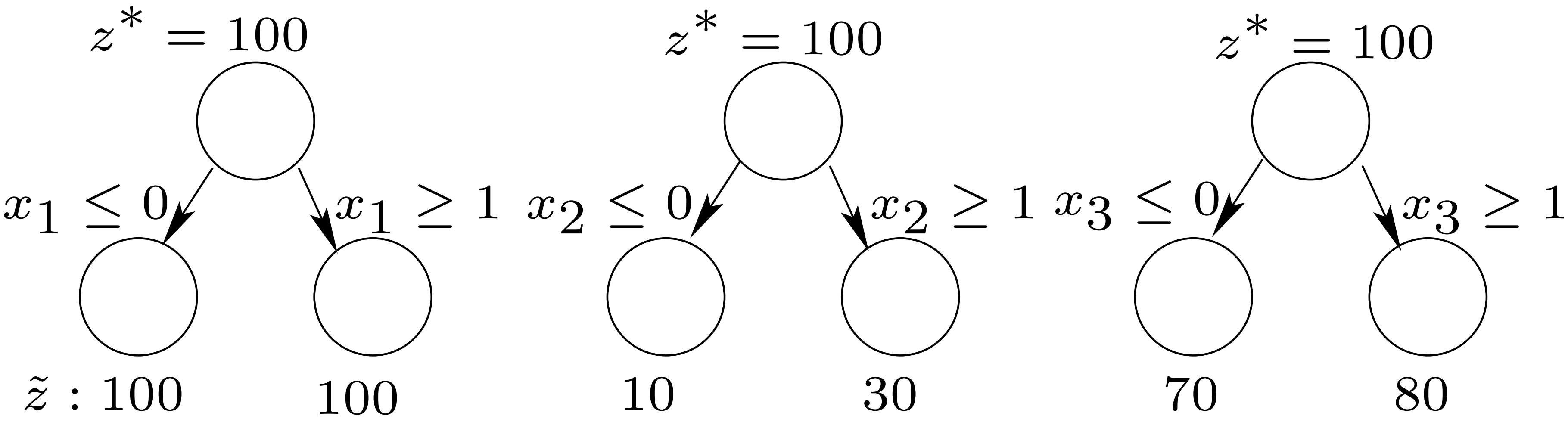
- Strong branching provides a more accurate estimate, but is computationally very expensive.
- Branching decisions near the top of the tree are the most important. (why?)
- Unfortunately, pseudocost branching is least effective at this stage.
- What can we do?
  - Evaluate each branching candidate by "pre-solving" the subproblem resulting from each candidate.
  - Branch on the "best" observed candidate.
- Can be costly. How to moderate this down?
  - Do only a limited number of dual-simplex pivots for each candidate for each child.
  - Use this estimate to choose the candidate.
- Empirically, this reduces number of nodes, but this must be traded against the computational expense.

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.

Comparing Branching Candidates

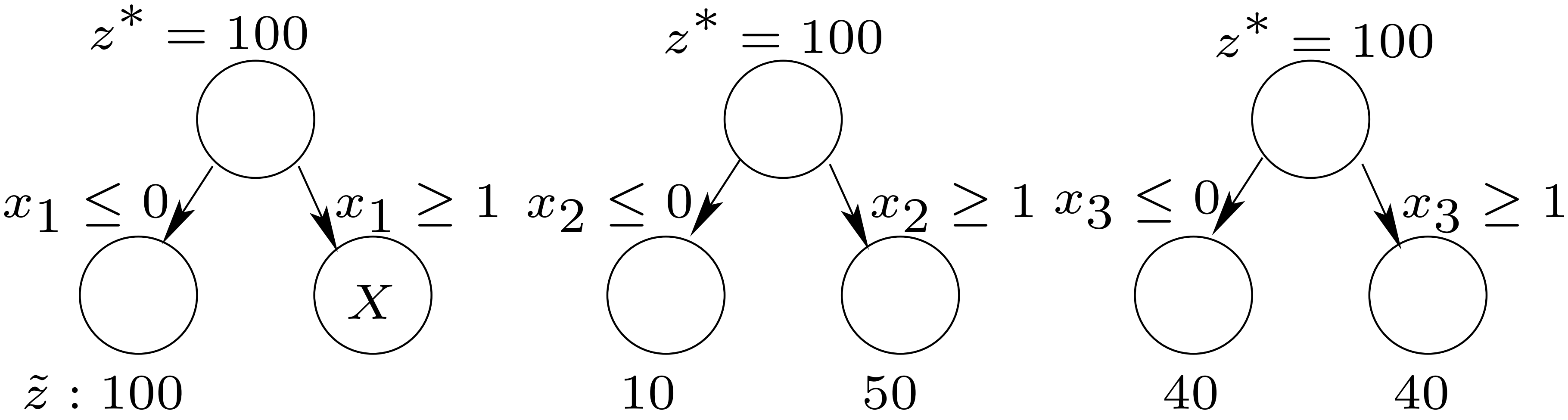
- So far we have seen how to evaluate each child node in several ways.
- Sometimes the choice of candidate is clear after this evaluation.



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

Comparing Candidates (cont.)

- However, the choice may not always be 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 Branching Candidates (cont.)

- The fractionality of the variables (after strong branching) may also be taken into account.
- Criteria based on the structure of constraints are discussed in *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.