Integer cuts - Example

Example 2.3: Build the integer cut for $\tilde{\boldsymbol{y}} = (1,0,0,1,1)^T$

$$y_1 + y_4 + y_5 - y_2 - y_3 \leq 2 \quad \text{(IC)}$$

Show that Eq. (IC) is infeasible for $\tilde{\boldsymbol{y}} = (1,0,0,1,1)^T$

Pick any other combination of $\boldsymbol{y}$ and show that Eq. (IC) is feasible

Eq. (IC) excludes $\tilde{\boldsymbol{y}}$ only

SOLVING PROBLEMS WITH BINARY VARIABLES

Mixed Integer Programming (MIP)

$$\begin{aligned} \min_{\boldsymbol{x},\boldsymbol{y}} \quad & f(\boldsymbol{x},\boldsymbol{y}) \\ \text{s.t.} \quad & h(\boldsymbol{x},\boldsymbol{y}) = 0 \\ & g(\boldsymbol{x},\boldsymbol{y}) \leq 0 \\ & \boldsymbol{x} \in \mathbb{R}^n \\ & \boldsymbol{y} \in \{0,1\}^q \end{aligned} \qquad (MIP)$$

- $\boldsymbol{x}$**:** variable vector represents the continuous decisions (flowrates, equipment sizes, pressure, temperature, heat duties)
- $\boldsymbol{y}$**:** binary variables represent the existence or non-existence of process units.

Mixed Integer Linear Programming (MILP)

$$\begin{aligned} \min_{\boldsymbol{x},\boldsymbol{y}} \quad & c_x^T\boldsymbol{x} + c_y^T\boldsymbol{y} \\ \text{s.t.} \quad & A\boldsymbol{x} + B\boldsymbol{y} \leq 0 \\ & \boldsymbol{x} \geq 0 \\ & \boldsymbol{y} \in \{0,1\}^q \end{aligned} \qquad (MILP)$$

- Commonly used in planning and scheduling problems, assignment problems.
- Several efficient MILP algorithms exist (e.g., CPLEX, XPRESS, GUROBI)
  - can solve problems with millions of binary variables.
  - are guaranteed to identify the best solution (if enough time and memory are provided).

MILP - Brute Force Approach

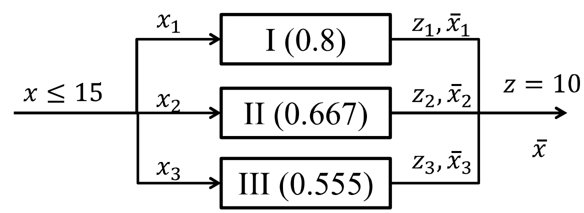
- Based on the complete enumeration of all combinations of the binary variables.
- When binary variables are fixed → LP problem
  - can be solved through Simplex or Interior-Point methods
  - global solution can be found by comparing solutions of LPs
- 2^q combinations to be tested
- Combinatorial explosion:

# of binaries: q	2	5	10	20	50	100
Combinations: 2^q	4	32	1024	10^6	10^{15}	10^{30}

➢ Considering 50 binary variables, and assuming that it takes only 10 ms to solve each LP, it would take 31 years to try all combinations

MILP - Brute Force Approach

Example 2.4: Reactor Selection (Pistikopoulos, 2016)



Reactor	Cost
I	$80 + 35x_1$
II	$54 + 30x_2$
III	$27 + 25x_3$

$$\begin{aligned}
\min_{x,y}\quad & 80y_1 + 35x_1 + 54y_2 + 30x_2 + 27y_3 + 25x_3 \\
\text{s.t.}\quad & 0.8x_1 + 0.667x_2 + 0.555x_3 = 10 \\
& x_1 + x_2 + x_3 \le 15 \\
& x_1 - 15y_1 \le 0,\quad x_2 - 15y_2 \le 0,\quad x_3 - 15y_3 \le 0 \\
& x_1, x_2, x_3 \ge 0,\ y_1, y_2, y_3 \in \{0,1\}
\end{aligned}$$

MILP - Brute Force Approach

Example: Reactors Selection → Complete Enumeration

– Three 0-1 variables → $2^3 = 8$ combinations

y_1	y_2	y_3	Solution
0	0	0	Infeasible
1	0	0	
0	1	0	
0	0	1	Infeasible
1	1	0	
1	0	1	
0	1	1	
1	1	1	

- Many 0-1 combinations
- Some combinations infeasible – not required

MILP - Relaxation and Rounding Approach

LP relaxation

- Relax MILP by removing integrality condition on $\mathbf{y}$ variables
  - $\mathbf{y}$ vary continuously between 0 and 1
  - resulting problem is an LP
- Solution of relaxed problem cannot be greater than solution of original MILP
- In some cases solution of LP is equal to that of MILP
  - the matrix B in problem $(MILP)$ must be unimodular (i.e., every square non-singular matrix of B has a determinant equal to 1).
- In general unstructured MILPs:
  - at solution of relaxed LP some y variables will be non-integer
  - usually the case in process synthesis.

MILP - Relaxation and Rounding Approach
Rounding scheme

- General case: solution of relaxed problem non-integer
  - apply a rounding scheme → round solution to the nearest integer.
  - may result in sub-optimal solution, or infeasible combination

<u>Example</u>:

$$\min_{y} \; z = -1.2y_1 - y_2$$
$$\text{s.t.} \quad y_1 + y_2 = 1$$
$$1.2y_1 + 0.5y_2 \le 1$$
$$(y_1, y_2) \in \{0,1\}^2$$

<u>Relaxed LP</u>: $(0 \le y_1, y_2 \le 1) : \begin{cases} y_1 = 0.715 \\ y_2 = 0.285 \end{cases} \quad z = -1.148$

<u>Rounding</u>: $\begin{matrix} y_1 = 1 \\ y_2 = 0 \end{matrix} : \begin{matrix} 1.2y_1 + 0.5y_2 \le 1 \\ \to 1.2 \cdot 1 + 0.5 \cdot 0 \le 1 \\ \to 1.2 \le 1 \end{matrix}$ (Infeasible)

<u>Optimal</u>: $\begin{matrix} y_1 = 0 \\ y_2 = 1 \end{matrix} \quad (z = -1.0)$

Note: $z_{\text{MILP}} > z_{\text{LP}}$

MILP - Branch-and-Bound (B&B) Techniques

<u>Main idea</u>:

- Use a divide and conquer approach to decision making
- Generate partial solutions to the problem
- Eliminate unpromising regions of the solution space

<u>Remarks</u>:

- Exhaustive enumeration of all 0-1 combinations of brute-force approach can be avoided
- Guaranteed to find the solution.

LP relaxations

- Subproblem P_i: LP relaxation of problem $(MILP)$:
  - binary variables in some set J_i have been fixed $(\dim\{J_i\} < q)$
  - the rest are allowed to vary between 0 and 1.

$$\begin{aligned} \min_{x,y} \quad & c_x^T \boldsymbol{x} + c_y^T \boldsymbol{y} \\ \text{s.t.} \quad & A\boldsymbol{x} + B\boldsymbol{y} \le d \\ & \boldsymbol{x} \ge 0, 0 \le \boldsymbol{y} \le 1 \\ & y_k \ \text{fixed}, k \in J_i \end{aligned} \qquad (P_i)$$

- Consider the relaxed subproblem (P_j) derived from problem (P_i):
  - set of fixed variables J_j contains set J_i → $J_i \subset J_j$ $(\dim\{J_i\} \le \dim\{J_j\} \le q)$

$$\begin{aligned} \min_{x,y} \quad & c_x^T \boldsymbol{x} + c_y^T \boldsymbol{y} \\ \text{s.t.} \quad & A\boldsymbol{x} + B\boldsymbol{y} \le d \\ & \boldsymbol{x} \ge 0, 0 \le \boldsymbol{y} \le 1 \\ & y_k \ \text{fixed}, k \in J_j \end{aligned} \qquad (P_j)$$

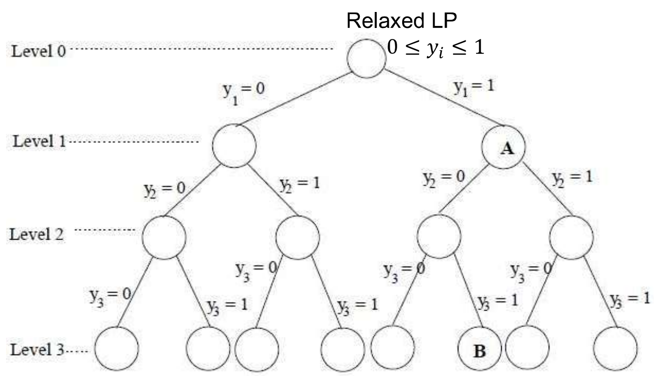
Properties of LP relaxations

- Let f^*, f_i^* and f_j^* be objective function values at the solution of problems $(MILP)$, (P_i) and (P_j), respectively.
- Properties:
  - If (P_i) infeasible → (P_j) infeasible
  - If(P_j) feasible →(P_i) feasible
  - If(P_j) feasible → $f_j^* \ge f_i^*$
  - If $\boldsymbol{y}$ is integer at the solution of (P_j) → $f_j^* \ge f^*$

At feasible point, objective function is always greater than or equal to f^*

- B&B algorithms use above properties to explore solution space efficiently.

A branch-and-bound tree



# of end-nodes: 2^q

# of nodes in Tree: $2^{q+1} - 1$

but usually not all nodes tested

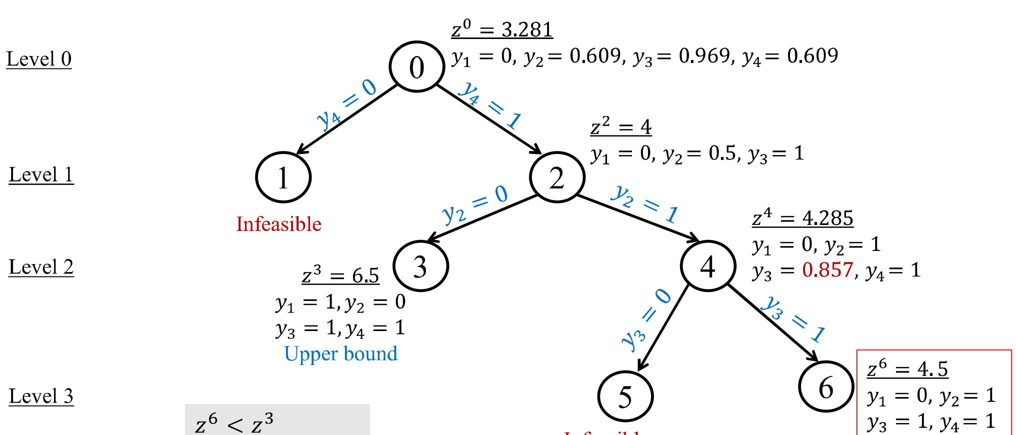
Branch-and-bound tree – example

Example 2.5

$$\min_{\boldsymbol{y}} \quad z = 3y_1 + y_2 + 1.5y_3 + 2y_4$$
$$s.t. \; 5y_1 + 4y_2 + 3y_3 + 6y_4 \ge 9$$
$$2y_1 + y_2 + 4y_3 + 2y_4 \ge 3$$
$$y_2 - y_4 \le 0$$
$$y_1 + 2y_2 + 7y_3 \ge 8$$
$$y_1, y_2, y_3, y_4 \in \{0,1\}^4$$

Solve the MILP using the best bound rule and branching on the most fractional variable

Example 2.5 – solution (end!)



$z^6 < z^3$

z^6=4.5 optimal

Branch and bound – pseudo code 1/3

Iteration 1

Step 1: Original MILP problem (P_0)

- Initialise upper-lower bounds on the solution → $U = +\infty,\ L = -\infty$
- Set iteration counter $k = 0$
- Initialise the list of subproblems → $\mathcal{L} = \emptyset$
- Set convergence tolerance, ϵ

Step 2: Solve an LP relaxation of (P_0) → (RP_0)

- If (RP_0) infeasible → (P_0) infeasible → terminate
- Else f^0 solution of (RP_0)
- If all binary vars are integer at f^0 → optimal solution of (P_0) found → terminate
- otherwise, f^0 a lower bound on problem (P_0) → $L = f^0$ and set $k = 1$

Branch-and-bound – pseudo code 2/3

Iteration 2

Step 3: Select binary variable for branching

- Select one binary variable, y^1
- Create 2 children nodes: $(P_{k,0})$, $(P_{k,1})$ as subproblems of (P_0)
  - nodes $(P_{k,0})$ → $y^1 = 0$ and $(P_{k,1})$ → $y^1 = 1$
  - no other binary variables are fixed at these nodes

Step 4: Solve LP relaxations of $(P_{k,0})$ and $(P_{k,1})$ → $(RP_{k,0})$ and $(RP_{k,1})$

- If LP infeasible move to next subproblem
- Two new tighter lower bounds, $f^{k,0}$ and $f^{k,1}$
  - smallest known lower bound on the problem is equal to $\min\{f^{k,0}, f^{k,1}\}$
- If all binary vars integer → upper bound → $U = f^{k,i}$ → move to next subproblem
- If a lower bound greater than U → fathom node (cannot lead to optimal solution)
- Add nodes with lower bound less than U to list $\mathcal{L}$

Branch-and-bound – pseudo code 3/3

Step 5: Problem (P_{k+1})

- Remove problem with lowest LB from $\mathcal{L}$
- Set $L = f^{k+1}$

Step 6

- If $U - L \le \epsilon$ → terminate → Solution is U
- Otherwise, set $k = k + 1$ and return to step 3

Select one node from the list

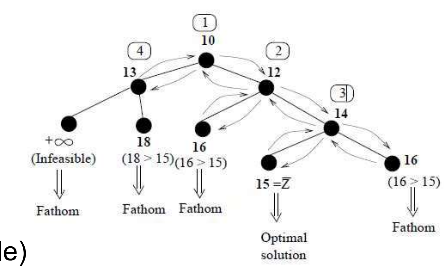
- fix another binary variable → create 2 new nodes
- solve subproblems - generate two new tighter lower bounds
- update the upper bound
- test for convergence
- eliminate any node with lower bound greater than upper bound
- add remaining nodes to the list to be analysed further

Branch-and-bound – Branching options

- Select one or more binary variables to be fixed at the new children nodes
  - most algorithms only fix one
  - ideally, fix variable that leads to the greatest increase of lower bound → speed up convergence
  - difficult to detect such a variable *a priori*
- Some simple branching strategies:
  - choose a variable randomly, or choose 1st variable not fixed yet from the list
  - choose most-fractional variable,
    - binary variable closest to 0.5 when solving the relaxation of the parent
    - e.g., if the solution at the root node is $y = (1{,}0.8{,}0.5)^T$ → choose y_3
  - choose the variable with the greatest sensitivity

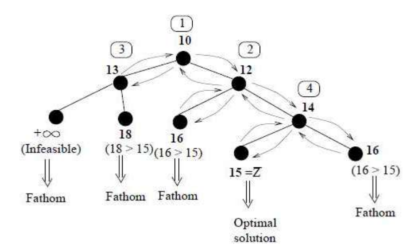
Branch-and-bound – Node selection rules

- Newest bound rule (depth first, LIFO)
  →expand the most recently created node
  - capable of generating a large number of feasible solutions
  - relatively small storage requirements



- Best bound rule ("breadth first", priority rule)
  →expand node with the smallest lower bound
  - relatively large storage requirement

➢ Combinations of these rules (and many others) are used in commercial codes



Performance of B&B algorithms

Worst-case performance

Maximum number of nodes explored in a B&B tree: $2^{q+1} - 1$

- greater than the number of combinations
- no guarantee B&B algorithm will terminate before entire tree explored
- worst-case behaviour is rare

Efficient solution of subproblems

B&B requires solution of similar LP problems

- information obtained from solving parent node can speed up solutions at children nodes
- increase efficiency of algorithm

Importance of formulation

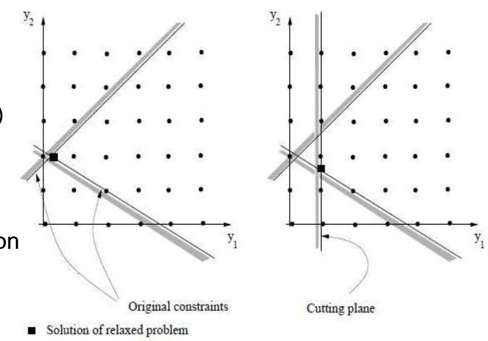
- Formulation can have a great impact on performance
- Make formulation as tight as possible
  - if big-M used → M as small as possible
- Numbers of variables/constraints affect computational requirements
  - Greatest impact → # of binary var.
  - 2nd → # of constraints
  - 3rd → # of continuous var.

MILP - Cutting-Plane Algorithms

- Start from an LP relaxation of original problem.
- Add (linear) constraints
  - relaxation successively tightened - narrows down the feasible region
- Apply a sufficient number of cutting planes
  - solution of relaxed problem becomes integer - equal to solution of original problem

Process for a two-variable *integer* problem

1. feasible region delimited by original constraints
2. only integer combinations y_1 and y_2 (black dots)
3. non-integer solution for the relaxed LP
4. solution satisfies original constraints apart from integrality
5. impose another constraint → removes 1st solution from feasible region
6. solve → 2nd non-integer solution
7. continue until an integer solution is found



Pros and cons of cutting plane approaches

- No feasible solution can be identified before finding optimal solution
- Difficult to prove finite convergence
  - if solution procedure is very slow → may end up with no solution
  - B&B can usually identify at least one feasible solution in reasonable time
- No single strategy to construct cutting planes
- Cutting plane algorithms not as popular as B&B
  - possible to combine both approaches
  - Branch-and-cut algorithms

Summary – Lecture 2

- Superstructures can be used to represent multiple choices
- Logical choices can be modelled via binary variables and algebraic constraints
- Branch-and-bound approaches can be used to solve MILPs