Mixed Integer Linear Programming

Combinatorial Problem Solving (CPS)

Javier Larrosa  Albert Oliveras  Enric Rodríguez-Carbonell

May 8, 2020
A mixed integer linear program (MILP, MIP) is of the form

\[
\begin{align*}
\min & \quad c^T x \\
Ax & = b \\
x & \geq 0 \\
x_i & \in \mathbb{Z} \quad \forall i \in I
\end{align*}
\]

If all variables need to be integer, it is called a (pure) integer linear program (ILP, IP).

If all variables need to be 0 or 1 (binary, boolean), it is called a 0–1 linear program.
Complexity: LP vs. IP

- Including integer variables increases enormously the modeling power, at the expense of more complexity.

- LP’s can be solved in polynomial time with interior-point methods (ellipsoid method, Karmarkar’s algorithm).

- Integer Programming is an NP-complete problem. So:
  - There is no known polynomial-time algorithm.
  - There are little chances that one will ever be found.
  - Even small problems may be hard to solve.

- What follows is one of the many approaches (and one of the most successful) for attacking IP’s.
Given a MIP

\[
\begin{align*}
\text{(IP)} & \quad \min c^T x \\
& \quad Ax = b \\
& \quad x \geq 0 \\
& \quad x_i \in \mathbb{Z} \quad \forall i \in \mathcal{I}
\end{align*}
\]

its \textbf{linear relaxation} is the LP obtained by dropping integrality constraints:

\[
\begin{align*}
\text{(LP)} & \quad \min c^T x \\
& \quad Ax = b \\
& \quad x \geq 0
\end{align*}
\]

Can we solve \textit{IP} by solving \textit{LP}? By rounding?
The optimal solution of

\[
\begin{align*}
\max & \quad x + y \\
-2x + 2y & \geq 1 \\
-8x + 10y & \leq 13 \\
x, y & \geq 0 \\
x, y & \in \mathbb{Z}
\end{align*}
\]

is \((x, y) = (1, 2)\), with objective 3

The optimal solution of its LP relaxation
is \((x, y) = (4, 4.5)\), with objective 9.5

No direct way of getting from \((4, 4.5)\) to \((1, 2)\) by rounding!

Something more elaborate is needed: branch & bound
\[ y \geq 0 \]
\[ -8x + 10y \leq 13 \]
\[ x \geq 0 \]
\[ -2x + 2y \geq 1 \]
\[ (0, 1) \]
\[ (1, 2) \]
\[ (4, 4.5) \]
\[ \max x + y \]
Branch & Bound

- Assume variables are bounded, i.e., have lower and upper bounds
- Let $P_0$ be the initial problem, $LP(P_0)$ be the LP relaxation of $P_0$
- If in optimal solution of $LP(P_0)$ all integer variables take integer values then it is also an optimal solution to $P_0$
- Else
  - Let $x_j$ be integer variable whose value $\beta_j$ at optimal solution of $LP(P_0)$ is such that $\beta_j \notin \mathbb{Z}$.
    - Define
      \[
      P_1 := P_0 \land x_j \leq \lfloor \beta_j \rfloor \\
      P_2 := P_0 \land x_j \geq \lceil \beta_j \rceil 
      \]
  - feasibleSols($P_0$) = feasibleSols($P_1$) $\cup$ feasibleSols($P_2$)
  - Idea: solve $P_1$, solve $P_2$ and then take the best
Let $x_j$ be integer variable whose value $\beta_j$ at optimal solution of $LP(P_0)$ is such that $\beta_j \notin \mathbb{Z}$.

Each of the problems

$$P_1 := P_0 \land x_j \leq \lfloor \beta_j \rfloor \quad P_2 := P_0 \land x_j \geq \lceil \beta_j \rceil$$

can be solved recursively

- We can build a binary tree of subproblems whose leaves correspond to pending problems still to be solved
- This procedure terminates as integer vars have finite bounds and, at each split, the domain of $x_j$ becomes strictly smaller
- If $LP(P_i)$ has optimal solution where integer variables take integer values then solution is stored
- If $LP(P_i)$ is infeasible then $P_i$ can be discarded (pruned, fathomed)
Example

\[
\begin{align*}
\text{min} & \quad -x - y \\
-2x + 2y & \geq 1 \\
-8x + 10y & \leq 13 \\
x, y & \geq 0 \\
x, y & \in \mathbb{Z}
\end{align*}
\]
Example

\[
\begin{align*}
\text{min} & \quad -x - y \\
-2x + 2y & \geq 1 \\
-8x + 10y & \leq 13 \\
x, y & \geq 0 \\
x, y & \in \mathbb{Z}
\end{align*}
\]
Example

Min obj: \(- x - y\)
Subject To
c1: \(-2 x + 2 y \geq 1\)
c2: \(-8 x + 10 y \leq 13\)
End

CPLEX> optimize
Primal simplex - Optimal: Objective = \(-8.5000000000e+00\)
Solution time = 0.00 sec. Iterations = 0 (0)
Deterministic time = 0.00 ticks (0.37 ticks/sec)

CPLEX> display solution variables x
Variable Name     Solution Value
x                  4.000000

CPLEX> display solution variables y
Variable Name     Solution Value
y                  4.500000
Example

\[
\begin{align*}
\text{min} & \quad -x - y \\
-2x + 2y & \geq 1 \\
-8x + 10y & \leq 13 \\
x, y & \geq 0 \\
x, y & \in \mathbb{Z}
\end{align*}
\]
Example

\begin{align*}
\min & \quad -x - y \\
-2x + 2y & \geq 1 \\
-8x + 10y & \leq 13 \\
x, y & \geq 0 \\
x, y & \in \mathbb{Z}
\end{align*}
Example

Min obj: - x - y
Subject To
c1: -2 x + 2 y >= 1
c2: -8 x + 10 y <= 13
Bounds
y >= 5
End

====================================================================

CPLEX> optimize
Bound infeasibility column 'x'.
Presolve time = 0.00 sec. (0.00 ticks)
Presolve - Infeasible.
Solution time = 0.00 sec.
Deterministic time = 0.00 ticks (1.67 ticks/sec)
Example

\[
\begin{align*}
\text{min} & \quad -x - y \\
-2x + 2y & \geq 1 \\
-8x + 10y & \leq 13 \\
x, y & \geq 0 \\
x, y & \in \mathbb{Z}
\end{align*}
\]
Example

\[
\begin{align*}
\text{min } & -x - y \\
& -2x + 2y \geq 1 \\
& -8x + 10y \leq 13 \\
x, y & \geq 0 \\
x, y & \in \mathbb{Z}
\end{align*}
\]
Example

Min obj: \( -x - y \)
Subject To
\[ c1: -2x + 2y \geq 1 \]
\[ c2: -8x + 10y \leq 13 \]
Bounds
\( y \leq 4 \)
End

==================================

CPLEX> optimize
Dual simplex - Optimal: Objective = \(-7.5000000000e+00\)
Solution time = 0.00 sec. Iterations = 0 (0)
Deterministic time = 0.00 ticks (2.68 ticks/sec)

CPLEX> display solution variables x
Variable Name        Solution Value
x                    3.500000

CPLEX> display solution variables y
Variable Name        Solution Value
y                    4.000000
Example

\[
\begin{align*}
\text{min} & \quad -x - y \\
-2x + 2y & \geq 1 \\
-8x + 10y & \leq 13 \\
x, y & \geq 0 \\
x, y & \in \mathbb{Z}
\end{align*}
\]
Example

\[\begin{align*}
\min & \quad -x - y \\
-2x + 2y & \geq 1 \\
-8x + 10y & \leq 13 \\
x, y & \geq 0 \\
x, y & \in \mathbb{Z}
\end{align*}\]
Example

Min obj: \(-x - y\)
Subject To
\[c1: \ -2x + 2y \geq 1\]
\[c2: \ -8x + 10y \leq 13\]
Bounds
\[x \geq 4\]
\[y \leq 4\]
End

=================================

CPLEX> optimize
Row 'c1' infeasible, all entries at implied bounds.
Presolve time = 0.00 sec. (0.00 ticks)
Presolve - Infeasible.
Solution time = 0.00 sec.
Deterministic time = 0.00 ticks (1.11 ticks/sec)
Example

\[
\begin{align*}
\text{min} & \quad -x - y \\
-2x + 2y & \geq 1 \\
-8x + 10y & \leq 13 \\
x, y & \geq 0 \\
x, y & \in \mathbb{Z}
\end{align*}
\]
Example

\[ \begin{align*}
\text{min} & \quad -x - y \\
-2x + 2y & \geq 1 \\
-8x + 10y & \leq 13 \\
x, y & \geq 0 \\
x, y & \in \mathbb{Z}
\end{align*} \]
Example

Min obj: \(-x - y\)
Subject To
\(c1: -2x + 2y \geq 1\)
\(c2: -8x + 10y \leq 13\)
Bounds
\(x \leq 3\)
\(y \leq 4\)
End

-----------------------------------

CPLEX> optimize
Dual simplex - Optimal: Objective = \(-6.7000000000e+00\)
Solution time = 0.00 sec. Iterations = 0 (0)
Deterministic time = 0.00 ticks (2.71 ticks/sec)

CPLEX> display solution variables x
Variable Name         Solution Value
x                      3.000000

CPLEX> display solution variables y
Variable Name         Solution Value
y                      3.700000
Example

\[
\begin{align*}
\text{min} & \quad -x - y \\
-2x + 2y & \geq 1 \\
-8x + 10y & \leq 13 \\
x, y & \geq 0 \\
x, y & \in \mathbb{Z}
\end{align*}
\]
Example

\[
\min -x - y \\
-2x + 2y \geq 1 \\
-8x + 10y \leq 13 \\
x, y \geq 0 \\
x, y \in \mathbb{Z}
\]
Example

Min obj: \(- x - y\)
Subject To
\(c1: -2 x + 2 y \geq 1\)
\(c2: -8 x + 10 y \leq 13\)
Bounds
\(x \leq 3\)
\(y = 4\)
End

===============================================

CPLEX> optimize
Bound infeasibility column ’x’.
Presolve time = 0.00 sec. (0.00 ticks)
Presolve - Infeasible.
Solution time = 0.00 sec.
Deterministic time = 0.00 ticks (1.12 ticks/sec)
Example

\[
\begin{align*}
\min & \quad -x - y \\
-2x + 2y & \geq 1 \\
-8x + 10y & \leq 13 \\
x, y & \geq 0 \\
x, y & \in \mathbb{Z}
\end{align*}
\]
Example

\[
\begin{align*}
\min \ & -x - y \\
-2x + 2y & \geq 1 \\
-8x + 10y & \leq 13 \\
x, y & \geq 0 \\
x, y & \in \mathbb{Z}
\end{align*}
\]
Example

Min obj: \(-x - y\)
Subject To
\(c1: -2x + 2y \geq 1\)
\(c2: -8x + 10y \leq 13\)
Bounds
\(x \leq 3\)
\(y \leq 3\)
End

====================================================================

CPLEX> optimize
Dual simplex - Optimal: Objective = \(-5.5000000000e+00\)
Solution time = 0.00 sec. Iterations = 0 (0)
Deterministic time = 0.00 ticks (2.71 ticks/sec)

CPLEX> display solution variables x
Variable Name Solution Value
\(x\) 2.500000

CPLEX> display solution variables y
Variable Name Solution Value
\(y\) 3.000000
Example

\[
\begin{align*}
\text{min} & \quad -x - y \\
-2x + 2y & \geq 1 \\
-8x + 10y & \leq 13 \\
x, y & \geq 0 \\
x, y & \in \mathbb{Z}
\end{align*}
\]
Example

\[
\begin{align*}
\text{min } & -x - y \\
-2x + 2y & \geq 1 \\
-8x + 10y & \leq 13 \\
x, y & \geq 0 \\
x, y & \in \mathbb{Z}
\end{align*}
\]
Example

Min obj: \(- x - y\)
Subject To
\[ c1: -2 x + 2 y \geq 1 \]
\[ c2: -8 x + 10 y \leq 13 \]
Bounds
\[ x = 3 \]
\[ y \leq 3 \]
End

====================================================================

CPLEX> optimize
Bound infeasibility column 'y'.
Presolve time = 0.00 sec. (0.00 ticks)
Presolve - Infeasible.
Solution time = 0.00 sec.
Deterministic time = 0.00 ticks (1.11 ticks/sec)
Example

\[
\begin{align*}
\min & \quad -x - y \\
-2x + 2y & \geq 1 \\
-8x + 10y & \leq 13 \\
x, y & \geq 0 \\
x, y & \in \mathbb{Z}
\end{align*}
\]
Example

\[
\begin{align*}
\min & \quad -x - y \\
-2x + 2y & \geq 1 \\
-8x + 10y & \leq 13 \\
x, y & \geq 0 \\
x, y & \in \mathbb{Z}
\end{align*}
\]
Min obj: \(- x - y\)
Subject To
\begin{align*}
c1: \quad & -2 x + 2 y \geq 1 \\
c2: \quad & -8 x + 10 y \leq 13 \\
\end{align*}
Bounds
\begin{align*}
x & \leq 2 \\
y & \leq 3 \\
\end{align*}
End

---

CPLEX> optimize
Dual simplex - Optimal: Objective = - 4.9000000000e+00
Solution time = 0.00 sec. Iterations = 0 (0)
Deterministic time = 0.00 ticks (2.71 ticks/sec)

CPLEX> display solution variables x
Variable Name       Solution Value
x                    2.000000

CPLEX> display solution variables y
Variable Name       Solution Value
y                    2.900000
Example

\[ \begin{align*}
\min \ & -x - y \\
-2x + 2y & \geq 1 \\
-8x + 10y & \leq 13 \\
x, y & \geq 0 \\
x, y & \in \mathbb{Z}
\end{align*} \]
Example

\[
\begin{align*}
&\text{min } -x - y \\
&-2x + 2y \geq 1 \\
&-8x + 10y \leq 13 \\
&x, y \geq 0 \\
&x, y \in \mathbb{Z}
\end{align*}
\]
Example

Min obj: - x - y
Subject To
  c1: -2 x + 2 y >= 1
  c2: -8 x + 10 y <= 13
Bounds
  x <= 2
  y = 3
End

====================================================================

CPLEX> optimize
Bound infeasibility column 'x'.
Presolve time = 0.00 sec. (0.00 ticks)
Presolve - Infeasible.
Solution time = 0.00 sec.
Deterministic time = 0.00 ticks (1.12 ticks/sec)
Example

\[
\begin{align*}
\min & \quad -x - y \\
-2x + 2y & \geq 1 \\
-8x + 10y & \leq 13 \\
x, y & \geq 0 \\
x, y & \in \mathbb{Z}
\end{align*}
\]
Example

\[
\begin{align*}
\text{min} & \quad -x - y \\
-2x + 2y & \geq 1 \\
-8x + 10y & \leq 13 \\
x, y & \geq 0 \\
x, y & \in \mathbb{Z}
\end{align*}
\]
Example

Min obj: $-x - y$
Subject To
$\begin{align*}
c1: & \quad -2x + 2y \geq 1 \\
c2: & \quad -8x + 10y \leq 13
\end{align*}$
Bounds
$x \leq 2$
$y \leq 2$
End

====================================================================

CPLEX> optimize
Dual simplex - Optimal: Objective = $-3.5000000000e+00$
Solution time = 0.00 sec. Iterations = 0 (0)
Deterministic time = 0.00 ticks (2.71 ticks/sec)

CPLEX> display solution variables x
Variable Name       Solution Value
x                    1.500000

CPLEX> display solution variables y
Variable Name       Solution Value
y                    2.000000
Example

min $-x - y$
$-2x + 2y \geq 1$
$-8x + 10y \leq 13$
$x, y \geq 0$
$x, y \in \mathbb{Z}$
Example

\[
\begin{align*}
\text{min} & \quad -x - y \\
-2x + 2y & \geq 1 \\
-8x + 10y & \leq 13 \\
x, y & \geq 0 \\
x, y & \in \mathbb{Z}
\end{align*}
\]
Example

Min obj: $- x - y$
Subject To
c1: $-2 x + 2 y \geq 1$
c2: $-8 x + 10 y \leq 13$
Bounds
$x = 2$
y $\leq 2$
End

================================

CPLEX> optimize
Bound infeasibility column 'y'.
Presolve time = 0.00 sec. (0.00 ticks)
Presolve - Infeasible.
Solution time = 0.00 sec.
Deterministic time = 0.00 ticks (1.11 ticks/sec)
Example

\[ \begin{align*}
\text{min } & -x - y \\
-2x + 2y & \geq 1 \\
-8x + 10y & \leq 13 \\
x, y & \geq 0 \\
x, y & \in \mathbb{Z}
\end{align*} \]
Example

\[ \begin{align*}
\text{min} & \quad -x - y \\
-2x + 2y & \geq 1 \\
-8x + 10y & \leq 13 \\
x, y & \geq 0 \\
x, y & \in \mathbb{Z}
\end{align*} \]
Example

Min obj: \(-x - y\)
Subject To
\(c1: -2x + 2y \geq 1\)
\(c2: -8x + 10y \leq 13\)
Bounds
\(x \leq 1\)
\(y \leq 2\)
End

=================================

CPLEX> optimize
Dual simplex - Optimal: Objective = \(-3.0000000000e+00\)
Solution time = 0.00 sec. Iterations = 0 (0)
Deterministic time = 0.00 ticks (2.40 ticks/sec)

CPLEX> display solution variables x
Variable Name | Solution Value
---------------|----------------
x             | 1.000000

CPLEX> display solution variables y
Variable Name | Solution Value
---------------|----------------
y             | 2.000000
Example

\[
\begin{align*}
\text{min } & -x - y \\
-2x + 2y & \geq 1 \\
-8x + 10y & \leq 13 \\
x, y & \geq 0 \\
x, y & \in \mathbb{Z}
\end{align*}
\]
Pruning in Branch & Bound

- We have already seen that if relaxation is infeasible, the problem can be pruned.

- Now assume an (integral) solution has been previously found.

- If solution has cost $Z$ then any pending problem $P_j$ whose relaxation has optimal value $\geq Z$ can be ignored, since

\[
\text{cost}(P_j) \geq \text{cost}(\text{LP}(P_j)) \geq Z
\]

The optimum will not be in any descendant of $P_j$!

- This cost-based pruning of the search tree has a huge impact on the efficiency of Branch & Bound.
Branch & Bound: Algorithm

\[ S := \{P_0\} \]  
\[ Z := +\infty \]  /* set of pending problems */  /* best cost found so far */

while \( S \neq \emptyset \) do

remove \( P \) from \( S \)
solve \( LP(P) \)

if \( LP(P) \) is feasible then  /* if unfeasible \( P \) can be pruned */

let \( \beta \) be optimal basic solution of \( LP(P) \)

if \( \beta \) satisfies integrality constraints then

if \( \text{cost}(\beta) < Z \) then store \( \beta \); update \( Z \)

else

if \( \text{cost}(LP(P)) \geq Z \) then continue  /* \( P \) can be pruned */

let \( x_j \) be integer variable such that \( \beta_j \notin \mathbb{Z} \)

\[ S := S \cup \{ P \land x_j \leq \lfloor \beta_j \rfloor, \ P \land x_j \geq \lceil \beta_j \rceil \} \]

return \( Z \)
Heuristics in Branch & Bound

Possible choices in Branch & Bound

Choice of the pending problem

- Depth-first search
- Breadth-first search
- Best-first search: assuming a relaxation is solved when it is added to the set of pending problems, select the one with best cost value
Heuristics in Branch & Bound

Possible choices in Branch & Bound

◆ Choice of the pending problem

■ Depth-first search
■ Breadth-first search
■ Best-first search: assuming a relaxation is solved when it is added to the set of pending problems, select the one with best cost value

◆ Choice of the branching variable: one that is

■ closest to halfway two integer values
■ most important in the model (e.g., 0-1 variable)
■ biggest in a variable ordering
■ the one with the largest/smallest cost coefficient
Heuristics in Branch & Bound

Possible choices in Branch & Bound

Choice of the pending problem

- Depth-first search
- Breadth-first search
- Best-first search: assuming a relaxation is solved when it is added to the set of pending problems, select the one with best cost value

Choice of the branching variable: one that is

- closest to halfway two integer values
- most important in the model (e.g., 0-1 variable)
- biggest in a variable ordering
- the one with the largest/smallest cost coefficient

No known strategy is best for all problems!
Remarks on Branch & Bound

- If integer variables are not bounded, Branch & Bound may not terminate:

\[
\begin{align*}
\text{min} & \quad 0 \\
1 & \leq 3x - 3y \leq 2 \\
x, y & \in \mathbb{Z}
\end{align*}
\]

is infeasible but Branch & Bound loops forever looking for solutions!
Remarks on Branch & Bound

- If integer variables are not bounded, Branch & Bound may not terminate:

\[
\begin{align*}
\text{min} & \quad 0 \\
1 & \leq 3x - 3y \leq 2 \\
x, y & \in \mathbb{Z}
\end{align*}
\]

is infeasible but Branch & Bound loops forever looking for solutions!

- E.g., we first find a solution with \( x = \frac{2}{3} \).
 Remarks on Branch & Bound

- If integer variables are not bounded, Branch & Bound may not terminate:

\[
\begin{align*}
\text{min } & 0 \\
1 & \leq 3x - 3y \leq 2 \\
x, y & \in \mathbb{Z}
\end{align*}
\]

is infeasible but Branch & Bound loops forever looking for solutions!

- E.g., we first find a solution with \( x = \frac{2}{3} \).
- In the subproblem with \( x \geq 1 \) we get a solution with \( y = \frac{1}{3} \).
Remarks on Branch & Bound

- If integer variables are not bounded, Branch & Bound may not terminate:

\[
\begin{align*}
\min & \quad 0 \\
& 1 \leq 3x - 3y \leq 2 \\
& x, y \in \mathbb{Z}
\end{align*}
\]

is infeasible but Branch & Bound loops forever looking for solutions!

- E.g., we first find a solution with \( x = \frac{2}{3} \).
- In the subproblem with \( x \geq 1 \) we get a solution with \( y = \frac{1}{3} \).
- In the subproblem with \( x \geq 1, y \geq 1 \) we get a solution with \( x = \frac{5}{3} \).
Remarks on Branch & Bound

- If integer variables are not bounded, Branch & Bound may not terminate:

\[
\begin{align*}
\text{min} & \quad 0 \\
1 & \leq 3x - 3y \leq 2 \\
x, y & \in \mathbb{Z}
\end{align*}
\]

is infeasible but Branch & Bound loops forever looking for solutions!

- E.g., we first find a solution with \( x = \frac{2}{3} \).
- In the subproblem with \( x \geq 1 \) we get a solution with \( y = \frac{1}{3} \).
- In the subproblem with \( x \geq 1, y \geq 1 \) we get a solution with \( x = \frac{5}{3} \).
- In the subproblem with \( x \geq 2, y \geq 1 \) we get a solution with \( y = \frac{4}{3} \).
Remarks on Branch & Bound

■ If integer variables are not bounded, Branch & Bound may not terminate:

\[
\begin{align*}
\text{min } & \quad 0 \\
1 & \leq 3x - 3y \leq 2 \\
x, y & \in \mathbb{Z}
\end{align*}
\]

is infeasible but Branch & Bound loops forever looking for solutions!

■ E.g., we first find a solution with \( x = \frac{2}{3} \).
■ In the subproblem with \( x \geq 1 \) we get a solution with \( y = \frac{1}{3} \).
■ In the subproblem with \( x \geq 1, y \geq 1 \) we get a solution with \( x = \frac{5}{3} \).
■ In the subproblem with \( x \geq 2, y \geq 1 \) we get a solution with \( y = \frac{4}{3} \).
■ In the subproblem with \( x \geq 2, y \geq 2 \) we get a solution with \( x = \frac{8}{3} \).
Remarks on Branch & Bound

If integer variables are not bounded, Branch & Bound may not terminate:

\[
\begin{align*}
\min & \quad 0 \\
1 & \leq 3x - 3y \leq 2 \\
x, y & \in \mathbb{Z}
\end{align*}
\]

is infeasible but Branch & Bound loops forever looking for solutions!

- E.g., we first find a solution with \(x = \frac{2}{3}\).
- In the subproblem with \(x \geq 1\) we get a solution with \(y = \frac{1}{3}\).
- In the subproblem with \(x \geq 1, y \geq 1\) we get a solution with \(x = \frac{5}{3}\).
- In the subproblem with \(x \geq 2, y \geq 1\) we get a solution with \(y = \frac{4}{3}\).
- In the subproblem with \(x \geq 2, y \geq 2\) we get a solution with \(x = \frac{8}{3}\).
- ...

Remarks on Branch & Bound

- After solving the relaxation of $P$, we have to solve the relaxations of $P \land x_j \leq \lfloor \beta_j \rfloor$ and $P \land x_j \geq \lceil \beta_j \rceil$

- These problems are similar. Do we have to start from scratch? Can be reuse somehow the computation for $P$?

- Idea: start from the optimal solution of the parent problem
Remarks on Branch & Bound

Let us assume that $P$ is of the form

$$\begin{align*}
\min & \ c^T x \\
Ax &= b \\
x &\geq 0, \quad x_i \in \mathbb{Z} \quad \forall i \in I
\end{align*}$$

Let $B$ be an optimal basis of the relaxation

Let $x_j$ be integer variable which at optimal solution is assigned $\beta_j \notin \mathbb{Z}$

Note that $x_j$ must be basic

Let us consider the problem $P_1 = P \land x_j \leq \lfloor \beta_j \rfloor$

We add a fresh slack variable $s$ and a new equation: $P \land x_j + s = \lfloor \beta_j \rfloor$

Since $s$ is fresh we have $(x_B, s)$ defines a basis for the relaxation of $P_1$
Remarks on Branch & Bound

\[
\begin{align*}
\min & -x - y \\
-2x + 2y & \geq 1 \\
-8x + 10y & \leq 13 \\
x, y & \geq 0 \\
x, y & \in \mathbb{Z}
\end{align*}
\Rightarrow
\begin{align*}
\min & -x - y \\
-2x + 2y - s_1 & = 1 \\
-8x + 10y + s_2 & = 13 \\
x, y & \geq 0 \\
x, y & \in \mathbb{Z}
\end{align*}
\]

- Optimal basis of the linear relaxation is \( \mathcal{B} = (x, y) \) with tableau

\[
\begin{align*}
\min & -\frac{17}{2} + \frac{9}{2} s_1 + s_2 \\
x & = 4 - \frac{5}{2} s_1 - \frac{1}{2} s_2 \\
y & = \frac{9}{2} - 2 s_1 - \frac{1}{2} s_2
\end{align*}
\]

- For the subproblem with \( y \leq 4 \) we add equation \( y + s = 4 \)

\( \mathcal{B} = (x, y, s) \) is a basis for this subproblem with tableau

\[
\begin{align*}
\min & -\frac{17}{2} + \frac{9}{2} s_1 + s_2 \\
x & = 4 - \frac{5}{2} s_1 - \frac{1}{2} s_2 \\
y & = \frac{9}{2} - 2 s_1 - \frac{1}{2} s_2 \\
s & = 4 - y = -\frac{1}{2} + 2s_1 + \frac{1}{2} s_2
\end{align*}
\]
Remarks on Branch & Bound

- \((x_B, s)\) defines a basis for the relaxation of \(P_1\)

- This basis is **not feasible**: the value in the basic solution assigned to \(s\) is \(\lfloor \beta_j \rfloor - \beta_j < 0\). We would need a Phase I to apply the primal simplex method!

- But since \(s\) is a slack the reduced costs have not changed: \((x_B, s)\) satisfies the optimality conditions!

- **Dual simplex method** can be used: basis \((x_B, s)\) is already **dual feasible**, no need of (dual) Phase I

- In practice often the dual simplex only needs very few iterations to obtain the optimal solution to the new problem
Let us consider a MIP of the form

$$\min_{x \in S} c^T x$$

where \( S = \{ x \in \mathbb{R}^n \mid Ax = b, x \geq 0, x_i \in \mathbb{Z} \ \forall i \in \mathcal{I} \} \)

and its linear relaxation

$$\min_{x \in P} c^T x$$

where \( P = \{ x \in \mathbb{R}^n \mid Ax = b, x \geq 0 \} \)

Let \( \beta \) be such that \( \beta \in P \) but \( \beta \notin S \).

A cut for \( \beta \) is a linear inequality \( \hat{a}^T x \leq \hat{b} \) such that

- \( \hat{a}^T \sigma \leq \hat{b} \) for any \( \sigma \in S \) (feasible solutions of the MIP respect the cut)
- and \( \hat{a}^T \beta > \hat{b} \) (\( \beta \) does not respect the cut)
Cutting Planes

\[
\begin{align*}
\text{max } x + y \\
-8x + 10y &\leq 13 \\
(1, 2) \\
(0, 1) \\
x \geq 0
\end{align*}
\]

\[
\begin{align*}
x + y &\leq 6 \\
x + y &\geq 0 \\
x, y &\in \mathbb{Z}
\end{align*}
\]

\[
\begin{align*}
\text{max } x + y \\
-2x + 2y &\geq 1 \\
-8x + 10y &\leq 13 \\
x, y &\geq 0 \\
x, y &\in \mathbb{Z}
\end{align*}
\]

\[x + y \leq 6 \text{ is a cut}\]
Using Cuts for Solving MIP’s

- Let $\hat{a}^T x \leq \hat{b}$ be a cut. Then the MIP

$$\min_{x \in S'} c^T x \quad \text{where} \quad S' = \left\{ x \in \mathbb{R}^n \mid \begin{array}{l} Ax = b \\ \hat{a}^T x \leq \hat{b} \\ x \geq 0 \\ x_i \in \mathbb{Z} \quad \forall i \in \mathcal{I} \end{array} \right\}$$

has the same set of feasible solutions $S$ but its LP relaxation is strictly more constrained.

- Instead of splitting into subproblems (Branch & Bound), one can add the cut and solve the relaxation of the new problem.

- In practice cuts are used together with Branch & Bound: If after adding some cuts no integer solution is found, then branch.

This technique is called Branch & Cut.
Gomory Cuts

- There are several techniques for deriving cuts
- Some are problem-specific (e.g., for the travelling salesman problem)
- Here we will see a generic technique: **Gomory cuts**
- Let us consider a basis $B$ and let $\beta$ be the associated basic solution. Note that for all $j \in \mathcal{R}$ we have $\beta_j = 0$
- Let $x_i$ be a basic variable such that $i \in \mathcal{I}$ and $\beta_i \notin \mathbb{Z}$
- E.g., this happens in the optimal basis of the relaxation when the basic solution does not meet the integrality constraints
- Let the row of the tableau corresponding to $x_i$ be of the form

\[
x_i = \beta_i + \sum_{j \in \mathcal{R}} \alpha_{ij} x_j
\]
Gomory Cuts

- Let $x \in S$. Then $x_i \in \mathbb{Z}$ and

$$x_i = \beta_i + \sum_{j \in \mathcal{R}} \alpha_{ij} x_j$$

$$x_i - \beta_i = \sum_{j \in \mathcal{R}} \alpha_{ij} x_j$$

- Let $\delta = \beta_i - \lfloor \beta_i \rfloor$. Then $0 < \delta < 1$

- Hence

$$x_i - \lfloor \beta_i \rfloor = x_i - \beta_i + \beta_i - \lfloor \beta_i \rfloor$$

$$= x_i - \beta_i + \delta$$

$$= \delta + x_i - \beta_i$$

$$= \delta + \sum_{j \in \mathcal{R}} \alpha_{ij} x_j$$
Gomory Cuts

\[ \delta = \beta_i - \lfloor \beta_i \rfloor \quad x_i - \lfloor \beta_i \rfloor = \delta + \sum_{j \in R} \alpha_{ij} x_j \]

Let us define

\[ R^+ = \{ j \in R \mid \alpha_{ij} \geq 0 \} \quad R^- = \{ j \in R \mid \alpha_{ij} < 0 \} \]

Assume \[ \sum_{j \in R} \alpha_{ij} x_j \geq 0. \]
Gomory Cuts

\[ \delta = \beta_i - \lfloor \beta_i \rfloor \quad x_i - \lfloor \beta_i \rfloor = \delta + \sum_{j \in \mathcal{R}} \alpha_{ij} x_j \]

- Let us define

\[ \mathcal{R}^+ = \{ j \in \mathcal{R} \mid \alpha_{ij} \geq 0 \} \quad \mathcal{R}^- = \{ j \in \mathcal{R} \mid \alpha_{ij} < 0 \} \]

- Assume \( \sum_{j \in \mathcal{R}} \alpha_{ij} x_j \geq 0 \).

Then \( \delta + \sum_{j \in \mathcal{R}} \alpha_{ij} x_j > 0 \) and \( x_i - \lfloor \beta_i \rfloor \in \mathbb{Z} \) imply

\[ \delta + \sum_{j \in \mathcal{R}} \alpha_{ij} x_j \geq 1 \]

\[ \sum_{j \in \mathcal{R}^+} \alpha_{ij} x_j \geq \sum_{j \in \mathcal{R}} \alpha_{ij} x_j \geq 1 - \delta \]

\[ \sum_{j \in \mathcal{R}^+} \frac{\alpha_{ij}}{1 - \delta} x_j \geq 1 \]
Gomory Cuts

\[ \delta = \beta_i - \lfloor \beta_i \rfloor \quad x_i - \lfloor \beta_i \rfloor = \delta + \sum_{j \in \mathcal{R}} \alpha_{ij} x_j \]

- Let us define

\[ \mathcal{R}^+ = \{ j \in \mathcal{R} \mid \alpha_{ij} \geq 0 \} \quad \mathcal{R}^- = \{ j \in \mathcal{R} \mid \alpha_{ij} < 0 \} \]

- Assume \( \sum_{j \in \mathcal{R}} \alpha_{ij} x_j \geq 0 \).

Then \( \delta + \sum_{j \in \mathcal{R}} \alpha_{ij} x_j > 0 \) and \( x_i - \lfloor \beta_i \rfloor \in \mathbb{Z} \) imply

\[ \delta + \sum_{j \in \mathcal{R}} \alpha_{ij} x_j \geq 1 \]

\[ \sum_{j \in \mathcal{R}^+} \alpha_{ij} x_j \geq \sum_{j \in \mathcal{R}} \alpha_{ij} x_j \geq 1 - \delta \]

\[ \sum_{j \in \mathcal{R}^+} \frac{\alpha_{ij}}{1 - \delta} x_j \geq 1 \]

Moreover \( \sum_{j \in \mathcal{R}^-} \left( \frac{-\alpha_{ij}}{\delta} \right) x_j \geq 0 \)
Gomory Cuts

\[\delta = \beta_i - \lfloor \beta_i \rfloor\quad x_i - \lfloor \beta_i \rfloor = \delta + \sum_{j \in \mathcal{R}} \alpha_{ij} x_j\]

- Let us define

\[\mathcal{R}^+ = \{ j \in \mathcal{R} \mid \alpha_{ij} \geq 0 \}\quad \mathcal{R}^- = \{ j \in \mathcal{R} \mid \alpha_{ij} < 0 \}\]

- Assume \( \sum_{j \in \mathcal{R}} \alpha_{ij} x_j < 0 \).
Gomory Cuts

\[ \delta = \beta_i - \lfloor \beta_i \rfloor \quad x_i - \lfloor \beta_i \rfloor = \delta + \sum_{j \in \mathcal{R}} \alpha_{ij} x_j \]

- Let us define

\[ \mathcal{R}^+ = \{ j \in \mathcal{R} \mid \alpha_{ij} \geq 0 \} \quad \mathcal{R}^- = \{ j \in \mathcal{R} \mid \alpha_{ij} < 0 \} \]

- Assume \( \sum_{j \in \mathcal{R}} \alpha_{ij} x_j < 0 \).

Then \( \delta + \sum_{j \in \mathcal{R}} \alpha_{ij} x_j < 1 \) and \( x_i - \lfloor \beta_i \rfloor \in \mathbb{Z} \) imply

\[ \delta + \sum_{j \in \mathcal{R}} \alpha_{ij} x_j \leq 0 \]

\[ \sum_{j \in \mathcal{R}^-} \alpha_{ij} x_j \leq \sum_{j \in \mathcal{R}} \alpha_{ij} x_j \leq -\delta \]

\[ \sum_{j \in \mathcal{R}^-} \left( \frac{-\alpha_{ij}}{\delta} \right) x_j \geq 1 \]
Let us define

\[ R^+ = \{ j \in R \mid \alpha_{ij} \geq 0 \} \quad R^- = \{ j \in R \mid \alpha_{ij} < 0 \} \]

Assume \( \sum_{j \in R} \alpha_{ij} x_j < 0 \).

Then \( \delta + \sum_{j \in R} \alpha_{ij} x_j < 1 \) and \( x_i - \lfloor \beta_i \rfloor \in \mathbb{Z} \) imply

\[ \delta + \sum_{j \in R} \alpha_{ij} x_j \leq 0 \]

\[ \sum_{j \in R^-} \alpha_{ij} x_j \leq \sum_{j \in R} \alpha_{ij} x_j \leq -\delta \]

\[ \sum_{j \in R^-} \left( \frac{-\alpha_{ij}}{\delta} \right) x_j \geq 1 \]

Moreover \( \sum_{j \in R^+} \frac{\alpha_{ij}}{1-\delta} x_j \geq 0 \)
Gomory Cuts

- In any case
  \[
  \sum_{j \in \mathcal{R}^-} \left( \frac{-\alpha_{ij}}{\delta} \right) x_j + \sum_{j \in \mathcal{R}^+} \frac{\alpha_{ij}}{1-\delta} x_j \geq 1
  \]
  for any \( x \in \mathcal{S} \).

However, when \( x = \beta \) this inequality is not satisfied (set \( x_j = 0 \) for \( j \in \mathcal{R} \))

- In the example:

\[
\begin{align*}
\min & -\frac{17}{2} + \frac{9}{2} s_1 + s_2 \\
x & = 4 - \frac{5}{2} s_1 - \frac{1}{2} s_2 \\
y & = \frac{9}{2} - 2 s_1 - \frac{1}{2} s_2
\end{align*}
\]

\( y \) violates the integrality condition,

we have \( \delta = \frac{1}{2} \), \( \sum_{j \in \mathcal{R}} \alpha_{ij} x_j = -2 s_1 - \frac{1}{2} s_2 \)

The cut is \( 4 s_1 + s_2 \geq 1 \), which projected on \( x, y \) is \( y \leq 4 \).
Ensuring All Vertices Are Integer

- Let us assume \( A, b \) have coefficients in \( \mathbb{Z} \).
- Sometimes it is possible to ensure for an IP that all vertices of the relaxation are integer.
- For instance, when the matrix \( A \) is totally unimodular: the determinant of every square submatrix is 0 or \( \pm 1 \).
Ensuring All Vertices Are Integer

- Let us assume $A, b$ have coefficients in $\mathbb{Z}$
- Sometimes it is possible to ensure for an IP that all vertices of the relaxation are integer
- For instance, when the matrix $A$ is totally unimodular: the determinant of every square submatrix is 0 or $\pm 1$

In that case all bases have inverses with integer coefficients

Recall **Cramer’s rule**: if $B$ is an invertible matrix, then

$$B^{-1} = \frac{1}{\det(B)} \text{adj}(B)$$

where $\text{adj}(B)$ is the **adjugate** matrix of $B$

Recall also that

$$\text{adj}(B) = ((-1)^{i+j} \det(M_{ji}))_{1 \leq i,j \leq n},$$

where $M_{ij}$ is matrix $B$ after removing the $i$-th row and the $j$-th column
Ensuring All Vertices Are Integer

- Sufficient condition for total unimodularity of a matrix $A$:
  (Hoffman & Gale’s Theorem)

1. Each element of $A$ is 0 or $\pm 1$
2. No more than two non-zeros appear in each column
3. Rows can be partitioned in two subsets $R_1$ and $R_2$ s.t.

   (a) If a column contains two non-zeros of the same sign, the row of one of them belongs to one subset, and the row of the other, to the other subset

   (b) If a column contains two non-zeros of different signs, the rows of both of them belong to the same subset
Assignment Problem

- \( n = \# \) of workers = \( \# \) of tasks
- Each worker must be assigned to exactly one task
- Each task is to be performed by exactly one worker
- \( c_{ij} \) = cost when worker \( i \) performs task \( j \)
Assignment Problem

- \( n = \# \) of workers = \# of tasks
- Each worker must be assigned to exactly one task
- Each task is to be performed by exactly one worker
- \( c_{ij} = \) cost when worker \( i \) performs task \( j \)

\[
x_{ij} = \begin{cases} 
1 & \text{if worker } i \text{ performs task } j \\
0 & \text{otherwise}
\end{cases}
\]

\[
\min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}
\]

\[
\sum_{j=1}^{n} x_{ij} = 1 \quad \forall i \in \{1, \ldots, n\}
\]

\[
\sum_{i=1}^{n} x_{ij} = 1 \quad \forall j \in \{1, \ldots, n\}
\]

\( x_{ij} \in \{0, 1\} \quad \forall i, j \in \{1, \ldots, n\} \)

- This problem satisfies Hoffman & Gale’s conditions
Ensuring All Vertices Are Integer

- Several kinds of IP’s satisfy Hoffman & Gale’s conditions:
  - Assignment
  - Transportation
  - Maximum flow
  - Shortest path
  - ...

- Usually ad-hoc network algorithms are more efficient for these problems than the simplex method as presented here.
Ensuring All Vertices Are Integer

- Several kinds of IP’s satisfy Hoffman & Gale’s conditions:
  - Assignment
  - Transportation
  - Maximum flow
  - Shortest path
  - ...

- Usually ad-hoc network algorithms are more efficient for these problems than the simplex method as presented here.

- But:
  - The simplex method can be specialized: network simplex method
  - Simplex techniques can be applied if the problem is not a purely network one but has extra constraints
Expressing Logical Constraints

- Sometimes we want to have an indicator variable of a constraint: a 0/1 variable equal to 1 iff the constraint is true (= reification in CP)

- E.g., let us to encode $\delta = 1 \iff a^T x \leq b$, where $\delta$ is a 0/1 var
Expressing Logical Constraints

- Sometimes we want to have an indicator variable of a constraint: a 0/1 variable equal to 1 iff the constraint is true (= reification in CP)

- E.g., let us to encode $\delta = 1 \iff a^T x \leq b$, where $\delta$ is a 0/1 var

- Assume $a^T x \in \mathbb{Z}$ for all feasible solution $x$
  
  Let $U$ be an upper bound of $a^T x - b$ for all feasible solutions
  
  Let $L$ be a lower bound of $a^T x - b$ for all feasible solutions
Expressing Logical Constraints

- Sometimes we want to have an indicator variable of a contraint: a 0/1 variable equal to 1 iff the constraint is true (= reification in CP)

- E.g., let us to encode $\delta = 1 \iff a^T x \leq b$, where $\delta$ is a 0/1 var

- Assume $a^T x \in \mathbb{Z}$ for all feasible solution $x$
  
  Let $U$ be an upper bound of $a^T x - b$ for all feasible solutions
  
  Let $L$ be a lower bound of $a^T x - b$ for all feasible solutions

1. $\delta = 1 \rightarrow a^T x \leq b$
   
   can be encoded with $a^T x - b \leq U(1 - \delta)$
Expressing Logical Constraints

- Sometimes we want to have an indicator variable of a constraint: a 0/1 variable equal to 1 iff the constraint is true (= reification in CP)

- E.g., let us to encode \( \delta = 1 \iff a^T x \leq b \), where \( \delta \) is a 0/1 var

- Assume \( a^T x \in \mathbb{Z} \) for all feasible solution \( x \)

  Let \( U \) be an upper bound of \( a^T x - b \) for all feasible solutions
  Let \( L \) be a lower bound of \( a^T x - b \) for all feasible solutions

1. \( \delta = 1 \rightarrow a^T x \leq b \)
   can be encoded with \( a^T x - b \leq U(1 - \delta) \)

2. \( \delta = 1 \iff a^T x \leq b \)
   \( \delta = 0 \rightarrow a^T x > b \)
   \( \delta = 0 \rightarrow a^T x \geq b + 1 \)
   can be encoded with \( a^T x - b \geq (L - 1)\delta + 1 \)
Expressing Logical Constraints

- We want to encode $\delta = 1 \leftrightarrow a^T x \leq b$, where $\delta$ is a 0/1 var.

- Now assume that $a^T x$ is real-valued.

  Let $U$ be an upper bound of $a^T x - b$ for all feasible solutions.

  Let $L$ be a lower bound of $a^T x - b$ for all feasible solutions.

1. $\delta = 1 \rightarrow a^T x \leq b$

   can be encoded with $a^T x - b \leq U(1 - \delta)$
Expressing Logical Constraints

- We want to encode $\delta = 1 \iff a^T x \leq b$, where $\delta$ is a 0/1 var

- Now assume that $a^T x$ is real-valued.

  Let $U$ be an upper bound of $a^T x - b$ for all feasible solutions

  Let $L$ be a lower bound of $a^T x - b$ for all feasible solutions

1. $\delta = 1 \rightarrow a^T x \leq b$
   
   can be encoded with $a^T x - b \leq U(1 - \delta)$

2. $\delta = 1 \leftarrow a^T x \leq b$
   
   $\delta = 0 \rightarrow a^T x > b$  
   Can only be modeled if we allow for a tolerance $\epsilon$
We want to encode $\delta = 1 \leftrightarrow a^T x \leq b$, where $\delta$ is a 0/1 var

Now assume that $a^T x$ is real-valued.

Let $U$ be an upper bound of $a^T x - b$ for all feasible solutions

Let $L$ be a lower bound of $a^T x - b$ for all feasible solutions

1. $\delta = 1 \rightarrow a^T x \leq b$
   can be encoded with $a^T x - b \leq U(1 - \delta)$

2. $\delta = 1 \leftarrow a^T x \leq b$
   $\delta = 0 \rightarrow a^T x > b$   Can only be modeled if we allow for a tolerance $\epsilon$
   $\delta = 0 \rightarrow a^T x \geq b + \epsilon$
   can be encoded with $a^T x - b \geq (L - \epsilon)\delta + \epsilon$
Expressing Logical Constraints

- We want to encode $\delta = 1 \leftrightarrow a^T x = b$, where $\delta$ is a 0/1 var
- Assume that $a^T x$ is real-valued.
  Let $U$ be upper bound of $a^T x - b$ for all feasible solutions
  Let $L$ be lower bound of $a^T x - b$ for all feasible solutions
Expressing Logical Constraints

- We want to encode $\delta = 1 \leftrightarrow a^T x = b$, where $\delta$ is a 0/1 var
- Assume that $a^T x$ is real-valued.

Let $U$ be upper bound of $a^T x - b$ for all feasible solutions
Let $L$ be lower bound of $a^T x - b$ for all feasible solutions

1. $\delta = 1 \rightarrow a^T x \leq b \quad \Rightarrow \quad a^T x - b \leq U (1 - \delta)$
Expressing Logical Constraints

- We want to encode $\delta = 1 \leftrightarrow a^T x = b$, where $\delta$ is a 0/1 var
- Assume that $a^T x$ is real-valued.

Let $U$ be upper bound of $a^T x - b$ for all feasible solutions
Let $L$ be lower bound of $a^T x - b$ for all feasible solutions

1. $\delta = 1 \rightarrow a^T x \leq b \quad \Rightarrow \quad a^T x - b \leq U(1 - \delta)$
2. $\delta = 1 \rightarrow a^T x \geq b \quad \Rightarrow \quad a^T x - b \geq L(1 - \delta)$
Expressing Logical Constraints

- We want to encode $\delta = 1 \leftrightarrow a^T x = b$, where $\delta$ is a 0/1 var
- Assume that $a^T x$ is real-valued.

Let $U$ be upper bound of $a^T x - b$ for all feasible solutions
Let $L$ be lower bound of $a^T x - b$ for all feasible solutions

1. $\delta = 1 \rightarrow a^T x \leq b \quad \Rightarrow \quad a^T x - b \leq U(1 - \delta)$
2. $\delta = 1 \rightarrow a^T x \geq b \quad \Rightarrow \quad a^T x - b \geq L(1 - \delta)$
3. $\delta = 1 \leftrightarrow a^T x = b$
   $\delta = 0 \rightarrow a^T x \neq b$
   $\delta = 0 \rightarrow a^T x < b \lor a^T x > b$
Expressing Logical Constraints

- We want to encode $\delta = 1 \leftrightarrow a^Tx = b$, where $\delta$ is a 0/1 var.
- Assume that $a^Tx$ is real-valued.

Let $U$ be upper bound of $a^Tx - b$ for all feasible solutions.
Let $L$ be lower bound of $a^Tx - b$ for all feasible solutions.

1. $\delta = 1 \rightarrow a^Tx \leq b \quad \Rightarrow \quad a^Tx - b \leq U(1 - \delta)$
2. $\delta = 1 \rightarrow a^Tx \geq b \quad \Rightarrow \quad a^Tx - b \geq L(1 - \delta)$
3. $\delta = 1 \leftarrow a^Tx = b$
   
   $\delta = 0 \rightarrow a^Tx \neq b$
   
   $\delta = 0 \rightarrow a^Tx < b \lor a^Tx > b$

Let $\epsilon$ be the tolerance, $\delta', \delta''$ auxiliary 0/1 vars.

$\delta = 0 \rightarrow \delta' = 0 \lor \delta'' = 0 \quad \Rightarrow \quad \delta' + \delta'' - \delta \leq 1$

$\delta' = 0 \rightarrow a^Tx \leq b - \epsilon \quad \Rightarrow \quad a^Tx - b \leq (U + \epsilon)\delta' - \epsilon$

$\delta'' = 0 \rightarrow a^Tx \geq b + \epsilon \quad \Rightarrow \quad a^Tx - b \geq (L - \epsilon)\delta'' + \epsilon$
Expressing Logical Constraints

- Boolean expressions can be modeled with 0/1 vars

- If $x_i$ is a 0/1 variable,
  let $X_i$ be a boolean variable such that $X_i$ is true iff $x_i = 1$

<table>
<thead>
<tr>
<th>$X_1 \lor X_2$</th>
<th>iff</th>
<th>$x_1 + x_2 \geq 1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X_1 \land X_2$</td>
<td>iff</td>
<td>$x_1 = x_2 = 1$</td>
</tr>
<tr>
<td>$\neg X_1$</td>
<td>iff</td>
<td>$x_1 = 0$</td>
</tr>
<tr>
<td>$X_1 \Rightarrow X_2$</td>
<td>iff</td>
<td>$x_1 \leq x_2$</td>
</tr>
<tr>
<td>$X_1 \Leftrightarrow X_2$</td>
<td>iff</td>
<td>$x_1 = x_2$</td>
</tr>
</tbody>
</table>
Example

Let $X_i$ represent “Ingredient $i$ is in the blend”, $i \in \{A, B, C\}$.
Express the sentence

“If ingredient $A$ is in the blend, then ingredient $B$ or $C$ (or both) must also be in the blend”

with linear constraints.
Example

Let $X_i$ represent “Ingredient $i$ is in the blend”, $i \in \{A, B, C\}$. Express the sentence

“If ingredient $A$ is in the blend,
then ingredient $B$ or $C$ (or both) must also be in the blend”

with linear constraints.

- We need to express $X_A \to (X_B \lor X_C)$.
- Equivalently, $\neg X_A \lor X_B \lor X_C$.
- $\neg X_A \lor X_B \lor X_C$ is equivalent to $(1 - x_A) + x_B + x_C \geq 1$.
- So $x_B + x_C \geq x_A$
Example (Fixed Setup Charge)

Let $x$ be the quantity of a product with unit production cost $c_1$. If the product is manufactured at all, there is a setup cost $c_0$

$\text{Cost of producing } x \text{ units} = \begin{cases} 
0 & \text{if } x = 0 \\
 c_0 + c_1 x & \text{if } x > 0 
\end{cases}$

Want to minimize costs. Model as a MIP? (for simplicity, additional constraints are not specified and can be omitted)
Example (Fixed Setup Charge)

Let $x$ be the quantity of a product with unit production cost $c_1$.
If the product is manufactured at all, there is a setup cost $c_0$

\[
\text{Cost of producing } x \text{ units} = \begin{cases} 
0 & \text{if } x = 0 \\
 c_0 + c_1 x & \text{if } x > 0 
\end{cases}
\]

Want to minimize costs. Model as a MIP?
(for simplicity, additional constraints are not specified and can be omitted)

Let $\delta$ be 0/1 var such that $x > 0 \rightarrow \delta = 1$ (i.e., $\delta = 0 \rightarrow x \leq 0$):
add constraint $x - U\delta \leq 0$, where $U$ is the upper bound on $x$

Then the cost is $c_0\delta + c_1x$.

No need to express $x > 0 \leftarrow \delta = 1$, i.e. $x = 0 \rightarrow \delta = 0$
Minimization will make $\delta = 0$ if possible (i.e., if $x = 0$)
Example (Capacity Expansion)

Let $a^T x$ be the consumption of a limited resource in a production process.

Want to relax the constraint $a^T x \leq b$ by increasing capacity $b$.

Capacity can be expanded to $b_i$

\[
b = b_0 < b_1 < b_2 < \cdots < b_t\]

with costs, respectively,

\[
0 = c_0 < c_1 < c_2 < \cdots < c_t\]

Want to minimize costs. Model as a MIP? (for simplicity, additional constraints are not specified and can be omitted)
Example (Capacity Expansion)

Let $a^Tx$ be the consumption of a limited resource in a production process.

Want to relax the constraint $a^Tx \leq b$ by increasing capacity $b$.

Capacity can be expanded to $b_i$

$$b = b_0 < b_1 < b_2 < \cdots < b_t$$

with costs, respectively,

$$0 = c_0 < c_1 < c_2 < \cdots < c_t$$

Want to minimize costs. Model as a MIP?

(for simplicity, additional constraints are not specified and can be omitted)

Let 0/1 variables $\delta_i$ mean “capacity expanded to $b_i$”. Then:

- $\sum_{i=0}^{t} \delta_i = 1$
- $a^Tx \leq \sum_{i=0}^{t} b_i \delta_i$
- Cost function: $\sum_{i=0}^{t} c_i \delta_i$