Mixed Integer Linear Programming

Feodor Pisnichenko

July 15, 2025

Summary

MILP formulation

Mixed Integer Linear Programs

A mixed integer linear program (MILP, MIP) is of the form:

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t.} \quad & Ax = b \\ & x \geq 0 \\ & x_i \in \mathbb{Z} \quad \forall i \in \mathcal{I} \end{aligned}$$

- If all variables must be integers, it is called a (pure) **integer linear program** (ILP, IP).
- If all variables must be 0 or 1 (binary, boolean), it is called a **binary (0-1) linear program**.

Mixed Integer Linear Programs

A mixed integer linear program (MILP, MIP) is of the form:

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t.} \quad & Ax = b \\ & x \geq 0 \\ & x_i \in \mathbb{Z} \quad \forall i \in \mathcal{I} \end{aligned}$$

- If all variables must be integers, it is called a (pure) **integer linear program** (ILP, IP).
- If all variables must be 0 or 1 (binary, boolean), it is called a **binary (0-1) linear program**.

Mixed Integer Linear Programs

A mixed integer linear program (MILP, MIP) is of the form:

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t.} \quad & Ax = b \\ & x \geq 0 \\ & x_i \in \mathbb{Z} \quad \forall i \in \mathcal{I} \end{aligned}$$

- If all variables must be integers, it is called a (pure) **integer linear program** (ILP, IP).
- If all variables must be 0 or 1 (binary, boolean), it is called a **binary (0-1) linear program**.

Complexity: LP vs. IP

- Including integer variables enormously increases the modeling power, at the expense of more complexity.
- LPs can be solved in polynomial time with interior-point methods (e.g., ellipsoid method, Karmarkar's algorithm).
- Integer Programming is an NP-hard 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.

LP Relaxation of a MIP

Given a MIP:

$$
\begin{aligned}
\text{(IP)} \quad \min \quad & c^T x \\
\text{s.t.} \quad & Ax = b \\
& x \geq 0 \\
& x_i \in \mathbb{Z} \quad \forall i \in \mathcal{I}
\end{aligned}
$$

Its **linear relaxation** is the LP obtained by dropping integrality constraints:

$$
\begin{aligned}
\text{(LP)} \quad \min \quad & c^T x \\
\text{s.t.} \quad & Ax = b \\
& x \geq 0
\end{aligned}
$$

Can we solve the IP by solving the LP? By rounding?

LP Relaxation of a MIP

Given a MIP:

$$
\begin{aligned}
\text{(IP)} \quad \min \quad & c^T x \\
\text{s.t.} \quad & Ax = b \\
& x \geq 0 \\
& x_i \in \mathbb{Z} \quad \forall i \in \mathcal{I}
\end{aligned}
$$

Its **linear relaxation** is the LP obtained by dropping integrality constraints:

$$
\begin{aligned}
\text{(LP)} \quad \min \quad & c^T x \\
\text{s.t.} \quad & Ax = b \\
& x \geq 0
\end{aligned}
$$

Can we solve the IP by solving the LP? By rounding?

LP Relaxation of a MIP

Given a MIP:

$$
\begin{aligned}
\text{(IP)} \quad \min \quad & c^T x \\
\text{s.t.} \quad & Ax = b \\
& x \geq 0 \\
& x_i \in \mathbb{Z} \quad \forall i \in \mathcal{I}
\end{aligned}
$$

Its **linear relaxation** is the LP obtained by dropping integrality constraints:

$$
\begin{aligned}
\text{(LP)} \quad \min \quad & c^T x \\
\text{s.t.} \quad & Ax = b \\
& x \geq 0
\end{aligned}
$$

Can we solve the IP by solving the LP? By rounding?

Branch & Bound

Branch & Bound

The optimal solution of

$$\begin{aligned} \max \quad & x + y \\ \text{s.t.} \quad & -2x + 2y \geq 1 \\ & -8x + 10y \leq 13 \\ & x, y \geq 0 \\ & x, y \in \mathbb{Z} \end{aligned}$$

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

- The optimal solution of its LP relaxation is $(x, y) = (4, 4.5)$, with objective 8.5.
- No direct way of getting from $(4, 4.5)$ to $(1, 2)$ by rounding!
- Something more elaborate is needed: **branch & bound**.

Branch & Bound

The optimal solution of

$$\begin{aligned} \max \quad & x + y \\ \text{s.t.} \quad & -2x + 2y \geq 1 \\ & -8x + 10y \leq 13 \\ & x, y \geq 0 \\ & x, y \in \mathbb{Z} \end{aligned}$$

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

- The optimal solution of its LP relaxation is $(x, y) = (4, 4.5)$, with objective 8.5.
- No direct way of getting from $(4, 4.5)$ to $(1, 2)$ by rounding!
- Something more elaborate is needed: **branch & bound**.

Branch & Bound

The optimal solution of

$$
\begin{aligned}
\max \quad & x + y \\
\text{s.t.} \quad & -2x + 2y \geq 1 \\
& -8x + 10y \leq 13 \\
& x, y \geq 0 \\
& x, y \in \mathbb{Z}
\end{aligned}
$$

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

- The optimal solution of its LP relaxation is $(x, y) = (4, 4.5)$, with objective 8.5.
- No direct way of getting from $(4, 4.5)$ to $(1, 2)$ by rounding!
- Something more elaborate is needed: **branch & bound**.

Branch & Bound

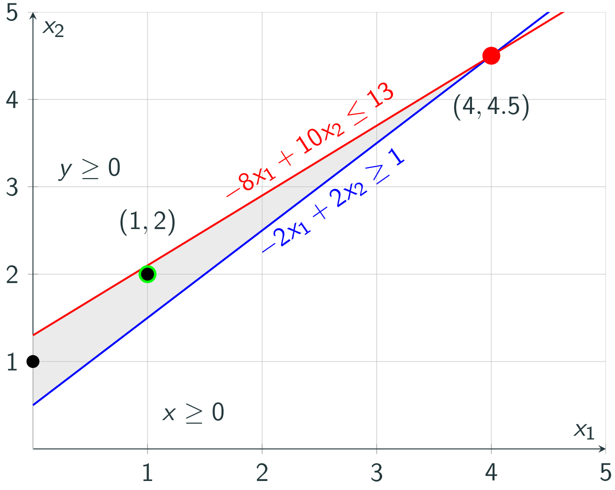
The optimal solution of

$$\begin{aligned} \max \quad & x + y \\ \text{s.t.} \quad & -2x + 2y \geq 1 \\ & -8x + 10y \leq 13 \\ & x, y \geq 0 \\ & x, y \in \mathbb{Z} \end{aligned}$$

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

- The optimal solution of its LP relaxation is $(x, y) = (4, 4.5)$, with objective 8.5.
- No direct way of getting from $(4, 4.5)$ to $(1, 2)$ by rounding!
- Something more elaborate is needed: **branch & bound**.

Branch & Bound



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 the optimal solution of $LP(P_0)$ has all integer variables taking integer values, then it is also an optimal solution to P_0.
- Else:
  - Let x_j be an integer variable whose value β_j at the optimal solution of $LP(P_0)$ is such that $\beta_j \notin \mathbb{Z}$.
  - Define:

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

  - $\text{feasibleSols}(P_0) = \text{feasibleSols}(P_1) \cup \text{feasibleSols}(P_2)$
  - Idea: solve P_1, solve P_2, and then take the best.

Branch & Bound

- Let x_j be an integer variable whose value β_j at the optimal solution of $LP(P_0)$ is such that $\beta_j \notin \mathbb{Z}$.
- Each of the problems

$$P_1 := P_0 \wedge x_j \leq \lfloor \beta_j \rfloor \qquad P_2 := P_0 \wedge x_j \geq \lceil \beta_j \rceil$$

  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 variables have finite bounds and, at each split, the domain of x_j becomes strictly smaller.
- If $LP(P_i)$ has an optimal solution where integer variables take integer values, then the solution is stored.
- If $LP(P_i)$ is infeasible, then P_i can be discarded (pruned, fathomed).

Branch & Bound Example

- Initial LP relaxation of P_0. Sol: $(4, 4.5)$, Obj: -8.5. Branch on y.
- Process $P_2(y \geq 5)$. Infeasible. Prune.
- Process $P_1(y \leq 4)$. Sol: $(3.5, 4)$, Obj: -7.5. Branch on x.
- Process $P_4(x \geq 4)$. Infeasible. Prune.
- Process $P_3(x \leq 3)$. Sol: $(3, 3.7)$, Obj: -6.7. Branch on y.
- Process $P_6(y \geq 4)$. Infeasible. Prune.
- Process $P_5(y \leq 3)$. Sol: $(2.5, 3)$, Obj: -5.5. Branch on x.
- Process $P_8(x \geq 3)$. Infeasible. Prune.
- Process $P_7(x \leq 2)$. Sol: $(2, 2.9)$, Obj: -4.9. Branch on y.
- Process $P_{10}(y \geq 3)$. Infeasible. Prune.
- Process $P_9(y \leq 2)$. Sol: $(1.5, 2)$, Obj: -3.5. Branch on x.
- Process $P_{12}(x \geq 2)$. Infeasible. Prune.
- Process $P_{11}(x \leq 1)$. Sol: $(1, 2)$, Obj: -3.0. **Integer Solution!**. Update best bound Z=3.0.

Branch & Bound Example

- Initial LP relaxation of P_0. Sol: $(4, 4.5)$, Obj: -8.5. Branch on y.
- Process $P_2(y \geq 5)$. Infeasible. Prune.
- Process $P_1(y \leq 4)$. Sol: $(3.5, 4)$, Obj: -7.5. Branch on x.
- Process $P_4(x \geq 4)$. Infeasible. Prune.
- Process $P_3(x \leq 3)$. Sol: $(3, 3.7)$, Obj: -6.7. Branch on y.
- Process $P_6(y \geq 4)$. Infeasible. Prune.
- Process $P_5(y \leq 3)$. Sol: $(2.5, 3)$, Obj: -5.5. Branch on x.
- Process $P_8(x \geq 3)$. Infeasible. Prune.
- Process $P_7(x \leq 2)$. Sol: $(2, 2.9)$, Obj: -4.9. Branch on y.
- Process $P_{10}(y \geq 3)$. Infeasible. Prune.
- Process $P_9(y \leq 2)$. Sol: $(1.5, 2)$, Obj: -3.5. Branch on x.
- Process $P_{12}(x \geq 2)$. Infeasible. Prune.
- Process $P_{11}(x \leq 1)$. Sol: $(1, 2)$, Obj: -3.0. **Integer Solution!**. Update best bound Z=3.0.

Branch & Bound Example

- Initial LP relaxation of P_0. Sol: $(4, 4.5)$, Obj: -8.5. Branch on y.
- Process $P_2(y \geq 5)$. Infeasible. Prune.
- Process $P_1(y \leq 4)$. Sol: $(3.5, 4)$, Obj: -7.5. Branch on x.
- Process $P_4(x \geq 4)$. Infeasible. Prune.
- Process $P_3(x \leq 3)$. Sol: $(3, 3.7)$, Obj: -6.7. Branch on y.
- Process $P_6(y \geq 4)$. Infeasible. Prune.
- Process $P_5(y \leq 3)$. Sol: $(2.5, 3)$, Obj: -5.5. Branch on x.
- Process $P_8(x \geq 3)$. Infeasible. Prune.
- Process $P_7(x \leq 2)$. Sol: $(2, 2.9)$, Obj: -4.9. Branch on y.
- Process $P_{10}(y \geq 3)$. Infeasible. Prune.
- Process $P_9(y \leq 2)$. Sol: $(1.5, 2)$, Obj: -3.5. Branch on x.
- Process $P_{12}(x \geq 2)$. Infeasible. Prune.
- Process $P_{11}(x \leq 1)$. Sol: $(1, 2)$, Obj: -3.0. **Integer Solution!**. Update best bound Z=3.0.

Branch & Bound Example

- Initial LP relaxation of P_0. Sol: $(4, 4.5)$, Obj: -8.5. Branch on y.
- Process $P_2(y \geq 5)$. Infeasible. Prune.
- Process $P_1(y \leq 4)$. Sol: $(3.5, 4)$, Obj: -7.5. Branch on x.
- Process $P_4(x \geq 4)$. Infeasible. Prune.
- Process $P_3(x \leq 3)$. Sol: $(3, 3.7)$, Obj: -6.7. Branch on y.
- Process $P_6(y \geq 4)$. Infeasible. Prune.
- Process $P_5(y \leq 3)$. Sol: $(2.5, 3)$, Obj: -5.5. Branch on x.
- Process $P_8(x \geq 3)$. Infeasible. Prune.
- Process $P_7(x \leq 2)$. Sol: $(2, 2.9)$, Obj: -4.9. Branch on y.
- Process $P_{10}(y \geq 3)$. Infeasible. Prune.
- Process $P_9(y \leq 2)$. Sol: $(1.5, 2)$, Obj: -3.5. Branch on x.
- Process $P_{12}(x \geq 2)$. Infeasible. Prune.
- Process $P_{11}(x \leq 1)$. Sol: $(1, 2)$, Obj: -3.0. **Integer Solution!**. Update best bound Z=3.0.

Branch & Bound Example

- Initial LP relaxation of P_0. Sol: $(4, 4.5)$, Obj: -8.5. Branch on y.
- Process $P_2(y \geq 5)$. Infeasible. Prune.
- Process $P_1(y \leq 4)$. Sol: $(3.5, 4)$, Obj: -7.5. Branch on x.
- Process $P_4(x \geq 4)$. Infeasible. Prune.
- Process $P_3(x \leq 3)$. Sol: $(3, 3.7)$, Obj: -6.7. Branch on y.
- Process $P_6(y \geq 4)$. Infeasible. Prune.
- Process $P_5(y \leq 3)$. Sol: $(2.5, 3)$, Obj: -5.5. Branch on x.
- Process $P_8(x \geq 3)$. Infeasible. Prune.
- Process $P_7(x \leq 2)$. Sol: $(2, 2.9)$, Obj: -4.9. Branch on y.
- Process $P_{10}(y \geq 3)$. Infeasible. Prune.
- Process $P_9(y \leq 2)$. Sol: $(1.5, 2)$, Obj: -3.5. Branch on x.
- Process $P_{12}(x \geq 2)$. Infeasible. Prune.
- Process $P_{11}(x \leq 1)$. Sol: $(1, 2)$, Obj: -3.0. **Integer Solution!**. Update best bound Z=3.0.

Branch & Bound Example

- Initial LP relaxation of P_0. Sol: $(4, 4.5)$, Obj: -8.5. Branch on y.
- Process $P_2(y \geq 5)$. Infeasible. Prune.
- Process $P_1(y \leq 4)$. Sol: $(3.5, 4)$, Obj: -7.5. Branch on x.
- Process $P_4(x \geq 4)$. Infeasible. Prune.
- Process $P_3(x \leq 3)$. Sol: $(3, 3.7)$, Obj: -6.7. Branch on y.
- Process $P_6(y \geq 4)$. Infeasible. Prune.
- Process $P_5(y \leq 3)$. Sol: $(2.5, 3)$, Obj: -5.5. Branch on x.
- Process $P_8(x \geq 3)$. Infeasible. Prune.
- Process $P_7(x \leq 2)$. Sol: $(2, 2.9)$, Obj: -4.9. Branch on y.
- Process $P_{10}(y \geq 3)$. Infeasible. Prune.
- Process $P_9(y \leq 2)$. Sol: $(1.5, 2)$, Obj: -3.5. Branch on x.
- Process $P_{12}(x \geq 2)$. Infeasible. Prune.
- Process $P_{11}(x \leq 1)$. Sol: $(1, 2)$, Obj: -3.0. **Integer Solution!**. Update best bound Z=3.0.

Branch & Bound Example

- Initial LP relaxation of P_0. Sol: $(4, 4.5)$, Obj: -8.5. Branch on y.
- Process $P_2(y \geq 5)$. Infeasible. Prune.
- Process $P_1(y \leq 4)$. Sol: $(3.5, 4)$, Obj: -7.5. Branch on x.
- Process $P_4(x \geq 4)$. Infeasible. Prune.
- Process $P_3(x \leq 3)$. Sol: $(3, 3.7)$, Obj: -6.7. Branch on y.
- Process $P_6(y \geq 4)$. Infeasible. Prune.
- Process $P_5(y \leq 3)$. Sol: $(2.5, 3)$, Obj: -5.5. Branch on x.
- Process $P_8(x \geq 3)$. Infeasible. Prune.
- Process $P_7(x \leq 2)$. Sol: $(2, 2.9)$, Obj: -4.9. Branch on y.
- Process $P_{10}(y \geq 3)$. Infeasible. Prune.
- Process $P_9(y \leq 2)$. Sol: $(1.5, 2)$, Obj: -3.5. Branch on x.
- Process $P_{12}(x \geq 2)$. Infeasible. Prune.
- Process $P_{11}(x \leq 1)$. Sol: $(1, 2)$, Obj: -3.0. **Integer Solution!**. Update best bound Z=3.0.

Branch & Bound Example

- Initial LP relaxation of P_0. Sol: $(4, 4.5)$, Obj: -8.5. Branch on y.
- Process $P_2(y \geq 5)$. Infeasible. Prune.
- Process $P_1(y \leq 4)$. Sol: $(3.5, 4)$, Obj: -7.5. Branch on x.
- Process $P_4(x \geq 4)$. Infeasible. Prune.
- Process $P_3(x \leq 3)$. Sol: $(3, 3.7)$, Obj: -6.7. Branch on y.
- Process $P_6(y \geq 4)$. Infeasible. Prune.
- Process $P_5(y \leq 3)$. Sol: $(2.5, 3)$, Obj: -5.5. Branch on x.
- Process $P_8(x \geq 3)$. Infeasible. Prune.
- Process $P_7(x \leq 2)$. Sol: $(2, 2.9)$, Obj: -4.9. Branch on y.
- Process $P_{10}(y \geq 3)$. Infeasible. Prune.
- Process $P_9(y \leq 2)$. Sol: $(1.5, 2)$, Obj: -3.5. Branch on x.
- Process $P_{12}(x \geq 2)$. Infeasible. Prune.
- Process $P_{11}(x \leq 1)$. Sol: $(1, 2)$, Obj: -3.0. **Integer Solution!**. Update best bound Z=3.0.

Branch & Bound Example

- Initial LP relaxation of P_0. Sol: $(4, 4.5)$, Obj: -8.5. Branch on y.
- Process $P_2(y \geq 5)$. Infeasible. Prune.
- Process $P_1(y \leq 4)$. Sol: $(3.5, 4)$, Obj: -7.5. Branch on x.
- Process $P_4(x \geq 4)$. Infeasible. Prune.
- Process $P_3(x \leq 3)$. Sol: $(3, 3.7)$, Obj: -6.7. Branch on y.
- Process $P_6(y \geq 4)$. Infeasible. Prune.
- Process $P_5(y \leq 3)$. Sol: $(2.5, 3)$, Obj: -5.5. Branch on x.
- Process $P_8(x \geq 3)$. Infeasible. Prune.
- Process $P_7(x \leq 2)$. Sol: $(2, 2.9)$, Obj: -4.9. Branch on y.
- Process $P_{10}(y \geq 3)$. Infeasible. Prune.
- Process $P_9(y \leq 2)$. Sol: $(1.5, 2)$, Obj: -3.5. Branch on x.
- Process $P_{12}(x \geq 2)$. Infeasible. Prune.
- Process $P_{11}(x \leq 1)$. Sol: $(1, 2)$, Obj: -3.0. **Integer Solution!**. Update best bound Z=3.0.

Branch & Bound Example

- Initial LP relaxation of P_0. Sol: $(4, 4.5)$, Obj: -8.5. Branch on y.
- Process $P_2(y \geq 5)$. Infeasible. Prune.
- Process $P_1(y \leq 4)$. Sol: $(3.5, 4)$, Obj: -7.5. Branch on x.
- Process $P_4(x \geq 4)$. Infeasible. Prune.
- Process $P_3(x \leq 3)$. Sol: $(3, 3.7)$, Obj: -6.7. Branch on y.
- Process $P_6(y \geq 4)$. Infeasible. Prune.
- Process $P_5(y \leq 3)$. Sol: $(2.5, 3)$, Obj: -5.5. Branch on x.
- Process $P_8(x \geq 3)$. Infeasible. Prune.
- Process $P_7(x \leq 2)$. Sol: $(2, 2.9)$, Obj: -4.9. Branch on y.
- Process $P_{10}(y \geq 3)$. Infeasible. Prune.
- Process $P_9(y \leq 2)$. Sol: $(1.5, 2)$, Obj: -3.5. Branch on x.
- Process $P_{12}(x \geq 2)$. Infeasible. Prune.
- Process $P_{11}(x \leq 1)$. Sol: $(1, 2)$, Obj: -3.0. **Integer Solution!**. Update best bound Z=3.0.

Branch & Bound Example

- Initial LP relaxation of P_0. Sol: $(4, 4.5)$, Obj: -8.5. Branch on y.
- Process $P_2(y \geq 5)$. Infeasible. Prune.
- Process $P_1(y \leq 4)$. Sol: $(3.5, 4)$, Obj: -7.5. Branch on x.
- Process $P_4(x \geq 4)$. Infeasible. Prune.
- Process $P_3(x \leq 3)$. Sol: $(3, 3.7)$, Obj: -6.7. Branch on y.
- Process $P_6(y \geq 4)$. Infeasible. Prune.
- Process $P_5(y \leq 3)$. Sol: $(2.5, 3)$, Obj: -5.5. Branch on x.
- Process $P_8(x \geq 3)$. Infeasible. Prune.
- Process $P_7(x \leq 2)$. Sol: $(2, 2.9)$, Obj: -4.9. Branch on y.
- Process $P_{10}(y \geq 3)$. Infeasible. Prune.
- Process $P_9(y \leq 2)$. Sol: $(1.5, 2)$, Obj: -3.5. Branch on x.
- Process $P_{12}(x \geq 2)$. Infeasible. Prune.
- Process $P_{11}(x \leq 1)$. Sol: $(1, 2)$, Obj: -3.0. **Integer Solution!**. Update best bound Z=3.0.

Branch & Bound Example

- Initial LP relaxation of P_0. Sol: $(4, 4.5)$, Obj: -8.5. Branch on y.
- Process $P_2(y \geq 5)$. Infeasible. Prune.
- Process $P_1(y \leq 4)$. Sol: $(3.5, 4)$, Obj: -7.5. Branch on x.
- Process $P_4(x \geq 4)$. Infeasible. Prune.
- Process $P_3(x \leq 3)$. Sol: $(3, 3.7)$, Obj: -6.7. Branch on y.
- Process $P_6(y \geq 4)$. Infeasible. Prune.
- Process $P_5(y \leq 3)$. Sol: $(2.5, 3)$, Obj: -5.5. Branch on x.
- Process $P_8(x \geq 3)$. Infeasible. Prune.
- Process $P_7(x \leq 2)$. Sol: $(2, 2.9)$, Obj: -4.9. Branch on y.
- Process $P_{10}(y \geq 3)$. Infeasible. Prune.
- Process $P_9(y \leq 2)$. Sol: $(1.5, 2)$, Obj: -3.5. Branch on x.
- Process $P_{12}(x \geq 2)$. Infeasible. Prune.
- Process $P_{11}(x \leq 1)$. Sol: $(1, 2)$, Obj: -3.0. **Integer Solution!**. Update best bound Z=3.0.

Branch & Bound Example

- Initial LP relaxation of P_0. Sol: $(4, 4.5)$, Obj: -8.5. Branch on y.
- Process $P_2(y \geq 5)$. Infeasible. Prune.
- Process $P_1(y \leq 4)$. Sol: $(3.5, 4)$, Obj: -7.5. Branch on x.
- Process $P_4(x \geq 4)$. Infeasible. Prune.
- Process $P_3(x \leq 3)$. Sol: $(3, 3.7)$, Obj: -6.7. Branch on y.
- Process $P_6(y \geq 4)$. Infeasible. Prune.
- Process $P_5(y \leq 3)$. Sol: $(2.5, 3)$, Obj: -5.5. Branch on x.
- Process $P_8(x \geq 3)$. Infeasible. Prune.
- Process $P_7(x \leq 2)$. Sol: $(2, 2.9)$, Obj: -4.9. Branch on y.
- Process $P_{10}(y \geq 3)$. Infeasible. Prune.
- Process $P_9(y \leq 2)$. Sol: $(1.5, 2)$, Obj: -3.5. Branch on x.
- Process $P_{12}(x \geq 2)$. Infeasible. Prune.
- Process $P_{11}(x \leq 1)$. Sol: $(1, 2)$, Obj: -3.0. **Integer Solution!**. Update best bound Z=3.0.

Pruning in Branch & Bound

- We have already seen that if a relaxation is infeasible, the problem can be pruned.
- Now assume an (integral) solution has been previously found.
- If the solution has cost Z, then any pending problem P_j whose relaxation has an optimal value $\geq Z$ (for a min problem) can be ignored, since

$$\text{cost}(P_j) \geq \text{cost}(LP(P_j)) \geq Z$$

- 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

- $S := \{P_0\}$ */* set of pending problems */*
- $Z := +\infty$ */* best cost found so far */*
- **while** $S \neq \emptyset$ **do**
-   remove P from S
-   solve $LP(P)$
-   **if** $LP(P)$ is feasible **then**
-     let β be the optimal basic solution of $LP(P)$
-     **if** β satisfies integrality constraints **then**
-       **if** cost(β) $<$ Z **then** store β; update Z
-     **else**
-       **if** cost($LP(P)$) $\geq$ Z **then** continue */* P can be pruned */*
-       let x_j be an integer variable such that $\beta_j \notin \mathbb{Z}$
-       $S := S \cup \{P \wedge x_j \leq \lfloor \beta_j \rfloor, P \wedge x_j \geq \lceil \beta_j \rceil\}$
- **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 the best cost value.

- **Choice of the branching variable:** one that is
  - closest to halfway between 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!

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 the best cost value.

- **Choice of the branching variable:** one that is
  - closest to halfway between 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!

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 the best cost value.

- **Choice of the branching variable:** one that is
  - closest to halfway between 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.
- After solving the relaxation of P, we have to solve the relaxations of $P \wedge x_j \leq \lfloor \beta_j \rfloor$ and $P \wedge x_j \geq \lceil \beta_j \rceil$. Do we start from scratch?
- Idea: start from the optimal solution of the parent problem. We add a new constraint, which makes the parent's optimal basis primal infeasible but dual feasible.
- **Dual simplex method** can be used for reoptimization. It is often very fast.

Remarks on Branch & Bound

- If integer variables are not bounded, Branch & Bound may not terminate.
- After solving the relaxation of P, we have to solve the relaxations of $P \wedge x_j \leq \lfloor \beta_j \rfloor$ and $P \wedge x_j \geq \lceil \beta_j \rceil$. Do we start from scratch?
- Idea: start from the optimal solution of the parent problem. We add a new constraint, which makes the parent's optimal basis primal infeasible but dual feasible.
- **Dual simplex method** can be used for reoptimization. It is often very fast.

Remarks on Branch & Bound

- If integer variables are not bounded, Branch & Bound may not terminate.
- After solving the relaxation of P, we have to solve the relaxations of $P \wedge x_j \leq \lfloor \beta_j \rfloor$ and $P \wedge x_j \geq \lceil \beta_j \rceil$. Do we start from scratch?
- Idea: start from the optimal solution of the parent problem. We add a new constraint, which makes the parent's optimal basis primal infeasible but dual feasible.
- **Dual simplex method** can be used for reoptimization. It is often very fast.

Remarks on Branch & Bound

- If integer variables are not bounded, Branch & Bound may not terminate.
- After solving the relaxation of P, we have to solve the relaxations of $P \wedge x_j \leq \lfloor \beta_j \rfloor$ and $P \wedge x_j \geq \lceil \beta_j \rceil$. Do we start from scratch?
- Idea: start from the optimal solution of the parent problem. We add a new constraint, which makes the parent's optimal basis primal infeasible but dual feasible.
- **Dual simplex method** can be used for reoptimization. It is often very fast.

Cutting Planes

Cutting Planes

- A **cut** is a linear inequality $p^T x \leq q$ that is satisfied by all integer solutions but violated by the current fractional solution β.
- By adding cuts, we tighten the LP relaxation without removing any feasible integer solutions.
- **Branch & Cut**: a hybrid method where cuts are added at nodes of the B&B tree to improve bounds and prune more of the tree.

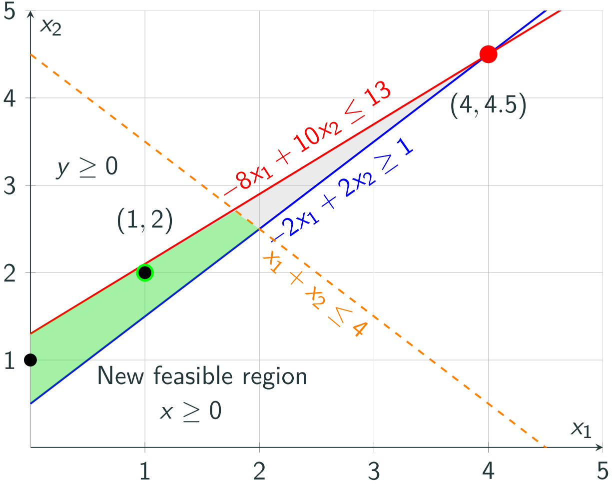
Cutting Planes

- A **cut** is a linear inequality $p^T x \leq q$ that is satisfied by all integer solutions but violated by the current fractional solution β.
- By adding cuts, we tighten the LP relaxation without removing any feasible integer solutions.
- **Branch & Cut**: a hybrid method where cuts are added at nodes of the B&B tree to improve bounds and prune more of the tree.

Cutting Planes

- A **cut** is a linear inequality $p^T x \leq q$ that is satisfied by all integer solutions but violated by the current fractional solution β.
- By adding cuts, we tighten the LP relaxation without removing any feasible integer solutions.
- **Branch & Cut**: a hybrid method where cuts are added at nodes of the B&B tree to improve bounds and prune more of the tree.

Cut



5
4
3
2
1
x_2
x_1
1
2
3
4
5
$-8x_1 + 10x_2 \leq 13$
$-2x_1 + 2x_2 \geq 1$
$x_1 + x_2 \leq 4$
$(4, 4.5)$
$(1, 2)$
$y \geq 0$
New feasible region
$x \geq 0$

Gomory Cuts

- Gomory cuts are a general method for generating cutting planes.
- From a simplex tableau row for a basic variable x_i with a fractional value β_i:
$$x_i = \beta_i + \sum_{j \in \mathcal{R}} \alpha_{ij} x_j$$
where $\mathcal{R}$ are the non-basic variables.
- We can rewrite this as:
$$x_i - \sum_{j \in \mathcal{R}} \lfloor \alpha_{ij} \rfloor x_j = \lfloor \beta_i \rfloor + \{\beta_i\} + \sum_{j \in \mathcal{R}} \{\alpha_{ij}\} x_j$$
where $\{\cdot\}$ is the fractional part.
- Since the LHS is integer, the RHS must be too. This leads to the Gomory cut:
$$\sum_{j \in \mathcal{R}} \{\alpha_{ij}\} x_j \geq \{\beta_i\}$$
This cut is violated by the current solution (where $x_j = 0$ for non-basic j).

Gomory Cuts

- Gomory cuts are a general method for generating cutting planes.
- From a simplex tableau row for a basic variable x_i with a fractional value β_i:

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

  where $\mathcal{R}$ are the non-basic variables.
- We can rewrite this as:

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

  where $\{\cdot\}$ is the fractional part.
- Since the LHS is integer, the RHS must be too. This leads to the Gomory cut:

$$\sum_{j \in \mathcal{R}} \{\alpha_{ij}\} x_j \geq \{\beta_i\}$$

  This cut is violated by the current solution (where $x_j = 0$ for non-basic j).

Gomory Cuts

- Gomory cuts are a general method for generating cutting planes.
- From a simplex tableau row for a basic variable x_i with a fractional value β_i:
$$x_i = \beta_i + \sum_{j \in \mathcal{R}} \alpha_{ij} x_j$$
where $\mathcal{R}$ are the non-basic variables.
- We can rewrite this as:
$$x_i - \sum_{j \in \mathcal{R}} \lfloor \alpha_{ij} \rfloor x_j = \lfloor \beta_i \rfloor + \{\beta_i\} + \sum_{j \in \mathcal{R}} \{\alpha_{ij}\} x_j$$
where $\{\cdot\}$ is the fractional part.
- Since the LHS is integer, the RHS must be too. This leads to the Gomory cut:
$$\sum_{j \in \mathcal{R}} \{\alpha_{ij}\} x_j \geq \{\beta_i\}$$
This cut is violated by the current solution (where $x_j = 0$ for non-basic j).

Gomory Cuts

- Gomory cuts are a general method for generating cutting planes.
- From a simplex tableau row for a basic variable x_i with a fractional value β_i:
$$x_i = \beta_i + \sum_{j \in \mathcal{R}} \alpha_{ij} x_j$$
where $\mathcal{R}$ are the non-basic variables.
- We can rewrite this as:
$$x_i - \sum_{j \in \mathcal{R}} \lfloor \alpha_{ij} \rfloor x_j = \lfloor \beta_i \rfloor + \{\beta_i\} + \sum_{j \in \mathcal{R}} \{\alpha_{ij}\} x_j$$
where $\{\cdot\}$ is the fractional part.
- Since the LHS is integer, the RHS must be too. This leads to the Gomory cut:
$$\sum_{j \in \mathcal{R}} \{\alpha_{ij}\} x_j \geq \{\beta_i\}$$
This cut is violated by the current solution (where $x_j = 0$ for non-basic j).

Total Unimodularity

Ensuring All Vertices Are Integer

- Consider an IP of the form:

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t.} \quad & Ax = b \\ & x \geq 0 \\ & x \in \mathbb{Z}^n \end{aligned}$$

- Let us assume A and b have integer coefficients.
- Are there any sufficient conditions to ensure that the simplex algorithm will directly provide an integer solution, without needing branch & bound or cuts?

Ensuring All Vertices Are Integer

- Consider an IP of the form:

$$\begin{aligned} \min \quad & c^T x \\ \text{s.t.} \quad & Ax = b \\ & x \geq 0 \\ & x \in \mathbb{Z}^n \end{aligned}$$

- Let us assume A and b have integer coefficients.
- Are there any sufficient conditions to ensure that the simplex algorithm will directly provide an integer solution, without needing branch & bound or cuts?

Ensuring All Vertices Are Integer

- We will see sufficient conditions to ensure that all vertices of the relaxation are integer.
- For instance, this occurs when the matrix A is **totally unimodular**: the determinant of every square submatrix is 0, +1, or -1.

Ensuring All Vertices Are Integer

- We will see sufficient conditions to ensure that all vertices of the relaxation are integer.
- For instance, this occurs when the matrix A is **totally unimodular**: the determinant of every square submatrix is 0, $+1$, or -1.

Ensuring All Vertices Are Integer

- If A is totally unimodular, all bases have inverses with integer coefficients.
- Recall Cramer's rule: if B is an invertible matrix, then

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

  where $\operatorname{adj}(B)$ is the adjugate matrix of B.
- The adjugate matrix is composed of determinants of submatrices, which are integers if A is an integer matrix.
- If $\det(B)$ is ± 1 and the coefficients of A and b are integers, then the basic solution $x_B = B^{-1} b$ will be integer.

Ensuring All Vertices Are Integer

- If A is totally unimodular, 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.
- The adjugate matrix is composed of determinants of submatrices, which are integers if A is an integer matrix.
- If $\det(B)$ is ± 1 and the coefficients of A and b are integers, then the basic solution $x_B = B^{-1} b$ will be integer.

Ensuring All Vertices Are Integer

- If A is totally unimodular, 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.
- The adjugate matrix is composed of determinants of submatrices, which are integers if A is an integer matrix.
- If $\det(B)$ is ± 1 and the coefficients of A and b are integers, then the basic solution $x_B = B^{-1}b$ will be integer.

Ensuring All Vertices Are Integer

- If A is totally unimodular, all bases have inverses with integer coefficients.
- Recall Cramer's rule: if B is an invertible matrix, then

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

  where $\mathrm{adj}(B)$ is the adjugate matrix of B.
- The adjugate matrix is composed of determinants of submatrices, which are integers if A is an integer matrix.
- If $\det(B)$ is ± 1 and the coefficients of A and b are integers, then the basic solution $x_B = B^{-1}b$ will be integer.

Ensuring All Vertices Are Integer

Sufficient condition for total unimodularity (Hoffman & Gale's Theorem)

A matrix A is totally unimodular if it satisfies the following conditions:

1. Each element of A is 0, $+1$, or -1.
2. No more than two non-zeroes appear in each column.
3. The rows can be partitioned into two subsets, R_1 and R_2, such that for each column:
   - If the column contains two non-zeroes of the **same sign**, the row of one is in R_1 and the row of the other is in R_2.
   - If the column contains two non-zeroes of **different signs**, the rows of both are in the same subset.

Applications of Total Unimodularity

- Several kinds of IPs satisfy Hoffman & Gale's conditions:
  - Assignment
  - Transportation
  - Maximum flow
  - Shortest path
- Usually, ad-hoc graph algorithms are more efficient for these problems than the simplex method as presented here.
- **But:**
  - The simplex method can be specialized (e.g., the network simplex method).
  - Simplex techniques can be applied if the problem is not a purely network one but has extra constraints.

Applications of Total Unimodularity

- Several kinds of IPs satisfy Hoffman & Gale's conditions:
  - Assignment
  - Transportation
  - Maximum flow
  - Shortest path
- Usually, ad-hoc graph algorithms are more efficient for these problems than the simplex method as presented here.
- **But:**
  - The simplex method can be specialized (e.g., the network simplex method).
  - Simplex techniques can be applied if the problem is not a purely network one but has extra constraints.

Modeling

Modeling with 0-1 Variables

- Sometimes we want to have an indicator variable of a constraint: a 0/1 variable equal to 1 if and only if the constraint is true.
- E.g., let's encode $\delta = 1 \leftrightarrow a^T x \leq b$, where δ is a 0/1 variable.
- Assume $a^T x$ is an integer for all feasible solutions 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. $\delta = 1 \rightarrow a^T x \leq b$
   - This can be encoded with the "big-M" constraint: $a^T x - b \leq U(1 - \delta)$

2. $a^T x \leq b \rightarrow \delta = 1$
   - This is equivalent to $\delta = 0 \rightarrow a^T x > b$.
   - Since $a^T x$ is integer, this is $\delta = 0 \rightarrow a^T x \geq b + 1$.
   - This can be encoded with: $a^T x - b \geq (L - 1)\delta + 1$

Modeling with 0-1 Variables

- Sometimes we want to have an indicator variable of a constraint: a 0/1 variable equal to 1 if and only if the constraint is true.
- E.g., let's encode $\delta = 1 \leftrightarrow a^T x \leq b$, where δ is a 0/1 variable.
- Assume $a^T x$ is an integer for all feasible solutions 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. $\delta = 1 \rightarrow a^T x \leq b$
   - This can be encoded with the "big-M" constraint: $a^T x - b \leq U(1 - \delta)$
2. $a^T x \leq b \rightarrow \delta = 1$
   - This is equivalent to $\delta = 0 \rightarrow a^T x > b$.
   - Since $a^T x$ is integer, this is $\delta = 0 \rightarrow a^T x \geq b + 1$.
   - This can be encoded with: $a^T x - b \geq (L - 1)\delta + 1$

Modeling with 0-1 Variables

- Sometimes we want to have an indicator variable of a constraint: a 0/1 variable equal to 1 if and only if the constraint is true.
- E.g., let's encode $\delta = 1 \leftrightarrow a^T x \leq b$, where δ is a 0/1 variable.
- Assume $a^T x$ is an integer for all feasible solutions 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. $\delta = 1 \rightarrow a^T x \leq b$
   - This can be encoded with the "big-M" constraint: $a^T x - b \leq U(1 - \delta)$

2. $a^T x \leq b \rightarrow \delta = 1$
   - This is equivalent to $\delta = 0 \rightarrow a^T x > b$.
   - Since $a^T x$ is integer, this is $\delta = 0 \rightarrow a^T x \geq b + 1$.
   - This can be encoded with: $a^T x - b \geq (L - 1)\delta + 1$

Modeling with 0-1 Variables

- Sometimes we want to have an indicator variable of a constraint: a 0/1 variable equal to 1 if and only if the constraint is true.
- E.g., let's encode $\delta = 1 \leftrightarrow a^T x \leq b$, where δ is a 0/1 variable.
- Assume $a^T x$ is an integer for all feasible solutions 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. $\delta = 1 \rightarrow a^T x \leq b$
   - This can be encoded with the "big-M" constraint: $a^T x - b \leq U(1 - \delta)$
2. $a^T x \leq b \rightarrow \delta = 1$
   - This is equivalent to $\delta = 0 \rightarrow a^T x > b$.
   - Since $a^T x$ is integer, this is $\delta = 0 \rightarrow a^T x \geq b + 1$.
   - This can be encoded with: $a^T x - b \geq (L - 1)\delta + 1$

Modeling with 0-1 Variables

- Sometimes it is convenient to think of constraints from a logical perspective, and then translate them into linear inequalities.
- If $x_1, ..., x_n, y_1, ..., y_m$ are 0/1 (Boolean) variables, then the logical clause:

$$x_1 \vee \ldots \vee x_n \vee \neg y_1 \vee \ldots \vee \neg y_m$$

is equivalent to the linear constraint:

$$x_1 + \cdots + x_n + (1 - y_1) + \cdots + (1 - y_m) \geq 1$$

Modeling with 0-1 Variables

- Sometimes it is convenient to think of constraints from a logical perspective, and then translate them into linear inequalities.
- If $x_1, ..., x_n, y_1, ..., y_m$ are 0/1 (Boolean) variables, then the logical clause:

$$x_1 \vee \ldots \vee x_n \vee \neg y_1 \vee \ldots \vee \neg y_m$$

  is equivalent to the linear constraint:

$$x_1 + \cdots + x_n + (1 - y_1) + \cdots + (1 - y_m) \geq 1$$

Examples

Example (Logical Constraints)

Problem

Let x_i represent "Ingredient i is in the blend", for $i \in \{A, B, C\}$.
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 $x_A \to (x_B \vee x_C)$.
- This is logically equivalent to $\neg x_A \vee x_B \vee x_C$.
- This is equivalent to the linear constraint $(1 - x_A) + x_B + x_C \geq 1$.
- Simplifying gives the final constraint: $x_B + x_C \geq x_A$.

Example (Logical Constraints)

Problem

Let x_i represent "Ingredient i is in the blend", for $i \in \{A, B, C\}$.
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 $x_A \rightarrow (x_B \vee x_C)$.
- This is logically equivalent to $\neg x_A \vee x_B \vee x_C$.
- This is equivalent to the linear constraint $(1 - x_A) + x_B + x_C \geq 1$.
- Simplifying gives the final constraint: $x_B + x_C \geq x_A$.

Example (Logical Constraints)

Problem

Let x_i represent "Ingredient i is in the blend", for $i \in \{A, B, C\}$.
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 $x_A \to (x_B \vee x_C)$.
- This is logically equivalent to $\neg x_A \vee x_B \vee x_C$.
- This is equivalent to the linear constraint $(1 - x_A) + x_B + x_C \geq 1$.
- Simplifying gives the final constraint: $x_B + x_C \geq x_A$.

Example (Logical Constraints)

Problem

Let x_i represent "Ingredient i is in the blend", for $i \in \{A, B, C\}$.
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 $x_A \rightarrow (x_B \vee x_C)$.
- This is logically equivalent to $\neg x_A \vee x_B \vee x_C$.
- This is equivalent to the linear constraint $(1 - x_A) + x_B + x_C \geq 1$.
- Simplifying gives the final constraint: $x_B + x_C \geq x_A$.

Example (Logical Constraints)

Problem

Let x_i represent "Ingredient i is in the blend", for $i \in \{A, B, C\}$.

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 $x_A \rightarrow (x_B \vee x_C)$.
- This is logically equivalent to $\neg x_A \vee x_B \vee x_C$.
- This is equivalent to the linear constraint $(1 - x_A) + x_B + x_C \geq 1$.
- Simplifying gives the final constraint: $x_B + x_C \geq x_A$.

Example (Fixed Setup Charge)

Problem

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. The total cost is:

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

How do we model this in a MIP?

- Let δ be a 0/1 variable such that $x > 0 \rightarrow \delta = 1$. This is equivalent to $\delta = 0 \rightarrow x \leq 0$.
- Add the constraint $x - U\delta \leq 0$, where U is a valid upper bound on x.
- The cost function to minimize becomes: $c_0\delta + c_1 x$.
- Note: We do not need to model the implication $\delta = 1 \rightarrow x > 0$. The minimization of cost will naturally force δ to be 0 whenever

Example (Fixed Setup Charge)

Problem

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. The total cost is:

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

How do we model this in a MIP?

- Let δ be a 0/1 variable such that $x > 0 \to \delta = 1$. This is equivalent to $\delta = 0 \to x \leq 0$.
- Add the constraint $x - U\delta \leq 0$, where U is a valid upper bound on x.
- The cost function to minimize becomes: $c_0\delta + c_1 x$.
- Note: We do not need to model the implication $\delta = 1 \to x > 0$. The minimization of cost will naturally force δ to be 0 whenever

Example (Fixed Setup Charge)

Problem

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. The total cost is:

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

How do we model this in a MIP?

- Let δ be a 0/1 variable such that $x > 0 \rightarrow \delta = 1$. This is equivalent to $\delta = 0 \rightarrow x \leq 0$.
- Add the constraint $x - U\delta \leq 0$, where U is a valid upper bound on x.
- The cost function to minimize becomes: $c_0\delta + c_1 x$.
- Note: We do not need to model the implication $\delta = 1 \rightarrow x > 0$. The minimization of cost will naturally force δ to be 0 whenever

Example (Fixed Setup Charge)

Problem

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. The total cost is:

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

How do we model this in a MIP?

- Let δ be a $0/1$ variable such that $x > 0 \rightarrow \delta = 1$. This is equivalent to $\delta = 0 \rightarrow x \leq 0$.
- Add the constraint $x - U\delta \leq 0$, where U is a valid upper bound on x.
- The cost function to minimize becomes: $c_0\delta + c_1 x$.
- Note: We do not need to model the implication $\delta = 1 \rightarrow x > 0$. The minimization of cost will naturally force δ to be 0 whenever

Example (Fixed Setup Charge)

Problem

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. The total cost is:

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

How do we model this in a MIP?

- Let δ be a 0/1 variable such that $x > 0 \to \delta = 1$. This is equivalent to $\delta = 0 \to x \leq 0$.
- Add the constraint $x - U\delta \leq 0$, where U is a valid upper bound on x.
- The cost function to minimize becomes: $c_0\delta + c_1 x$.
- Note: We do not need to model the implication $\delta = 1 \to x > 0$. The minimization of cost will naturally force δ to be 0 whenever

Example (Capacity Expansion)

Problem

The consumption of a resource is $a^T x$. We want to relax the constraint $a^T x \leq b$ by increasing the capacity b. We can choose from discrete capacity levels

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

with corresponding costs

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

How do we model this choice?

Example (Capacity Expansion)

- Let δ_i be a $0/1$ variable for choosing capacity level b_i.
- Choose exactly one capacity level:

$$\sum_{i=0}^{t} \delta_i = 1$$

- The resource constraint becomes:

$$a^T x \leq \sum_{i=0}^{t} b_i \delta_i$$

- The cost to be minimized must include the expansion cost:

$$\cdots + \sum_{i=0}^{t} c_i \delta_i$$

Example (Capacity Expansion)

- Let δ_i be a $0/1$ variable for choosing capacity level b_i.
- Choose exactly one capacity level:

$$\sum_{i=0}^{t} \delta_i = 1$$

- The resource constraint becomes:

$$a^T x \leq \sum_{i=0}^{t} b_i \delta_i$$

- The cost to be minimized must include the expansion cost:

$$\cdots + \sum_{i=0}^{t} c_i \delta_i$$

Example (Capacity Expansion)

- Let δ_i be a $0/1$ variable for choosing capacity level b_i.
- Choose exactly one capacity level:

$$\sum_{i=0}^{t} \delta_i = 1$$

- The resource constraint becomes:

$$a^T x \leq \sum_{i=0}^{t} b_i \delta_i$$

- The cost to be minimized must include the expansion cost:

$$\cdots + \sum_{i=0}^{t} c_i \delta_i$$

Example (Capacity Expansion)

- Let δ_i be a $0/1$ variable for choosing capacity level b_i.
- Choose exactly one capacity level:

$$\sum_{i=0}^{t} \delta_i = 1$$

- The resource constraint becomes:

$$a^T x \leq \sum_{i=0}^{t} b_i \delta_i$$

- The cost to be minimized must include the expansion cost:

$$\cdots + \sum_{i=0}^{t} c_i \delta_i$$