Branch-and-bound

YANG, Hanbin

School of Data Science (SDS)
The Chinese University of Hong Kong, Shenzhen

May 5, 2023

Modeling

Introduction: Integer Programming

$$\begin{aligned}
\min / \max \quad & c_1 x_1 + c_2 x_2 + \cdots + c_n x_n \\
\text{s.t.} \quad & a_{i1} x_1 + a_{i2} x_2 + \cdots + a_{in} x_n \left\{ \begin{matrix} \leq \\ = \\ \geq \end{matrix} \right\} b_i && i = 1, 2, \ldots, m \\
& 0 \leq x_j \leq u_j && j = 1, 2, \ldots, n \\
& x_j \text{ is integer for some or all } j = 1, 2, \ldots, n.
\end{aligned}$$

- Mixed integer program (MIP)
- Pure integer program
- Binary decision variables
- Binary integer program

Motivation: Why Should We Study IP?

- Sometimes we need to have integer variables due to realistic requirements: divisibility assumption not acceptable.
- Binary decisions used to model logical constraints.

We will see examples for both points.

Motivation: A Scheduling Problem

Employee scheduling: 6 shifts requiring full-time equivalent employees:

Shift	Time	FTEEs
1	0 - 4	15
2	4 - 8	10
3	8 - 12	40
4	12 - 16	70
5	16 - 20	40
6	20 - 0	35

- Full-time employees: 8-hour shifts, \$15.20 per hour wage.
- Part-time employees, 4-hour shifts, \$12.95 per hour wage.
- At least $\frac{2}{3}$ employees working any time must be full-time.
- Part-time answers only 5 calls in the time full-time answers 6 calls.
- Minimize staff costs.

Motivation: A Scheduling Problem

$$\min_{x,y} \quad 121.6(x_1 + x_2 + \cdots + x_6) + 51.8(y_1 + y_2 + \cdots + y_6)$$

s.t.

$x_6 + x_1 + \frac{5}{6}y_1 \geq 15$	(12am)	$x_6 + x_1 \geq \frac{2}{3}(x_6 + x_1 + y_1)$
$x_1 + x_2 + \frac{5}{6}y_2 \geq 10$	(4am)	$x_1 + x_2 \geq \frac{2}{3}(x_1 + x_2 + y_2)$
$x_2 + x_3 + \frac{5}{6}y_3 \geq 40$	(8am)	$x_2 + x_3 \geq \frac{2}{3}(x_2 + x_3 + y_3)$
$x_3 + x_4 + \frac{5}{6}y_4 \geq 70$	(12pm)	$x_3 + x_4 \geq \frac{2}{3}(x_3 + x_4 + y_4)$
$x_4 + x_5 + \frac{5}{6}y_5 \geq 40$	(4pm)	$x_4 + x_5 \geq \frac{2}{3}(x_4 + x_5 + y_5)$
$x_5 + x_6 + \frac{5}{6}y_6 \geq 35$	(8pm)	$x_5 + x_6 \geq \frac{2}{3}(x_5 + x_6 + y_6)$

$$x_t \geq 0, y_t \geq 0 \qquad t = 1, 2, \ldots, 6.$$

Motivation: A Scheduling Problem

An optimal solution to this LP:

	12am	4am	8am	12pm	4pm	8pm
Full-time employees	7.1	0	40	23.2	16.8	7.9
Part-time employees	0	3.5	0	8.1	0	12.4

with a total cost of \$12795.2. Note that the solution uses 95 full-time employees and 24 part-time employees.

Issue: not integer! We cannot hire half a person.

Motivation: A Scheduling Problem

Intuition: how about rounding the solution?

	12am	4am	8am	12pm	4pm	8pm
Full-time employees	8	0	40	24	17	8
Part-time employees	0	3	0	8	0	12

- Is it feasible? Check the demand requirements and two-thirds rules.
- What change could we expect in the objective value? Upper bound: \$12986.6

Motivation: A Scheduling Problem

- Is this solution optimal? What information could we get from this solution?
  - Lower bound: $z^*_{LP} = 12795.2$
  - Upper bound: $\hat{z} = 12986.6$
  - The optimal value: $z^*_{LP} \leq z^*_{IP} \leq \hat{z}$
  - Optimality gap:
$$\frac{\hat{z} - z^*_{IP}}{z^*_{IP}} \leq \frac{\hat{z} - z^*_{LP}}{z^*_{LP}} = 0.015$$
- The optimal solution: $z^*_{IP} = 12795.2$

	12am	4am	8am	12pm	4pm	8pm	Total
Full-time employees	10	0	40	10	30	5	95
Part-time employees	0	0	0	24	0	0	24

Motivation: A Scheduling Problem

What could happen in an alternative universe?

- Not always $z^*_{IP} = z^*_{LP}$. More often $z^*_{IP} > z^*_{LP}$.
- May not always have such a tight bound.
- May not be able to obtain a feasible solution by rounding.
- Simple heuristics may still be valuable.

Modeling Logical Constraints

- True or False $\rightarrow$ 0 or 1
- Use binary variables to identify which condition is true, $x, y, z \in \{0, 1\}$
- If x is true, then y is true: $x \leq y$
- Only one of x and y is true: $x + y = 1$
  What about at most one of them is true? At least one of them is true?
- y indicates whether $f(x) \leq b$ is violated:

$$f(x) \leq b + My$$

M is a sufficiently large number (parameter).

Wait... Sometimes It Can Be Both 0 and 1

y indicates whether $f(x) \leq b$ is violated:

$$f(x) \leq b + My$$

If $f(x) > b$, then y has to be 1. If $f(x) \leq b$, y can also be 1, but the statement is not violated.

$\rightarrow$ Usually we have other ways to "push" y to take the right value:

- When violated, there is a penalty term
- The sum of y has to be smaller than a constant

Either-Or Constraints

$$\text{Either } f_1(x_1, \ldots, x_n) \leq b_1,$$
$$\text{or } f_2(x_1, \ldots, x_n) \leq b_2;$$

$$y \in \{0, 1\}$$
$$f_1(x_1, \ldots, x_n) \leq b_1 + My$$
$$f_2(x_1, \ldots, x_n) \leq b_2 + M(1 - y)$$

- If $y = 0$ the first constraint must hold and the second constraint is vacuous.
- If $y = 1$ the second constraint must hold and the first constraint is vacuous.
- This is non-exclusive.

K-out-of-N Must Hold

Out of the N following constraints, K must be satisfied:

$$
\begin{aligned}
& f_1(x_1, \ldots, x_n) \leq b_1 \\
& f_2(x_1, \ldots, x_n) \leq b_2 \\
& \vdots \\
& f_N(x_1, \ldots, x_n) \leq b_N
\end{aligned}
$$

Use N binary variables $y_j,\ j = 1, \ldots, N$ to indicate whether constraint j holds.

K-out-of-N Must Hold

$$f_1(x_1, \ldots, x_n) \leq b_1 + M(1 - y_1)$$
$$f_2(x_1, \ldots, x_n) \leq b_2 + M(1 - y_2)$$
$$\vdots$$
$$f_N(x_1, \ldots, x_n) \leq b_N + M(1 - y_N)$$
$$\sum_{j=1}^{N} y_j = K$$
$$y_j \in \{0, 1\} \qquad j = 1, 2, \ldots, N.$$

Compound Alternatives

Suppose we have three regions and we can only select one region to operate. Each region will have its own operation constraints:

$$
\begin{aligned}
&\text{Region 1 Constraints} \begin{cases} f_1(x_1, \ldots, x_n) \leq b_1 + M(1 - y_1) \\ f_2(x_1, \ldots, x_n) \leq b_2 + M(1 - y_1) \end{cases} \\
&\text{Region 2 Constraints} \begin{cases} f_3(x_1, \ldots, x_n) \leq b_3 + M(1 - y_2) \\ f_4(x_1, \ldots, x_n) \leq b_4 + M(1 - y_2) \end{cases} \\
&\text{Region 3 Constraints} \begin{cases} f_5(x_1, \ldots, x_n) \leq b_5 + M(1 - y_3) \\ f_6(x_1, \ldots, x_n) \leq b_6 + M(1 - y_3) \end{cases} \\
&y_1 + y_2 + y_3 = 1 \\
&y_1, y_2, y_3 \in \{0, 1\}.
\end{aligned}
$$

If-then Constraints - Linearization

The **trick**

- Model the constraint: if $z = 0$ then $a^\top x \leq b$;
- Let M be an upper bound for $a^\top x - b$;
- Write $a^\top x - b \leq Mz$;
- If $z = 0$,then $a^\top x - b \leq 0$ asrequired.
  Otherwise, we get $a^\top x - b \leq M$, which is always true.

If-then Constraints - Linearization

Slight change: if $z = 1$ then $a^\top x \leq b$

- Again, let M be an upper bound for $a^\top x - b$;
- Write: $a^\top x - b \leq M(1 - z)$.

Reversed inequality: if $z = 0$ then $a^\top x \geq b$

- Write constraint as $-a^\top x + b \leq 0$;
- Let m be an upper bound on $-a^\top x + b$;
- Write: $-a^\top x + b \leq mz$. Same as: $a^\top x - b \geq -mz$;
- Note: $-m$ is a lower bound on $a^\top x - b$.

If-then Constraints - Linearization

The converse: if $a^\top x \leq b$ then $z = 1$

- Equivalent to: if $z = 0$ then $a^\top x > b$ (contrapositive);
- The strict inequality is not really enforceable. Instead, write: if $z = 0$ then $a^\top x \geq b + \varepsilon$ where ε is small;
- Let m be a lower bound for $a^\top x - b$ and we obtain the equivalent constraint: $a^\top x - b \geq mz + \varepsilon(1 - z)$;
- If $z = 0$, we get $a^\top x \geq b + \varepsilon$, as required. Otherwise, we get: $a^\top x - b \geq m$, which is always true.

Note: If a, x, b are integer-valued, we may set $\varepsilon = 1$.

If-then Constraints - Example

Example

If $T_1 \leq t \leq T_2$, then $y = 1$; otherwise, $y = 0$.

We need to use two anxiliary binary variables a and b:

- If $t \geq T_1$, then $a = 1$;
- If $t \leq T_2$, then $b = 1$;
- If $a + b = 2$, then $y = 1$.

Example: Knapsack Problem

We have a knapsack and we want to select from a set of items $i \in I$ to fill the knapsack.

- Knapsack has a weight capacity b.
- Each item weighs a_i and carries a value of c_i.
- We would like to maximize the total value of selected items, while keeping the weight below the knapsack capacity.

$$\begin{aligned} \max_{x} \quad & \sum_{i \in I} c_i x_i \\ \text{s.t.} \quad & \sum_{i \in I} a_i x_i \leq b \\ & x_i \in \{0, 1\} \qquad i \in I. \end{aligned}$$

Example: Knapsack Problem

Extensions:

- What if you can select an item more than once? $0 \leq x_i \leq u_i, \quad x_i \in \mathbb{Z}$.
- Knapsack problems have many other applications. For example, in capital budgeting, we want to decide whether we select a project to invest in.
  - Multiple years: $\sum_{i \in I} a_{it} x_i \leq b_t \qquad t \in T$
  - Some projects are mutually exclusive: $x_1 + x_2 \leq 1 \rightarrow \sum_{i \in S \subseteq I} x_i \leq 1$
  - At least one project has to be selected: $x_1 + x_2 \geq 1 \rightarrow \sum_{i \in S \subseteq I} x_i \geq 1$
  - Prerequisite: $x_1 \leq x_2, x_1 \leq x_3$

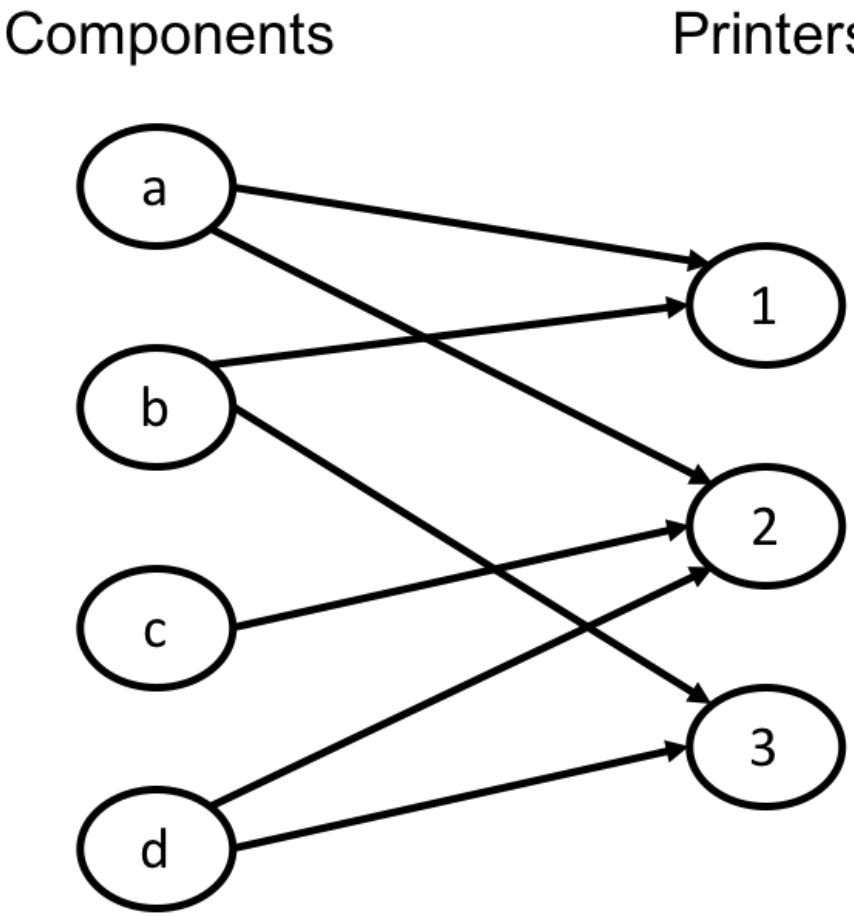
More Application Example: Selection Problem

HP would like to select a set of components to produce so that they can support as many printers of previous model as possible.

- A printer of type $j \in J$ uses a set of components $I_j \subseteq I$.
- There is a budget b for HP to produce/maintain inventory of components.
- Supporting the type j printer will bring a value of v_j.
- Maximize the value of supporting printers by selecting a set of components to produce, while obeying the budget constraint.

Key: identify this prerequisite relationship: supporting printer type j requires producing components in set I_j.

Example: Selection Problem



x_i: producing component

y_j: supporting printers

$$\max_{x,y} \quad \sum_{j \in J} v_j y_j$$

$$\text{s.t.} \quad y_j \leq x_i \qquad j \in J, i \in I_j$$

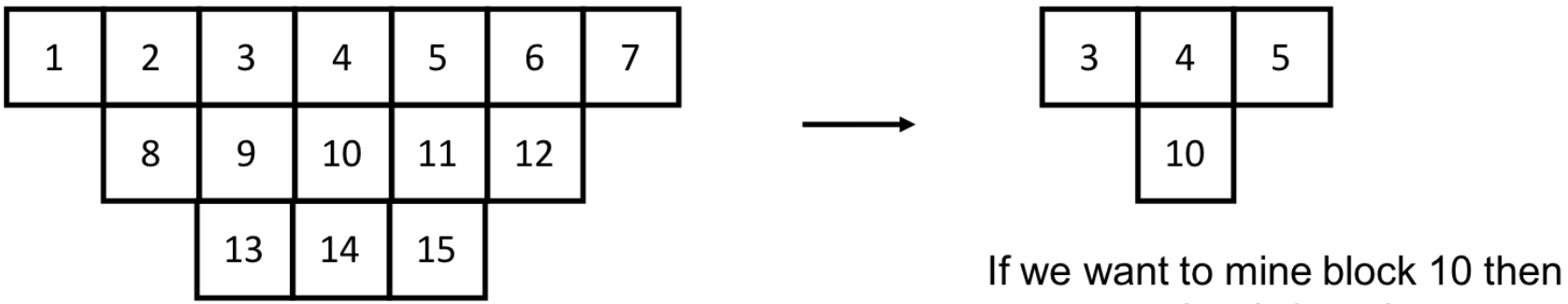
$$\sum_{i \in I} c_i x_i \leq b$$

$$x_i \in \{0,1\} \qquad i \in I$$

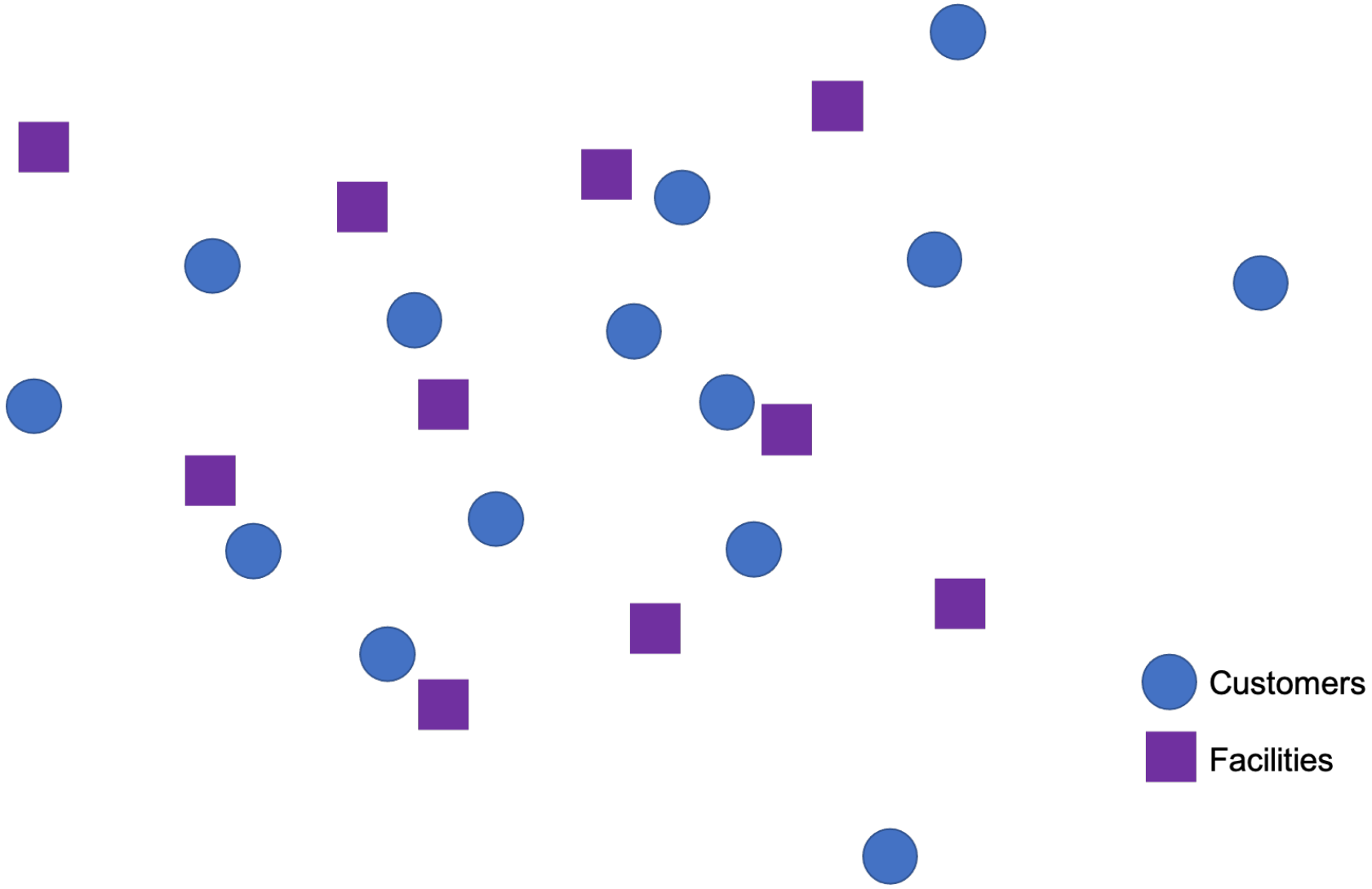
$$y_j \in \{0,1\} \qquad j \in J.$$

Example: Selection Problem

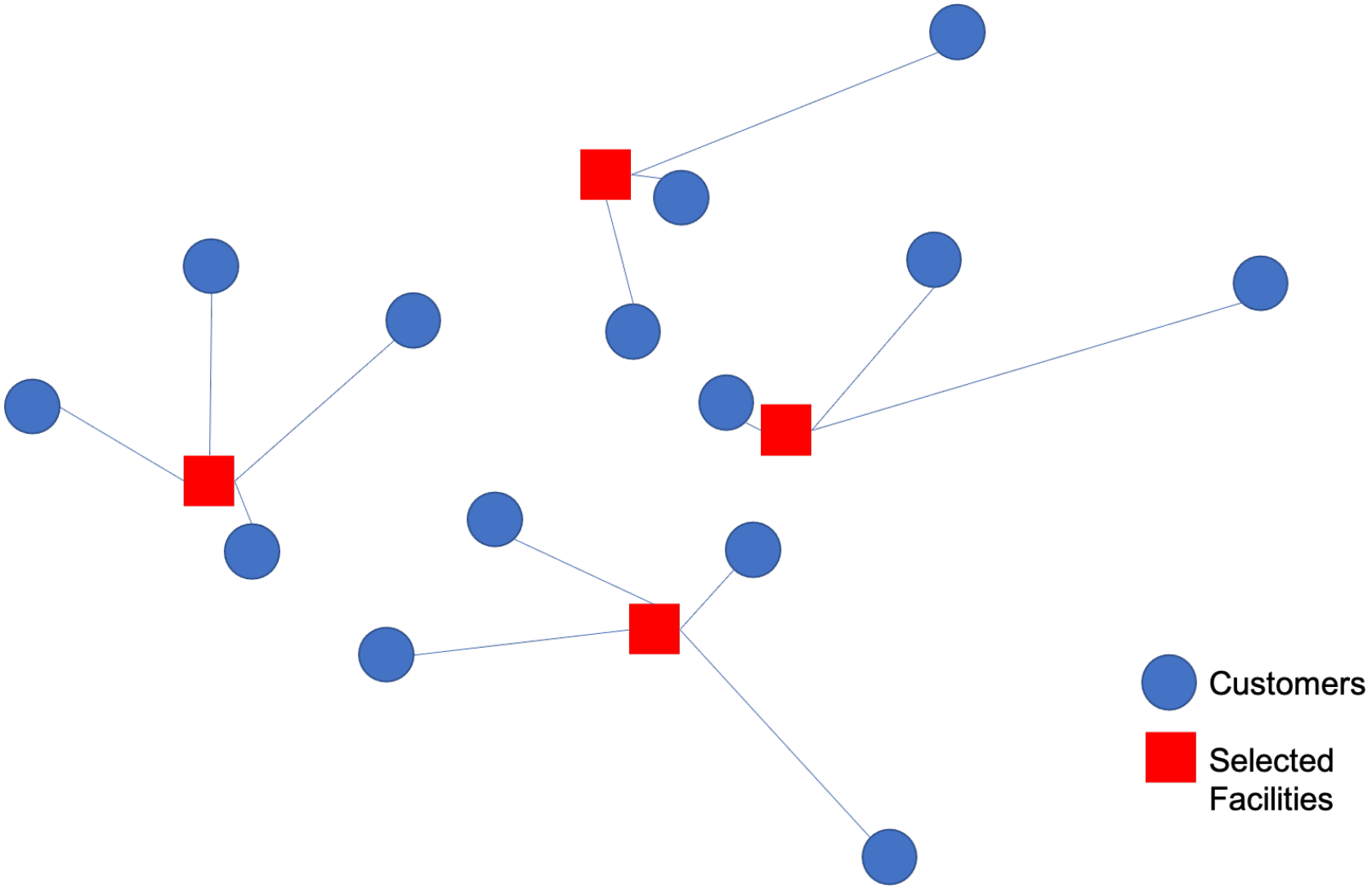
Another example of prerequisite conditions: open-pit mining



Example: Facility Location Problem



Example: Facility Location Problem



Example: Facility Location Problem

- Select facilities $i \in I$ to open within the budget.
- Transport x_{ij} from facility i to customer $j \in J_i$.
- Satisfies the demand at customer $i \in I$ and supply capacity constraints at facilities $j \in J$.
- Minimize the total transportation cost

Example: Facility Location Problem

x_{ij}: amount shipped from facility i to customer j.
y_i: whether we open facility i

$$
\begin{aligned}
\min_{x,y} \quad & \sum_{i \in I} \sum_{j \in J_i} c_{ij} x_{ij} \\
\text{s.t.} \quad & \sum_{i \in I} f_i y_i \leq b \\
& \sum_{i \in I_j} x_{ij} \geq d_j \qquad j \in J \\
& \sum_{j \in J_i} x_{ij} \leq s_i y_i \qquad i \in I \\
& y_i \in \{0,1\} \qquad i \in I \\
& x_{ij} \geq 0 \qquad i \in I, j \in J_i.
\end{aligned}
$$

Example: Facility Location Problem

$$\min_{x,y} \quad \sum_{i \in I} \sum_{j \in J_i} c_{ij} x_{ij}$$

$$\text{s.t.} \quad \sum_{i \in I} f_i y_i \leq b$$

$$\sum_{i \in I_j} x_{ij} \geq d_j \qquad j \in J$$

$$\sum_{j \in J_i} x_{ij} \leq s_i y_i \qquad i \in I$$

$$y_i \in \{0,1\} \qquad i \in I$$

$$x_{ij} \geq 0 \qquad i \in I, j \in J_i.$$

- Both installation costs and shipping costs in the objective?
- Customer can only source from one facility?
- No supply or demand, but just assignment?
- Minimize the worst customer's cost?

Example: Facility Location Problem

$$
\begin{aligned}
\min_{x,y} \quad & \sum_{i \in I} \sum_{j \in J_i} c_{ij} x_{ij} \\
\text{s.t.} \quad & \sum_{i \in I} f_i y_i \leq b \\
& \sum_{i \in I_j} x_{ij} \geq d_j \qquad j \in J \\
& \sum_{j \in J_i} x_{ij} \leq s_i y_i \qquad i \in I \\
& y_i \in \{0,1\} \qquad i \in I \\
& x_{ij} \geq 0 \qquad i \in I, j \in J_i.
\end{aligned}
$$

Both installation costs and shipping costs in the objective?

$$
\begin{aligned}
\min_{x,y} \quad & \sum_{i \in I} \sum_{j \in J_i} c_{ij} x_{ij} + \sum_{i \in I} f_i y_i \\
\text{s.t.} \quad & \sum_{i \in I_j} x_{ij} \geq d_j \qquad j \in J \\
& \sum_{j \in J_i} x_{ij} \leq s_i y_i \qquad i \in I \\
& y_i \in \{0,1\} \qquad i \in I \\
& x_{ij} \geq 0 \qquad i \in I, j \in J_i.
\end{aligned}
$$

Example: Facility Location Problem

$$
\begin{aligned}
\min_{x,y} \quad & \sum_{i \in I} \sum_{j \in J_i} c_{ij} x_{ij} \\
\text{s.t.} \quad & \sum_{i \in I} f_i y_i \leq b \\
& \sum_{i \in I_j} x_{ij} \geq d_j \qquad j \in J \\
& \sum_{j \in J_i} x_{ij} \leq s_i y_i \qquad i \in I \\
& y_i \in \{0,1\} \qquad i \in I \\
& x_{ij} \geq 0 \qquad i \in I, j \in J_i.
\end{aligned}
$$

Customer can only source from one facility?

$$
\begin{aligned}
\min_{x,y} \quad & \sum_{i \in I} \sum_{j \in J_i} c_{ij} d_j x_{ij} \\
\text{s.t.} \quad & \sum_{i \in I} f_i y_i \leq b \\
& \sum_{j \in J_i} d_j x_{ij} \leq s_i y_i \qquad i \in I \\
& \sum_{i \in I} x_{ij} = 1 \qquad j \in J \\
& y_i \in \{0,1\} \qquad i \in I \\
& x_{ij} \in \{0,1\} \qquad i \in I, j \in J_i.
\end{aligned}
$$

Example: Facility Location Problem

$$\min_{x,y} \quad \sum_{i\in I}\sum_{j\in J_i} c_{ij}x_{ij}$$

$$\text{s.t.} \quad \sum_{i\in I} f_i y_i \le b$$

$$\sum_{i\in I_j} x_{ij} \ge d_j \qquad j \in J$$

$$\sum_{j\in J_i} x_{ij} \le s_i y_i \qquad i \in I$$

$$y_i \in \{0,1\} \qquad i \in I$$

$$x_{ij} \ge 0 \qquad i \in I, j \in J_i.$$

No supply or demand, but just assignment?

$$\min_{x,y} \quad \sum_{i\in I}\sum_{j\in J_i} c_{ij}x_{ij}$$

$$\text{s.t.} \quad \sum_{i\in I} f_i y_i \le b$$

$$x_{ij} \le y_i \qquad i \in I$$

$$\sum_{i\in I} x_{ij} = 1 \qquad j \in J$$

$$y_i \in \{0,1\} \qquad i \in I$$

$$x_{ij} \in \{0,1\} \qquad i \in I, j \in J_i.$$

Example: Facility Location Problem

$$\begin{aligned}
\min_{x,y} \quad & \sum_{i\in I}\sum_{j\in J_i} c_{ij}x_{ij} \\
\text{s.t.} \quad & \sum_{i\in I} f_i y_i \le b \\
& \sum_{i\in I_j} x_{ij} \ge d_j \qquad j \in J \\
& \sum_{j\in J_i} x_{ij} \le s_i y_i \qquad i \in I \\
& y_i \in \{0,1\} \qquad i \in I \\
& x_{ij} \ge 0 \qquad i \in I, j \in J_i.
\end{aligned}$$

Minimize the worst customer's cost?

$$\begin{aligned}
\min_{x,y,V} \quad & V \\
\text{s.t.} \quad & V \ge \sum_{i\in I} c_{ij}x_{ij} \qquad j \in J \\
& \sum_{i\in I} f_i y_i \le b \\
& \sum_{i\in I_j} x_{ij} \ge d_j \qquad j \in J \\
& \sum_{j\in J_i} x_{ij} \le s_i y_i \qquad i \in I \\
& y_i \in \{0,1\} \qquad i \in I \\
& x_{ij} \ge 0 \qquad i \in I, j \in J_i.
\end{aligned}$$

How Do We Solve an IP?

For a minimization problem like staff scheduling, where we can solve the LP relaxation to obtain a lower bound and use some heuristics (rounding) to generate an upper bound/a feasible solution.

- When is LP relaxation tight? $\rightarrow$ total unimodularity
- Can we find a systematic way to solve IP? $\rightarrow$ branch-and-bound

Remember for the staff scheduling problem, we have:

- Lower bound: $z^*_{LP} = 12795.2$
- Upper bound: $\hat{z} = 12986.6$
- The optimal value: $z^*_{LP} \leq z^*_{IP} \leq \hat{z}$
- Optimality gap:

$$\frac{\hat{z} - z^*_{IP}}{z^*_{IP}} \leq \frac{\hat{z} - z^*_{LP}}{z^*_{LP}} = 0.015$$

Bounding

Theorem (LP Relaxation as a Bound for IP)

1. *For a maximization problem, the optimal value of the LP relaxation provides an upper bound for the IP optimal value. An integer feasible solution provides a lower bound for the IP optimal value.*
2. *For a minimization problem, the optimal value of the LP relaxation provides a lower bound for the IP optimal value. An integer feasible solution provides an upper bound for the IP optimal value.*

There are some cases where solving the LP relaxation can give very good solutions:

Theorem

If the optimal solution to the LP relaxation is integral, then the solution must be optimal to the IP problem.

When Does LP Have Integral Solutions

Remember for LP, there must exist an optimal solution that is a basic feasible solution.

- If we can guarantee that every BFS is integral, then the LP must have an integer optimal solution.
- This criterion has nothing to do with the objective function.
- Some sufficient conditions for this — totally unimodularity (TU).
- We have mentioned MCNF has TU matrices in the constraints.

Totally Unimodularity

We introduce a condition under which all BFS must be integral — totally unimodularity.

Definition

A matrix A is said to be *totally unimodular* if the determinant of each submatrix of A is either 0, 1, or -1.

Theorem

If the constraint matrix A is totally unimodular and b is integral, then all the BFS are integers and the LP must have an optimal solution that is an integral solution.

Totally unimodularity (TU)

- In order for A to be TU, it can only have 0, 1 or -1 entries
- But it needs much more

One Sufficient Condition for TU

Theorem

Let A be an $m \times n$ matrix. Then the following conditions together are sufficient for A to be totally unimodular:

1. *Every column of A contains at most two non-zero entries;*
2. *Every entry in A is 0, $+1$, or -1;*
3. *The rows of A can be partitioned into two disjoint sets B and C such that two nonzero entries in a column are in the same set of rows if they have different signs and in different sets of rows if they have the same sign.*

Corollary

A $0, +1, -1$ matrix A is totally unimodular if it contains no more than one +1 and no more than one -1 in each column.

Example: Matching Problem

For the matching problem:

$$\begin{aligned} \max \quad & \sum_{i=1}^{n} \sum_{j=1}^{n} v_{ij} x_{ij} & \\ \text{s.t.} \quad & \sum_{i=1}^{n} x_{ij} = 1 & \forall j \\ & \sum_{j=1}^{n} x_{ij} = 1 & \forall i \\ & x_{ij} \in \{0,1\} & \end{aligned}$$

The constraint matrix is TU. An example when $n = 2$ (in the order of $x_{11}, x_{12}, x_{21}, x_{22}$):

$$A = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}$$

One can choose $B = \{1, 2\}$, $C = \{3, 4\}$. The condition holds and A is TU.

Example: MCNF

$$
\begin{aligned}
\min \quad & \sum_{(i,j)\in A} c_{ij} x_{ij} & & \\
\text{s.t.} \quad & \sum_{(i,j)\in FS(i)} x_{ij} - \sum_{(j,i)\in RS(i)} x_{ji} = b_i & & i \in N \\
& \ell_{ij} \le x_{ij} \le u_{ij} & & (i,j) \in A.
\end{aligned}
$$

Each column corresponds to an arc (i, j) and we have only one $+1$ element and only one -1 element in such column. Therefore, the matrix A is TU.

What If We Do Not Have TU?

TU is a nice property, but not common in practical problems. We need a systematic way to solve IP: Branch-and-bound.

What do we know so far?

- We can solve LP using simplex method.
- We have shown that solving the LP relaxation problem provides a lower bound for the minimization IP problem (an upper bound for the maximization problem).
- When we obtain a feasible solution using some heuristics. That feasible solution offers an upper bound.
- Can we improve the lower/upper bounds?

Branch-and-bound

Introduction

Suppose $\mathcal{S}$ is the feasible set for an MILP and we wish to compute

$$\max_{x \in \mathcal{S}} \quad c^\top x.$$

- Consider a partition of $\mathcal{S}$ into subsets $\mathcal{S}_1, \ldots, \mathcal{S}_k$. Then

$$\max_{x \in \mathcal{S}} \quad c^\top x = \max_{1 \le i \le k} \max_{x \in \mathcal{S}_i} \quad c^\top x.$$

- Let $\{X_i\}_{i=1}^k$ is an admissible disjunction. Then let $\mathcal{S}_i = \mathcal{S} \cap X_i$.
- We only consider linear disjunctions so that the subproblems remain MILPs after branching.
- Generally speaking, we want $\hat{x} \notin \cup_{1 \le i \le k} X_i$, where $\hat{x}$ is the infeasible solution produced by solving the bounding problem (LP relaxation subproblem).

What is Branch-and-bound?

- Branch and bound is not so much a complete algorithm as a framework;
- A wide variety of different algorithms can be obtained by implementing the constituent procedures in different ways;
- The most fundamental constituent procedures are
  - A method for obtaining upper and lower bounds on the value of the optimal solution (usually by solving a relaxation or dual); and
  - A method for producing a valid disjunction violated by a given (infeasible) solution.
- Dividing the original problem into subproblems is called branching;

Branch-and-bound Algorithm

- When we generate an LP relaxation solution, we may fall on a fractional solution.
- For example:

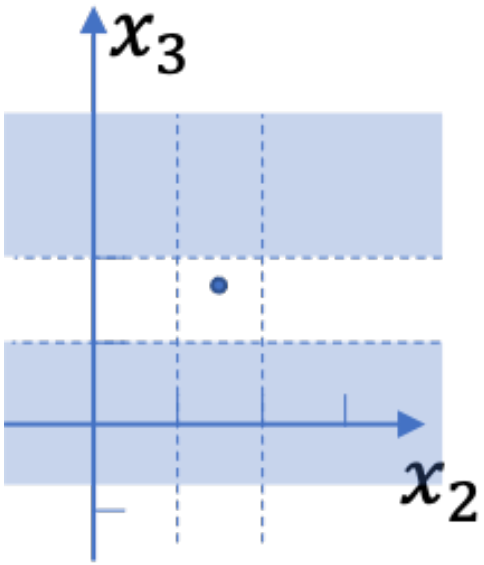
$$\begin{aligned} \max \quad & 4x_1 - 2x_2 + 7x_3 - x_4 \\ \text{s.t.} \quad & x_1 + 5x_3 \leq 10 \\ & x_1 + x_2 - x_3 \leq 1 \\ & 6x_1 - 5x_2 \leq 0 \\ & -x_1 + 2x_3 - 2x_4 \leq 3 \\ & x_1, x_2, x_3, x_4 \geq 0 \\ & x_1, x_2, x_3 \in \mathbb{Z}. \end{aligned}$$

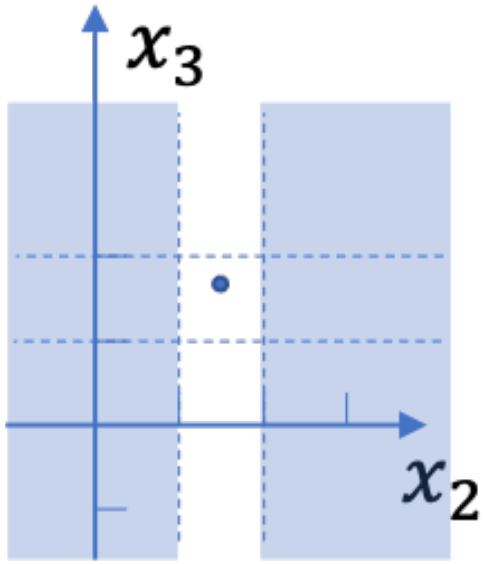
If we solve the LP relaxation, we would obtain a solution

$$(x_1^{LP}, x_2^{LP}, x_3^{LP}, x_4^{LP}) = (\frac{5}{4}, \frac{3}{2}, \frac{7}{4}, 0)$$

Branch-and-bound Algorithm

For a solution like $(x_1^{LP}, x_2^{LP}, x_3^{LP}, x_4^{LP}) = (\frac{5}{4}, \frac{3}{2}, \frac{7}{4}, 0)$, we could split the feasible region into something like $x_2 \geq 2$ and $x_2 \leq 1$, or we could split the feasible region into something like $x_3 \geq 2$ and $x_3 \leq 1$, because nothing in between is feasible. Of course, it can also be done on x_1.





Branch-and-bound Algorithm

Let's try this divide-and-conquer approach:

0. (Initialize)
   $\epsilon \geq 0$: desired solution tolerance. Let $z^* = -\infty$.
   Solve the LP relaxation of the model to obtain x^{LP} and z^{LP}.
   - LP relaxation infeasible: the MIP is infeasible.
   - $x_j^{LP}, j \in I$ are integer valued: x^{LP} an optimal solution to the MIP.
   - Otherwise go to Step 1, with the current LP labeled as unfathomed (unpruned).
1. (Branching)
   Select an unfathomed subproblem, and select a fractional$x_j^{LP}, j \in I$.
   Create two new subproblems by adding the respective constraints:

$$X_1 = \{x \in \mathbb{R}^n : x_j \leq \lfloor x_j^{LP} \rfloor\} \quad \text{and} \quad X_2 = \{x \in \mathbb{R}^n : x_j \geq \lfloor x_j^{LP} \rfloor + 1\}.$$

Branch-and-bound Algorithm

2. (Bounding)
   Solve the LP relaxations of the two new subproblems, and for each obtain x^{LP} and z^{LP}, or discover it is infeasible.
3. (Fathoming)
   For each new subproblem declare the subproblem to be fathomed if any of the following tests are passed.
   - (better solution exists) If $z^{LP} \leq z^*$, where z^* corresponds to the incumbent (current best) solution, then fathom the subproblem.
   - (infeasible) If the LP is infeasible, then fathom the subproblem.
   - (integer solution) If $x_j^{LP}, j \in I$ are all integer valued, then fathom. If $z^{LP} > z^*$ then update the incumbent $x^* \leftarrow x^{LP}$ and $z^* \leftarrow z^{LP}$. Re-apply test 1 to all unfathomed subproblems.

Branch-and-bound Algorithm

4. (Terminating)
   - If all nodes are fathomed and $z^* = -\infty$ then stop: infeasible.
   - Otherwise let $\bar{z}$ be the maximum of z^{LP} across all unfathomed subproblems.
   - If $\bar{z} - z^* \leq \epsilon \cdot z^*$, then stop: x^* is a solution within $(100 \cdot \epsilon)\%$ of optimal. Otherwise, go to Step 1.

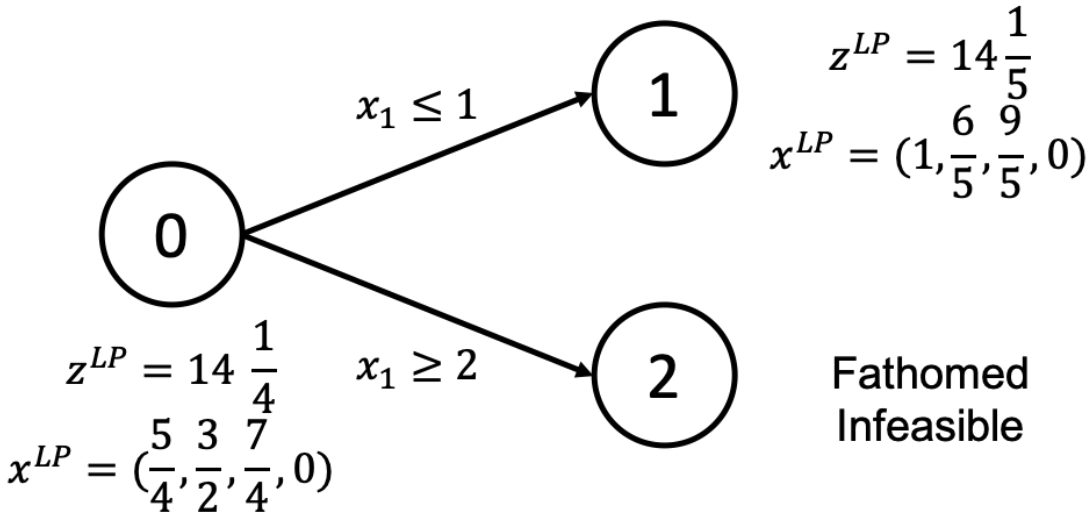
Branch-and-bound Algorithm: Example

$$
\begin{aligned}
\max \quad & 4x_1 - 2x_2 + 7x_3 - x_4 \\
\text{s.t.} \quad & x_1 + 5x_3 \leq 10 \\
& x_1 + x_2 - x_3 \leq 1 \\
& 6x_1 - 5x_2 \leq 0 \\
& -x_1 + 2x_3 - 2x_4 \leq 3 \\
& x_1, x_2, x_3, x_4 \geq 0 \\
& x_1, x_2, x_3 \in \mathbb{Z}.
\end{aligned}
$$

Let $\epsilon = 0$ (although you would not do this in a large-scale MIP). Solve the LP relaxation and obtain

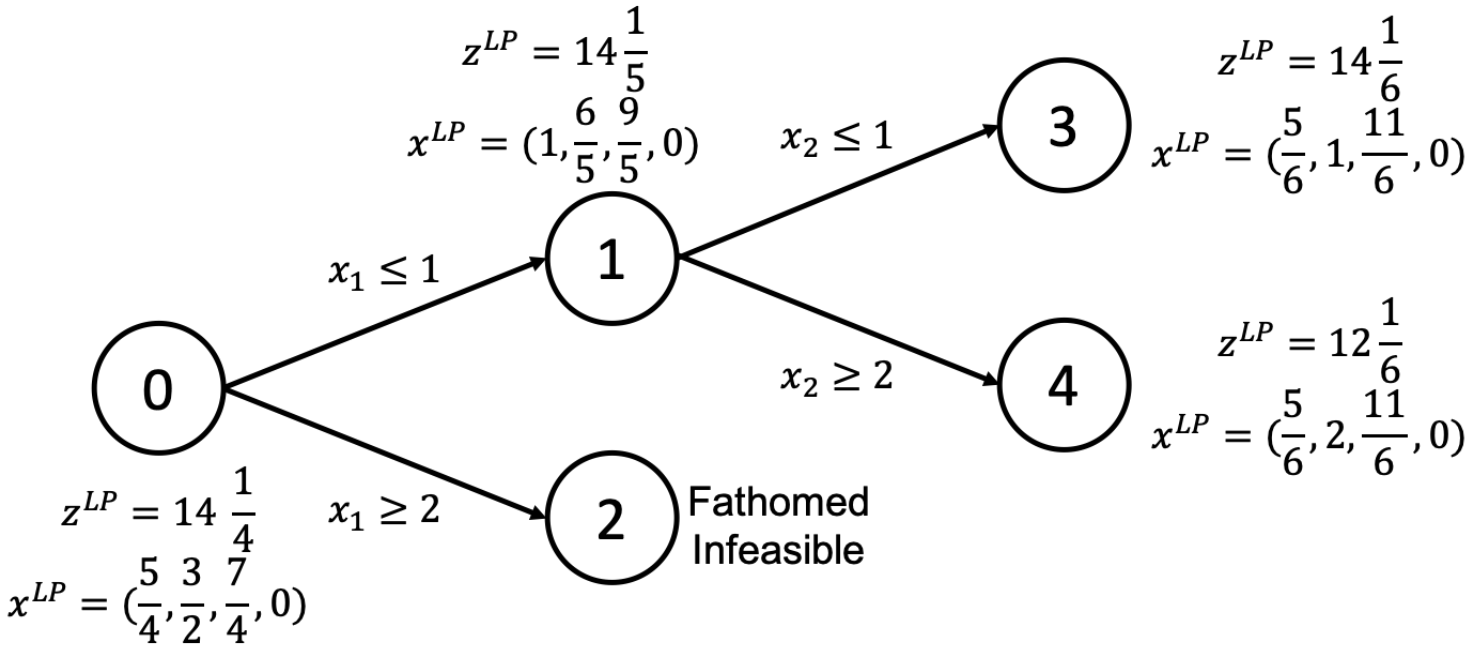
$$(x_1^{LP}, x_2^{LP}, x_3^{LP}, x_4^{LP}) = (\frac{5}{4}, \frac{3}{2}, \frac{7}{4}, 0) \quad \text{and} \quad z^{LP} = 14\frac{1}{4}$$

Branch-and-bound Algorithm: First Branch



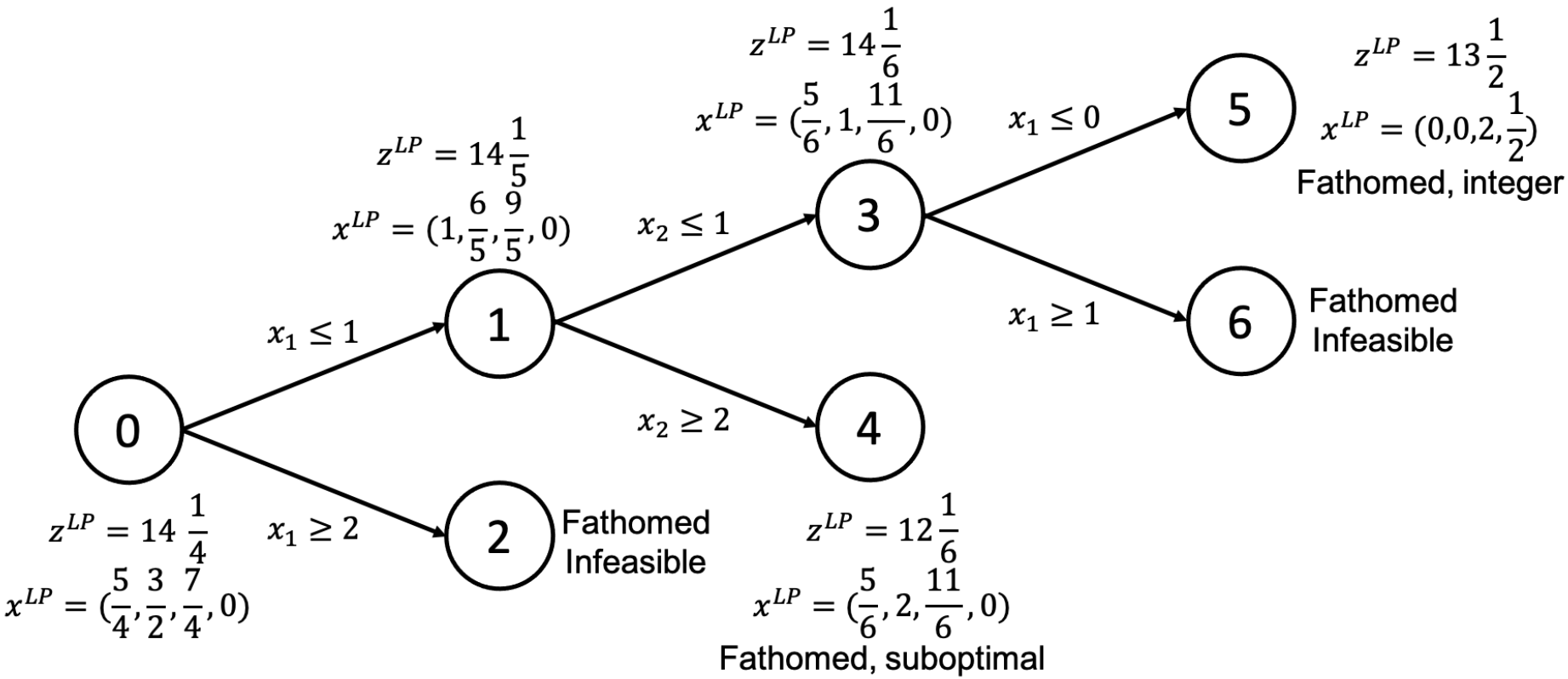
- Branch on x_1: $x_1 \leq 1$, $x_1 \geq 2$
- Check the subproblems' LP relaxation for fathoming
- Fathom infeasible node 2

Branch-and-bound Algorithm: Second Branch



- Branch on x_2: $x_2 \leq 1$, $x_2 \geq 2$
- Check the subproblems' LP relaxation for fathoming: cannot fathom anything
- Choose the more promising node: $14\frac{1}{6} > 12\frac{1}{6}$, so node 3

Branch-and-bound Algorithm: Third Branch



- Branch on x_1: $x_1 \leq 0$, $x_1 \geq 1$
- Check the subproblems' LP relaxation for fathoming: fathom every node left

Order of Computing the Branches

When we do B&B, we may have two choices to do at each step

- In the example, both node 3 and node 4 yield non-integral solutions. Which one should we choose to branch?
- We can proceed with branching node 3 and thus yield two branches (node 5 and 6).
- Then we need to decide to go for one of these two branches next or try node 4 instead.

An option is to choose the node with the best "potential". For example, in the previous example, when it comes to the decision whether to branch on node 3 or 4. We choose node 3 as the LP relaxation upper bound is higher for node 3 compared to that for node 4. This is called a best-first search rule.

Order of Computing the Branches

Generally speaking, we need to decide if we want to *go deep* into one branch first or *go wide* to solve all problems on a given level first. In the B&B algorithm, sometimes it is beneficial to *go deep*.

- Most integral solutions lie deep in the tree; it's good to have integral feasible solutions early, so we can use it in the *bounding* procedure
- It is also memory-efficient, since each LP is obtained from its parent by merely adding one constraint
- It's also easier to code (recursion)

The policy to choose which fractional variable to branch on involves some state-of-the-art research. Some papers have utilized machine learning!

Branch-and-bound Algorithm: Implementation

Most modern commercial MIP solvers, such as Gurobi, CPLEX, and COPT, use B&B algorithms with enhancements:

- preprocessing: tightening bounds/constraints, eliminating variables/constraints
- heuristic search for good feasible solutions
- cutting-planes

It is also important to decide:

- a proper ϵ
- which variable do we branch on
- which node do we explore next

Formulation is important!!! Tight formulation $\rightarrow$ fewer explored nodes/faster feasible solution

Geometric Representation of IP

Solving (M)IP is in general hard. How hard? NP-hard.

- We want to know what is actually happening when solving an IP
- We have seen how to graphically represent an LP: polyhedron/polytope
- We have seen some graphical representation of the branching process in B&B algorithm

Geometric Representation of IP

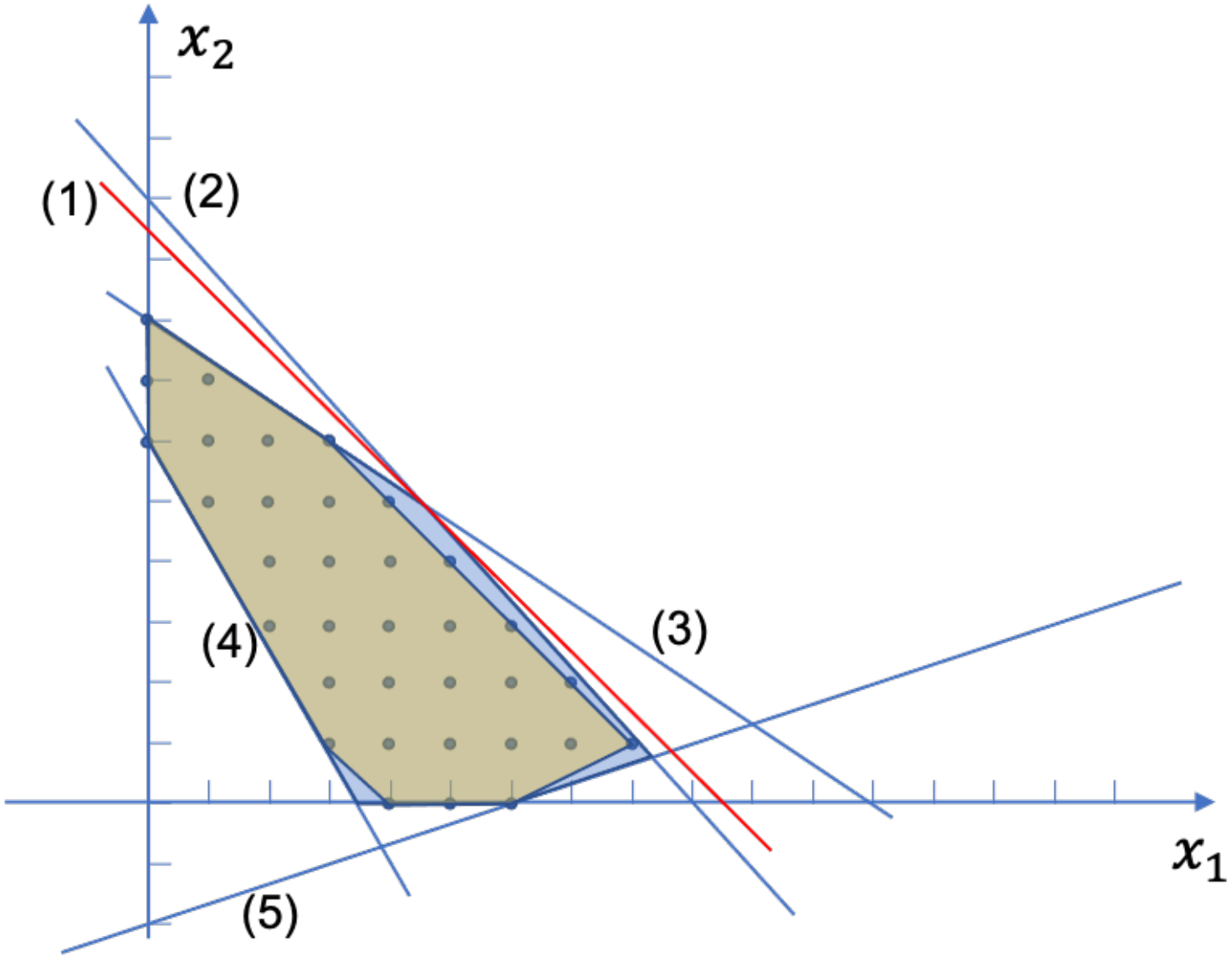
You may now ask:

- How different can solving a MIP be from solving an LP?
- Can we use the method of solving an LP in solving a MIP?

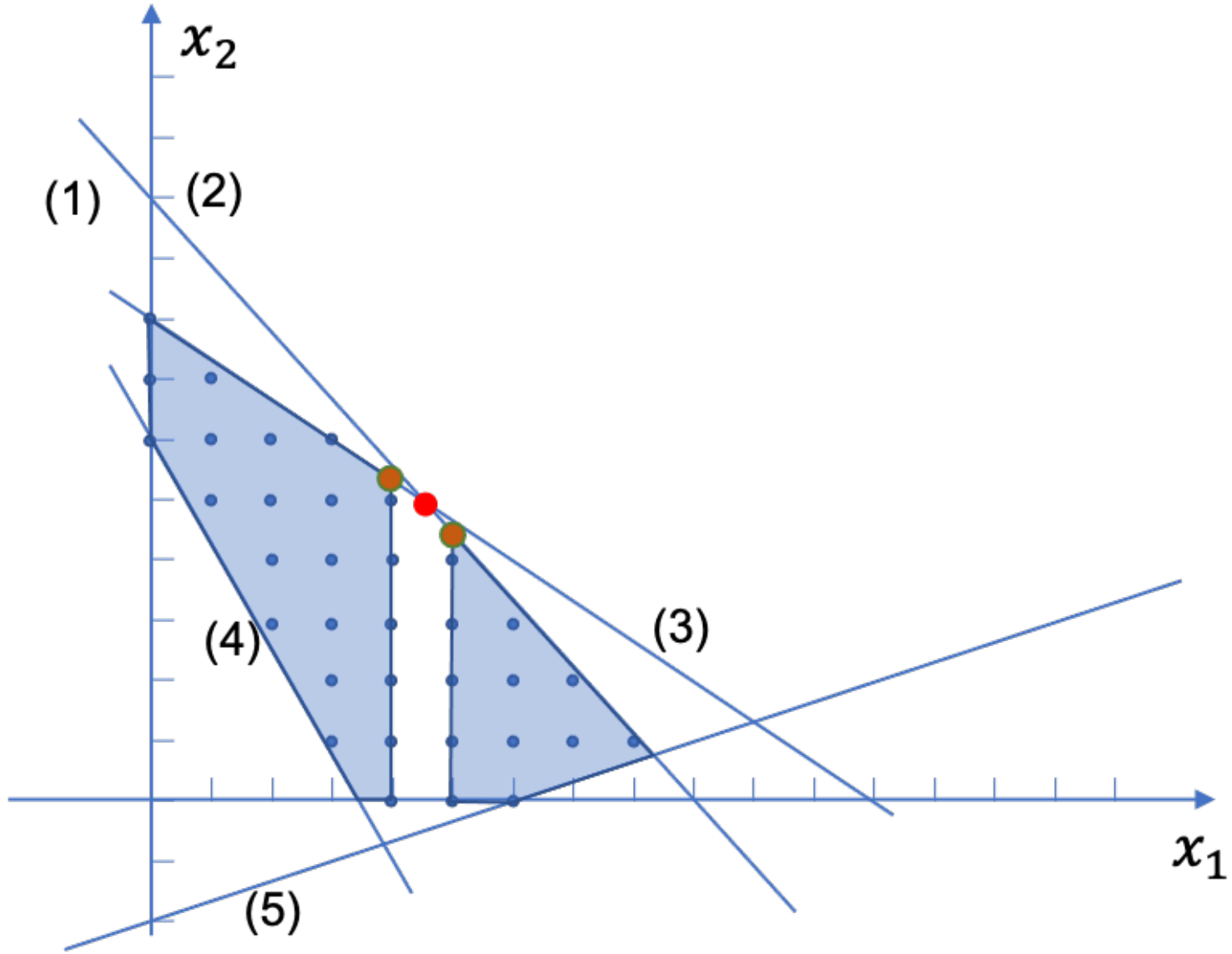
Let's look at an IP as follows:

$$
\begin{aligned}
\max \quad & x_1 + x_2 && (1)\\
\text{s.t.} \quad & 11x_1 + 10x_2 \leq 100 && (2)\\
& 2x_1 + 3x_2 \leq 24 && (3)\\
& 7x_1 + 4x_2 \geq 24 && (4)\\
& x_1 - 3x_2 \leq 6 && (5)\\
& x_1, x_2 \in \mathbb{Z}^+. && (6)
\end{aligned}
$$

Geometric Representation of IP



Geometric Representation of IP



Cutting-plane Method

Reformulation

A MIP of the form

$$\begin{aligned} \max \quad & c^\top x \\ \text{s.t.} \quad & x \in \mathcal{S} \end{aligned}$$

can be equivalent to the following problem

$$\begin{aligned} \max \quad & c^\top x \\ \text{s.t.} \quad & x \in \text{conv}(\mathcal{S}), \end{aligned}$$

which is a LP since $\text{conv}(\mathcal{S})$ is a polyhedron. Unfortunately, $\text{conv}(\mathcal{S})$ is necessarily of exponential size in general, so there can be no way of generating them efficiently.

Cutting-plane Algorithm

We will focus on one type of cutting plane: the Gomory cutting plane.

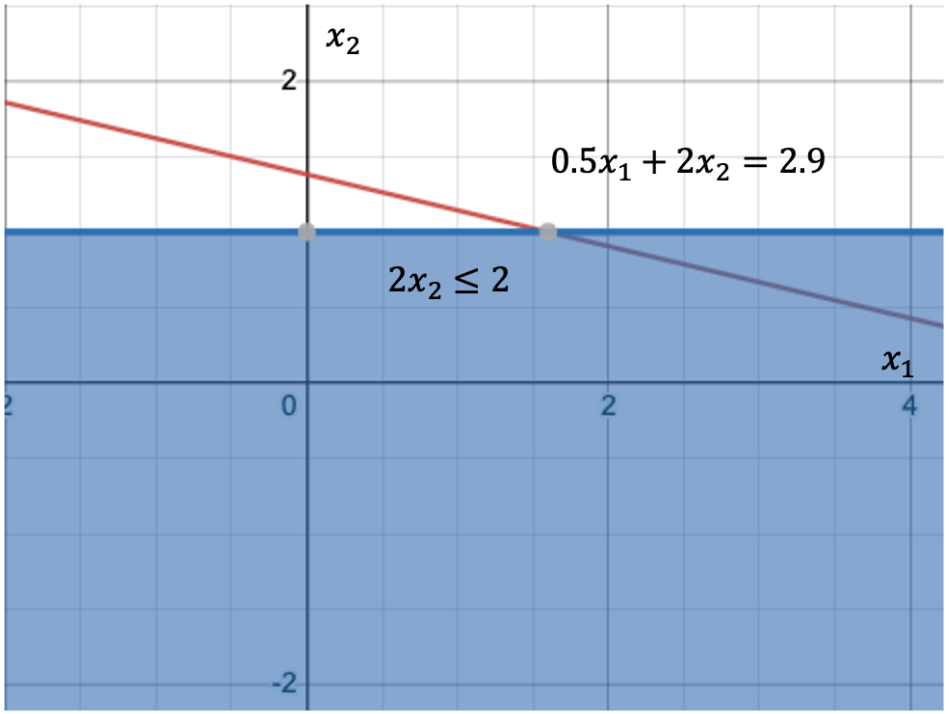
- All variables are integers
- For a constraint like:

$$\sum_{j=1}^{n} a_j x_j = b,$$

  a_j, b are fractional and $x_j \in \mathbb{Z}$
- Valid inequality: cuts off fractional points but all integer solutions remain feasible

$$\sum_{j=1}^{n} a_j x_j = b \rightarrow \sum_{j=1}^{n} \lfloor a_j \rfloor x_j \leq b \rightarrow \sum_{j=1}^{n} \lfloor a_j \rfloor x_j \leq \lfloor b \rfloor.$$

Gomory Cut



The Gomory cut can be appended when solving the LP relaxation.

- It can be used independently: a pure cutting-plane method.
- It can be used within a branch-and-bound process: branch-and-cut.

Derive Gomory cut based on this valid inequality in a simplex iteration?

Gomory Cut

- For an LP problem, we solve it with the simplex method
- At a simplex iteration, we have

$$A_B x_b + A_N x_N = b \rightarrow x_B + A_B^{-1} A_N x_N = A_B^{-1} b = \bar{b}$$

- One row within this set of equations looks like:

$$x_k + \sum_{j \in N} \bar{a}_{ij} x_j = \bar{b}_i$$

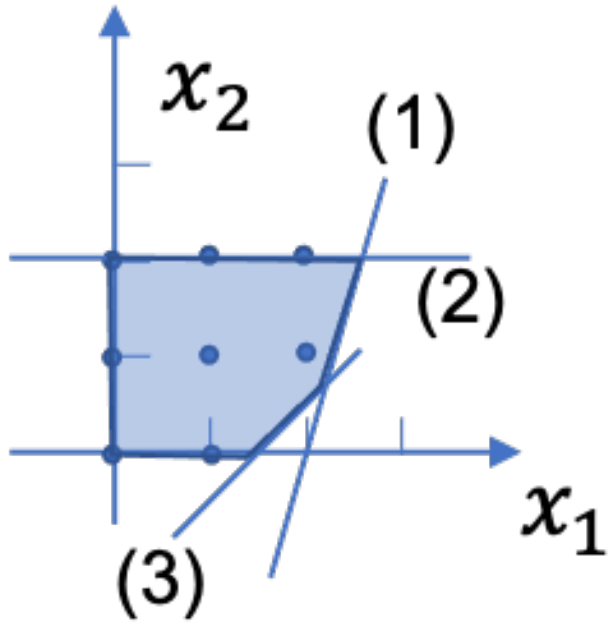
- When $\bar{a}_{ij}$ and $\bar{b}_i$ are fractional, generate a Gomory cut:

$$x_k + \sum_{j \in N} \lfloor \bar{a}_{ij} \rfloor x_j \leq \lfloor \bar{b}_i \rfloor$$

- Where do I get $\bar{a}_{ij}$ and $\bar{b}_i$? $\rightarrow$ Simplex tableau.

Gomory Cut Example

$$
\begin{aligned}
\max \quad & 4x_1 - x_2 \\
\text{s.t.} \quad & 7x_1 - 2x_2 \leq 14 && (1) \\
& x_2 \leq 3 && (2) \\
& 2x_1 - 2x_2 \leq 3 && (3) \\
& x_1, x_2 \geq 0 && (4) \\
& x_1, x_2 \in \mathbb{Z}. && (5)
\end{aligned}
$$



Standard form:

$$
\begin{aligned}
\min \quad & -4x_1 + x_2 \\
\text{s.t.} \quad & 7x_1 - 2x_2 + x_3 = 14 \\
& x_2 + x_4 = 3 \\
& 2x_1 - 2x_2 + x_5 = 3 \\
& x_1, x_2, x_3, x_4, x_5 \geq 0.
\end{aligned}
$$

Gomory Cut Example

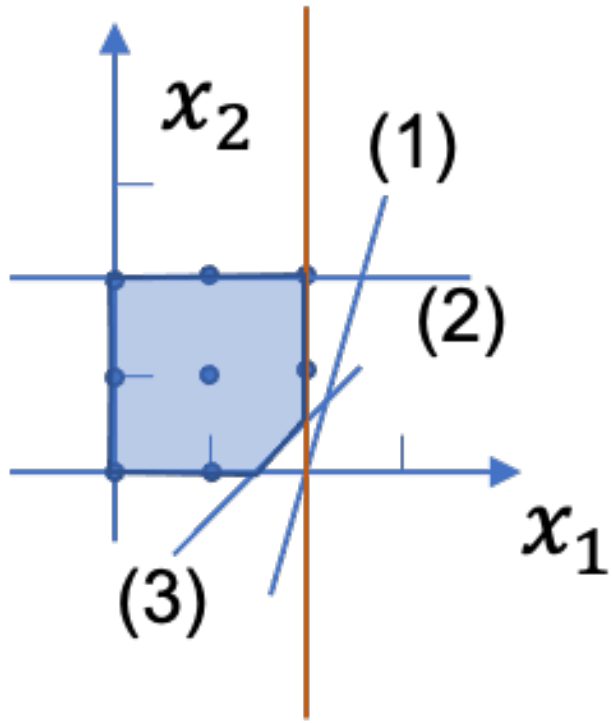
Optimal BV: x_1, x_2, x_5, NBV: x_3, x_4

$$x_B + A_B^{-1} A_N x_N = A_B^{-1} b$$

$$A_B^{-1} A_N = \begin{bmatrix} \frac{1}{7} & \frac{2}{7} \\ 0 & 1 \\ -\frac{2}{7} & \frac{10}{7} \end{bmatrix} \rightarrow$$

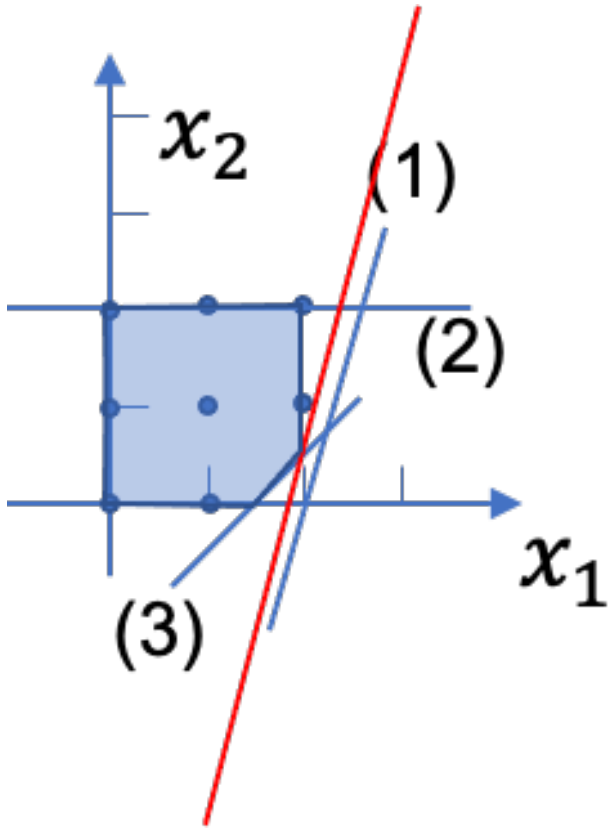
$$\begin{cases} x_1 + \frac{1}{7} x_3 + \frac{2}{7} x_4 = \frac{20}{7} \\ x_2 + x_4 = 3 \\ -\frac{2}{7} x_3 + \frac{10}{7} x_4 + x_5 = \frac{23}{7} \end{cases} \rightarrow$$

$$\begin{cases} x_1 \leq 2 \\ -x_3 + x_4 + x_5 \leq 3. \end{cases}$$



Gomory Cut Example

- Resolve LP relaxation:
- $x = \begin{bmatrix} 2 \\ 0.5 \\ 1 \\ 2.5 \\ 0 \end{bmatrix}$
- After adding the first constraint, adding the second constraint does not cut down any fractional solution in (x_1, x_2) space



- $x_3 = 14 - 7x_1 + 2x_2, x_4 = 3 - x_2, x_5 = 3 - 2x_1 + 2x_2$
- $-x_3 + x_4 + x_5 = -14 + 7x_1 - 2x_2 + 3 - x_2 + 3 - 2x_1 + 2x_2$
  $= -8 + 5x_1 - x_2 \leq 3 \rightarrow 5x_1 - x_2 \leq 11.$