Integer and Mixed-integer Linear Optimization Models

Shixuan Zhang

ISEN 320-501, Fall 2023

1 Mixed-integer Linear Optimization and Computer Tools

Given problem data $c = (c_1, \ldots, c_n) \in \mathbb{R}^n$, $b = (b_1, \ldots, b_m) \in \mathbb{R}^m$, and

$$A = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix} \in \mathbb{R}^{m \times n},$$

a (mixed-)integer linear optimization (MILO) model can be defined as

$$\begin{aligned} \min \,/\, \max \quad & c^\top x \\ \text{s.t.} \quad & Ax \leq b, \\ & x \in \mathbb{R}^{n_1} \times \mathbb{Z}^{n_2}, \end{aligned} \tag{1}$$

where $n_1 + n_2 = n$. If $n = n_1$ and $n_2 = 0$, then the problem (1) reduces to the usual linear optimization (LO) model. If $n_2 = n$, then we say it is a (pure) integer linear optimization, and otherwise a mixed-integer linear optimization when $n_2 < n$. While integer and mixed-integer linear optimization models may have different algorithmic aspects, we do not distinguish them in this course because we focus mainly on the modeling part. We see a simple example of a MILO model as follows.

Example 1. *A company produces two types of baby carriers, non-reversible and reversible.*

- *Each non-reversible carrier sells for \$22, requires 2 linear yards of a solid color fabric, and costs \$7 to manufacture.*
- *Each reversible carrier sells for \$35, requires 2 linear yards of a printed fabric as well as 2 linear yards of a solid color fabric, and costs \$10 to manufacture.*

The company has 900 linear yards of solid color fabrics and 600 linear yards of printed fabrics available for its new carrier collection. It can spend up to \$4,000 on manufacturing the carriers. The demand is such that all reversible carriers made are projected to sell, whereas at most 350

non-reversible carriers can be sold. The goal of the company is to maximize its profit (e.g., the difference of revenues and expenses) resulting from manufacturing and selling the new carrier collection.

We define $x_1, x_2 \geq 0$ to be the numbers of non-reversible and reversible carriers to manufacture. If we do not require x_1, x_2 to take integer values, then we have a LO model as

$$\begin{array}{lrcrcll}
\max & 15x_1 & + & 25x_2 & & & \textit{(profit)} \\
\text{s.t.} & x_1 & + & x_2 & \leq & 450 & \textit{(solid color fabric constraint)} \\
 & & & x_2 & \leq & 300 & \textit{(printed fabric constraint)} \\
 & 7x_1 & + & 10x_2 & \leq & 4,000 & \textit{(budget constraint)} \\
 & x_1 & & & \leq & 350 & \textit{(demand constraint)} \\
 & & & x_1, x_2 & \geq & 0. & \textit{(nonnegativity constraints).}
\end{array}$$

We can use `OR-Tools` *and* `GLOP` *to find the solution, as coded in* `lin_model_production.py`*, and get the following result.*

```
The maximum profit for the baby carrier production is 9642.9.
x1 = 142.86 (non-reversible carriers)
x2 = 300.00 (reversible carriers)
```

We can see that the solution is not integral. In fact, since the objective function consists of integer coefficients and the optimal value here is non-integer, there does not exist an integer solution. This example suggests that it is not always practical to relax the integrality requirements and solve the LO model as an substitute.

We can still use the `Python` module `OR-Tools` for MILO modeling. As usual, we import it using the following command.

```
from ortools.linear_solver import pywraplp
```

Next we declare a MILO solver `SCIP` instead of the LO solver `GLOP` that we used for LO models.

```
solver = pywraplp.Solver.CreateSolver("SCIP")
```

For Example 1, we can define our integer variables using the `IntVar` functions, the argument for which is the same as the function `NumVar` for continuous variables.

```
x1 = solver.IntVar(0.0, solver.infinity(), 'x1')
x2 = solver.IntVar(0.0, solver.infinity(), 'x2')
```

To be more specific, the first argument specifies the lower bound, the second one specifies the upper bound (where `solver.infinity()` gives ∞), and the last one gives the variable a name used in model printing or exporting. After adding the variables, each linear constraint can be added similar to what we did for LO models.

```
# define the solid color fabric constraint
solver.Add(x1 + x2 <= 450)

# define the printed fabric constraint
solver.Add(x2 <= 300)

# define the budget constraint
solver.Add(7*x1 + 10*x2 <= 4000)

# define the demand constraint
solver.Add(x1 <= 350)
```

We set linear objective function.

```
# create the objective of maximizing the profit
solver.Maximize(15*x1 + 25*x2)
```

And now we may invoke the solver.

```
# call the solver
status = solver.Solve()
```

The result (from the script `int_model_production.py`) is printed below.

```
The maximum profit for the baby carrier production is 9635.0.
x1 = 144.00 (non-reversible carriers)
x2 = 299.00 (reversible carriers)
```

By comparing the MILO and LO solutions in Example 1, we see that the MILO solution is not just a rounded solution of the LO model. The following example further illustrates that a rounded solution may not be feasible to the MILO problem.

Example 2. *A hospital uses a 12-hour shift schedule for its nurses, with each nurse working either day shifts (7:00 am-7:00 pm) or night shifts (7:00 pm-7:00 am). Each nurse works 3*

Table 1: Number of Required Nurses for Day Shifts

Day of week/shift	Nurses required
Monday (Mo)	16
Tuesday (Tu)	12
Wednesday (We)	18
Thursday (Th)	13
Friday (Fr)	15
Saturday (Sa)	9
Sunday (Su)	8

consecutive day shifts or 3 consecutive night shifts and then has 4 days off. The minimum

number of nurses required for each day shift during a week is given in the following table: In addition, it is required that at least two thirds of the day-shift nurses have weekends (Saturday and Sunday) off. The hospital is aiming to design a schedule for day-shift nurses that minimizes the total number of nurses employed.

To formulate a MILO model for the hospital scheduling problem, we define the decision variables ($\mathbb{Z}_{\geq 0}$ means nonnegative integers) as follows.

$x_1 \in \mathbb{Z}_{\geq 0}$: *number of nurses on Mo-Tu-We schedule*
$x_2 \in \mathbb{Z}_{\geq 0}$: *number of nurses on Tu-We-Th schedule*
$x_3 \in \mathbb{Z}_{\geq 0}$: *number of nurses on We-Th-Fr schedule*
$x_4 \in \mathbb{Z}_{\geq 0}$: *number of nurses on Th-Fr-Sa schedule*
$x_5 \in \mathbb{Z}_{\geq 0}$: *number of nurses on Fr-Sa-Su schedule*
$x_6 \in \mathbb{Z}_{\geq 0}$: *number of nurses on Sa-Su-Mo schedule*
$x_7 \in \mathbb{Z}_{\geq 0}$: *number of nurses on Su-Mo-Tu schedule*

On Monday, there are $x_1 + x_6 + x_7$ nurses working, so by requirement we should have

$$x_1 + x_6 + x_7 \geq 16.$$

Similarly, for the other days of the week, we have constraints

$$\begin{aligned} x_1 + x_2 + x_7 &\geq 12, \\ x_1 + x_2 + x_3 &\geq 18, \\ x_2 + x_3 + x_4 &\geq 13, \\ x_3 + x_4 + x_5 &\geq 15, \\ x_4 + x_5 + x_6 &\geq 9, \\ x_5 + x_6 + x_7 &\geq 8. \end{aligned}$$

Clearly $\sum_{i=1}^{7} x_i \geq 1$. Thus the requirement that two thirds of the day-shift nurses have weekends off can be expressed as

$$\frac{x_1 + x_2 + x_3}{\sum_{i=1}^{7} x_i} \geq \frac{2}{3}.$$

This can be transformed into a linear *constraint*

$$x_1 + x_2 + x_3 - 2x_4 - 2x_5 - 2x_6 - 2x_7 \geq 0.$$

The objective is to minimize the total number of nurses $\sum_{i=1}^{7} x_i$*, so the model can be written as*

$$\begin{aligned}
\min \quad & \sum_{i=1}^{7} x_i \\
\text{s.t.} \quad & x_1 + x_6 + x_7 \geq 16, \\
& x_1 + x_2 + x_7 \geq 12, \\
& x_1 + x_2 + x_3 \geq 18, \\
& x_2 + x_3 + x_4 \geq 13, \\
& x_3 + x_4 + x_5 \geq 15, \\
& x_4 + x_5 + x_6 \geq 9, \\
& x_5 + x_6 + x_7 \geq 8, \\
& \sum_{i=1}^{3} x_i - 2 \sum_{i=4}^{7} x_i \geq 0, \\
& x_i \in \mathbb{Z}_{\geq 0}, \quad i = 1, \ldots, 7.
\end{aligned}$$

If we relax the integrality constraints, the result (from the script `lin_model_scheduling.py`*) is printed below.*

```
The minimum number of nurses is 31.3.
x1 = 10.29
x2 = 0.29
x3 = 10.29
x4 = 2.43
x5 = 2.29
x6 = 4.29
x7 = 1.43
```

If we round them to a nearest integer solution $\bar{x} = (10, 0, 10, 2, 2, 4, 1)$*, we see that*

$$\begin{aligned}
\bar{x}_1 + \bar{x}_6 + \bar{x}_7 &= 15 \not\geq 16, \\
\bar{x}_1 + \bar{x}_2 + \bar{x}_7 &= 11 \not\geq 12, \\
\bar{x}_1 + \bar{x}_2 + \bar{x}_3 &= 20 \geq 18, \\
\bar{x}_2 + \bar{x}_3 + \bar{x}_4 &= 12 \not\geq 13, \\
\bar{x}_3 + \bar{x}_4 + \bar{x}_5 &= 14 \not\geq 15, \\
\bar{x}_4 + \bar{x}_5 + \bar{x}_6 &= 8 \not\geq 9, \\
\bar{x}_5 + \bar{x}_6 + \bar{x}_7 &= 7 \not\geq 8.
\end{aligned}$$

In other words, this rounded solution $\bar{x}$ *is* infeasible*. Instead, we should directly solve the MILO model in the script* `int_model_scheduling.py`*, the result for which is printed below.*

```
The minimum number of nurses is 32.0.
x1 = 11.00
x2 = 0.00
x3 = 11.00
x4 = 2.00
x5 = 3.00
x6 = 4.00
x7 = 1.00
```

One of the most important features of MILO modeling is the use of $0/1$ variables (or sometimes called *binary* variables). An example is the following location covering problem.

Example 3. *A city is planning to set up new emergency centers at different possible locations. Due to distances and one-way streets, an emergency center at*

- *location 1 can cater to patients in locations 1, 2, 4, 7;*
- *a center at location 2 can cater to patients in locations 2, 3, 5;*
- *a center at location 3 can cater to patients in locations 1, 3, 6;*
- *a center at location 4 can cater to patients in locations 2, 3, 4, 5;*
- *a center at location 5 can cater to patients in locations 1, 5, 6;*
- *a center at location 6 can cater to patients in locations 3, 4, 6;*
- *a center at location 7 can cater to patients in locations 2, 3, 7.*

We need to cater to patients at all locations and would like to set up the minimum number of emergency centers.
To model this problem, we can define our decision variables for $i = 1, \ldots, 7$ by

$$x_i = \begin{cases} 1 & \text{if an emergency center is set up at location } i, \\ 0 & \text{otherwise.} \end{cases}$$

To cover patients at the location 1, there are only three possible locations: 1, 3, and 5. Thus we need

$$x_1 + x_3 + x_5 \geq 1.$$

Similarly, we can write the covering constraints at other locations as

$$\begin{aligned} x_1 + x_2 + x_4 + x_7 &\geq 1, \\ x_2 + x_3 + x_4 + x_6 + x_7 &\geq 1, \\ x_1 + x_4 + x_6 &\geq 1, \\ x_2 + x_4 + x_5 &\geq 1, \\ x_3 + x_5 + x_6 &\geq 1, \\ x_1 + x_7 &\geq 1. \end{aligned}$$

The goal is to minimize the sum

$$\min \quad \sum_{i=1}^{7} x_i.$$

We can code this model in the script `model_covering.py` *and get the following result.*

```
The minimum number of emergency centers is 3.0.
x1 = 1.0
x2 = 1.0
x3 = 1.0
x4 = 0.0
x5 = 0.0
x6 = 0.0
x7 = 0.0
```

The following *knapsack problem* is a very well-known problem in integer optimization.

Example 4. *One would like to carry different items to the camping ground in my knapsack. They have a choice of n items. Item i produces an utility of u_i for them. The volume of item i is v_i. The volume of the knapsack is V. The goal is to maximize the total utility of items put in the knapsack.*
To formulate a MILO model, we define our variables for each item $i = 1, \ldots, n$ as

$$x_i \in \{0, 1\} : \text{ whether or not item } i \text{ is chosen.}$$

Then the knapsack volume constraint can be written as

$$\sum_{i=1}^{n} v_i x_i \leq V.$$

The objective function is to maximize the total utility

$$\max \quad \sum_{i=1}^{n} u_i x_i.$$

2 Logical Constraints on 0/1 Variables

In MILO models, $0/1$ variables are often used to indicate whether a certain condition happens or not. For example, for a variable z_i, we say that condition i is satisfied if $z_i = 1$, and $z_i = 0$ otherwise, for some $i = 1, 2$, or 3. We can impose logical constraints on these $0/1$ as follows.

- "If-then" constraint: if condition 1 is satisfied, then we also need to satisfy condition 2

$$z_2 \geq z_1, \quad z_1, z_2 \in \{0, 1\}.$$

- Nonexclusive "either-or" constraint: either condition 1 is satisfied, or condition 2 is satisfied, and both of them can be satisfied at the same time

$$z_1 + z_2 \geq 1, \quad z_1, z_2 \in \{0, 1\}.$$

- Exclusive "either-or" constraint: either condition 1 is satisfied, or condition 2 is satisfied, but they cannot be satisfied at the same time

$$z_1 + z_2 = 1, \quad z_1, z_2 \in \{0, 1\}.$$

Sometimes we may compose our conditions with logical operations before in "if-then" statement. The following cases are some examples on how we can do this.

- "Not" operation: to say that condition 2 is satisfied if condition 1 is *not* satisfied, we can use

$$z_2 \geq 1 - z_1, \quad z_1, z_2 \in \{0, 1\}.$$

- "And" operation: to say that condition 3 is satisfied if conditions 1 *and* 2 are satisfied, we can use

$$z_3 \geq z_1 + z_2 - 1, \quad z_1, z_2, z_3 \in \{0, 1\}.$$

- "Or" operation: to say that condition 3 is satisfied if conditions 1 *or* 2 is satisfied, we can use

$$z_3 \geq z_1, \quad z_3 \geq z_2, \quad z_1, z_2, z_3 \in \{0, 1\}.$$

- "Xor" operation: to say that condition 3 is satisfied if conditions 1 *xor* 2 is satisfied

(i.e., exactly one of conditions 1 and 2 is satisfied), we can use

$$z_3 \geq z_3', \quad z_1 + z_2 + z_3' = 2z_3'', \quad z_1, z_2, z_3, z_3', z_3'' \in \{0, 1\}.$$

Here, z_3' and z_3'' are auxiliary variables introduced to model the "xor" operation.

The following resource allocation problem is an example of the knapsack problem with logical constraints.

Example 5. *There are 6 projects considered for potential investment of the \$100,000 budget for the upcoming year. The required investment and end-of-year payout amounts are described in the following table. We assume that partial investments are not allowed, that is, if a project*

	Project					
	1	2	3	4	5	6
Investment (\$·1000)	10	25	35	45	50	60
Payout (\$·1000)	12	30	41	55	65	77

is selected, then we must invest the full amount in it. Additionally, the following requirements need to be satisfied.

- *No more than three projects can be selected.*
- *If project 6 is chosen, then project 1 must also be chosen.*
- *Project 5 can be chosen only if project 2 is chosen.*
- *If project 3 is chosen, then project 4 cannot be chosen.*

The objective is to maximize the total end-of-year payout from the investment.
To define our decision variables, for each project $i = 1, \ldots, 6$, let

$$x_i = \begin{cases} 1, & \text{if we choose project } i, \\ 0, & \text{otherwise.} \end{cases}$$

Without the additional requirements, we only have the budget constraint

$$10x_1 + 25x_2 + 35x_3 + 45x_4 + 50x_5 + 60x_6 \leq 100.$$

The objective can be written as

$$\max \quad 12x_1 + 30x_2 + 41x_3 + 55x_4 + 65x_5 + 77x_6.$$

Each of the requirements can be written as follows.

- *No more than three projects can be selected.*

$$x_1 + x_2 + x_3 + x_4 + x_5 + x_6 \leq 3.$$

- *If project 6 is chosen, then project 1 must also be chosen.*

$$x_6 \leq x_1.$$

- *Project 5 can be chosen only if project 2 is chosen.*

$$x_5 \leq x_2.$$

- *If project 3 is chosen, then project 4 cannot be chosen.*

$$x_4 \leq 1 - x_3.$$

We code this MILO model in `model_allocation.py` *and use the solver* `SCIP` *to get the following result.*

```
The maximum payout is $119000.00.
The investment on each project is shown below.
x1: 1.0
x2: 1.0
x3: 0.0
x4: 0.0
x5: 0.0
x6: 1.0
```

The logical constraints can appear even when there are continuous variables, as shown in the next example.

Example 6. *Suppose that in Example 5, we now allow fractional investment but with a minimum for each project. The updated project information is listed in the table below. To update*

	Project					
	1	2	3	4	5	6
Investment ($·1000)	10	25	35	45	50	60
Payout ($·1000)	12	30	41	55	65	77
Min. Amount ($·1000)	2	5	4	9	6	7

the model, we define the following continuous decision variables in addition to $x_1, \ldots, x_6$,

$$y_i \geq 0 : \text{ the amount invested in the project } i, \quad i = 1, \ldots, 6.$$

We can impose constraints

$$m_i x_i \leq y_i \leq v_i x_i, \quad i = 1, \ldots, 6,$$

where $m_i > 0$ is the minimum investment ammount for project i, while v_i is the full investment amount. In this way, we invest in the project i if and only if $x_i = 1$. Accordingly, the budget constraint becomes

$$y_1 + y_2 + y_3 + y_4 + y_5 + y_6 \leq 100.$$

The objective can be written as

$$\max \quad \frac{12}{10}y_1 + \frac{30}{25}y_2 + \frac{41}{35}y_3 + \frac{55}{45}y_4 + \frac{65}{50}y_5 + \frac{77}{60}y_6.$$

We code this MILO model in the script `model_fractional_allocation.py` *and get the following result.*

```
The maximum payout is $126000.00.
The investment on each project is shown below.
x1: 0.0
y1: $0.00
x2: 1.0
y2: $5000.00
x3: 0.0
y3: $0.00
x4: 1.0
y4: $45000.00
x5: 1.0
y5: $50000.00
x6: 0.0
y6: $0.00
```

We see that the total payout increases compared with Example 5 because we have more flexible investment options for each project. We are now investing in projects 2, 4, 5, which are different from the projects 1, 2, 6 previously.

In Example 6, we are enforcing the continuous variables y_i to be zero if the corresponding 0/1 variable x_i is zero. The modeling of such optional variable bound/-linear constraint can be done in a more general setting. For some linear constraint $j = 1, \ldots, m$,

$$\sum_{i=1}^{n} a_{ji} x_i \leq b_j,$$

we can "switch on or off" this constraint by an additional integer variable $z_j \in \{0, 1\}$ by

$$\sum_{i=1}^{n} a_{ji} x_i \leq b_j + M z_j,$$

where the parameter M is a "sufficiently large" number. Theoretically speaking, we

should choose M such that during any algorithmic step (e.g., an simplex method iteration), the sign of any linear expression involving M only depends on the sign of M. In practice, assuming that our problem is bounded, we may adaptively increase the value of M by a fixed multiplicative factor each time, until our objective value does not change any more. There are also many problems where we can naturally get a good choice of M from the problem data, as shown in the following example.

Example 7. *A wholesale company specializing in one product considers the possibility of opening up to m warehouses W_i of capacity b_i, for $i = 1, \dots, m$, to serve n retail locations R_j, for $j = 1, \dots, n$. The fixed cost of opening warehouse W_i is f_i. The capacity of b_i means that we can ship up to b_i units of product from warehouse W_i. Transporting one unit of the product from W_i to R_j costs c_{ij} dollars. To satisfy the demand, at least d_j units of the product must be delivered to R_j. The goal is to decide which of the m warehouses to open and how many units of the product should be shipped from each opened warehouse to each retail location, minimizing the overall cost for the company.*
As in the usual transportation problem, we define the continuous variables for each $i = 1, \dots, m$ and $j = 1, \dots, n$

$$x_{ij} \geq 0 : \text{ the product quantity shipped from } W_i \text{ to } R_j.$$

To model the fixed cost, we introduce for each $i = 1, \dots, m$

$$y_i = \begin{cases} 1, & \text{if warehouse } i \text{ is open,} \\ 0, & \text{otherwise.} \end{cases}$$

The objective is to minimize the total cost of transportation plus the fixed charges of opening warehouse:

$$\min \quad \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} + \sum_{i=1}^{m} f_i y_i.$$

To satisfy the demand at R_j, we can write

$$\sum_{i=1}^{m} x_{ij} \geq d_j, \quad j = 1, \dots, n.$$

We also need to make sure that the number of units shipped out of W_i does not exceed the capacity:

$$\sum_{j=1}^{n} x_{ij} \leq b_i, \quad i = 1, \dots, m.$$

In addition, if a warehouse is not opened, then no units can be shipped out of it:

$$\text{if } y_i = 0, \quad \text{then } x_{ij} = 0, \quad j = 1, \dots, m.$$

Note that this is an optional linear constraint, and b_i is a natural choice for the big-M for every x_{ij}, $j = 1, \ldots, n$. Thus we can write our MILO model as

$$\begin{aligned}
\min \quad & \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} + \sum_{i=1}^{m} f_i y_i, \\
\text{s. t.} \quad & \sum_{i=1}^{m} x_{ij} \geq d_j, \quad j = 1, \ldots, n, \\
& \sum_{j=1}^{n} x_{ij} \leq b_i, \quad i = 1, \ldots, m, \\
& x_{ij} \leq b_i y_i, \quad i = 1, \ldots, m,\ j = 1, \ldots, n, \\
& x_{ij} \geq 0,\ y_i \in \{0, 1\}, \quad i = 1, \ldots, m,\ j = 1, \ldots, n.
\end{aligned}$$

An important application of MILO is to model piecewise linear functions that are not convex nor concave. For simplicity, we focus on modeling continuous piecewise linear functions on an interval I of finite length. Recall that a function f is piecewise linear on $I := \{x \in \mathbb{R} : \underline{a} \leq x \leq \bar{a}\}$ if we can find points $\underline{a} = a_0 < a_1 < \cdots < a_l = \bar{a}$, such that on each subinterval $I_k := \{x \in \mathbb{R} : a_{k-1} < x < a_k\}$, $k = 1, \ldots, l$, $f(x)$ is an affine linear function, i.e., there exist $b_k, c_k \in \mathbb{R}$ such that

$$f(x) = b_k x + c_k, \quad \forall\, x \in I_k, \quad k = 1, \ldots, l.$$

The continuity assumption then translates into conditions

$$f(a_k) = a_k b_k + c_k = a_k b_{k+1} + c_{k+1}, \quad \forall\, k = 1, \ldots, l-1.$$

Suppose we would like to model the equation $y = f(x)$, $x \in I$. We can define auxiliary variables

$$z_k = \begin{cases} 1, & \text{if } x \text{ lies in the interval } I_k, \\ 0, & \text{otherwise.} \end{cases}$$

By definition, we should have

$$\sum_{k=1}^{l} z_k = 1.$$

Note that if $x = a_k$ for some $k = 1, \ldots, k-1$, then we can allow either $z_k = 1$ or $z_{k+1} = 1$. We need to enforce the linear constraints $y = b_k x + c_k$ if $z_k = 1$, which can be written as

$$\begin{aligned}
y &\leq b_k x + c_k + M(1 - z_k), \\
y &\geq b_k x + c_k - M(1 - z_k),
\end{aligned}$$

for some sufficiently large $M > 0$, e.g., $M = \max\{\sup_{x \in I} f(x), -\inf_{x \in I} f(x)\}$. In practice, using this big-M model can sometimes lead to inefficiency in the solution step

(see the next section). Thus we describe an alternative way to model piecewise linear functions.

We observe that if f is a continuous piecewise linear function, then on each subinterval I_k, the value y should be a convex combination of $f(a_{k-1}), f(a_k)$ in the same way x is a convex combination of a_{k-1}, a_k. Thus we define more auxiliary variables $0 \leq w_0, w_1, \ldots, w_k \leq 1$ as convex combination coefficients.

- If $x \notin I_1$, then $w_0 = 0$, which can be expressed as

$$w_0 \leq z_1.$$

- If $x \notin I_k$ *and* $x \notin I_{k+1}$, $k \leq l-1$, then $w_k = 0$, which can be expressed as

$$w_k \leq z_k + z_{k+1}, \quad k = 1, \ldots, l-1.$$

- If $x \notin I_l$, then $w_l = 0$, which can be expressed as

$$w_l \leq z_l.$$

Then we can write x, y as convex combinations

$$\begin{aligned} 1 &= \sum_{k=0}^{l} w_k, \\ x &= \sum_{k=0}^{l} w_k a_k, \\ y &= \sum_{k=0}^{l} w_k f(a_k). \end{aligned}$$

Example 8. *A large-scale grocery retailer must purchase onions for two of their stores. Onions can be purchased from three farms. Here are the relevant details: Store 1 requires at least 1000 units and store 2 requires at least 2000 units of onions. Farm 1 sells onions at \$3 per unit and farm 2 sells onions at \$4 per unit. Farm 3 sells onions in the following fashion. The first 300 units are sold for \$3 per unit, the next 400 units are sold at a discounted rate of \$2.5 per unit. However, the price goes up after that to \$5, as the farm believes that there may be more demand than supply. For example, if 800 units are purchased from farm 3, then the cost is*

$$\$3 \cdot 300 + \$2.5 \cdot 400 + \$5 \cdot 100 = \$2400.$$

The transportation cost per unit from the farms to the stores are given below.

	Farm 1	*Farm 2*	*Farm 3*
Store 1	*\$1*	*\$1*	*\$2*
Store 2	*\$2*	*\$1*	*\$1*

The goal is to determine how many units of onions to purchase from each farm such that the total cost is minimized. For $i = 1, 2, 3$ *and* $j = 1, 2$*, let*

$$x_{ij} \geq 0 : \textit{number of units of onions to purchase from farm } i \textit{ for store } j.$$

Let $y \in \mathbb{R}$ *denote the cost of purchase from farm 3. The store demand constraints are*

$$\begin{aligned} x_{11} + x_{12} + x_{13} &\geq 1000, \\ x_{21} + x_{22} + x_{23} &\geq 2000. \end{aligned}$$

The objective is to minimize the total purchase and transportation cost

$$\begin{aligned} \min \quad & 3(x_{11} + x_{21}) + 4(x_{12} + x_{22}) + y \\ & + x_{11} + x_{12} + 2x_{13} + 2x_{21} + x_{22} + x_{23}. \end{aligned}$$

To model the function $y = f(x_{13} + x_{23})$*, note that*

$$f(0) = 0, \quad f(300) = 900, \quad f(700) = 1900, \quad f(3000) = 13400.$$

For $k = 1, 2, 3$*, let*

$$z_k \in \{0, 1\} \textit{ denote whether } x_{13} + x_{23} \textit{ lies in the interval } I_k,$$

and for $k = 0, 1, 2, 3$*, let*

$$0 \leq w_k \leq 1 \textit{ be the kth convex combination coefficient}.$$

We can write the constraint $y = f(x_{13} + x_{23})$ *through the following ones:*

$$\begin{aligned} w_0 &\leq z_1, \\ w_1 &\leq z_1 + z_2, \\ w_2 &\leq z_2 + z_3, \\ w_3 &\leq z_3, \\ z_1 + z_2 + z_3 &= 1, \\ w_0 + w_1 + w_2 + w_3 &= 1, \\ 300w_1 + 700w_2 + 3000w_3 &= x_{13} + x_{23}, \\ 900w_1 + 1900w_2 + 13400w_3 &= y. \end{aligned}$$

We code the MILO model in the script `model_purchase.py` *and use* `SCIP` *to solve it. The result is printed below.*

```
The minimum total cost for onion purchase is 13100.000000000004.
x11 = 1000.0000000000001
x21 = 0.0
x31 = 0.0
x12 = 0.0
x22 = 1300.0000000000002
x32 = 700.0
```

3 A Glance at MILO Solution Methods

There are two major families of solution methods in solving MILO problems: branch-and-bound methods and cutting plane methods. Each of them deserve much more than what we can spend in this course. Our goal here is modest: to give an idea of how MILO problems can be solved and how the solution methods are related to the simplex method we used for LO problems. For any MILO problem (1), we define its LO relaxation as

$$\begin{aligned} \min \text{ / } \max \quad & c^\mathsf{T} x \\ \text{s.t.} \quad & Ax \le b, \\ & x \in \mathbb{R}^{n_1+n_2}. \end{aligned} \tag{2}$$

3.1 The Branch-and-bound Methods

Given our MILO problem, suppose we found a solution $\bar{x}$ to the LO relaxation.

- If $\bar{x} \in \mathbb{R}^{n_1} \times \mathbb{Z}^{n_2}$, then $\bar{x}$ is feasible to the MILO problem;
- otherwise there exists $n_1 < i \le n_1 + n_2$ such that $\bar{x}_i$ is fractional. In this case, we need to solve the problem with either of the two constraints
  - $x_i \ge \lceil \bar{x}_i \rceil$,
  - $x_i \le \lfloor \bar{x}_i \rfloor$,

Note that at least one of them contains the true optimal solution to our MILO. This *branching* procedure leads to a *tree* of LO subproblems. For example, for the LO relaxation of our MILO problem

$$\text{LO}_0 : \max_{x \in \mathbb{R}^{n_1+n_2}} \quad c^\mathsf{T} x \quad \text{s.t. } Ax \le b,$$

if we get a fractional solution $\bar{x}$ such that $\bar{x}_i \notin \mathbb{Z}$, we then get two LO problems

$$\text{LO}_1 : \max_{x \in \mathbb{R}^{n_1+n_2}} \quad c^\mathsf{T} x \quad \text{s.t. } Ax \le b, x_i \ge \lceil \bar{x}_i \rceil,$$

and

$$\mathrm{LO}_2 : \max_{x \in \mathbb{R}^{n_1+n_2}} \quad c^\mathsf{T} x \quad \text{s. t. } Ax \leq b, x_i \leq \lfloor \bar{x}_i \rfloor.$$

Then again if we get a fractional solution $\tilde{x}$ to LO_1, with $\tilde{x}_j \notin \mathbb{Z}$, we can continue the *branching* procedure and get

$$\mathrm{LO}_3 : \max_{x \in \mathbb{R}^{n_1+n_2}} \quad c^\mathsf{T} x \quad \text{s. t. } Ax \leq b, x_i \geq \lceil \bar{x}_i \rceil, x_j \geq \lceil \tilde{x}_j \rceil,$$

and

$$\mathrm{LO}_4 : \max_{x \in \mathbb{R}^{n_1+n_2}} \quad c^\mathsf{T} x \quad \text{s. t. } Ax \leq b, x_i \geq \lceil \bar{x}_i \rceil, x_j \leq \lfloor \tilde{x}_j \rfloor.$$

While it seems that we may need to enumerate all possible integers for $x_{n_1+1}, \ldots, x_{n_1+n_2}$ in our feasible region, many of the LO problems in this procedure do not need any further branching. We can *prune* the problem LO_i if

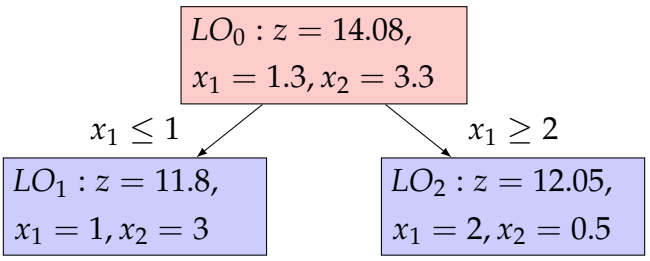
- we find a solution $x \in \mathbb{R}^{n_1} \times \mathbb{Z}^{n_2}$ to the LO problem LO_i;
- the LO problem LO_i is infeasible; or
- the optimal value of LO_i is worse than the best *bound* (the objective value of the best solution we have found).

The pruning step, especially when we already obtained a high-quality bound, would often greatly save our computational effort. We illustrate the branch-and-bound method by the following simple example.

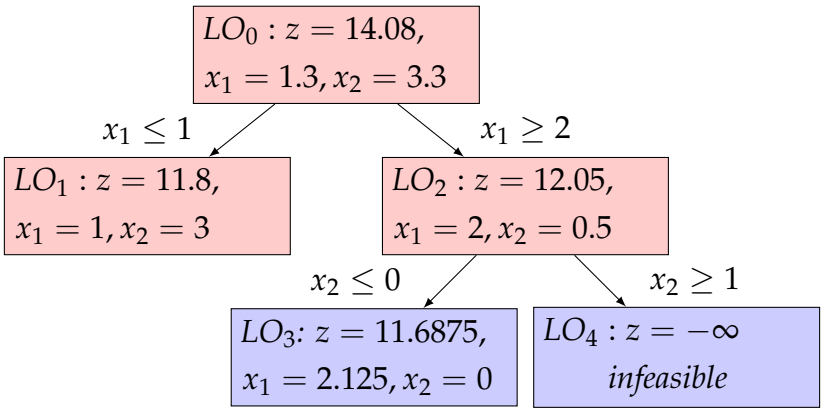
Example 9. *Consider the MILO problem*

$$\begin{aligned} \max \quad & 5.5x_1 + 2.1x_2 \\ \text{s. t.} \quad & -x_1 + x_2 \leq 2, \\ & 8x_1 + 2x_2 \leq 17, \\ & x_1, x_2 \in \mathbb{Z}_{\geq 0}. \end{aligned}$$

The solution of the LO relaxation is $x_1 = 1.3, x_2 = 3.3$, while the LO optimal value is $z = 14.08$, which gives an upper bound on our maximization problem. We can branch on the variable x_1: $x_1 \leq 1$ and $x_2 \geq 2$.



Here, the branch LO_1 is pruned by integrality. From LO_2 we branch on x_2: $x_2 \leq 0$ and $x_2 \geq 1$.



Now the branch LO_3 is pruned by bound, and the branch LO_4 is pruned by infeasibility. The optimal value of the MILO problem is thus $z = 11.8$ with an optimal solution $x_1 = 1$, $x_2 = 3$.

The numerical performance of the branch-and-bound methods will often depend on the *branching rule*, i.e., which LO problem to solve next. There are numerous branching rules and heuristics for high-quality solutions for pruning based on the structure of the MILO problem. We remark that in each branching step, we are simply resolving the LO problem with one additional inequality constraint. Thus we may use the current primal-dual solution information in our simplex tableau to warm start the dual simplex method.

3.2 The Cutting Plane Methods

Recall that the feasible regions of LO problems are polyhedra. If all of the vertices of the feasible region of the LO relaxation (2) are integer points for the last n_2 coordinates, then the simplex method will find an optimal solution to our MILO problem (1). Such polyhedra are called *integral*, but they are uncommon in practice. The main idea of the cutting plane methods is to artificially add linear constraints that are *valid* for all (mixed-)integer points while being able to "cut off" non-integer vertices.

As an simple illustration, we consider the following discrete (pure-integer) optimization case (i.e., $n_1 = 0$, $n = n_2$). Suppose that we have found a basic optimal solution $\bar{x}$ to the standard form of the LO relaxation (2), where $\bar{x}_i = \bar{b}_i \notin \mathbb{Z}$ for some $i \in B$. The corresponding constraint in the tableau is

$$x_i + \sum_{j \in N} \bar{a}_{ij} x_j = \bar{b}_i. \tag{3}$$

We claim that the following inequality is valid for all feasible solutions to the MILO problem (1)

$$\bar{b}_i - \lfloor \bar{b}_i \rfloor - \sum_{j \in N} (\bar{a}_{ij} - \lfloor \bar{a}_{ij} \rfloor) x_j \leq 0. \tag{4}$$

To see this, note that since $x_j \in \mathbb{Z}$ for each $j \in N$, we have

$$\lfloor \bar{b}_i \rfloor = \left\lfloor \sum_{j \in N} \bar{a}_{ij} x_j \right\rfloor \geq \sum_{j \in N} \lfloor \bar{a}_{ij} \rfloor x_j.$$

By substituting (3) in the above inequality, we obtain the Gomory fractional cut (4).

Example 10. *Consider the same MILO problem*

$$\begin{aligned} \max \quad & 5.5x_1 + 2.1x_2 \\ \text{s. t.} \quad & -x_1 + x_2 \leq 2, \\ & 8x_1 + 2x_2 \leq 17, \\ & x_1, x_2 \in \mathbb{Z}_{\geq 0}. \end{aligned}$$

We may introduce slack variables $x_3, x_4 \in \mathbb{Z}_{\geq 0}$ for the inequality constraints, due to the integrality of the left-hand side coefficients and the right-hand sides.

$$\begin{aligned} \max \quad & 5.5x_1 + 2.1x_2 \\ \text{s. t.} \quad & -x_1 + x_2 + x_3 = 2, \\ & 8x_1 + 2x_2 + x_4 = 17, \\ & x_1, x_2, x_3, x_4 \in \mathbb{Z}_{\geq 0}. \end{aligned}$$

Suppose we have found the simplex tableau associated with the optimal solution $x_1 = 1.3, x_2 = 3.3$ to its LO relaxation.

z	x_1	x_2	x_3	x_4	*rhs*	*basis*
1	0	0	0.58	0.76	14.08	z
0	0	1	0.8	0.1	3.3	x_2
0	1	0	−0.2	0.1	1.3	x_1

In particular, the second row can be written as

$$x_2 + 0.8x_3 + 0.1x_4 = 3.3.$$

The Gomory fractional cut can be applied with $i = 2$, $N = \{3, 4\}$, $\bar{b}_2 = 3.3$, $\bar{a}_{23} = 0.8$, and $\bar{a}_{24} = 0.1$

$$0.8x_3 + 0.1x_4 \geq 0.3.$$

Since $x_3 = 2 + x_1 - x_2$ and $x_4 = 17 - 8x_1 - 2x_2$, this yields

$$x_2 \leq 3.$$

We plot this cut in Figure 1.

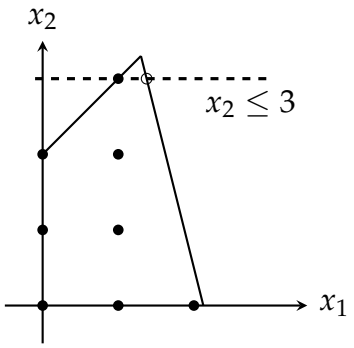


Figure 1: Illustration of a Gomory fractional cut

The idea of rounding coefficients can be used to generate cuts for the mixed-integer case, which is known as the *Gomory mixed-integer cuts*. Suppose that we have found a basic feasible solution $\bar{x}$ to the LO relaxation (2), with the corresponding row (3) where $i \in B \cap \{n_1 + 1, \ldots, n\}$. Let $N_1 := N \cap \{1, \ldots, n_1\}$, $N_2 := N \cap \{n_1 + 1, \ldots, n_1 + n_2\}$, $f_0 := \bar{b}_i - \lfloor \bar{b}_i \rfloor$, and $f_j := \bar{a}_{ij} - \lfloor \bar{a}_{ij} \rfloor$, for $j \in N$.
The Gomory mixed integer cut can be written as

$$\sum_{\substack{j \in N_2: \\ f_j \leq f_0}} \frac{f_j}{f_0} x_j + \sum_{\substack{j \in N_2: \\ f_j > f_0}} \frac{1 - f_j}{1 - f_0} x_j + \sum_{\substack{j \in N_1: \\ \bar{a}_{ij} \geq 0}} \frac{\bar{a}_{ij}}{f_0} x_j - \sum_{\substack{j \in N_1: \\ \bar{a}_{ij} < 0}} \frac{\bar{a}_{ij}}{1 - f_0} x_j \geq 1. \tag{5}$$

Example 10 (continued)**.** *Here we have* $f_0 = 0.3$, $f_3 = 0.8$, *and* $f_4 = 0.1$. *Note that we do not have any continuous variable* $N_1 = \varnothing$, *so the formula* (5)

$$\sum_{\substack{j \in N_2: \\ f_j \leq f_0}} \frac{f_j}{f_0} x_j + \sum_{\substack{j \in N_2: \\ f_j > f_0}} \frac{1 - f_j}{1 - f_0} x_j \geq 1$$

then becomes

$$\frac{1 - 0.8}{1 - 0.3} x_3 + \frac{0.1}{0.3} x_4 \geq 1 \iff 6x_3 + 7x_4 \geq 21.$$

Using $x_3 = 2 + x_1 - x_2$ *and* $x_4 = 17 - 8x_1 - 2x_2$, *we can write the cut in terms of* x_1 *and* x_2 *as*

$$5x_1 + 2x_2 \leq 11.$$

We plot this cut in Figure 2. Compared with Figure 1, it shows that Gomory mixed-integer cut is stronger, *in the sense that it "cuts off" more non-integer points in the LO relaxation.*

The proof of the Gomory mixed-integer cuts (5) consists of a sequence of mixed-integer cuts, which are presented below.

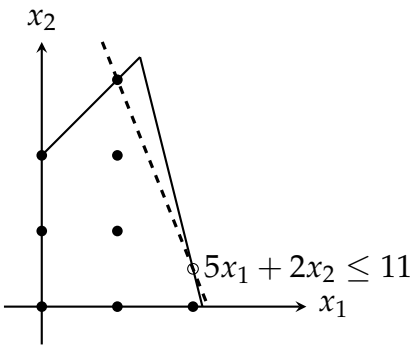


Figure 2: Illustration of a Gomory mixed-integer cut

Lemma 1. *If* $x_1 + \sum_{j=2}^{n} x_j \geq b$, $x_1 \geq 0$, $x_2, \ldots, x_n \in \mathbb{Z}$, *and* $f := b - \lfloor b \rfloor > 0$, *then*

$$\frac{x_1}{f} + \sum_{j=2}^{n} x_j \geq \lceil b \rceil.$$

Proof. Note that

$$f\left(\lceil b \rceil - \sum_{j=2}^{n} x_j\right) = f + f\left(\lfloor b \rfloor - \sum_{j=2}^{n} x_j\right) \leq f + \left(\lfloor b \rfloor - \sum_{j=2}^{n} x_j\right) = b - \sum_{j=2}^{n} x_j \leq x_1.$$

Divide both sides by f and we are done. □

Lemma 2. *If* $\sum_{j=2}^{n} x_j \leq b + x_1$, $x_1 \geq 0$, $x_2, \ldots, x_n \in \mathbb{Z}$, *and* $f := b - \lfloor b \rfloor > 0$, *then*

$$\sum_{j=2}^{n} x_j \leq \lfloor b \rfloor + \frac{x_1}{1-f}.$$

Proof. Apply Lemma 1 with $-b$ and $-x_j$ for $j = 2, \ldots, n$. Note that $\lfloor -b \rfloor = -\lceil b \rceil$ and $-b + \lfloor -b \rfloor = 1 - f$. □

Lemma 3. *Consider* $\sum_{j=2}^{n} a_j x_j \leq b + x_1$, *where* $x_1 \geq 0$ *and* $x_j \in \mathbb{Z}_{\geq 0}$ *for* $j = 2, \ldots, n$. *Let* $f := b - \lfloor b \rfloor$ *and* $f_j := a_j - \lfloor a_j \rfloor$ *for* $j = 2, \ldots, n$. *Then*

$$\sum_{j: f_j \leq f} \lfloor a_j \rfloor x_j + \sum_{j: f_j > f} \left(\lfloor a_j \rfloor + \frac{f_j - f}{1-f}\right) x_j \leq \lfloor b \rfloor + \frac{x_1}{1-f}.$$

Proof. Note that

$$\sum_{j: f_j \leq f} \lfloor a_j \rfloor x_j + \sum_{j: f_j > f} \lceil a_j \rceil x_j \leq b + x_1 + \sum_{j: f_j > f} (1 - f_j) x_j.$$

The left-hand side are all integers, so we can apply Lemma 2 with $x_1 + \sum_{j: f_j > f} (1 - f_j) x_j \geq 0$ regarded as the continuous value. □

Proposition 4 (Gomory mixed-integer cuts). *If $x \in \mathbb{R}^{n_1}_{\geq 0} \times \mathbb{Z}^{n_2}_{\geq 0}$ satisfies (3), then it also satisfies (5). Moreover, the point $x_i = \bar{b}_i$ with $x_j = 0$ for all $j \in N$ does not.*

Proof. Note that the equation (3) implies

$$x_i + \sum_{j \in N_2} \bar{a}_{ij} x_j \leq \bar{b}_i + \sum_{\substack{j \in N_1: \\ \bar{a}_{ij} < 0}} -\bar{a}_{ij} x_j.$$

Now apply Lemma 3 and we get

$$x_i + \sum_{\substack{j \in N_2: \\ f_j \leq f}} \lfloor \bar{a}_{ij} \rfloor x_j + \sum_{\substack{j \in N_2: \\ f_j > f}} \left(\lfloor \bar{a}_{ij} \rfloor + \frac{f_j - f_0}{1 - f_0} \right) \leq \lfloor \bar{b}_i \rfloor + \sum_{\substack{j \in N_1: \\ \bar{a}_{ij} < 0}} \frac{-\bar{a}_{ij}}{1 - f_0} x_j.$$

Now substitute x_i using the equation (3) and we are done. The last claim follows from $f_0 > 0$. □

We remark that there are many more types cuts used in MILO problems, most of which exploits some further structure of the data A and b. Due to the limit of this course, we refer any interested reader to the textbook [1] for further study.

4 More on Integer Modeling

Example 11. *A salesperson wants to visit n cities. The distance between the different cities $i, j \in \{1, \ldots, n\}$ is denoted as d_{ij}. Starting at city 1, the salesperson must visit each city* exactly once *and then return to city 1 (see Figure 3). The goal is to minimize the total distance traveled. To model this problem, we can define variables for each $i \neq j$, $i, j \in \{1, \ldots, n\}$*

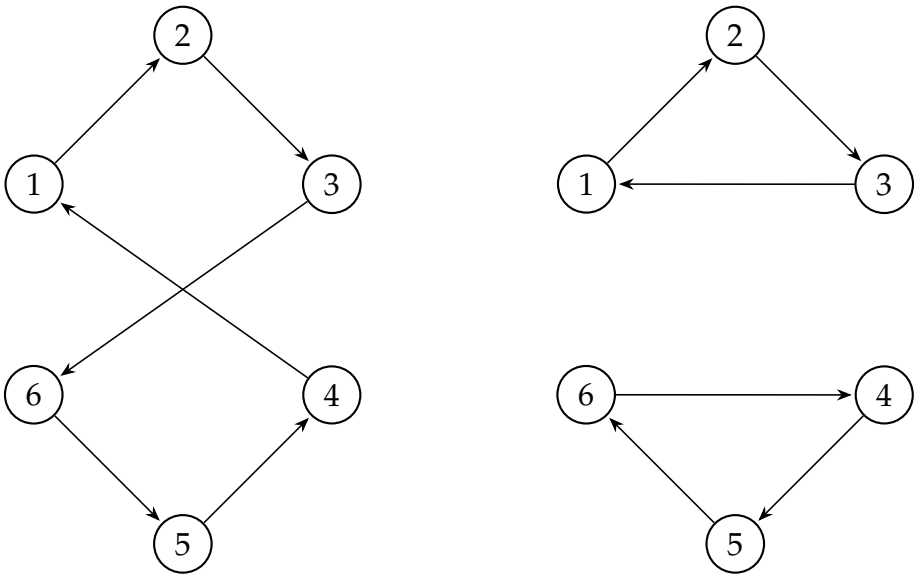
$$x_{ij} = \begin{cases} 1, & \text{if the salesperson travels from city } i \text{ to city } j, \\ 0, & \text{otherwise.} \end{cases}$$

The objective is to minimize the total distance

$$\min \quad \sum_{i=1}^{n} \sum_{j \neq i} d_{ij} x_{ij}.$$

Note that the salesperson should arrive at and leave from each city exactly once, *which can be written as the constraints*

$$\sum_{j \neq i} x_{ij} = 1, \quad i = 1, \ldots, n,$$
$$\sum_{i \neq j} x_{ij} = 1, \quad j = 1, \ldots, n.$$



(a) a feasible travel plan (b) an infeasible travel plan

Figure 3: Traveling plans of visiting 6 cities

The travel plan in Figure 3b satisfies all these constraints, but is still infeasible as we go back to city 1 (where we started) before we visited all of the 6 cities. We call the paths such as $1 \to 2 \to 3 \to 1$ subtours, *which need to be eliminated by additional constraints*

$$\sum_{i \in S, j \notin S} x_{ij} \geq 1, \quad \text{for all subsets } S \subsetneq \{1, \ldots, n\}, \quad 2 \leq |S| \leq n-2.$$

The number of the subtour elimination constraints in Example 11 is typically huge: for $n \geq 4$, we would have $2^n - 2 - 2n$ possible subtours, which is over 1 million for a mere number of 20 cities! People thus often use a technique called *constraint generation* to solve this problem. The idea is to solve the problem with only a subset of the constraints and dynamically add the rest as needed. In Example 11, once we get a solution $\bar{x}_{ij}$, we can start out tour from city 1 and move to the city j such that $\bar{x}_{1j} = 1$. By repeating the procedure, we would either visit all the cities before going back to city 1, or find a subtour $\bar{S}$. In the former case, we have successfully solved the traveling salesperson problem, while in the latter we need to add the subtour elimination constraint for $\bar{S}$ and resolve the problem.

Example 12. *A paper mill produces large rolls of paper of width W, which are then cut into rolls of various smaller widths in order to meet demand. Let m be the number of different widths that the mill produces. The mill receives an order for b_i rolls of width w_i for $i = 1, \ldots, m$, where $w_i \leq W$. The goal is to find the smallest number of large rolls needed to meet the demand.*
One way to formulate our MILO model is as follows. Suppose p is an upper bound on the number of paper rolls, such as $p = \sum_{i=1}^{m} b_i$. We can define our decision variables for $j =$

$1, \ldots, n$ *as*

$$y_j = \begin{cases} 1, & \text{if the large roll } j \text{ is used,} \\ 0, & \text{otherwise,} \end{cases}$$

and for $i = 1, \ldots, m$, $j = 1, \ldots, p$,

$$z_{ij} \in \mathbb{Z}_{\geq 0} : \text{ the number of rolls of width } w_i \text{ to be cut out of roll } j.$$

Then our MILO model can be written as

$$\begin{aligned} \min \quad & \sum_{j=1}^{p} y_j \\ \text{s. t.} \quad & \sum_{i=1}^{m} w_i z_{ij} \leq W y_i, \quad i = 1, \ldots, p, \\ & \sum_{j=1}^{p} z_{ij} \geq b_i, \quad i = 1, \ldots, m, \\ & y_j \in \{0, 1\}, \quad j = 1, \ldots, p, \\ & z_{ij} \in \mathbb{Z}_{\geq 0}, \quad i = 1, \ldots, m, \ j = 1, \ldots, p. \end{aligned} \tag{6}$$

This model has some disadvantages: its LO relaxation is usually weak, and there is no easy way to round fractional solutions from the LO relaxation into feasible integer solutions because we have inequality constraints of both directions. Alternatively, people consider the following formulation. Let $s \in \mathbb{Z}^m$ *denote a cutting pattern, where* s_i *rolls of width* w_i *are cut out of the large paper roll. The set of all cutting patterns is*

$$\mathcal{S} := \left\{ s \in \mathbb{Z}_{\geq 0}^m : \sum_{i=1}^{m} w_i s_i \leq W \right\}.$$

Now we can define for each $s \in \mathcal{S}$,

$$x_s \in \mathbb{Z}_{\geq 0} : \text{ the number of rolls cut according to the pattern } s.$$

The only constraints are

$$\sum_{s \in \mathcal{S}} s_i x_s \geq b_i, \quad i = 1, \ldots, m.$$

Thus our alternative MILO model is

$$\begin{aligned} \min \quad & \sum_{s \in \mathcal{S}} x_s \\ \text{s. t.} \quad & \sum_{s \in \mathcal{S}} s_i x_s \geq b_i, \quad i = 1, \ldots, m, \\ & x_s \in \mathbb{Z}_{\geq 0}, \quad s \in \mathcal{S}. \end{aligned} \tag{7}$$

The number of variables $|\mathcal{S}|$ can be potentially large, which motivates people to use the column generation *algorithm, as outlined below. The dual of its LO relaxation is*

$$\begin{aligned} \max \quad & \sum_{i=1}^{m} b_i u_i \\ \text{s. t.} \quad & \sum_{i=1}^{m} s_i u_i \le 1, \quad \forall s \in \mathcal{S}, \\ & u_1, \dots, u_m \ge 0. \end{aligned} \tag{8}$$

Suppose we use a subset $\mathcal{S}'$ of $\mathcal{S}$ and find a primal optimal solution $\bar{x}_s$, $s \in \mathcal{S}'$ and a dual optimal solution $\bar{u}_1, \dots, \bar{u}_m$. We can extend $\bar{x}_s$ to all of $\mathcal{S}$ by setting $\bar{x}_s = 0$ for all $s \in \mathcal{S} \setminus \mathcal{S}'$. Thus when the dual solution $\bar{u}$ is feasible, we know that $\bar{x}$ is optimal by the weak duality. This is equivalent to check that the constraints

$$\sum_{i=1}^{m} s_i \bar{u}_i \le 1$$

are satisfied for all $s \in \mathcal{S}$. Or equivalently, if the following problem

$$\max \quad \sum_{i=1}^{m} \bar{u}_i s_i, \quad s \in \mathcal{S} \tag{9}$$

has an optimal value is no more than 1, then $\bar{u}$ is feasible. Otherwise any $s \in \mathcal{S}$ such that $\sum_{i=1}^{m} \bar{u}_i s_i > 1$ can be added as a new variable $S' \leftarrow S' \cup \{s\}$. The problem (9) is in fact a knapsack problem (cf. Example 4) by the definition of $\mathcal{S}$.

Example 13. *The famous puzzle game* sudoku *can be formulated as an integer optimization problem. The goal is to fill numbers $\{1, 2, \dots, 9\}$ in the empty locations and the rule is that every number should appear only once in each row, in each column, and in each 3×3 square. Some numbers are provided in the beginning of the puzzle that cannot be changed. We can define variables for $i, j, k \in \{1, 2, \dots, 9\}$,*

$$x_{i,j,k} = \begin{cases} 1, & \text{if the number } k \text{ is selected at the location } (i, j), \\ 0, & \text{otherwise.} \end{cases}$$

The constraints consist of the following. Exactly one number should be selected at each location:

$$\sum_{k=1}^{9} x_{i,j,k} = 1, \quad i, j = 1, \dots, 9.$$

For the given numbers, we fix the corresponding variables to 1:

$$x_{i,j,k} = 1, \quad \text{if } S_{ij} = k.$$

2	5			3		9		1
	1				4			
4		7				2		8
		5	2					
				9	8	1		
	4				3			
			3	6			7	2
	7							3
9		3				6		4

Figure 4: A sudoku puzzle

For each row and column, every number should appear only once:

$$\sum_{j=1}^{9} x_{i,j,k} = 1, \quad i,k = 1,\ldots,9,$$

and

$$\sum_{i=1}^{9} x_{i,j,k} = 1, \quad j,k = 1,\ldots,9.$$

For each block, every number should appear only once:

$$\sum_{i''=1}^{3}\sum_{j''=1}^{3} x_{3(i'-1)+i'',3(j'-1)+j'',k} = 1, \quad i',j' = 1,\ldots,3,\ k = 1,\ldots,9.$$

The objective is not needed, or we can set trivially

$$\min \quad 0.$$

We can code this model in `model_sudoku.py` *and get the following solution to the puzzle shown in Figure 4.*

```
Found a solution to the sudoku puzzle:
2 5 8 7 3 6 9 4 1
```

```
6 1 9 8 2 4 3 5 7
4 3 7 9 1 5 2 6 8
3 9 5 2 7 1 4 8 6
7 6 2 4 9 8 1 3 5
8 4 1 6 5 3 7 2 9
1 8 4 3 6 9 5 7 2
5 7 6 1 4 2 8 9 3
9 2 3 5 8 7 6 1 4
```

References

[1] Michele Conforti, Gérard Cornuéjols, and Giacomo Zambelli. Integer Programming. Springer International Publishing, 2014.