Mixed-Integer Linear Programming (MILP): Model Formulation

Benoît Chachuat <benoit@mcmaster.ca>

McMaster University
Department of Chemical Engineering

ChE 4G03: Optimization in Chemical Engineering

Mixed-Integer Linear Programming

Integer Programs (IP)

An optimization model is an **Integer Program** if *any of* its decision variables is discrete

- If *all* variables are discrete, the model is a **pure integer program**
- Otherwise, the model is a **mixed-integer program**

Integer variables appear in many problems:

- Trays in a distillation column
- Number of employees (1000's)

Can be solved continuous, then rounded to nearest integer

- Number of parallel chemical reactors
- Whether or not to operate boiler#2 on Monday
- Scheduling people and equipment to tasks over time

Not appropriate to solve continuous and round after

Mixed-Integer Linear Programming

Class Exercise: Give more examples of integer decisions in the field of Chemical Engineering:

1. Fluid flow:
   -
   -
2. Heat transfer:
   -
   -
3. Mass transfer:
   -
   -
4. Reactor design:
   -
   -

Mixed-Integer Linear Programming

Linear vs. Nonlinear Integer Programs

- An IP model is an **integer linear program** (**ILP**) if its (single) objective function and *all* its constraints are linear
- Otherwise, it is an **integer nonlinear program** (**INLP**)

Standard Mixed-Integer Linear Programming (MILP) Formulation:

$$\min_{\mathbf{x},\mathbf{y}} \quad z \triangleq \mathbf{c}^\mathsf{T}\mathbf{x} + \mathbf{d}^\mathsf{T}\mathbf{y}$$

$$\text{s.t.} \quad \mathbf{A}\mathbf{x} + \mathbf{E}\mathbf{y} \left\{ \begin{array}{c} \leq \\ = \\ \geq \end{array} \right\} \mathbf{b}$$

$$\mathbf{x}_{\min} \leq \mathbf{x} \leq \mathbf{x}_{\max}, \quad \mathbf{y} \in \{0,1\}^{n_y}$$

- Mixed-Integer Nonlinear Programs (MINLPs) are very difficult to solve
- This is a great motivation for the continued use of LP methods!

Outline

Modeling MILP requires knowledge and ingenuity — We will learn some typical MILP formulations here:

1. Native MILP Formulations
   - "Lumpy" Linear Programs
   - Knapsack Models
   - Assignment and Matching Models
   - Scheduling Models
2. Approximations of Nonlinear Models as MILPs
   - Separable Programming

For additional examples, see Rardin (1998), Chapter 11

"Lumpy" Linear Programs

One broad class of "lumpy" LPs is when either/or decisions are added to what is otherwise an LP model

Expressing All-or-Nothing Requirements

All-or-nothing variable requirements of the form

$$x_j = 0 \lor x_j = u_j$$

can be modeled by substituting

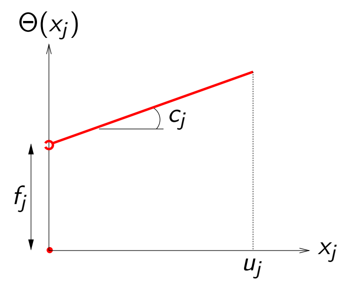
$$x_j \leftarrow u_j y_j, \quad \text{s.t. } y_j \in \{0,1\}$$

Example: In a blending model, use none or all of a given ingredient

$$\begin{aligned}
\min_{x_1,x_2,x_3} \ & 18x_1 + 3x_2 + 9x_3 \\
\text{s.t. } & 2x_1 + x_2 + 7x_3 \leq 150 \\
& 0 \leq x_1 \leq 60 \\
& 0 \leq x_2 \leq 30 \\
& x_3 = 0 \lor x_3 = 20
\end{aligned} \qquad \Longleftrightarrow$$

"Lumpy" Linear Programs (cont'd)

Another source of "lumpy" LP models is when the objective involves fixed charges (i.e., start-up cost)



Expressing Fixed-Charge Requirements

Fixed-cost requirements of the form: $\Theta(x_j) \triangleq \begin{cases} f_j + c_j x_j, & \text{if } x_j > 0 \\ 0, & \text{otherwise} \end{cases}$

with nonnegative fixed charge $f_j > 0$ and variable $x_j \leq u_j$, can be modeled by substituting

$$\Theta(x_j) \leftarrow f_j y_j + c_j x_j, \quad \text{s.t.} \quad x_j \leq u_j y_j \quad \text{and} \quad y_j \in \{0,1\}$$

Formulating Models for Fixed-Charged Objectives

Class Exercise: Consider a fixed-charge objective function

$$\min_{x_1,x_2} \Theta_1(x_1) + \Theta_2(x_2)$$

where

$$\Theta_1(x_1) \triangleq \begin{cases} 150 + 7x_1, & \text{if } x_1 > 0 \\ 0, & \text{otherwise} \end{cases}$$

$$\Theta_2(x_2) \triangleq \begin{cases} 110 + 9x_2, & \text{if } x_2 > 0 \\ 0, & \text{otherwise} \end{cases}$$

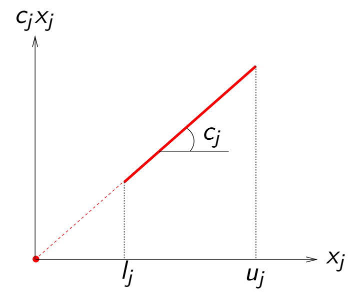
and x_1, x_2 satisfy

$$\begin{aligned} x_1 + x_2 &\geq 8 \\ 0 \leq x_1 &\leq 3 \\ 0 \leq x_2 &\leq 8. \end{aligned}$$

Formulate a corresponding MILP model.

"Lumpy" Linear Programs (cont'd)

Yet another source of "lumpy" LP models is when a decision variable is semi-continuous



Expressing Semi-Continuous Requirements

Semi-continuous variables of the form: $x_j = 0 \vee l_j \leq x_j \leq u_j$, with $l_j > 0$ can be modeled by introducing 2 new continuous variables $x_j^l, x_j^u \geq 0$, 1 new binary variable $y_j \in \{0,1\}$, and 2 additional constraints:

$$x_j = l_j x_j^l + u_j x_j^u$$
$$1 = y_j + x_j^l + x_j^u$$

Formulating Models for Semi-Continuous Variables

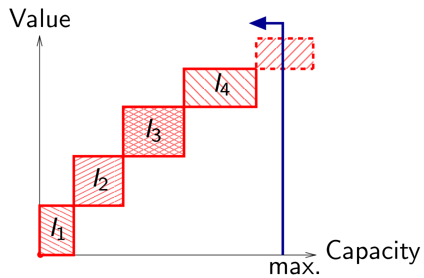
Class Exercise: Consider the following blending problem, where the ingredient x_3 is a semi-continuous variable:

$$\begin{aligned} \min_{x_1,x_2,x_3} \quad & 18x_1 + 3x_2 + 9x_3 \\ \text{s.t.} \quad & 2x_1 + x_2 + 7x_3 \leq 150 \\ & 0 \leq x_1 \leq 60 \\ & 0 \leq x_2 \leq 30 \\ & x_3 = 0 \vee 10 \leq x_3 \leq 20 \end{aligned}$$

Formulate a corresponding MILP model.

Knapsack Models

- The problem is to select a maximum value collection of items subject to limitations on resources consumed
- Knapsack models are the simplest of all (pure) integer linear programs (ILPs)



- Each element is either all in or all out of the selection:

$$\boxed{y_j \triangleq \begin{cases} 1, & \text{if item } j \text{ selected} \\ 0, & \text{otherwise} \end{cases}}$$

Example: Selection of projects subject to limitations on budgets

Class Exercise: Readily available Canadian coins are 1¢, 5¢, 10¢ and 25¢. Formulate an ILP model to minimize the number of coins needed to provide change amount for q¢.

Knapsack Models (cont'd)

1. **Dependence** of choice j on choice i can be enforced on corresponding binary variables by adding constraints of the form:

2. **Mutually exclusiveness** conditions allowing at most one of a set of choices J can be enforced by adding constraints of the form:

   Similarly, at most p of a set of choices J can be enforced as:

3. Constraints can also enforce selection of exactly p or at least p of a set of choices J

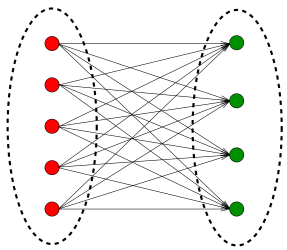
Formulating ILP Models

Class Exercise: The Department of Chemical Engineering at McMaster is acquiring optimization software for use in ChE 4G03.
The 4 codes available and the types of algorithms they provide are indicated by $\times$'s in the following table:

Algorithm Type	Code, j 1	2	3	4
LP	$\times$	$\times$	$\times$	$\times$
IP	—	$\times$	—	$\times$
NLP	—	—	$\times$	$\times$
Cost	3	4	6	14

1. Formulate a MILP model to acquire a minimum cost collection of codes providing LP, IP and NLP capability
2. Formulate a MILP model to acquire a minimum cost collection of codes with exactly one providing LP, one providing IP, and one providing NLP capability

Assignment Models



- The issue here is optimal matching or pairing of objects of two distinct types
- It is standard to model all assignment forms with the decision variables:

$$\boxed{y_{i,j} \triangleq \begin{cases} 1, & \text{if } i \text{ of the 1st set is matched with } j \text{ of the 2nd} \\ 0, & \text{otherwise} \end{cases}}$$

Examples: Assign sales personnel to customers, jobs to machine, etc.

- Assignment constraints forcing every i and/or every j to be assigned are of the form:

Assignment Models (cont'd)

- Generalized assignment constraints encompass cases where allocation of i to j requires fixed space $s_{i,j}$ within j capacity b_j:

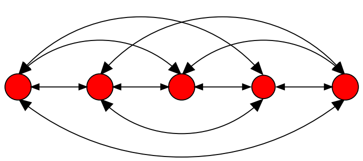
- Linear assignment models minimize/maximize a linear objective function of the form:

$$\boxed{\sum_{i\in I}\sum_{j\in J} c_{i,j}y_{i,j}}$$

where $c_{i,j}$ is the cost of assigning i to j

Class Exercise: Items $i = 1,\ldots,100$ of volume c_i are being stored in an automated warehouse. Storage location $j = 1,\ldots,20$ are at a distance d_j from the input/output station, and all have capacity b. Formulate a MILP model to store all items at minimum total travel distance.

Matching Models



- Unlike assignment models, matching models eliminate the distinction between the sets
- Decision variables of matching models typically are:

$$\boxed{y_{i,i'} \triangleq \begin{cases} 1, & \text{if } i \text{ is paired with } i' \\ 0, & \text{otherwise} \end{cases}}$$

where $i' > i$, by convention, to avoid double counting

- Matching constraints forcing some $i \in I$ to pair with and only with some other $i' \in I$ are of the form:

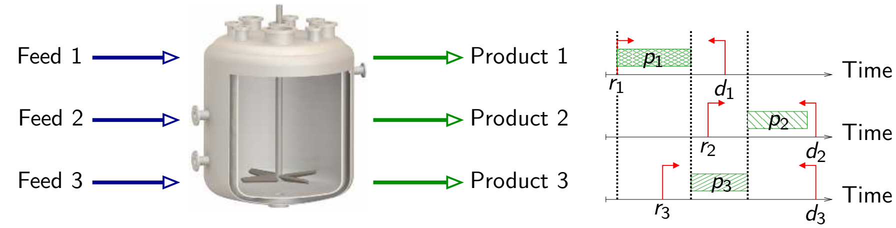
Matching Models (cont'd)

Class Exercise: The instructor of ChE 4G03 wants to assign his students to 2-person teams for a graded tutorial. each student s has scored his/her preference $p_{s,s'}$ for working with each other student s'.

1. Formulate a MILP model to form teams in a way that maximizes total preference.

Process Scheduling Models

- **Scheduling** is the allocation of resources over time
- **Single-process scheduling** problems seek an optimal sequence in which to complete a given collection of tasks on a single process that can accommodate only one job at a time



Typical Data:

- $p_j \triangleq$ estimated process time for task j
- $r_j \triangleq$ release time at which task j becomes available for processing
- $d_j \triangleq$ due date by which task j should be completed

Process Scheduling Models (cont'd)

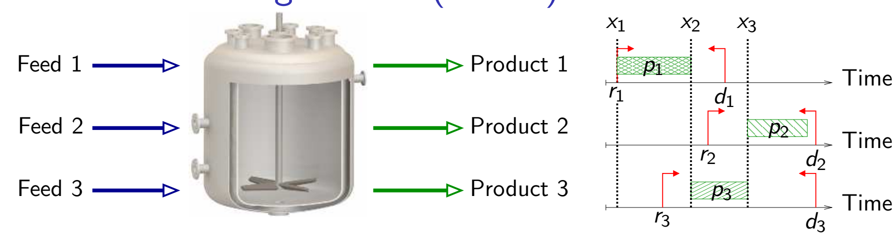
Time Decision Variables and Related Constraints:

- $x_j \triangleq$ start time for task j
- resource availability: $x_j \geq r_j, \ \forall j$

Handling of Due Dates:

- Due dates are usually handled as goals to be reflected in the objective function rather than explicit constraints
- This avoids the situation where there is no feasible schedule that meets all due dates
- Dates that must be met are termed **deadlines** to distinguish and can be easily enforced as: $x_j + p_j \leq d_j, \ \forall j$

Process Scheduling Models (cont'd)



Objective Functions — Often minimizes one of the following:

Max. completion time (makespan)	$\max\{x_j + p_j : j = 1, \ldots, n\}$
Mean completion time	$\frac{1}{n}\sum_{j=1}^{n}(x_j + p_j)$
Max. lateness	$\max\{x_j + p_j - d_j : j = 1, \ldots, n\}$
Mean lateness	$\frac{1}{n}\sum_{j=1}^{n}(x_j + p_j - d_j)$
Max. tardiness	$\max\{0, \max\{x_j + p_j - d_j : j = 1, \ldots, n\}\}$
Mean tardiness	$\frac{1}{n}\sum_{j=1}^{n}\max\{0, x_j + p_j - d_j\}$

Process Scheduling Models (cont'd)

Class Exercise: The following table shows the process time, release times, due dates, and scheduled starts for three jobs. Compute the corresponding value of each of the six objective functions listed previously.

	Job 1	**Job 2**	**Job 3**
Process time, p_i	15	6	9
Release time, r_i	5	10	0
Due Date, d_i	20	25	36
Scheduled start, x_i	9	24	0

1. Completion times $(x_i + p_i)$ are:

$$9 + 15 = 24, \qquad 24 + 6 = 30, \qquad 0 + 9 = 9$$

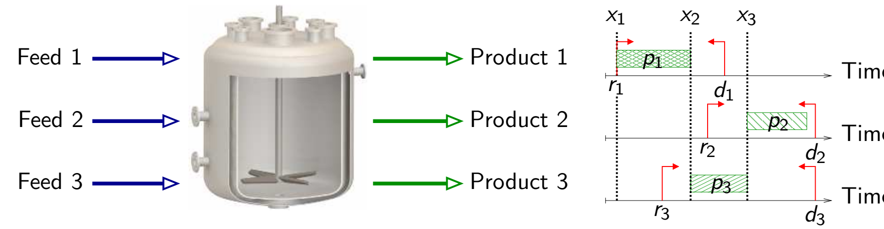
Maximum completion time is $\max\{24, 30, 9\} = 30$
Mean completion time is $\frac{1}{3}(24 + 30 + 9) = 21$

2. Lateness of the three jobs $(x_j + p_j - d_j)$ is:

$$9 + 15 - 20 = 4, \qquad 24 + 6 - 25 = 5, \qquad 0 + 9 - 36 = -27$$

Maximum lateness is $\max\{4, 5, -27\} = 5$
Mean completion time is $\frac{1}{3}(4 + 5 - 27) = -6$

Process Scheduling Models (cont'd)



- **Conflict Constraints:**

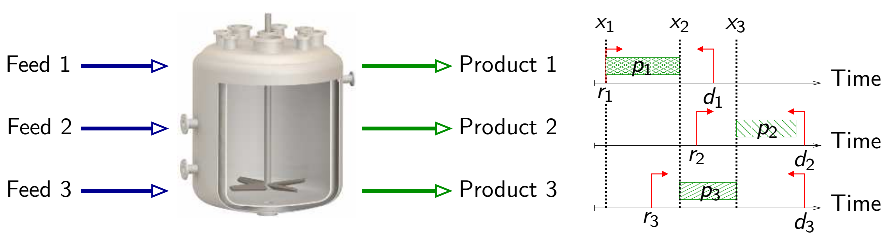
$$\boxed{x_j + p_j \le x_{j'} \quad \vee \quad x_{j'} + p_{j'} \le x_j, \quad \forall j, j' = 1, \ldots, n, \quad j \ne j'}$$

- **Reformulation Using Disjunctive Variables:**

$$\boxed{y_{j,j'} \triangleq \begin{cases} 1, & \text{if } j \text{ binding comes before } j' \\ 0, & \text{if } j' \text{ binding comes before } j \end{cases}}$$

Illustrative Example: $y_{1,2} = y_{1,3} = 1$, $y_{2,3} = 0$

Process Scheduling Models (cont'd)



Big-M Method

Conflicts between tasks j and j' can be prevented with **disjunctive constraint pairs**:

$$\boxed{\begin{aligned} x_j + p_j &\le x_{j'} + M(1 - y_{j,j'}) \\ x_{j'} + p_{j'} &\le x_j + M y_{j,j'} \end{aligned}}$$

where M is a large positive constant (e.g., the max. makespan)

Class Exercise: Formulate integer linear constraints for feasible schedules on a single process with 3 tasks having process times 14, 3, and 7.

Separable Programming

- Consider the following mathematical program:

$$\min_{\mathbf{x}} \quad z \triangleq \sum_{j=1}^{n} f_j(x_j) = f_1(x_1) + \cdots + f_n(x_n)$$

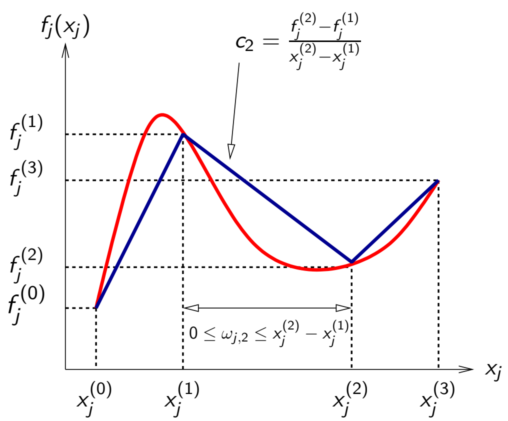
$$\text{s.t.} \quad \sum_{j=1}^{n} a_{i,j} x_j \le b_i, \quad i = 1, \ldots, m$$

$$x_j \ge 0, \quad j = 1, \ldots, n$$

  - The objective consists of n nonlinear, separable terms $f_j(x_j)$, each a function of a single variable only
  - The m constraints are linear
- When *each* f_j is convex or concave on the feasible region, the separable program can be approximated with an LP
- Otherwise, the separable program can be approximated with an **MILP**

Approximating General Separable Programs as MILPs

Piecewise affine approximation:
(N_j intervals)



- Define:

$$c_j^{(k)} = \frac{f_j^{(k)} - f_j^{(k-1)}}{x_j^{(k)} - x_j^{(k-1)}}$$

$$0 \leq \omega_{j,k} \leq x_j^{(k)} - x_j^{(k-1)}$$

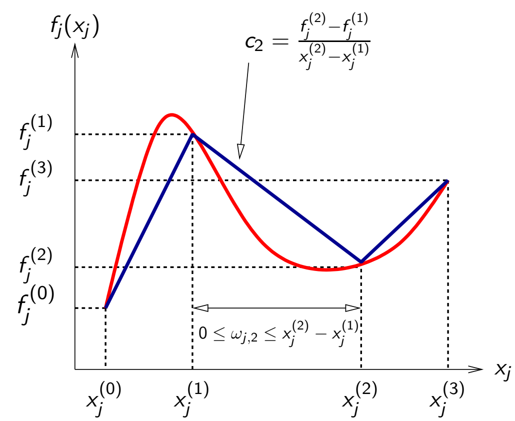
- Substitute *each* variable x_j and function f_j by:

$$x_j \leftarrow x_j^{(0)} + \sum_{k=1}^{N_j} \omega_{j,k}$$

$$f_j(x_j) \leftarrow f_j^{(0)} + \sum_{k=1}^{N_j} c_j^{(k)} \omega_{j,k}$$

Approximating General Separable Programs as MILPs

Piecewise affine approximation:
(N_j intervals)



- Disjunctive Variables:

$$y_{j,k} \triangleq \begin{cases} 0, & \text{if } \omega_{j,k} = 0 \\ 1, & \text{if } \omega_{j,k} > 0 \end{cases}$$

- Disjunctive Constraints:

$$\omega_{j,k} \leq \left[x_j^{(k)} - x_j^{(k-1)}\right] y_{j,k}, \quad \forall k = 1, \ldots, N_j$$

$$\omega_{j,k} \geq \left[x_j^{(k)} - x_j^{(k-1)}\right] y_{j,k+1}, \quad \forall k = 1, \ldots, N_j - 1$$

- Do you see why this works?

Approximating Separable Programs as MILPs (cont'd)

Approximate Mixed-Integer Linear Program (on N_j Intervals):

$$\begin{aligned}
\min_{\boldsymbol{\omega}, \mathbf{y}} \quad & \sum_{j=1}^{n} f_j^{(0)} + \sum_{j=1}^{n} \sum_{k=1}^{N_j} c_j^{(k)} \omega_{jk} \\
\text{s.t.} \quad & \sum_{j=1}^{n} \sum_{k=1}^{N_j} a_{i,j} \omega_{j,k} \leq b_i - \sum_{j=1}^{n} a_{i,j} x_j^{(0)}, \quad i = 1, \ldots, m \\
& \omega_{j,k} \leq \left[x_j^{(k)} - x_j^{(k-1)}\right] y_{j,k}, \quad k = 1, \ldots, N_j \\
& \omega_{j,k} \geq \left[x_j^{(k)} - x_j^{(k-1)}\right] y_{j,k+1}, \quad k = 1, \ldots, N_j - 1 \\
& y_{j,k} \in \{0, 1\}, \quad j = 1, \ldots, n, \quad k = 1, \ldots, N_j
\end{aligned}$$

- The solution can be made as accurate as desired by using enough intervals — One pays the price in terms of increased problem size!