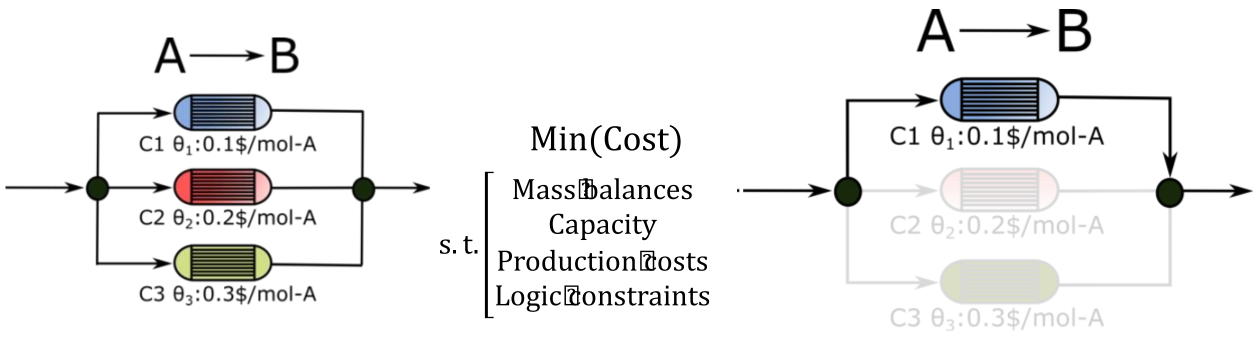
CHAPTER 8
FORMULATING MILP AND MINLP PROBLEMS

- In this chapter you should learn how to use binary variables to represent decision making in the design of chemical processes
- The second objective of the chapter is that you learn how to transform logic statements into equations.

Introduction

Superstructures

A superstructure can be understood as a network that contains all relevant alternatives to a design problem. Mathematically a superstructure can be typically represented using integer programming, that is the problems that result are either MILP or MINLPs. This representation mode was developed in the 70s by Edward Sargent and has been popularized as a way to represent design problems in chemical engineering. A trivial example is shown below; We are trying to model the selection of a catalyst and we have 3 options, we can formulate the selection process using a mathematical programming model, the solution to the problem, which is evident in this case is that catalyst 1 is optimal, that is what is shown on the right.



In chemical engineering countless problems have been tackled using this approach, a critical issue to render the approach effective is to be able to model using integers. This is the subject of this chapter.

Modeling with binaries

Binary variables are commonly used in modeling chemical processes, especially in design problems, because they are useful to represent decision making in situations where one faces mutually exclusive situations (e.g., should I install a piece of equipment or not). The type of problems where they appear are either MINLP (the most general case) or MILP problems.

MINLP

$$\begin{aligned} \min Z &= f(x,y) \\ s.t.\quad h(x,y) &= 0 \\ g(x,y) &\le 0 \\ x \in R^n \; & y \in \{0,1\}^m \end{aligned}$$

MILP

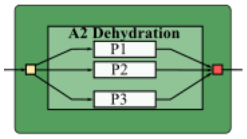
$$\begin{aligned} \min Z &= a^T x + b^T y \\ s.t.\quad Ax + By &\le d \\ x \in R^n \; & y \in \{0,1\}^m \end{aligned}$$

Conceptually, the idea is relatively simple, we should use binaries to derive constraints that enforce the decision-making process, this decision making respond to the nature of a binary variable, that is:

$$y_j = \begin{cases} 1 \rightarrow decision\ represented\ by\ y_j\ is\ made \\ 0 \rightarrow decision\ represented\ by\ y_j\ is\ not\ made \end{cases}$$

To show the use of binaries in the design of chemical processes (and other processes as well) let's explore some relevant examples:

- In many instances you have a set of choices and you want to enforce some conditions associated with these choices, for example, consider a set of catalyst that you can select to perform a reaction (P1,P2 and P3 in the figure):

  If you want to model decision making about the number of catalyst to be used, then you assign a binary variable to each of the catalysts, i.e., Y_{P1}, Y_{P2}, Y_{P3}. Then we can enforce some logic constraints such as:

  - Select at least one:

  $$\sum_{p \in P} Y_p \geq 1$$

  - Select exactly one:

  $$\sum_{p \in P} Y_p = 1$$

  - Select no more than one:

  $$\sum_{p \in P} Y_p \leq 1$$

  We will see later that the first of these constraints can be interpreted as an inclusive OR, the second one as an EXCLUSIVE OR, and the third one as a knapsack constraint.

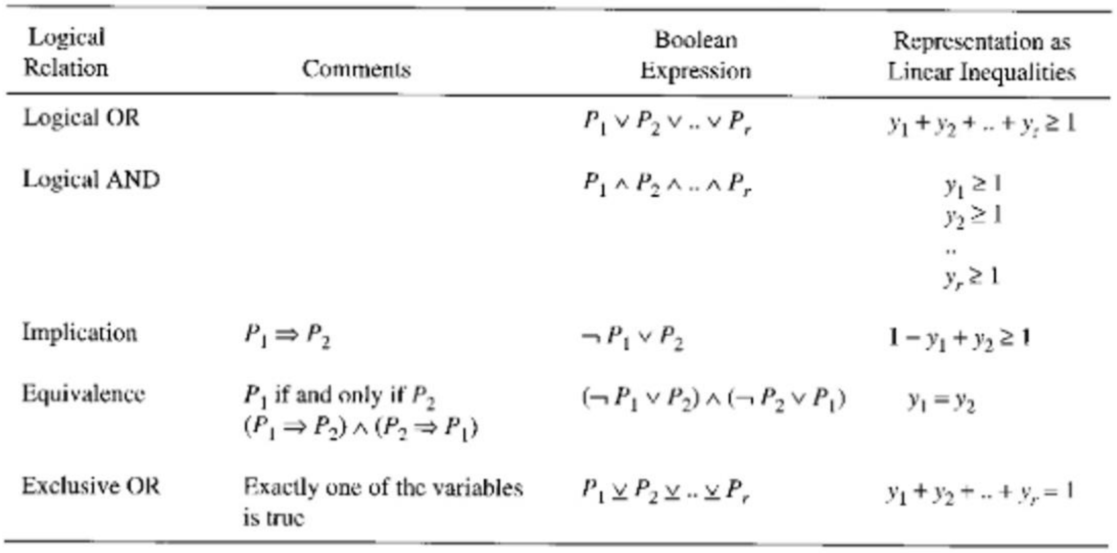
- In some instances you may face the modeling of situations where you want two or more statements to be satisfied, for example, you may want to develop a model in which the following logic is embedded: "If the adsorber is selected or the membrane is selected, then do not use cryogenic separation". If that is the case then we can introduce the following two constraints to enforce this statement:

$$y_A + y_{CS} \leq 1$$
$$y_M + y_{CS} \leq 1$$

Logic and the formulation of constraints

The fundamental question that we face now is how we can derive the constraints that we need in a systematic way. Toward this end, we can think of statement of interest as logical propositions, and we can assign binary variables y_i to each of these propositions. For each statement P_i, we assign a binary y_i. For example, to the statement P_i "select reactor i" we can assign binary y_i. If the statement is true (that is, if the reactor is selected), then $y_i = 1$, otherwise $y_i = 0$. Note that, if a statement P_i is true, then its negation $not\ P_i$ is false. This is captured by noting that if $\neg P_i \rightarrow (1 - y_i)$.

The following table shows the relation between logical statements and the equivalent linear inequalities that can be written to represent each case. The fundamental insight is that the inequalities are satisfied only if the desired proposition is true.



Logical Relation	Comments	Boolean Expression	Representation as Linear Inequalities
Logical OR		$P_1 \vee P_2 \vee .. \vee P_r$	$y_1 + y_2 + .. + y_r \geq 1$
Logical AND		$P_1 \wedge P_2 \wedge .. \wedge P_r$	$y_1 \geq 1$ $y_2 \geq 1$ .. $y_r \geq 1$
Implication	$P_1 \Rightarrow P_2$	$\neg P_1 \vee P_2$	$1 - y_1 + y_2 \geq 1$
Equivalence	P_1 if and only if P_2 $(P_1 \Rightarrow P_2) \wedge (P_2 \Rightarrow P_1)$	$(\neg P_1 \vee P_2) \wedge (\neg P_2 \vee P_1)$	$y_1 = y_2$
Exclusive OR	Exactly one of the variables is true	$P_1 \veebar P_2 \veebar .. \veebar P_r$	$y_1 + y_2 + .. + y_r = 1$

In the case of the OR statement, for example, we see that for it to be true, it is enough that only one of the statements is true (see truth table below). As a practical example, let's say that P1=choose reactor 1, and P2= choose reactor 2. Then the statement choose reactor 1 or reactor 2 can be represented as the following disjunction $P_1 \vee P_2$, this disjunction is true in three cases: if reactor 1 is selected, if reactor 2 is selected or if both reactors are selected. This fact is well represented by the results of the inequality, note that in the three first cases the inequality $(Y_1 + Y_2 \geq 1)$ is satisfied, while in the last one it is not, consequently, one may use this inequality into a mathematical programming formulation to enforce an OR logic, we will see later how to tie this logic statement with physical variables that ensure for example that no mass flows through equipment that is not selected.

P_1	P_2	$P_1 \vee P_2$	$Y_1 + Y_2$
TRUE	TRUE	TRUE	2
TRUE	FALSE	TRUE	1
FALSE	TRUE	TRUE	1
FALSE	FALSE	FALSE	0

In the case of the AND condition, the logic imposes that all relevant statements should be satisfied simultaneously (see truth table below). Following the same example as before, we can formulate a stamen that says choose reactor 1 AND reactor 2, this statement can be represented as the following conjunction $P_1 \wedge P_2$. This conjunction is only true if both reactor one and reactor 2 are selected. One can represent this proposition using two inequalities ($y_1 \geq 1$ and $y_2 \geq 1$), note that these inequalities are able to capture the required logic, the proposition is true only in the first case where both Y_1 and Y_2 are greater than 1.

P_1	P_2	$P_1 \wedge P_2$	Y_1	Y_2
TRUE	TRUE	TRUE	1	1
TRUE	FALSE	FALSE	1	0
FALSE	TRUE	FALSE	0	1
FALSE	FALSE	FALSE	0	0

The implication is the last logical operation that I want to discuss, rhetorically it is somehow simple to understand, if statement P_1 is true, then statement P_2 should also be true. Consider this example (completely out of the realm of chemE but still helpful to understand the meaning of the implication): $P_1 = I\ bite\ my\ tongue, P_2 = I\ cry$. Note that the implication $P_1 \rightarrow P_2$ makes sense in the following cases: (1) if $P_1$is true and P_2 is true, meaning that I bite my tongue and therefore I cry, this is absolutely the case (2) if $P_1$is false and P_2 is true, meaning that I did not bite my tongue but I am still crying, this is possible, I may be crying for some other reason (3) if P_1 is false and P_2 is also false, meaning that I did not bite my tongue, and I am not crying, this is also possible, this usually happens on a good day. The only case where the implication is false (meaning that the statement does not make sense) is if $P_1$is true and P_2 is false, meaning that I bite my tongue and I am not crying, this will not happen. We can capture this logic with the Boolean statement $\neg P_1 \vee P_2$ which translates into the inequality $(1 - y_1) + y_2 \geq 1 \rightarrow y_2 - y_1 \geq 0$. We can see this statement in the truth table shown below

P_1	P_2	$P_1 \rightarrow P_2$	$\neg P_1$	P_2	$\neg P_1 \vee P_2$	$y_2 - y_1$
TRUE	TRUE	TRUE	FALSE	TRUE	TRUE	0
TRUE	FALSE	FALSE	FALSE	FALSE	FALSE	-1
FALSE	TRUE	TRUE	TRUE	TRUE	TRUE	1
FALSE	FALSE	TRUE	TRUE	FALSE	TRUE	0

Transforming complex logical statements into inequalities

In many instances, more complex logical statements may be used to describe the relations between decisions in design problems. In these cases, it is still possible to derive representative logical equations. The procedure that we will describe makes use of two notions

1. We use the fact that a collection of propositions separated by an inclusive OR is a clause $Q_i = p_1 \vee p_2 \ldots \vee p_r$
2. A collection of clauses separated by AND statements constitute a conjunctive normal form:
$$Q_1 \wedge Q_2 \wedge \ldots \wedge Q_S$$
$$(P_{1,1} \vee P_{1,2} \vee \ldots \vee P_{1,r1}) \wedge (P_{2,1} \vee P_{2,2} \vee \ldots \vee P_{2,r2}) \wedge \ldots \wedge (P_{s,1} \vee P_{s,2} \vee \ldots \vee P_{s,rs})$$

The fundamental intuition is that this conjunctive form can be translated into a collection of inequalities, as follows (note that each clause is a disjunction, and we have a way to transform disjunctions into equations):

$$Y_{1,1} + Y_{1,2} + \cdots + Y_{1,r1} \geq 1$$

$$Y_{2,1} + Y_{2,2} + \cdots + Y_{2,r1} \geq 1$$
$$\vdots$$
$$Y_{s,1} + Y_{s,2} + \cdots + Y_{s,r1} \geq 1$$

The question is then how can we obtain a conjunctive normal form given a specific logic statement. To achive this we can use the following procedure which consists in applying the following three steps:

- Replace the implication by its equivalent disjunction

$$P_1 \rightarrow P_2 \Leftrightarrow \neg P_1 \vee P_2$$

- Move the negation inward by applying DeMorgan's theorem

$$\neg\,(P_1 \wedge P_2) \Leftrightarrow \neg\, P_1 \vee \neg P_2$$
$$\neg(P_1 \vee P_2) \Leftrightarrow \neg\, P_1 \wedge \neg P_2$$

- Recursively distribute the "OR" over the "AND" by using the following equivalence

$$(P_1 \wedge P_2) \vee P_3 \Leftrightarrow (P_1 \vee P_3) \wedge (P_2 \vee P_3)$$

Example: to illustrate the use of the previous procedure, let's consider the following statement from a design problem "if you select the flash unit, then select distillation and not the membrane", the logic associated with this statement is as follows:

$$P_{Flas} \rightarrow (P_{distillation} \wedge \neg P_{membrane})$$

The first step is to remove the implication by expressing it in terms of its equivalent disjunction, as follows:

$$\neg P_{Flas} \vee (P_{distillation} \wedge \neg P_{membrane})$$

The second step is to apply De'Morgan's theorem, but in this case is not necessary.

The final step consists in distributing the OR over the AND, as follows:

$$(\neg P_{Flas} \vee P_{distillation}) \wedge (\neg P_{Flash} \vee \neg P_{membrane})$$

This is a conjunctive form, with two clauses, that can be represented as two linear inequalities, as follows:

$$1 - y_{flash} + y_{distillation} \geq 1$$
$$1 - y_{flash} + 1 - y_{membrane} \geq 1$$

Example: let's consider another example: "if the absorber to recover the product is selected, or the membrane separator is selected, then do not use cryogenic separation." This can be stated as follows:

$$P_A \vee P_M \rightarrow \neg P_{CS}$$

First, we remove the implication as follows

$$\neg(P_A \vee P_M) \vee \neg P_{CS}$$

Second, we use De'Morgan's theorem:

$$(\neg P_A \wedge \neg P_M) \vee \neg P_{CS}$$

Finally, we apply the OR over the AND:

$$(\neg P_A \vee \neg P_{CS}) \wedge (\neg P_M \vee \neg P_{CS})$$

This translates into two inequalities

$$1 - y_A + 1 - y_{CS} \geq 1$$

$$1 - y_M + 1 - y_{CS} \geq 1$$

Activation of variables and constraints

Selection of units in the design of chemical processes (activation of variables): Binary variables can be used to activate or deactivate the continuous variables. For instance, if a process unit does not exists (i.e., $y_i = 0$) then the inlet flow rate to the unit *i* (F_i) should be zero, while if the process unit *i* exists (i.e., $y_i = 1$) then the inlet flow rate should be within given lower and upper bounds. This can be expressed as:

$$F_i^L y_i \leq F_i \leq F_i^U y_i$$

When several input streams with flow rates F_j are directed toward the same process unit, then there are two alternatives to express the inactivation constraint:

Alternative 1:

$$\sum_{j=1}^{m} F_j - U y_i \leq 0$$

$$F_j \geq 0$$

Alternative 2:

$$F_j - U y_i \leq 0,\ j = 1, \dots, m$$

$$F_j \geq 0$$

Note that these equations provide a direct link between the physics of the problem and the logic statements previously studied.

Activation of constraints: Let's consider a case such that if a unit exists one equality h(x) and one inequality g(x) must hold true. This situation can be modelled by introducing positive slack variables for the equality and inequality constraint such that the model is written as follows:

$$h(x) + s_1^+ - s_1^- = 0$$

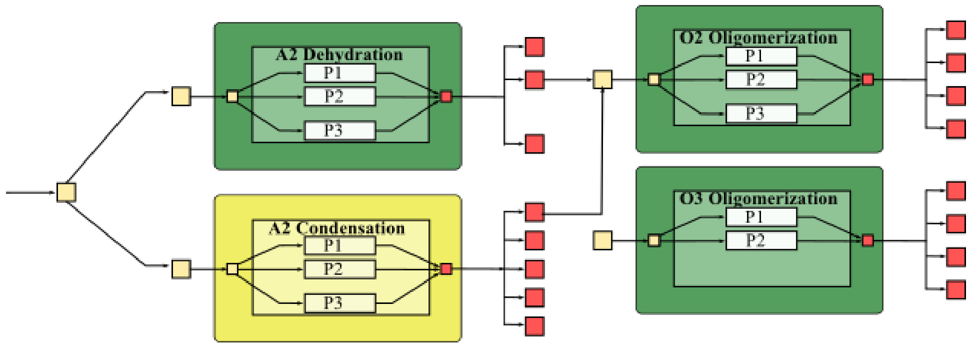
$$g(x) \leq s_2$$

$$s_1^+ + s_1^- \leq U_1(1 - y_i)$$

$$s_2 \leq U_2(1 - y_i)$$

$$s_1^+, s_1^-, s_2 \geq 0$$

Example: Can you write the logical and activation constraints associated with the following superstructure.



Modeling disjunctions

In a number of cases, we will have to enforce logic conditions that involve continuous variables. Those cases are known as disjunctions. For example, one may be interested in enforcing the following condition in a design problem:

If reactor 1 is selected, then pressure P must be between 5 to 10 atm
If reactor 2 is selected, then pressure P must be between 20 to 30 atm

This condition can be represented as a linear disjunction of the following form:

$$\vee_{i\in D}\,[A_i x \le b_i] = [A_1 x \le b_1] \vee [A_2 x \le b_2] \vee \ldots \vee \left[A_{|D|} x \le b_{|D|}\right]$$

If we apply the previous definition to the example, we obtain the following representation:

$$\begin{bmatrix} P \le 10 \\ -P \le -5 \end{bmatrix} \vee \begin{bmatrix} P \le 30 \\ -P \le -20 \end{bmatrix}$$

Where the first term is related to reactor 1, and the second term is related to reactor 2. This disjunction, can be transformed into a set of mixed integer constraints using two approaches:

1. The big M reformulation: In general, this reformulation can be presented as follows

$$\vee_{i\in D}\,[A_i x \le b_i]$$
$$A_i x \le b_i + M_i(1-y_i), i \in D$$
$$\sum_{i\in D} y_i = 1$$

Here M_i is an arbitrarily large value that renders the inequality redundant if $y_i = 0$. However, one must be careful, since the larger the value of M_i, the larger the feasible space, and therefore the computational time needed to solve the model.

For the example that we are dealing with, this reformulation looks as follows:

$$\begin{bmatrix} P \le 10 \\ -P \le -5 \end{bmatrix} \vee \begin{bmatrix} P \le 30 \\ -P \le -20 \end{bmatrix}$$
$$P \le 10 + M_1(1-y_1)$$
$$-P \le -5 + M_1(1-y_1)$$
$$P \le 30 + M_2(1-y_2)$$
$$-P \le -20 + M_2(1-y_2)$$
$$y_1 + y_2 = 1$$

2. The convex hull reformulation: this reformulation is useful because it avoids the use of the big M parameter. This idea was introduced by Egon Balas, a major figure in integer optimization during the 20th century and a professor at Carneggie Mellon University. In this approach we have that:

$$x = \sum_{i \in D} \tilde{x_i}$$
$$A_i \tilde{x_i} \leq b_i y_i, \forall i \in D$$
$$\sum_{i \in D} y_i = 1$$
$$0 \leq \tilde{x_i} \leq U y_i, \forall i \in D$$

The first of these constraints is rewriting the original variable in terms of disaggregated variables (one per each element of the disjunction). The second equation, consist in writing the inequalities in terms of the disaggregated variables and a binary variable y_i associated with each disjunction. The third constraint is included to indicate that only one of the disjunctions should be selected. The fourth and final constraint is optional, and it is included only if $y_i = 0$ in the second inequality does not imply $\tilde{x_i} = 0$.

For the example that we are dealing with, this approach looks like follows:

$$\begin{bmatrix} P \leq 10 \\ -P \leq -5 \end{bmatrix} \vee \begin{bmatrix} P \leq 30 \\ -P \leq -20 \end{bmatrix}$$
$$P = \tilde{P}_1 + \tilde{P}_2$$
$$\tilde{P}_1 \leq 10 y_1$$
$$-\tilde{P}_1 \leq -5 y_1$$
$$\tilde{P}_2 \leq 30 y_2$$
$$\tilde{P}_2 \leq -20 y_2$$
$$y_1 + y_2 = 1$$

Example: Consider the following disjunction

$$[x_1 - x_2 \leq -1] \vee [-x_1 + x_2 \leq -1]$$
$$x_1, x_2 \geq 0, x_1, x_2 \leq 4$$

Using the big M reformulation:

$$x_1 - x_2 \leq 1 + M(1 - y_1)$$
$$-x_1 + x_2 \leq 1 + M(1 - y_2)$$
$$y_1 + y_2 = 1$$
$$x_1, x_2 \geq 0; x_1, x_2 \leq 4$$

Using the convex hull reformulation

$$x_1 = z_1 + z_2$$
$$x_2 = w_1 + w_2$$
$$z_1 - w_1 \leq -y_1$$
$$-z_2 + w_2 \leq -y_2$$

$$y_1 + y_2 = 1$$

$$0 \leq z_1 \leq 4y_1$$
$$0 \leq z_2 \leq 4y_2$$
$$0 \leq w_1 \leq 4y_1$$
$$0 \leq w_2 \leq 4y_2$$

Generalized disjunctive programming

The idea of GDP is to formulate optimization problems in terms of constraints in algebraic form and in the form of disjunctions and logic propositions.

“The formulation of problems in this way can help to facilitate the formulation of a models because disjunctions and logic provide a higher level representation that does not require the use to pose the problem in terms of only algebraic equations as in MINLP” Grossmann

Written in this language GDP look as follows:

$$\min\left(Z = \sum_k c_k + f(x)\right)$$

$$r(x) \leq 0$$

$$\bigvee_{j\in K}\begin{bmatrix} Y_{jk} \\ g_{jk}(x) \leq 0 \\ c_k = \gamma_{jk} \end{bmatrix}, k \in K$$

$$\underline{\bigvee} Y_{j,k}$$

$$\Omega(Y) = true$$

$$0 \leq x \leq U$$

This equation can be translated into an MINLP using either the big M reformulation or the convex hull reformulation, in the first case we have:

$$\min\left(\sum_{k\in K}\sum_{j\in J_K} \gamma_k y_{jk} + f(x)\right)$$

$$r(x) \leq 0$$

$$g_{jk}(x) \leq M_j\left(1 - y_{j,k}\right), \forall j \in J_k, k \in K$$

$$Ay \leq a$$

$$\sum_{j\in J_K} y_{j,k} = 1$$

A few archetypical examples

Assignment problem: Given a set of n jobs i and a set of n machines j. Also, given the cost $c_{i,j}$ for assigning job I to machine j. Find the minimum cost assignment of the jobs to the machines. Assume that only one job per machine is possible and a job is assigned to only one machine.

$$\min z = \sum_{i \in I} \sum_{j \in J} c_{ij} y_{i,j}$$

$$\sum_{j=1}^{n} y_{i,j} = 1,\ i = 1, \dots, n$$

$$\sum_{i=1}^{n} y_{i,j} = 1,\ j = 1, \dots, n$$

$$y_{i,j} \in \{0,1\}$$

Knapsack problem: Given a set of n objects ($i \in I$) of given weight w_i and price p_i. Given is also a knapsack that can hold a maximum weight W of objects. Select those objects that can be accommodated in the knapsack and that maximize its value.

$$\max z = \sum_{i \in I} p_i y_i$$

$$\sum_{j=1}^{n} w_i y_{i,j} = \sum_{i \in I} w_i y_i \leq W$$

$$y_{i,j} \in \{0,1\}$$

Set covering problem (from Bierlaire formulation): After the World cup, Camille wants to complete her collection of stickers representing the players of each competing team. She can buy entire collections of stickers from schoolmates. In each collection there are stickers she needs and stickers she does not need. Camille must decide which stickers to purchase, in order to complete her own album, at minimum price.

$$\min z = \sum_{c \in Collections} c_c y_c$$

$$\sum_{c=1}^{n} a_{c,s} y_c \geq 1,\ s \in Stickers$$

Note that $a_{c,s}$ is a binary parameter that is equal to 1 if a sticker is present in a collection. The second constraint enforces the fact that each sticker must be in at least one collection.

Plant assignment problem: Consider *n* potential plants *i* that are to serve *m* customers *j* with demand d_j. Each plant has a fixed cost f_i and a maximum capacity U_i. Also, the combined transportation and manufacturing cost from a plant *i* to customer *j* is given by the cost c_{ij}. The problem consists in selecting the plants that serve the customers to satisfy the demands at minimum cost.

$$z = \sum_{i=1}^{n} f_i y_i + \sum_{i=1}^{n}\sum_{j=1}^{m} c_{i,j} x_{i,j}$$

$$s.t.\quad \begin{array}{l} \sum_{i=1}^{n} x_{ij} = d_j, \qquad \forall\, j = 1, \dots, m \\ \sum_{j=1}^{m} x_{ij} \leq U_i y_i, \qquad \forall\, i = 1, \dots, n \\ x_{i,j} \geq 0, \forall i = 1, \dots, n, j = 1, \dots, m \end{array}$$

In this problem we define binary y_i if plant i is selected and $x_{i,j}$ is the amount of material sent from plant i to customer j, finally, U_i represents the maximum capacity of plant i.