Modeling linear integer optimization problems
An introduction

Karen Aardal and Theresia van Essen

April 2020

Optimization models are systems of equations that describe a given problem. The decisions that need to be made are represented by *decision variables* which can take continuous, integer or binary values. The performance of a decision is given by a function in the decision variables, also referred to as *objective function*. The restrictions on the values that can be assigned to these decision variables can be expressed by inequalities or equations, and are referred to as *constraints*. The constants in the constraints and objective function are input and are often called *parameters*. Then, the optimization model represents the problem of finding the values of the decision variables that fulfill all constraints and that minimizes or maximizes the objective function.

In optimization, developing and analyzing models are key activities. Designing a model is a skill and requires a lot of practice before one can "see" what a natural model is. There are, however, some basic techniques one can learn in order to make the process easier. It is very important to be careful in all the steps and to use good notation.

In this short note, we go through some techniques and give some general hints with the aim of making it more easy and fun to design good models. We focus on linear models. The note consists of two sections. Section one concerns linear and linear integer models. The second section treats mixed-integer 0-1 models, where a subset of the variables are restricted to take the values 0 or 1.

1 Linear and linear integer optimization models

1.1 Introduction

We start with a description of a simple so-called product-mix problem [2].

Example 1. Giapetto's Woodcarving Inc., manufactures two types of wooden toys: soldiers and trains. A soldier sells for \$27 and uses \$10 worth of raw materials. Each soldier that is manufactured increases Giapetto's variable labor and overhead costs by \$14. A train sells for \$21 and uses \$9 worth of raw materials. Each train built increases Giapetto's variable labor and overhead costs by \$10. The manufacture of wooden soldiers and trains require two types of skilled labor: carpentry and finishing. A soldier requires 2 hours of finishing and 1 hour of carpentry labor while a train requires 1 hour of finishing and 1 hour of carpentry labor. Each week Giapetto can obtain all the needed raw material

but only 100 finishing hours and 80 carpentry hours. Demand for trains is unlimited, but at most 40 soldiers are bought each week. Giapetto wants to maximize the weekly profit (revenues − costs). Formulate a linear optimization model of Giapetto's situation that can be used to maximize Giapetto's weekly profit.

When designing a model, the first, very important, step is to separate input from the decision variables. The input represents what is known, whereas the variables represent the unknown. The variables are assigned values as a result of applying an algorithm to the model. In order to increase readability and clarity, it is important to use good notation. For very uncomplicated models, it can be acceptable to use letters such as x to represent variables, but the more complicated a model gets, the more useful it is to give names that provide a stronger association to the meaning of the variables. The same is true for the input.

We formulate this problem quite generally and start by defining the input. We use the index j to represent the products; $j = 1$ represents soldiers and $j = 2$ represents trains. Once we have looked at the general formulation, we write down the model for the specific input of Giapetto's.

Input. Let

$$
\begin{aligned}
r_j &= \text{the revenue per sold item of product } j,\ j = 1, 2,\\
m_j &= \text{the per unit costs of raw material for producing product } j,\ j = 1, 2,\\
l_j &= \text{the per unit costs of labor and overhead for producing product } j,\ j = 1, 2,\\
f_j &= \text{the number of hours of finishing for one unit of product } j,\ j = 1, 2,\\
c_j &= \text{the number of hours of carpentry for one unit of product } j,\ j = 1, 2,\\
F &= \text{the total number of finishing hours available per week,}\\
C &= \text{the total number of carpentry hours available per week,}\\
D_j &= \text{maximum demand per week for product } j,\ j = 1, 2.
\end{aligned}
$$

Next, we define the decision variables. The choice of decision variables for such a simple model is straightforward, but for more complicated models it could be quite complicated and clearly not unique. In order to come up with a good variable definition, it is useful to think in terms of what kind of answer we would like to get from the model. In this case, we would like to know how many soldiers and how many trains we would like to produce per week, and this is therefore the very natural choice.

Variable definition. Let

x_j = the number of product j produced per week, $j = 1, 2$.

Since we can only produce an integer number of trains and soldiers, we restrict the variables to take integer values. We are now ready to write down our optimization model:

$$
\begin{array}{lll}
\text{maximize} & z(x) = (r_1 - m_1 - l_1)x_1 + (r_2 - m_2 - l_2)x_2 & \\
\text{subject to} & f_1x_1 + f_2x_2 \;\leq\; F & \\
& c_1x_1 + c_2x_2 \;\leq\; C & \\
& x_j \;\leq\; D_j, \quad j = 1,2 & \\
& x_j \;\geq\; 0, \quad j = 1,2 & \\
& x_j \;\in\; \mathbb{Z}, \quad j = 1,2\,.
\end{array}
$$

We now go through all the elements of the model. The first expression is what we call the *objective function*:

$$\text{maximize } z(x) = (r_1 - m_1 - l_1)x_1 + (r_2 - m_2 - l_2)x_2.$$

We want to maximize the weekly profit for Giapetto's. The coefficient for variable x_1, which is the number of produced soldiers per week, is $(r_1 - m_1 - l_1)$, which is the revenue − the material cost − the labor cost per unit. Similarly for the production of trains. Instead of writing "maximize" (or "minimize") we typically write the abbreviations "max" (or "min"). Also, instead of writing $z(x)$ as the notation for the objective function, we typically just use z.

Next, we have three *constraints* that limit the possible values x_1 and x_2 can take. Sometimes we distinguish between the *structural constraints*, which in this case are the constraints

$$
\begin{array}{rcl}
f_1x_1 + f_2x_2 & \leq & F \\
c_1x_1 + c_2x_2 & \leq & C \\
x_j & \leq & D_j, \quad j = 1,2,
\end{array}
$$

the *nonnegativity constraints* $x_j \geq 0,\ j = 1,2$, and the *integrality constraints* $x_j \in \mathbb{Z},\ j = 1,2$, but this distinction is not really essential. The first structural constraint models the limitation we have in the number of available finishing hours, and the second constraint models the limitation we have in the number of available carpentry hours. The last set of structural constraints sets a limit on the number of sold items. Instead of writing out "subject to" we use the abbreviation "s.t.".

We can write this model even more formally, such that it is valid for n products. In this more general case, our product mix model looks as follows:

$$
\begin{array}{lll}
\max & z = \sum_{j=1}^{n}(r_j - m_j - l_j)x_j & \\
\text{s.t.} & \sum_{j=1}^{n} f_jx_j \;\leq\; F & \\
& \sum_{j=1}^{n} c_jx_j \;\leq\; C & \\
& x_j \;\leq\; D_j, \quad j = 1,\ldots,n & \\
& x_j \;\geq\; 0, \quad j = 1,\ldots,n. & \\
& x_j \;\in\; \mathbb{Z}, \quad j = 1,\ldots,n\,.
\end{array}
$$

You may think that it is a bit "overkill" to write this model so formally since we only have two variables here, but in general it is actually quite useful to have such a formal description of the model since we then clarify that the model in fact can be used for any number n of products.

Finally, let us actually write down the model for the specific case of Giapetto's with the relevant values of the input. Notice that $(r_1 - m_1 - l_1) = 27 - 10 - 14 = 3$, $(r_2 - m_2 - l_2) = 21 - 9 - 10 = 2$, and $D_2 = \infty$, which yields:

$$
\begin{array}{lrcrcrl}
\max & z = 3x_1 & + & 2x_2 & & & \\
\text{s.t.} & 2x_1 & + & x_2 & \leq & 100 & \\
& x_1 & + & x_2 & \leq & 80 & \\
& x_1 & & & \leq & 40 & \\
& x_1, & & x_2 & \geq & 0, & j = 1, 2 \\
& x_1, & & x_2 & \in & \mathbb{Z}, & j = 1, 2\,.
\end{array}
$$

1.2 Using variables with multiple indices

The product-mix problem we saw in the previous subsection was quite straightforward to model. Let us look at a slightly more challenging problem [2].

Example 2. Powerco has three power plants that supply the need of four cities. Each power plant can supply the following numbers of kilowatt hours (kWh) of electricity: plant 1: 40 million; plant 2: 50 million; plant 3: 40 million. The peak power demands in these cities, which occur at the same time (2 pm), are as follows (in kWh): city 1: 45 million; city 2: 20 million; city 3: 30 milion; city 4: 30 million. The costs of sending 1 million kWh of electricity from plant to city depend on the distance the electricity must travel. These costs, and the other input as well, are given in Table 1.

Formulate the problem of minimizing the total cost while meeting each city's peak power demand as a linear optimization problem.

From	City 1	City 2	City 3	City 4	Supply million kWh
Plant 1	\$ 8	\$ 6	\$ 10	\$ 9	40
Plant 2	\$ 9	\$ 12	\$ 13	\$ 7	50
Plant 3	\$ 14	\$ 9	\$ 16	\$ 5	40
Demand milion kWh	45	20	30	30	

Table 1: Input for the Powerco problem

Again, we start by introducing the notation for the input and the decision variables. In our previous example, we used one index for each product. Here, we could introduce one index j for each combination of plant and city obtaining $j = 1, \ldots, 12$. This, however, makes the model non-transparent to read, and when we interpret the outcome we have to translate the index back to the particular combination. This might be acceptable for 12 combinations, but imagine that we have 50 plants and 500 cities. What we typically

do in a case like this, is to introduce a double-index notation ij, where i represents a plant, and j a city.

Input. Let

c_{ij} = the cost of transporting one million kWh from plant i to city j, $i = 1, \ldots, 3,\ j = 1, \ldots, 4$

s_i = supply, in million kWh, at plant $i,\ i = 1, \ldots, 3$

d_j = peak demand, in million kWh, in city $j,\ j = 1, \ldots, 4$.

Variable definition. Let

x_{ij} = the amount of electricity, in million kWh, transported from plant i to city j, $i = 1, \ldots, 3,\ j = 1, \ldots, 4$.

Before formulating the problem, we illustrate the problem by a graph. This often helps in the formulation, not only in this case, but in most cases where some underlying graph structure is present. Assume that we have m plants and n cities.

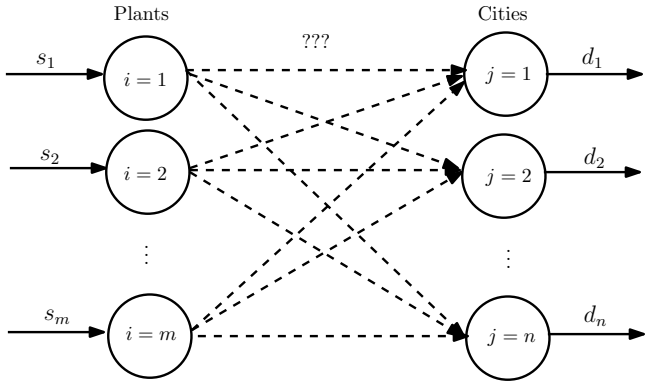


Figure 1: Network structure of Example 2.

We see from this network that the solid arrows represent input and the dashed lines represent what we would like to know: how much should be transported between the plants and the cities? This precisely corresponds to our choice of decision variables x_{ij}.

We now first formulate the problem in its general form, and then with the specific input for this instance.

$$
\begin{array}{lrcl}
\min & z = \sum_{i=1}^{m} \sum_{j=n}^{n} c_{ij} x_{ij} & & \\
\text{s.t.} & \sum_{j=1}^{n} x_{ij} & \leq & s_i, \quad i = 1, \ldots, m \\
& \sum_{i=1}^{m} x_{ij} & = & d_j, \quad j = 1, \ldots, n \\
& x_{ij} & \geq & 0, \quad i = 1, \ldots, m,\ j = 1, \ldots, n.
\end{array}
$$

Notice that for the demand constraints $\sum_{i=1}^{m} x_{ij} = d_j,\ j = 1, \ldots, n$, we could also have chosen to use the $\geq$-sign since all costs are positive, but for clarity we choose equality

here. Next, we give the formulation for the specific instance:

$$
\begin{array}{llrrrrrrrrrrrrcr}
\min & z = & 8x_{11}+ & 6x_{12}+ & 10x_{13}+ & 9x_{14}+ & 9x_{21}+ & 12x_{22}+ & 13x_{23}+ & 7x_{24}+ & 14x_{31}+ & 9x_{32}+ & 16x_{33}+ & 5x_{34} & & \\
\text{s.t.} & & x_{11}+ & x_{12}+ & x_{13}+ & x_{14} & & & & & & & & & \leq & 40 \\
 & & & & & & x_{21}+ & x_{22}+ & x_{23}+ & x_{24} & & & & & \leq & 50 \\
 & & & & & & & & & & x_{31}+ & x_{32}+ & x_{33}+ & x_{34} & \leq & 40 \\
 & & x_{11}+ & & & & x_{21}+ & & & & x_{31} & & & & = & 45 \\
 & & & x_{12}+ & & & & x_{22}+ & & & & x_{32} & & & = & 20 \\
 & & & & x_{13}+ & & & & x_{23}+ & & & & x_{33} & & = & 30 \\
 & & & & & x_{14}+ & & & & x_{24}+ & & & & x_{34} & = & 30 \\
 & & x_{11}, & x_{12}, & x_{13}, & x_{14}, & x_{21}, & x_{22}, & x_{23}, & x_{24}, & x_{31}, & x_{32}, & x_{33}, & x_{34}, & \geq & 0.
\end{array}
$$

If we solve this problem, we obtain the following solution x^*. (We typically denote the optimal solution by x^* and the optimal objective value by z^*.) The objective value of this solution is $z^* = 1,005 \times 10^6$:

$$
\begin{array}{rcrrcrrcrrcr}
x_{11}^* & = & 0 & x_{12}^* & = & 15 & x_{13}^* & = & 25 & x_{14}^* & = & 0 \\
x_{21}^* & = & 45 & x_{22}^* & = & 0 & x_{23}^* & = & 5 & x_{24}^* & = & 0 \\
x_{31}^* & = & 0 & x_{32}^* & = & 5 & x_{33}^* & = & 0 & x_{34}^* & = & 30
\end{array}
$$

Due to the choice of indices it is very easy to interpret the solution in terms of the graph from Figure 1, see Figure 2.

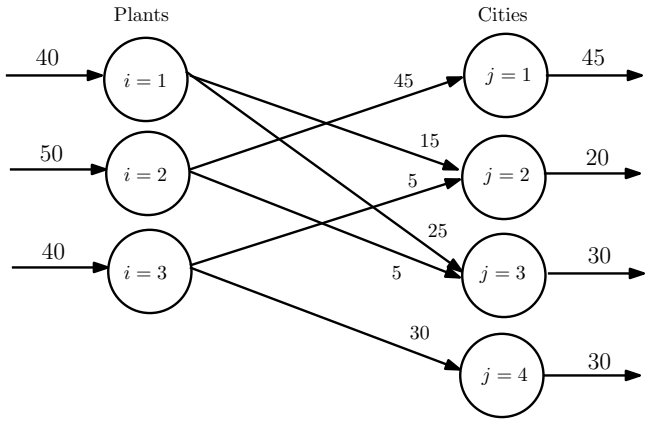


Figure 2: Solution of Example 2.

Here, we have seen an example of double-index variables x_{ij}. We can also of course have three or more indices, depending on the problem, for instance x_{ijt} = the number of products transported from i to j in time period t.

1.3 Modeling problems that have a sequential aspect

Many problems have a natural time sequence embedded in its structure. At the start, such problems might seem difficult to model, but again, it may help to use a network to illustrate the interaction between the time periods. We give an example below.

Example 3. An event accessories company delivers napkins for a conference that lasts for five days. The demand for napkins for the five days is 900, 1440, 1260, 1620, 1296, respectively. For this event, the napkins are delivered with a specific print, which renders the napkins worthless once the conference is over. The price for one napkin is \$2.50. The company has its own cleaning facility, where three different cleaning programs are used:

1. one-hour cleaning: the napkins are ready to use the next day,
2. fast cleaning: the napkins can be used again after two days,
3. standard cleaning: the napkins can be used again after three days.

The per unit costs for cleaning by the three different programs are \$0.50, \$0.25, and \$0.10, respectively. All the cleaning programs cause some discoloration of the special print on the napkins with the consequence that 40%, 20%, and 10%, respectively, of the napkins have to be discarded. The cleaning still has to be paid.

Model the problem of deciding how many new napkins should be bought, and how many napkins should be cleaned in the various cleaning programs each day, as a linear optimization problem.

Input. Let

c^n = the per unit cost of buying a new napkin,
c^o = the per unit cost of the one-hour cleaning,
c^f = the per unit cost of the fast cleaning,
c^s = the per unit cost of the standard cleaning,
d_t = demand for napkins on day t.

In order to decide on a good set of decision variables, we have to give it some thought. It is clear that we need to decide on how many new napkins we should buy and the number of napkins to be cleaned in the various cleaning programs each day:

Variable definition, part 1. Let

x_t^n = the number of new napkins bought for use on day t,
x_t^o = the number of napkins cleaned by one-hour cleaning at the end of day t,
x_t^f = the number of napkins cleaned by fast cleaning at the end of day t,
x_t^s = the number of napkins cleaned by standard cleaning at the end of day t.

Moreover, it is clear that we can only buy and clean an integer number of napkins. So, the variables defined above should be restricted to take integer values only. What makes the modeling a bit more tricky than we have seen so far, is that a certain number of napkins are destroyed in each of the cleaning processes. The number of napkins that are lost, and the number of napkins that "survive" the cleaning should also be integer. Since the number sent to cleaning is restricted to be integer, it is sufficient to require either the number of lost napkins or the number of surviving napkins to be integer. Here, we have decided on the number of lost napkins, but this choice is arbitrary. Either the surviving or the lost napkins therefore need to be introduced as decision variables that

will be restricted to take integer values. Since the number of lost and clean napkins are required to be integer, it might happen that we are forced to clean extra napkins just to achieve the integrality. We therefore allow for napkins to be wasted each day. We assume that we do not store wasted napkins to wash them at a later stage.

Variable definition, part 2. Let

l_t^o = the number of napkins destroyed in one-hour cleaning at the end of day t,
l_t^f = the number of napkins destroyed in the fast cleaning at the end of day t,
l_t^s = the number of napkins destroyed in the standard cleaning at the end of day t,
w_t = number of napkins thrown away after use on day t.

Before presenting the optimization model, we again illustrate the problem by a graph. The top node represents the buying of new napkins and has arcs to the nodes representing the five days. In these nodes, we see the number of the day $(1, \ldots, 5)$ and the number of napkins required for that day.

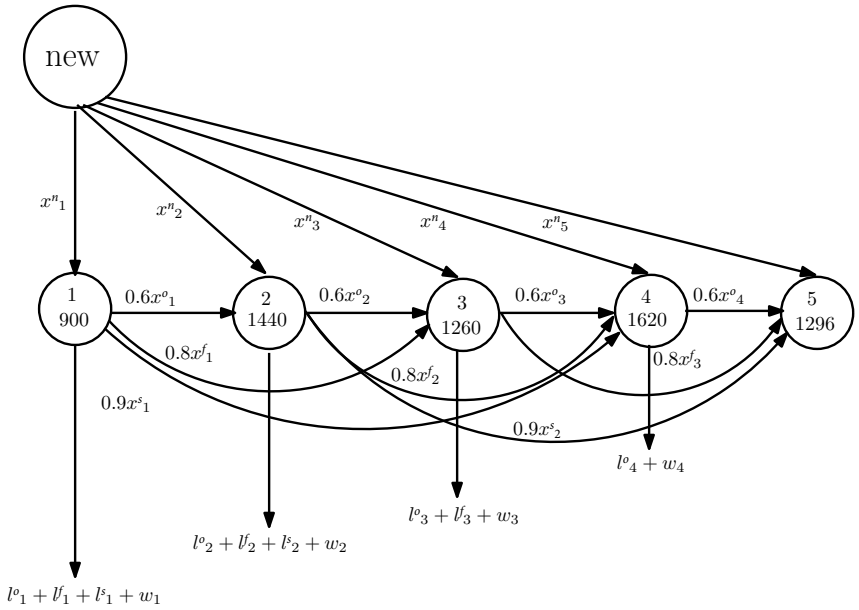


Figure 3: Graph representation of Example 3.

We now give the model for the general case of T days:

$$\min z = c^n \sum_{t=1}^{T} x_t^n + c^o \sum_{t=1}^{T-1} x_t^o + c^f \sum_{t=1}^{T-2} x_t^f + c^s \sum_{t=1}^{T-3} x_t^s$$

$$\begin{aligned}
\text{s.t. } l_t^o - 0.4x_t^o &= 0,\ t = 1, \ldots, T-1 \\
l_t^f - 0.2x_t^f &= 0,\ t = 1, \ldots, T-2 \\
l_t^s - 0.1x_t^s &= 0,\ t = 1, \ldots, T-3 \\
x_1^n &\geq d_1 \\
x_2^n + 0.6x_1^o &\geq d_2 \\
x_3^n + 0.6x_2^o + 0.8x_1^f &\geq d_3 \\
x_t^n + 0.6x_{t-1}^o + 0.8x_{t-2}^f + 0.9x_{t-3}^s &\geq d_t,\ t = 4, \ldots, T \\
x_1^n - x_1^o - x_1^f - x_1^s - w_1 &= 0 \\
x_2^n + 0.6x_1^o - x_2^o - x_2^f - x_2^s - w_2 &= 0 \\
x_3^n + 0.6x_2^o + 0.8x_1^f - x_3^o - x_3^f - x_3^s - w_3 &= 0 \\
x_t^n + 0.6x_{t-1}^o + 0.8x_{t-2}^f + 0.9x_{t-3}^s - x_t^o - x_t^f - x_t^s - w_t &= 0,\ t = 4, \ldots, T-1 \\
x_t^n,\ x_t^o,\ x_t^f,\ x_t^s,\ l_t^o,\ l_t^f,\ l_t^s,\ w_t &\quad \text{integer for relevant } t \\
\text{all variables} &\geq 0.
\end{aligned}$$

The first three constraints just define the variables l_t^o, l_t^f, and l_t^s. The following four constraints ensure that the inflow of napkins is large enough each day. The last four structural constraints guarantee that the inflow each day is equal to the outflow, so that no napkin is unaccounted for along the way. What we buy and what comes in from cleaning done in the previous days should be sent to cleaning at the end of the day or simply wasted if that turns out to be cheaper. As you can see, we do not only have the standard non-negativity constraint, but again also a constraint saying that all variables have to take integer values.

If we solve the model for the given input, we obtain the following result. We buy a total of 2,853 napkins divided over the days as follows.

	Day 1	Day 2	Day 3	Day 4	Day 5
New napkins	900	1,440	513	–	–

The choice of cleaning program is summarized in Table 2.

The cost of this solution is \$9,053.75. We illustrate the solution in the graph given in Figure 4. Notice that we only clean 1,600 napkins at the end of day 4. The remaining 20 are thrown away.

In the three models studied above, either all or none of the variables were restricted to take integer values. We can easily imagine similar models in which a subset of the variables are restricted to be integer whereas the remaining variables are continuous. We refer to such models as mixed-integer models. In the following section, we discuss examples of such models.

Cleaning Program	Day 1	Day 2	Day 3	Day 4
One hour	–	45	840	1,600
Fast	900	1,395	420	–
Standard	–	–	–	–
Total	900	1,440	1,260	1,600

Table 2: Choice of cleaning program, Example 3.

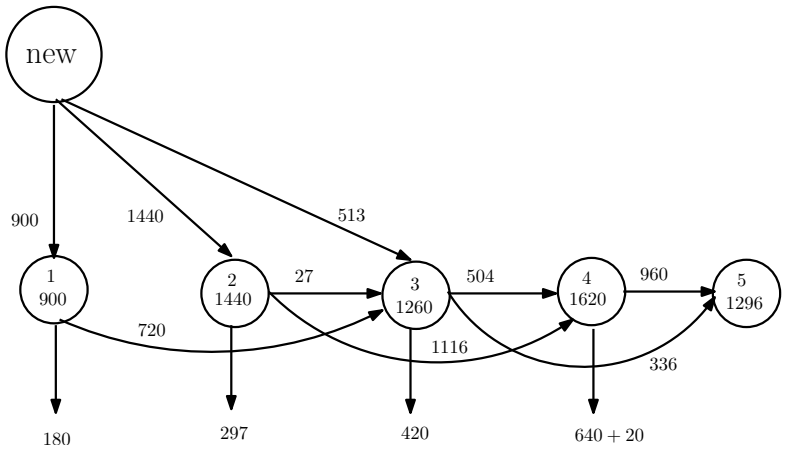


Figure 4: Solution of Example 3.

2 Linear mixed-integer 0-1 problems

Here, we focus on linear mixed-integer optimization models in which the integer variables are restricted to take the values 0 or 1. Another word for 0-1 variables is *binary variables*.

One of the most simple 0-1 optimization problems is the knapsack problem. Here, we are given a set $N = \{1, \ldots, n\}$ of items. Each item $j \in N$ has a weight and a "profit". We would like to decide on which items to bring in the "knapsack", which has a capacity of b, such that the total profit is maximized.

Input. Let

p_j = the profit of item j, $j \in N$,
a_j = the weight of item j, $j \in N$,
b = the capacity of the knapsack.

Variable definition. Let

$$x_j = \begin{cases} 1, & \text{if item } j \text{ is included in the knapsack} \\ 0, & \text{otherwise.} \end{cases}$$

We can now formulate the knapsack problem as follows:

$$\begin{aligned} \max z &= \sum_{j=1}^{n} p_j x_j \\ \text{s.t. } &\sum_{j=1}^{n} a_j x_j \leq b \\ &x_j \in \{0,1\},\ j = 1, \ldots, n \end{aligned}$$

We want to make three remarks:

1. In the definition of the input, we used the notation $j \in N$ whereas we in the problem formulation used $j = 1, \ldots, n$. These variants mean the same and can be used interchangeably.

2. If we replace the $\leq$-sign in the knapsack constraint by the $=$-sign, we obtain the so-called "subset sum" problem. Here, we require the sum of the weights of the subset of chosen items to add up exactly to b.

3. If we replace $x_j \in \{0,1\},\ j = 1, \ldots, n$ by x_j integer, $j = 1, \ldots, n$, we obtain the integer knapsack problem, where x_j denotes the number of item i that is included in the knapsack.

The knapsack problem might not occur in practice as a stand-alone problem, but it does form a relaxation of a vast number of 0-1 or integer optimization problems and is therefore mathematically interesting to study. We return to this simple structure later in this note.

Just one more remark before we move on to the next subsection. Typically, binary optimization problems fall into the category of *combinatorial optimization* problems. There is no unique definition of what a combinatorial optimization problem is, but it is quite usual to refer to a problem as a combinatorial optimization problem if we have a ground set N of items, such as in the knapsack problem, and we seek an optimal subset $S \subseteq N$ of this ground set, that is, we can write the problem as

$$\max \text{ (or } \min)_{S \subseteq N} \{ \sum_{j \in S} c_j \mid S \in \mathfrak{F} \},$$

where c_j is the objective coefficient of item j, and where $\mathfrak{F}$ is the set of feasible solutions.

In the knapsack problem, the objective coefficients are denoted by p_j and the feasible set is defined as

$$\mathfrak{F} = \{x \in \{0,1\}^n \mid \sum_{j \in N} a_j x_j \leq b\}\,.$$

2.1 Modeling fixed costs

In many applications, we are dealing with a cost structure as follows:

$$c(x) = \begin{cases} ax + f, & \text{if } x > 0 \\ 0, & \text{otherwise.} \end{cases}$$

Such a cost structure is present whenever we have to pay a fixed cost for setting up an activity plus a per-unit cost for using it, see Figure 5.

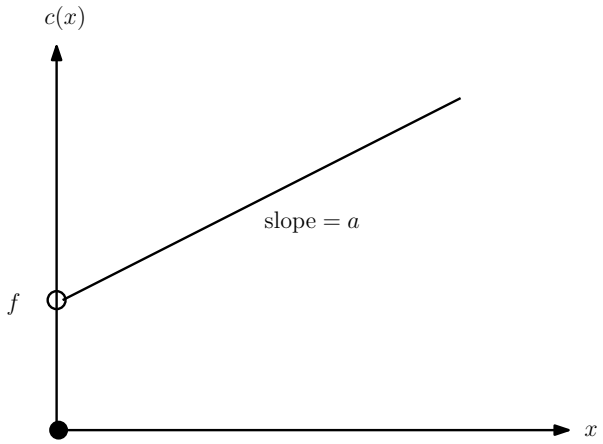


Figure 5: Discontinuous cost structure.

An example where such a cost structure is present is the uncapacitated facility location problem.

Example 4. We are given a set N of clients and a set M of possible sites where we can locate a facility. The cost of opening a facility at site $i \in M$ is f_i and the cost of supplying all of client j's demand from facility i is $c_{ij},\ i \in M,\ j \in N$. We want to decide on at which sites we shall open a facility, and from which open facilities each client is serviced. The goal is to minimize the total cost under the restriction that the demand of each client has to be met. Formulate the problem as a mixed-integer linear optimization problem. An example of an instance is given in Figure 6.

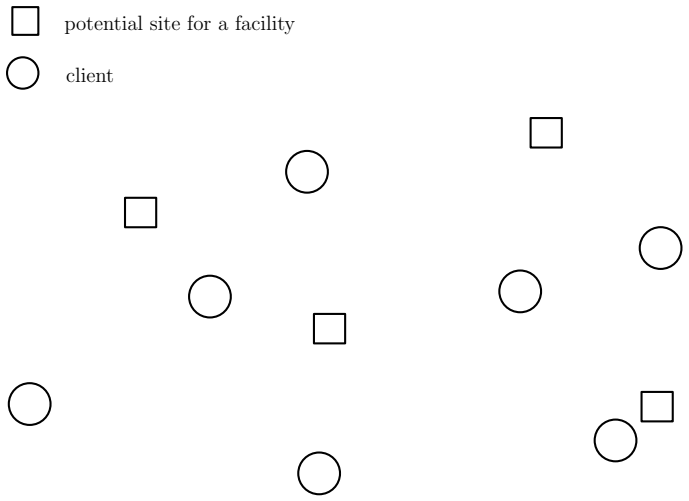


Figure 6: An example of an instance of the uncapacitated facility location problem.

Input. Let

f_i = the fixed cost of opening a facility at site $i,\ i \in M$,

c_{ij} = the cost of supplying all demand of client j from facility $i,\ i \in M,\ j,\ j \in N$.

The question now is which variables to introduce. It is natural to have one type of variable indicating the fraction of client j's demand supplied by facility i.

Variable definition, part 1. Let

x_{ij} = fraction of client j's demand supplied from facility $i\ i,\ i \in M,\ j,\ j \in N$.

Using this set of variables, we can model the restriction that the demand of each client has to be met:

$$\sum_{i \in M} x_{ij} = 1, \quad j \in N\,.$$

Moreover, we can model part of the objective function, namely $\sum_{i \in M} \sum_{j \in N} c_{ij} x_{ij}$, but, how can we model the fixed cost of opening a facility at site $i \in M$ using a linear objective function and constraints? The "trick" is to introduce a binary variable y_i indicating whether or not we open a facility at site $i \in M$.

Variable definition, part 2. Let

$$y_i = \begin{cases} 1, & \text{if a facility is opened at site } i \in M \\ 0, & \text{otherwise.} \end{cases}$$

Using the y_i-variables, we can model the other part of the objective function, namely the fixed costs: $\sum_{i \in M} f_i y_i$. If we leave it at this, all the y_i-variables will take the value

0 (assuming all fixed costs are positive), which is not what we want. What we need is a constraint that forces y_i to take value 1 if the $x_{ij} > 0$ for any $j \in N$. This is done by the constraints $y_i \geq x_{ij}$ for all $j \in N$. We are now ready to write down the complete model.

$$\begin{aligned} \min z = \sum_{i \in M} \sum_{j \in N} c_{ij} x_{ij} + \sum_{i \in M} f_i y_i & \\ \text{s.t.} \sum_{i \in M} x_{ij} = 1, & \quad j \in N \\ y_i - x_{ij} \geq 0, & \quad i \in M,\ j \in N \\ x_{ij} \geq 0, & \quad i \in M,\ j \in N \\ y_i \in \{0,1\}, & \quad i \in M\,. \end{aligned}$$

An example of a feasible solution to the instance illustrated in Figure 6, can be found in Figure 7.

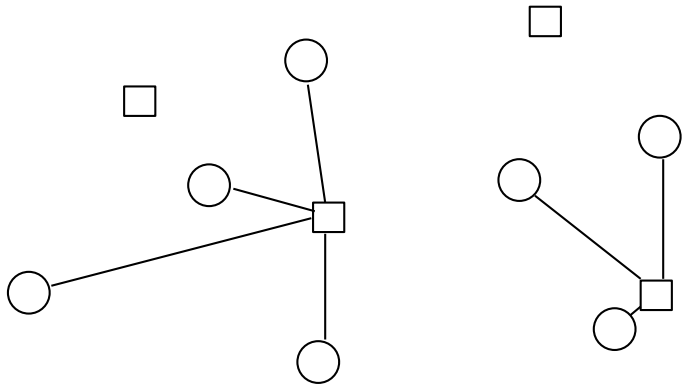


Figure 7: An illustration of a feasible solution to the instance given in Figure 6.

We see that two of the facility sites are not used and that, in this case, all clients are serviced by a single facility. In fact, it is easy to show that if a client is fractionally serviced by two or more facilities, then there exists a solution, with the same objective value, in which the client is serviced by a single facility. This is left as an exercise to the reader.

2.2 Modeling max/min problems

Example 5. A student wants to determine an optimal strategy for preparing for the next exam period in which he has n exams. For each course, he can choose among m levels of preparation. If he prepares course j at level i, he gets the grade c_{ij}, and in order to prepare course j at level i he needs t_{ij} hours. He has only T hours in total for his preparation. We assume that both c_{ij} and t_{ij} are integer for all i, j. His goal is to spend

his hours in such a way that the lowest grade that he obtains is as high as possible. Formulate the problem as a *linear* mixed-integer optimization problem.

Input. Let

c_{ij} = the grade obtained after preparing course j at level i, $i = 1, \ldots m$, $j = 1, \ldots, n$,

t_{ij} = the number of hours needed to prepare course j at level i, $i = 1, \ldots m$, $j = 1, \ldots, n$,

T = the total number of hours available.

This problem is a so-called "max/min" problem, and we have to think carefully how to model this using only linear constraints and a linear objective function. Let us first think about the variable definition. It is clear that we need a 0-1 variable x_{ij} indicating whether or not he prepares a course j at level i. Using this set of variables, we can model important parts of the problem namely:

- the student can prepare course j at exactly one level,
- the total time spent should not exceed T.

These two constraints can be written mathematically as:

$$\begin{aligned} \sum_{i=1}^{m} x_{ij} &= 1, \quad j = 1, \ldots, n\,, \qquad (1) \\ \sum_{i=1}^{m}\sum_{j=1}^{n} t_{ij} x_{ij} &\leq T\,. \end{aligned}$$

The question now is how to link the study efforts to the resulting grade and how to enforce the "max/min" aspect. The grade is really just an outcome of the preparation and we can therefore just define a variable z_j saying which grade the student gets depending on the preparation. Constraint (1) ensures that each course is prepared at exactly one level.

$$z_j = \sum_{i=1}^{m} c_{ij} x_{ij}, \quad j = 1, \ldots, n\,,$$

or, equivalently,

$$z_j - \sum_{i=1}^{m} c_{ij} x_{ij} = 0, \quad j = 1, \ldots, n\,,$$

With all the grades in place, we would like to determine $\max\{\min_j z_j\}$, but this is clearly not linear as it is written here. In order to obtain a linear problem we can, however, introduce an extra "helping variable" z that we maximize with the constraint that $z \leq z_j,\ j = 1, \ldots, n$. To summarize, we get the following set of variables:

Variable definition. Let

$$\begin{aligned} x_{ij} &= \text{1 if course } j \text{ is prepared at level } i \text{ and 0 otherwise, } i = 1, \dots m,\ j = 1, \dots, n, \\ z_j &= \text{the grade obtained for course } j,\ i = 1, \dots m,\ j = 1, \dots, n, \\ z &= \text{the lowest grade obtained.} \end{aligned}$$

The complete model now becomes:

$$\begin{aligned} \max z & \\ \text{s.t.} \sum_{i=1}^{m} x_{ij} &= 1, \quad j = 1, \dots, n\,, \\ \sum_{i=1}^{m}\sum_{j=1}^{n} t_{ij}x_{ij} &\le T\,, \\ z_j - \sum_{i=1}^{m} c_{ij}x_{ij} &= 0, \quad j = 1, \dots, n\,, \\ z - z_j &\le 0, \quad j = 1, \dots, n\,, \\ z,\ z_j &\ge 0, \quad j = 1, \dots, n\,, \\ x_{ij} &\in \{0,1\}, \quad i = 1, \dots, m\ j = 1, \dots, n\,. \end{aligned}$$

It is clear that min/max-problems can be modeled in a similar fashion.

2.3 Disjunctions

There are situations in which we want to model that only a subset of the constraints should be active. In a regular model, we require all constraints to be active, but in some cases it is relevant to make a choice. Typical examples where this occurs are machine scheduling problems, in which we want to determine in which order jobs are processed on machines. Consider two jobs i and j. Then, either i precedes j on a certain machine, or j precedes i. The question is how to model this using linear expressions.

Suppose we have two constraints:

$$\begin{aligned} a^1 x &\le b_1, \\ a^2 x &\le b_2\,, \end{aligned}$$

where a^1 and a^2 are n-dimensional row vectors, $x \in \mathbb{R}^n$ and $0 \le x_j \le u_j,\ j = 1, \dots, n$. To model that only one of the two constraints is active, we introduce one 0-1 variable for each of the constraints.

Variable definition. Let

$$y_i = \begin{cases} 1, & \text{if constraint } i \text{ is active} \\ 0, & \text{otherwise.} \end{cases} \qquad i = 1, 2$$

We also need to introduce a constant M that should have a value that is "large enough". Having the constant M and the binary variables $y_i,\ i = 1, 2$, we can model the disjunction

as follows:

$$\begin{aligned} a^1x - b_1 &\leq M(1-y_1) \\ a^2x - b_2 &\leq M(1-y_2) \\ y_1 + y_2 &= 1 \\ x_j &\leq u_j,\ j = 1,\ldots,n \\ x_j &\geq 0,\ j = 1,\ldots,n \\ y_i &\in \{0,1\},\ i = 1,2\,. \end{aligned} \tag{2}$$

Notice that in the case that we choose between two constraints like in the model above, it suffices to have one binary variable y. We define y as follows,

Alternative variable definition. Let

$$y = \begin{cases} 1, & \text{if constraint (2) is active} \\ 0, & \text{otherwise.} \end{cases}$$

The resulting model is then:

$$\begin{aligned} a^1x - b_1 &\leq M(1-y) \\ a^2x - b_2 &\leq My \\ x_j &\leq u_j,\ j = 1,\ldots,n \\ x_j &\geq 0,\ j = 1,\ldots,n \\ y &\in \{0,1\}\,. \end{aligned}$$

2.4 Modeling connectivity

In several combinatorial optimization problems, especially graph-related problems, we need to choose subsets of the variables such that this subset has certain properties. An example is the Minimum Spanning Tree (MST) problem.

Given an undirected graph $G = (V, E)$, a tree $G' = (V', T)$, $V' \subseteq V$ in G is a connected graph such that $|T| = |V'| - 1$. If $V' = V$, then the tree is a *spanning tree* in G, i.e., the tree spans all the vertices in G, see Figure 8. An MST in G is a spanning tree in which the sum of the edge lengths is minimum.

Input. Let
$G = (V, E)$, and undirected graph
l_e = the length of edge $e \in E$.

To model the MST-problem as a binary linear optimization problem, it is natural to use the following variables:

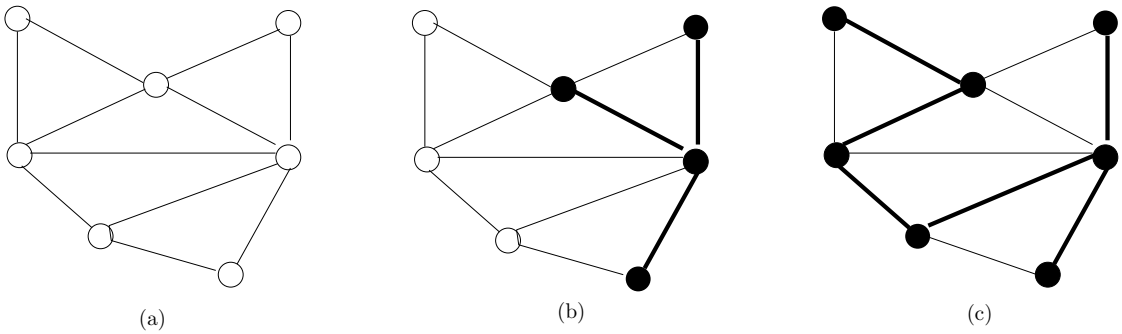


Figure 8: (a) An undirected graph G, (b) a tree in G, (c) a spanning tree in G.

Variable definition. Let

$$x_e = \begin{cases} 1, & \text{if edge } e \text{ is part of the MST} \\ 0, & \text{otherwise.} \end{cases} \qquad e \in E$$

It is quite straightforward to model both the objective function and the requirement that the number of edges is equal to $|V| - 1$:

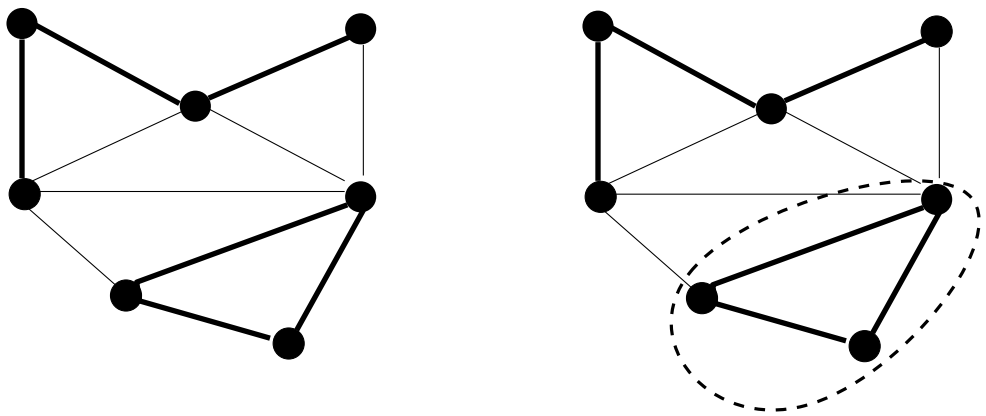
$$\begin{aligned} \min & \sum_{e \in E} l_e x_e \\ \text{s.t.} & \sum_{e \in E} x_e = |V| - 1 \qquad (3) \\ & x_e \in \{0,1\},\ e \in E\,. \qquad (4) \end{aligned}$$

This model is, however, not complete since we have not yet included constraints that enforce the resulting subgraph to be connected. If we consider the graph in Figure 8(a), the subgraph in Figure 9(a) is feasible with respect to Constraints (3) and (4), but it is not a spanning tree.

The difficulty in modeling the MST-problem lies in the modeling of the connectivity requirements. We need to enforce that, for each proper subset S of the vertices, there should be an MST-edge between S and $V \setminus S$. By a proper subset $S \subset V$ we mean that $S \neq \emptyset$ and $S \neq V$. The encircled set of vertices in Figure 9(b) does not have an MST-edge connecting it to the complement set of vertices.

The complete MST-model then becomes:

$$\begin{aligned} \min & \sum_{e \in E} l_e x_e \\ \text{s.t.} & \sum_{e \in E} x_e = |V| - 1 \\ & \sum_{\{e:=\{i,j\} \in E | i \in S,\ j \in V \setminus S\}} x_e \geq 1 \qquad \text{for all } S \subset V \qquad (5) \\ & x_e \in \{0,1\},\ e \in E\,. \end{aligned}$$



(a) (b)

Figure 9: (a) A subgraph satisfying Constraints (3) and (4), (b) A subset of the vertices with no adjacent MST-edge.

What Constraints (5) are saying is that if we look at each proper subset S of the vertices and we sum over all the variables representing edges having one endpoint in S and the other endpoint in the complement of S, then at least one of these variables should take value 1. Since there are exponentially many subsets of V, constraint class (5) is exponentially large. This might seem prohibitive, but in fact it is not in this case. Notice that we can also model the connectivity requirements as follows.

$$\sum_{e \in E(S)} x_e \leq |S| - 1 \quad \text{for all } S \subset V\,,$$

where $E(S)$ is the set of edges with both endpoints in S.

A second problem where similar constraints as (5) occur is the well-known Traveling Salesman Problem (TSP). Here, we again have an undirected graph $G = (V, E)$ and we know the length of each edge $e \in E$. We assume the the graph is *complete*, i.e., all the edges are present, possibly with length equal to infinity.

Input. Let
$G = (V, E)$, a complete undirected graph,
l_e = the length of edge $e \in E$.

The problem is to determine a circuit on all vertices in G such that the length of the circuit is minimized. A different way of interpreting the problem is to view the vertices as "cities" and requiring a "salesman" to visit each city exactly once and then returning to the starting city in such a way that the length of the resulting "tour" is minimized. An example of a TSP-tour is given in Figure 10(b).

Variable definition. Let

$$x_e = \begin{cases} 1, & \text{if edge } e \text{ is part of the TSP-circuit} \\ 0, & \text{otherwise.} \end{cases} \qquad e \in E$$

It is again easy to model the objective function, which is the same as in the MST-problem. To say that each city should be visited exactly once is the same as requiring that the number of TSP-edges adjacent to each vertex should be exactly 2; one is used to enter the "city", and one to leave it. Let $\delta(v)$ be the set of edges adjacent to vertex v, i.e., $\delta(v) = \{e \in E \mid v \text{ is one of the endpoints of } e\}$. We would then obtain:

$$\begin{aligned} \min & \sum_{e \in E} l_e x_e \\ \text{s.t.} & \sum_{e \in \delta(v)} x_e = 2 \quad \text{for all } v \in V \qquad (6) \\ & x_e \in \{0,1\},\ e \in E\,. \qquad (7) \end{aligned}$$

All TSP-circuits are feasible in this model, but we also allow for several so-called "subtours" that only encompass a subset of the vertices, see Figure 10(a). To exclude the

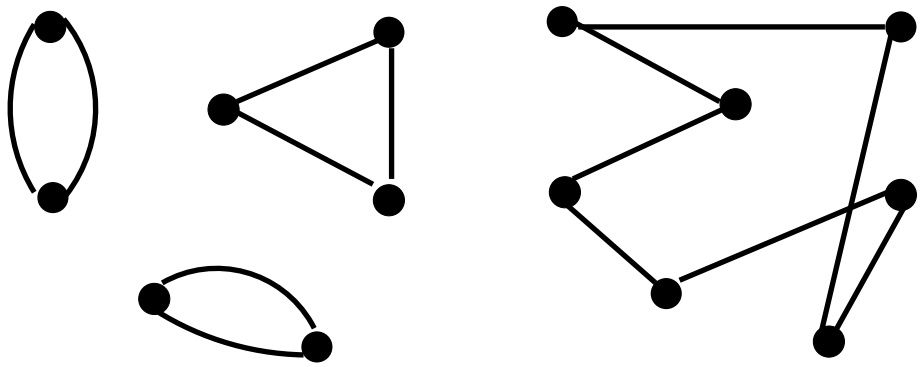


Figure 10: (a) Subtours satisfying Constraints (6) and (7), (b) A TSP-tour.

subtours we need to add so-called "subtour elimination constraints". These are very similar to the connectivity constraints that we added in the MST-problem. The full model of the TSP-problem is as follows:

$$\begin{aligned} \min & \sum_{e \in E} l_e x_e \\ \text{s.t.} & \sum_{e \in \delta(v)} x_e = 2 \quad \text{for all } v \in V \\ & \sum_{\{\{i,j\} \in E \mid i \in S,\ j \in V \setminus S\}} x_e \geq 2 \quad \text{for all } S \subset V \qquad (8) \\ & x_e \in \{0,1\},\ e \in E\,. \end{aligned}$$

Constraints (8) require that at least two edges have to be adjacent to any proper subset of vertices; the salesman has to enter the set at some city, and he has to leave the set at some city. He may also enter and leave the set more than one time, but this is allowed by Constraints (8).

2.5 Modeling integer variables by binary variables

Suppose we want to produce an item in certain batch sizes, say 0, 5, 15, 30, and 50. The natural thing to do is to introduce a variable x, representing the number of produced items, and require $x \in \{0, 5, 15, 30, 50\}$. But this does not immediately fit in our framework of linear integer optimization. The way to deal with this is to introduce one 0-1 variable for each of the four positive batch sizes:

Variable definition. Let

$$y_i = \begin{cases} 1, & \text{if we produce the } i\text{th positive batch size} \\ 0, & \text{otherwise,} \end{cases} \qquad i = 1, \ldots, 4\,,$$

where the batch sizes are 5,15,30,50 respectively. We then get the following constraints:

$$\begin{aligned} x - 5y_1 - 15y_2 - 30y_3 - 50y_4 &= 0 \\ \sum_{i=1}^{4} y_i &\leq 1 \\ y_i &\in \{0,1\}, \quad i = 1, \ldots, 4\,. \end{aligned}$$

We may also want to model general integer variables by binary variables due to possible computational advantages. Suppose we have a variable x that can take any nonnegative integer value, i.e. $x \in \mathbb{Z}_{\geq 0}$. Then we can write x as

$$\begin{aligned} x &= \sum_{i=0}^{k} 2^i y_i \\ y_i &\in \{0,1\}, \quad i = 1, \ldots, k\,. \end{aligned}$$

An important question here is which value we should choose for k if x does not have an easily computable bound. An answer to this question is given in the proof of Theorem 13.6 in the book by Papadimitriou and Steiglitz [1].

2.6 Elimination of products of variables

When modeling real-life applications, you may end up with a non-linear formulation which includes the product of variables. When this concerns the product of two binary variables or the product of a binary and a continuous variable, we can introduce an additional variable to eliminate this product in order to obtain a linear formulation.

First, let's consider the case where we want to eliminate the product of two binary variables x_1 and x_2. This product, x_1x_2, can be replaced by an additional variable y. The following linear constraints force y to take the value of x_1x_2:

$$\begin{aligned} y &\leq x_1 \\ y &\leq x_2 \\ y &\geq x_1 + x_2 - 1 \\ y &\in \{0,1\} \end{aligned}$$

The validity of these constraints can be checked by examining all possible situations listed in the following table.

x_1	x_2	x_1x_2	Constraints	Result
0	0	0	$y \leq 0$ $y \leq 0$ $y \geq -1$ $y \in \{0,1\}$	$y = 0$
0	1	0	$y \leq 0$ $y \leq 1$ $y \geq 0$ $y \in \{0,1\}$	$y = 0$
1	0	0	$y \leq 1$ $y \leq 0$ $y \geq 0$ $y \in \{0,1\}$	$y = 0$
1	1	1	$y \leq 1$ $y \leq 1$ $y \geq 1$ $y \in \{0,1\}$	$y = 1$

Second, we consider the case where we want to eliminate the product of a binary variable x_1 and a continuous variable x_2 for which $0 \leq x_2 \leq u$ holds. We can introduce a new continuous variable y to replace the product x_1x_2. The following constraints must be added to force y to take the value of x_1x_2:

$$\begin{aligned} y &\leq ux_1 \\ y &\leq x_2 \\ y &\geq x_2 - u(1 - x_1) \\ y &\geq 0 \end{aligned}$$

The validity of these constraints can be checked by examining all possible situations listed in the following table.

x_1	x_2	x_1x_2	Constraints	Result
0	$w : 0 \leq w \leq u$	0	$y \leq 0$ $y \leq w$ $y \geq w - u$ $y \geq 0$	$y = 0$
1	$w : 0 \leq w \leq u$	0	$y \leq u$ $y \leq w$ $y \geq w$ $y \geq 0$	$y = w$

References

[1] C.H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization, Algorithms and Complexity.* Prentice-Hall, Inc., Englewood Cliffs, NJ, 1982.

[2] W.L. Winston. *Operations Research: Applications and Algorithms.* Duxbury Press, 1994.

[3] L.A. Wolsey. *Integer Programming.* Wiley-Interscience series in discrete mathematics and optimization, John Wiley & Sons, Inc., Ney York, NY, 1998.

[4] J. Bisschop. *AIMMS - Optimization Modeling.* 2006.