International Series in
Operations Research & Management Science

José Manuel García Sánchez

Modelling in Mathematical Programming

Methodology and Techniques

Springer

José Manuel García Sánchez

Modelling in Mathematical Programming

Methodology and Techniques

Springer

José Manuel García Sánchez
IO and Business Management
University of Seville
Sevilla, Spain

ISSN 0884-8289 ISSN 2214-7934 (electronic)
International Series in Operations Research & Management Science
ISBN 978-3-030-57249-5 ISBN 978-3-030-57250-1 (eBook)
https://doi.org/10.1007/978-3-030-57250-1

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

To Carmen, for her unconditional support.
To Laura and Sara.
To my family.

Preface

Generally, modelling has been considered a little regulated technique, based on knowledge of the types of optimization problems and the experience of the modeler. This book presents the first methodology for the building of a mathematical model in an integral way, as well as new techniques that help us if we want to follow our own building criteria. The objective is to provide a simple work dynamic that facilitates the modelling process.

The book is a basic tool for learning to model in mathematical programming, from models without much complexity to complex system models. The book presents a structure the models, and complex constraints models more easily. It is a basic modelling guide for any system, and explains models already existing in the literature.

The book presents a structure that guides the orderly learning of the components that the methodology establishes in an optimization problem, within a system:

1. The elements: all the actors that participate in the system. They are diverse in nature, from people, tools, places, time, etc. They usually have associated information that we will call data, and that must be numerical information or to be defined numerically.

The elements participate in the actions that occur in the system and support its specifications. They are closely related to the activities that occur in the system, and the joint identification of both components is sometimes effective.

The elements are configured taking into account their quantitative nature, their associated data and their reference in the specifications. The quantitative nature of the elements will be used as a tool to help define decision activities. It is probably an unnecessary tool for an experienced modeler, but it may be useful for people who start modelling in mathematical programming.

2. Decision activities: direct actions that occur in the system for which it is necessary to decide their value, which is not determined. They are associated with the elements. The decision activities are simple actions. They cannot be the result

of a logical calculation, simple function or combination of other decision activities. Decision activities define the main variables of a model.

3. Calculations: based on the decision activities, a system may need additional information that is obtained from these decisions through a calculation process. Calculations can be generated from other calculations, not only from the base decisions of the system. The calculations are also represented as variables.

We differentiate between decision variables and calculations. This helps us to correctly model the relationship between variables. This is one of the important contributions of the book.

Expressing the relationship with the variables on which it depends can be through a mathematical function, as a result of a logical proposition, or acting as a bound of those variables. According to this, we have three types of calculations:

– Auxiliary calculations: represent the calculations obtained from the result of a linear mathematical function. They are used for a concept of comfort and clarity in the construction of the model, without being obliged to use it. They give rise to the auxiliary variables of a problem.
– Logical calculations: they are used to obtain the result of a logical proposition. Its use is mandatory. The variables generated will be referred to as logical variables.
– Upper/lower bound calculations: they are used to obtain the upper or lower bound of a set of variable values. The generated variables will be referred to as boundary variables. Their use is also necessary.

4. Specifications: regulations, impositions or actions of defined value that must be fulfilled in the system. They give rise to the constraints of the problem, but specification should not be confused with constraint. Constraint is a mathematical expression whereas specification is a statement that the system must fulfil and that is implemented in one or more constraints.

We provide a classification of the specifications of a system, where we highlight, as another important contribution, a complete guide for the formulation of specifications based on propositional logic. These types of specifications add complexity to the problems. The modelling guide reduces complexity in the modelling of this type of specification, as well as help to understand modelled problems without the use of a methodology. From the defined components, we will learn to identify the implicit specifications of a system.

5. Objective criterion: this can be understood as one more specification of the system. It expresses the criteria that guides the resolution of the system. The objective will give rise to a linear function of costs, costs that are associated to the elements of the system and that is known as Objective Function. In certain objectives, the criterion will also lead to the use of calculations and the definition of specific constraints. The book will present the modelling of all possible objectives that may arise in optimization problems.

This book follows a sequential approach to the learning of each of the components. However, to explain certain concepts that will appear throughout the methodology, it will sometimes be necessary to refer to components that have not yet been seen. We do this as simply as possible, leaving the most advanced concepts for when all the components have been studied.

Sevilla, Spain José Manuel García Sánchez

Contents

Chapter 1
Introduction to Modelling in Mathematical Programming

1.1 Model

In general, a model is a representation of reality (Colin 1973). In that order and in a more extensive way, Pidd (2010) proposes the model definition as "an explicit and external representation of part of reality as seen by people who want to use the model to understand, change, manage and control this part of reality". On the other hand, Aracil (1983) defines that "a model constitutes an abstract representation of a certain aspect of reality and has a structure that is formed by the elements that characterize the aspect of modelled reality and the relationships between these elements".

Based on these definitions, we can assume that the objective when creating a model must be to create a representation as complete and as close as possible to reality. And the representation of reality is the representation of the elements that participate in it and their relationships. Based on this, this book aims to be a tool to build models, mathematical models and, more specifically, mathematical programming models, also called optimization models.

García Sabater (2015) defines mathematical models as "formal models that use the language of mathematics to describe a system, expressing parameters, variables and relationships."

Lowry (1965) distinguishes three types of mathematical models:

Descriptive models: they are used to describe an existing situation. On a set of variables subject to some mathematical equation, results are obtained that inform us about a certain situation.

Prediction or forecasting models: they are used for the simulation of future events. They describe a system over time, which is why some authors include them within the descriptive models.

J. M. García Sánchez, *Modelling in Mathematical Programming*, International Series in Operations Research & Management Science 298,
https://doi.org/10.1007/978-3-030-57250-1_1

Planning or normative models: they are built based on goals and restrictions. They pursue the creation of a plan that better reaches a fixed objective and that is going to be subject to a series of restrictions.

The models of mathematical programming or optimization are planning models in which mathematical relationships are to be expressed through functions. Modelling in the field of mathematical programming corresponds to expressing through mathematical relationships, called constraints, the specifications of a system where a series of activities are performed and represented as variables and in which an optimization criterion or function is followed for its realization. The systems are formed by a set of related elements that favor both the activities of the system and its specifications.

Modelling is the tool that allows us to capture the reality of a system within a mathematical framework, perfectly usable by operational research to find solutions to the problem posed.

The term optimization is fundamental in the area of mathematical programming. Optimization consists in the maximization or minimization of a function, known as the objective function. Mathematical programming looks for valid solutions that represent the activity of a system and that can be evaluated with respect to that objective function.

Generally, modelling has been considered a little regulated technique, based on knowledge of the types of optimization problems and the experience of the modeller. This book aims to provide a methodology for the construction of a mathematical model in an integral way, as well as techniques that help us if we want to follow our own building criteria. The objective is to provide a simple work dynamic that facilitates the modelling process.

An important aspect in modelling is the description of the system to be modelled. It is necessary to perfectly identify the elements that are part of the problem and all the characteristics that are relevant. We should not describe parts of the system that are not part of the problem. In modelling, it is fundamental to start with an exhaustive and precise description and, from this, to lead the modeller to a correct definition of the components of the model.

The book does not deal with the search processes for solutions of the different resolution methodologies. For this there are countless bibliographical references that the reader can use.

1.2 Classical Components of a Mathematical Programming Model

A basic model of mathematical programming consists of four components (Castillo et al. 2002):

- *Data*: the deterministic values that the model handles and that are represented in a simplified way with the use of parameters.
- *Variables*: these define the decisions that occur in the problem. The variables always represent what there is to find out their value. Regarding the value, the variables of a model can be continuous or integer, and within the integer variables, a special and important type of variable is distinguished, the binary variables, which only take the values 1/0 (true/false).
- *Constraints*: equalities and mathematical inequalities that define the specifications and rules of the problem. The constraints are mathematical relationships between the variables and the problem data. The type of relationship determines the type of model as we will see in Sect. 1.3 of this chapter.

On the other hand, sign constraints associated with the variables of the problem are also imposed.

- *Objective Function*: this defines the optimization criterion, which will maximize or minimize a function.

As we will see next, within the types of mathematical programming models, we can work with variants regarding the type of data, the linear character of the mathematical functions, and the number of constraints and objective functions.

1.3 Classification of Mathematical Programming Models

The mathematical programming models or optimization models depend fundamentally on their resolution of the type of variables that are part of it and the linear or non-linear character of the mathematical expressions that compose it. However, there are other factors that also define the classification of the models:

- According to the model data:
  - Deterministic models: all model data are known and accurate.
  - Stochastic models: the data have a random or probabilistic component.

This methodology focuses exclusively on mathematical deterministic models, but independently of this, some of its techniques for the elaboration of probabilistic models could be used.

- According to the number of objectives of the problem and the number of restrictions:

We can work with mathematical models that only have constraints, optimization problems with an objective function (most common case), or optimization problems with several objective functions. Similarly, there are optimization problems that do not have constraints and only have an objective function.

- According to the type of functions that make up the model and its variables:
  - Linear Programming Models: all mathematical expressions are linear.
  - Non-linear Programming Models: there are some non-linear expressions, either in the constraints or in the objective function.

  Within the linear models, we distinguish between:

- (Continuous) Linear Programming Models: all the variables are continuous.
- Integer Linear Programming Models: there are integer variables. Within these, the following are distinguished:
  - Pure Integer Linear Programming Models: all are integer variables.
  - Mixed Integer Linear Programming Models: there are integer and continuous variables.
  - Binary Linear Programming Models: all variables are binary.

1.4 First Example

In order to quickly introduce the concept of modelling in mathematical programming, let us first take a look at an example of what it takes to obtain a mathematical model from the description of a system. For our first example, we will consider a very simplified production system.

Butter Production
Imagine a butter production factory that wants to optimize its daily production of butter. Two types of butter are made (Sweet and Raw). A kilo of sweet butter gives the manufacturer a profit of $10 and a kilo of raw a profit of $15. For the production of butter, two machines are used: a pasteurization machine and a whipping machine. The daily use time of the pasteurization machine is 3.5 hours and 6 hours for the whipping machine.

The time (in minutes) consumed by each machine to obtain a kilo of butter is shown in Table 1.1:

To gain simplicity, we ignore the input components for the production of butter (cream, water, salt, preservatives, etc.) and any cost generated in the process.

Let's move on to the identification of the basic components of the mathematical model (variables, constraints, and objective function). The Data component is displayed, while the constraints and objective function are identified.

1. Variables

As mentioned, the variables of a model represent the actions or activities that occur in the system and on which it is necessary to decide a value. In our case, the activity in this system is the production of butter, specifically, the activity of producing sweet butter and raw butter. What we need to find out is how much

Table 1.1 Butter processing times

	Sweet butter	Raw butter
Pasteurization	3	3
Whipping	3	6

sweet butter to produce and how much raw butter to produce. To represent these two actions, we define two variables:

- $x1$: Amount of sweet butter to be produced
- $x2$: Amount of raw butter to be produced

Since butter production is measured in kilograms, these two variables will take continuous values.

We assume that a negative number of kilos of butter cannot be produced, so the sign of the variables is established as:

$$x_1 \geq 0$$
$$x_2 \geq 0$$

2. Constraints

The activities of the system require two processes to be carried out: pasteurization and whipping. Each of them is carried out in a different machine. There are a pasteurization machine with a working capacity of 3.5 h/day and a whipping machine with a capacity of 6 h/day.

We need the use of these two machines to carry out our activities. Therefore, we could consider machines as resources of the production system.

Resources:

– Pasteurization machine
– Whipping machine

The operating specification that these resources impose on the system is its capacity. Therefore, the total consumption of these resources must not exceed their capacity. In other words:

Consumption in the pasteurization machine $\leq$ Capacity of the pasteurization machine
Consumption in the whipping machine $\leq$ Capacity of the whipping machine

The consumption of each resource is generated by each of the system's activities. Sweet butter production consumes time on each machine, as does the production of raw butter, according to the values shown in Table 1.1.

Focusing on the pasteurization machine, it is pointed out that:

1 Kg of sweet butter consumes 3 min of pasteurization; therefore:
2 Kg of sweet butter consumes 6 (3×2) min of pasteurization.
3 Kg of sweet butter consumes 6 (3×3) min of pasteurization.
. . .

so:
x_1 Kg of sweet butter consumes 3 × x_1 min of pasteurization.
Identical analysis for raw butter generates a consumption of $3x_2$.
The total consumption in minutes would be expressed therefore as:
$3x_1 + 3x_2$ min

The capacity of the pasteurization machine was 3.5 h/day = 210 min/day. We express capacity and consumption in the same unit, minutes, and the mathematical expression that defines the constraint of the use of that resource would be as:

$$\mathbf{3x_1 + 3x_2 \leq 210}$$

In a similar way, the constraint associated with the resource of the whipping machine would be developed, obtaining as a final set of system restrictions:

– Pasteurization machine ⇒ $\mathbf{3x_1 + 8x_2 \leq 210}$
– Whipping machine ⇒ $\mathbf{3x_1 + 6x_2 \leq 360}$

3. Objective Function

The optimization criterion of the system is the maximization of the benefit of production. The system has the profit produced by the two activities that constitute the production process, that is, it is known that a profit of \$10 is obtained for the production of each kilo of sweet butter and \$15 for the production of each kilo of raw butter. Mathematically defining profit expression:
Total Profit = Profit of sweet butter production + Profit of raw butter production
Profit of sweet butter production:

1 Kg of sweet butter ⇒ Profit = \$10
2 Kg of sweet butter ⇒ Profit = 10 × 2 = \$20
3 Kg of sweet butter ⇒ Profit = 10 × 3 = \$30
...
x_1 Kg of sweet butter ⇒ Profit = \$$10x_1$
Profit of raw butter production = $15x_1$
Total Profit = $10x_1 + 15x_2$

Therefore, the maximization of the benefit is defined as **Maximize $\mathbf{10x_1 + 15x_2}$.**

4. Complete Model

Complete model is defined as:

Maximize $10x_1 + 15x_2$
subject to
$3x_1 + 3x_2 \leq 210$
$3x_1 + 6x_2 \leq 360$
$x_1 \geq 0$
$x_2 \geq 0$
x_1, x_2 continuous

With this we have just expressed a first model with two continuous variables, two linear constraints and a linear objective function. A solution of the model is any combination of values of the problem variables (x_1, x_2) that satisfy the imposed specifications. Example ($x_1 = 20$, $x_2 = 30$) is a solution that follows the specifications:

$$\begin{aligned}
&3 \cdot 20 + 3 \cdot 30 \leq 210 \qquad \Rightarrow \qquad 150 \leq 210\\
&3 \cdot 20 + 6 \cdot 30 \leq 360 \qquad \Rightarrow \qquad 240 \leq 360\\
&20 \geq 0\\
&30 \geq 0\\
&\text{Solution profit : } 10 \times 20 + 15 \cdot 30 = \$650
\end{aligned}$$

The mission of the resolution methods is to find the optimal solution or one close to the optimum. As already mentioned, the study of resolution methods is not the subject of this book. For this problem, the optimal solution is reached for the production values ($x_1 = 20$, x_2=50) with a profit of \$950 (10 × 20 + 15 × 50).

This first example has served to establish the correspondence between System and Model, about what it means to transform the written description of simple production processes into a set of mathematical expressions. From the next chapter, we will define our methodology from a broader perspective.

1.5 Considerations on the Format of a Mathematical Model

The general format of a model can be represented as follows:

$$\begin{aligned}
&\text{Min} \quad f(X)\\
&\text{subject to}\\
&G_i(X) \approx b_i i = 1 \ldots m
\end{aligned}$$

where:
X: Vector of variables
f(*X*): Objective function
$\approx$: Sign of the constraints [$\leq$; $=$; $\geq$]
$b = (b_1 \ldots b_i \ldots b_m)$: Vector of independent terms
G_i (*X*) $\approx b_i$: Set of *m* constraints, i = 1 ... *m*

An extended representation of this format could be seen as follows:

$$
\begin{aligned}
&\text{Min } c_1x_1 + c_2x_2 + \ldots + c_jx_j + \ldots + c_nx_n \\
&s.t \\
&a_{11}x_1 + a_{12}x_2 + \ldots + a_{1j}x_j + \ldots + a_{1n}x_n \approx b_1 \\
&a_{21}x_1 + a_{22}x_2 + \ldots + a_{2j}x_j + \ldots + a_{2n}x_n \approx b_2 \\
&\ldots \\
&a_{i1}x_1 + a_{i2}x_2 + \ldots + a_{ij}x_j + \ldots + a_{in}x_n \approx b_i \\
&\ldots \\
&a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mj}x_j + \ldots + a_{mn}x_n \approx b_m \\
&x_1 \geq 0 \quad x_2 \geq 0 \ \ \ldots \ \ x_j \geq 0 \ \ \ldots \ \ x_n \geq 0
\end{aligned}
$$

The objective of the problem is considered in a generic way as to minimize the objective function. The expression of maximizing a function can be transformed into minimizing by a simple transformation:

$$\text{Max} f(x) \Rightarrow -\text{Min} - f(x)$$

The first sign does not affect the resolution of the model, only the value resulting from the objective function, in which case the sign should be changed, that is, the value of the objective function of any solution obtained through the minimize problem will have to change sign to convert it to the maximize problem.

The second sign involves changing the sign of the original objective function. Let us see the transformation of our first example:

$$
\begin{aligned}
\text{Maximize } 10x_1 + 15x_2 &\Rightarrow -\text{Minimize} - (10x_1 + 15x_2) \\
&\Rightarrow -\text{Minimize} - 10x_1 - 15x_2
\end{aligned}
$$

The second important aspect in the definition of any model is the sign of the constraints. Any constraint is limited to the use of three signs: $\leq$, $=$, and $\geq$.

In literature it is possible to find the generic definition of a model as:

$$
\begin{aligned}
&\text{Min} \quad f(X) \\
&\text{subject to} \\
&G_i(X) \leq b_i \nearrow i = 1 \ldots m
\end{aligned}
$$

This representation is even more simplified, but equivalent:

In the case of having a constraint with sign $G_i\ (X) \geq b_i$, it is sufficient to multiply the restriction by -1 to convert the sign: $-G_i(X) \leq -b_i$.

In the case of having a constraint with sign =, which is equivalent to two similar constraints, one with sign $\leq$ and one with sign $\geq$: $G_i(X) = b_i \Rightarrow G_i(X) \leq b_i$; $G_i(X) \geq b_i$.
With the signed restriction $\geq$, proceed as before: $-G_i(X) \leq -b_i$.
Therefore, $G_i(X) = b_i$ is equivalent to $G_i(X) \leq b_i$ and $-\mathrm{G_i}(X) \leq -b_i$.
So, any model can be expressed with sign $\leq$.

On the other hand, the use of the greater-than and the less-than sign is not allowed. In the case that a system finds a constraint that is defined with those signs, a small modification must be made to express it correctly. Imagine a generic constraint $G_i(X)<b_i$. The way of operating is as follows:

If all the variables are integer, the problem is solved by defining as independent term b_i-1, since it is the maximum value that the expression $G_i(X)$ could reach. In the case of the greater-than sign, $G_i(X)>b_i$, the equivalent constraint is defined as $G_i(X)\geq b_i + 1$.

If there are continuous variables, then for the case of <, it may be that the expression $G_i(X)$ will take values between $b_i - 1$ and b_i, so the previous transformation could not be performed. In this case, it is necessary to commit a minimum controlled error in the constraint, defining the value ε as small as we want, and the constraint is defined as $G_i(X) \leq b_i\text{-}\varepsilon$. In the case of >, $G_i(X) \geq b_i+\varepsilon$.

The adjustment of the value of ε is made according to the precision of the values that the continuous variables could take and therefore the function $G_i(X)$.

Finally, if an original variable is coming defined as a negative variable $x_j \leq 0$, then we do a substitution of variables. We let $y_i = -x_i$. Then $y_i \geq 0$. And we substitute $-y_i$ for x_i wherever x_i appears in the model. And in the case where x_i is a free variable, unconstrained in sign, we substitute in the model the free variable x_i by the difference of two nonnegative variables, $x_i = e_i - v_i$, $e_i \geq 0$ and $v_i \geq 0$. If necessary, we can impose a propositional logic specification in the model so as not to allow the two variables e_i and v_i to take positive values.

1.6 Justification of the Use of Mathematical Programming Models

The advancement of technology and optimization libraries contributes significantly to the importance of using mathematical models as a way of solving this kind of problems. For decades, mathematical models have been formulated to represent any optimization problem, but in most cases, these models have been used as an insubstantial mathematical contribution, which only allowed for obtaining solutions for instances with few data. Nowadays, thanks to technology, it is possible to solve larger problems. For this reason, optimization libraries have proliferated with a more advanced methodological development, which allow the possibility of solving optimization problems counting only on the mathematical model, without the intervention of the user and without the need to elaborate complex exact or heuristic methods to obtain solutions.

When we talk about solving mathematical programming models, we talk about the search for optimal solutions, that is, exact methods of resolution. In an auxiliary way, mathematical programming can participate in approximate procedures, but on subproblems of the original one, or using time as a factor in the completion of the search process of the optimal solution and collecting the best solution found. But in a primary way, the resolution of mathematical programming models is focused on the search for an optimal solution.

There are three fundamental factors that affect the choice of how to solve a problem:

- The computational complexity of the problem: referred to if the problem fits within the problems of class P (polynomial) or class NP (non-polynomial, and not class P) (Dean 2016).
- The size of the problem: the size of a problem is determined by the number of elements involved in it.
- The temporary nature of the problem: there are operational problems for which a solution is needed in a very limited time. The problem is solved very often, either because the solution is used in a short period of time or because it is necessary to test many alternatives with respect to the data. On the other hand, there are tactical and strategic problems in which the solution is going to be used for a long time and we are allowed the license to be able to wait a while until obtaining a solution.

The entire optimization problems framed in the computational class P can be solved optimally without the need of the mathematical model. Above them, the heuristic resolution does not fit, regardless of the size and temporary nature of the problem. Likewise, the use of the model to obtain optimal solutions is also usually a feasible option.

The problems of linear programming, only with continuous variables, are considered non-complex problems and are solved optimally with the mathematical model. All existing exact procedures are standards that make use of the model.

When the problem is not considered class P and it is considered NP, then the size of the problem and its temporary nature come into play. In the first place, it must be mentioned that each optimization problem in this typology needs its own analysis regarding the resolution time to obtain optimal solutions. There are NP problems for which the mathematical model presents very good behavior, even for instances of a considerable size. On the other hand, there are others where, even for a small size, the exact resolution procedure uses a lot of time (e.g., many scheduling problems). For these types of problems, the option of obtaining the optimal solution requires the mathematical model in the vast majority of cases. There is a very limited group on which other exact techniques can be used, such as dynamic programming, but this possibility is very limited.

The other resolution option for NP problems is the heuristic resolution, for which we do not need the mathematical model. It is the option to choose when there are time limitations and the exact method uses too much time. It may even be the best

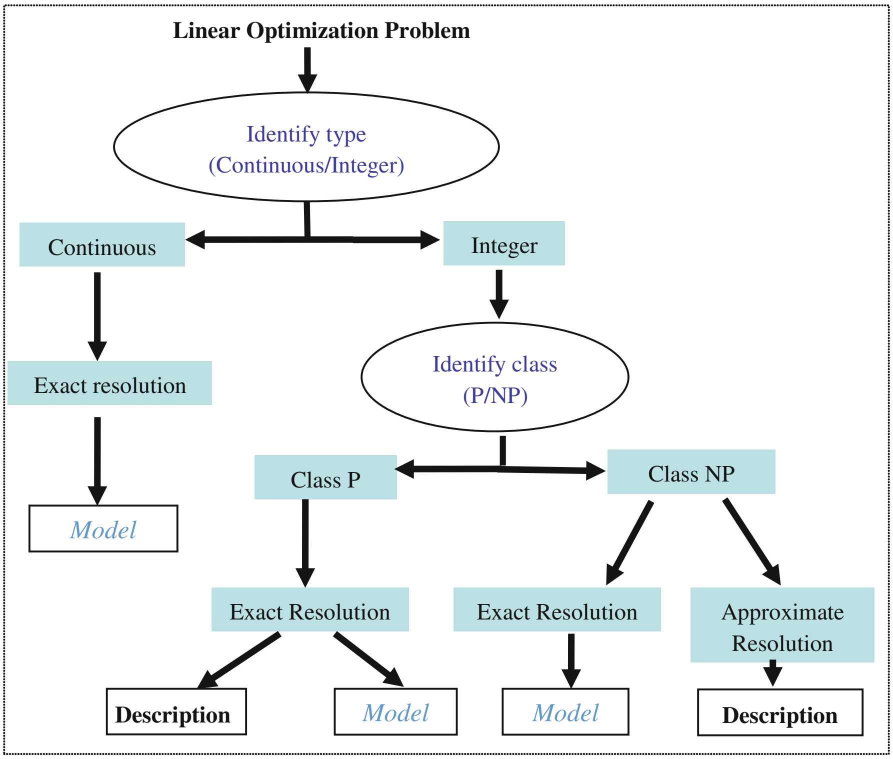


Fig. 1.1 Resolution alternatives

option if the system data are unreliable or simply because it is not so necessary to obtain the optimal solution.

Figure 1.1 summarizes the resolution options for an optimization problem.

Real systems tend to have a large volume of data and specifications. Almost all of them usually have a computational complexity of NP type. Class P problems represent a very small percentage of optimization problems. All this contributes to the fact that nowadays heuristic methodologies are the first option for solving a problem. In any case, I would always recommend the use of the mathematical model and the exact resolution with optimization libraries as a first resource to try to solve a problem.

From a futuristic perspective, both the arrival of increasingly faster processors and the evolution and the rise of computer networks that allow parallel and distributed computing will contribute to the exact resolution of problems using existing techniques (such as the branch and bound method) being an increasingly important option.

It is essential to model with the resolution in mind and to provide a model that obtains solutions as quickly as possible, so to make the most appropriate model for the later resolution is a job for the modeller.

References

Aracil, J. (1983). *Introducción a la dinámica de sistemas*. Madrid: Alianza Editorial.

Castillo, E., Conejo, P., & Pedregal P. (2002). Formulación y resolución de modelos de programación matemática en ingeniería y ciencia. Universidad de Castilla La Mancha Ediciones.

Colin, L. (1973). *Models in planning: An introduction to the use of quantitative models in planning*. Oxford: Pergamon Press.

Dean, W. (2016). Computational complexity theory. In *The Stanford encyclopedia of philosophy*. Stanford: Metaphysics Research Lab, Stanford University.

García Sabater, J. P. (2015). http://personales.upv.es/jpgarcia/LinkedDocuments/modeladomatematico.pdf. Modelado y Resolución de Problemas de Organización Industrial mediante Programación Matemática Lineal. Accessed May, 2020.

Lowry, I. S. (1965). A short course in model design. *Journal of the American Institute of Planners, 31*, 158.

Pidd, M. (2010). *Tools for thinking: Modelling in management science* (3rd ed.). Chichester: Wiley.

Chapter 2
Structure of a Mathematical Programming Model

2.1 Environment of an Optimization Problem

Optimization problems arise within diverse environments of any nature. In this book we will denominate the environment as a system, so that the system is where the optimization problem is defined, and it is the system that has the elements that participate in the problem, the specifications and the criterion that directs the optimization.

Aracil and Gordillo (1997) refer to the system as an aspect of reality endowed with a certain complexity because it consists of interacting parts. We can speak of a system when we refer to any organized environment on which a specific management of its elements is proposed, whose objective is to achieve goals under an objective criterion. In our environment, we will work with static systems, those in which the information of their elements does not change over time. There are also dynamic systems, which are solved with other techniques.

Static systems can be considered and solved by optimization techniques. The objective of this methodology is to convert the description of the system, that is, the description of an optimization problem, to a mathematical language usable by optimization techniques in mathematical programming, which is known in terms of operational research such as "modelling the problem."

Traditionally, whenever an optimization problem is defined, a management situation is described within a given environment. The model is the mathematical representation of the problem. It is perfectly admissible, and just as appropriate, to refer to the term "model the system" instead of "model the problem," meaning "modelling the system," how to model its operation or model its management. The result in any case will be a mathematical model that we can refer to as a mathematical programming model or optimization model, which will later be solved to obtain a solution for its management.

J. M. García Sánchez, *Modelling in Mathematical Programming*, International Series in Operations Research & Management Science 298,
https://doi.org/10.1007/978-3-030-57250-1_2

2.2 Components of an Optimization Problem

In a system where an optimization problem is defined, the following components are distinguished:

Elements
The elements are all the actors that participate in the system. They are diverse in nature, from people, tools, places, time, etc.

They usually have associated information that we will call data, and that must be numerical information or to be defined numerically.

The elements participate in the actions that occur in the system and also support its specifications. They are closely related to the activities that occur in the system, and sometimes the joint identification of both components is effective.

The elements are configured taking into account their nature, their associated data, and their reference in the specifications.

Decision Activities
These are direct actions that occur in the system for which it is necessary to decide their value, which is not determined. They are associated with the elements. The decision activities are simple actions. They cannot be the result of a logical calculation, simple function, or combination of other decision activities.

Decision activities define the main variables of a model. The values that a variable can have will be a binary value (1/0), an integer value, or a continuous value.

Calculations
Based on the decision activities, a system may need additional information that is obtained from these decisions through a calculation process. Even calculations can be generated from other calculations, not only from the base decisions of the system. The calculations are also represented as variables, and in classical formulation, they are also considered decision variables, although they are really only decision variables for the resolution processes not for the operation of a system.

The way of expressing the relationship with the variables on which it depends can be through a mathematical function, as a result of a logical proposition, or acting as a bound of those variables. According to this, we will have three types of calculations:

Auxiliary Calculations

Represent the calculations obtained from the result of a linear mathematical function. Normally, they are used for a concept of comfort and clarity in the construction of the model, without being obliged to use it, but they are mandatory when we want to discretize the value of a function, that is, when we want the value of a function to be integer. They give rise to the auxiliary variables of a problem.

Logical Calculations

They are used to obtain the result of a logical proposition (Klement 2006). Its use is mandatory. The variables generated will be referred to as logical variables.

Lower/Upper Bound Calculations

They are used to obtain the upper or lower bound of a set of variables. Its use is also necessary.

Specifications
Regulations, impositions or limitations that must be fulfilled in the system. They give rise to the constraints of the problem, but specification should not be confused with constraint. Constraint is a single mathematical expression, whereas the specification is a characteristic that the system must fulfill and that is implemented in one or more constraints.

Objective Criterion
This can be understood as one more specification of the system, expressing the criteria that guide the resolution of the system. The objective will give rise to a linear function of costs, costs that are associated to the elements of the system and that is known as objective function. In certain objectives, the criterion will also lead to the use of calculations and the definition of specific constraints.

This book will follow a sequential approach to the learning of each of the components, practically following the order presented now. However, to explain certain concepts that will appear throughout the methodology, it will sometimes be necessary to refer to components that have not yet been seen. We will try to do this as simply as possible, leaving the most advanced concepts for when all the components have been studied.

Figure 2.1 represents the relationships between the different components. The elements, owners of the system information, relate to all the components of the system with a special influence on the activities. On them fall the specifications, calculations, and the optimization criterion. And these components also determine the typology of the elements, so the relationship is bidirectional.

The decision activities participate in the specifications and the objective criterion. The calculations that are defined as variables also participate in specifications and the objective function. They can be necessary to define specifications and the objective criterion. Likewise, the definition of a calculation is represented as a specification (Fig. 2.1).

Relationship of Proposed Components and Classical Components
As is logical, the classical components (García Sabater 2015; Castillo et al. 2002) are quite equivalent to the components proposed in this methodology, with hints that are displayed in the following figure (Fig. 2.2):

The elements provide all the data, parameters and subscripts that the problem will have, with an organized structure that will allow its subsequent implementation in an optimization library. The elements are also used with special relevance in the definition of problem decision activities, and its quantitative nature will determine the type of variable. The quantitative nature will depend on its ability to be measured in the system, its degree of determination, its data, and how the element is identified in the system.

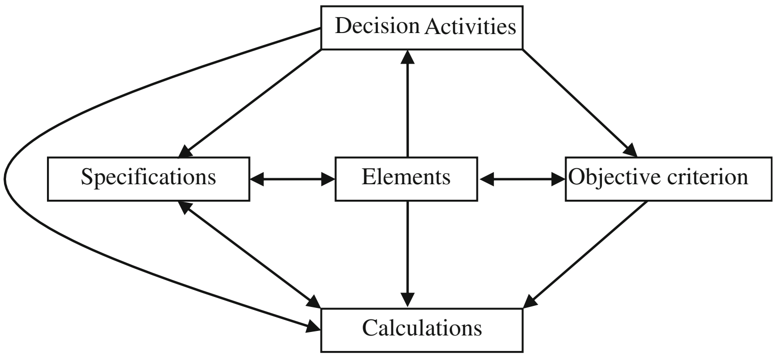


Fig. 2.1 Relationship between the proposed components

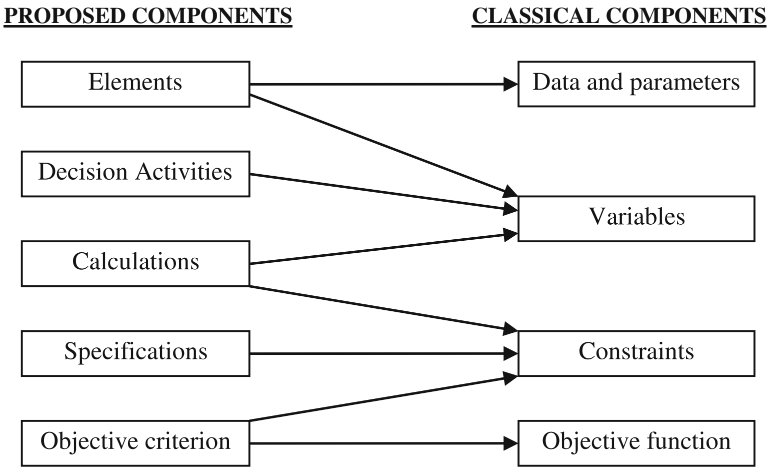


Fig. 2.2 Relationship between components

The decision activities correspond to variables of the problem. The calculations also generate variables and always have associated constraints. The specifications, as already mentioned, define constraints. The objective criterion defines the objective function and may also come with constraints.

2.3 Examples

To illustrate the components of a system, let us look at a couple of simple examples. The first one was modelled in Chap. 1. We use it here simply to repeat its modelling but with the components of this methodology. The second system is a sales system. Both will be used as illustrations for many explanations in future chapters.

Illustration 2.1: Production of Butter

A butter production factory wants to optimize its daily production of butter. Two types of butter are made (Sweet and Raw). A kilo of sweet butter gives the manufacturer a profit of $10 and a kilo of raw a profit of $15. For the production of butter, two machines are used, a pasteurization machine and a whipping machine. The daily use time of the pasteurization machine is 3.5 h and 6 h for the whipping machine.

The time (in minutes) consumed by each machine to obtain a kilo of butter is shown in the following table:

Table. Butter processing times (in minutes)

	Sweet butter	**Raw butter**
Pasteurization	3	3
Whipping	3	6

1. **Elements**

   Analyzing the statement, we can identify the following participating elements: Regarding the resources used in production:

- Pasteurization machine: The numerical information associated with this element in the system is:
  - The usage time = 3.5 h.
  - The time that a kilo of sweet butter uses in the machine = 3 min.
  - The time spent in the machine by kilo of raw butter = 3 min.
- Whipping machine: The associated information coincides with the information of the pasteurization machine, that is, the time of use (6 h), the time used by a kilo of sweet butter (3 min), and the time used by a kilo of raw butter (6 min).

  Regarding the concepts produced in the system:

- Sweet Butter: The information of this element in the system has already been partly reflected, and it is the time that a kilo of this element needs in each machine. In addition to these times, there is also a profit per kilo = $10.
- Raw Butter: Element similar to the other butter and with the same information.

We could also consider as an element another actor that appears in the text, the factory itself, which would play the role of system. We can verify that it does not have any characteristic that is translated into a certain value, nor will there be any activity that must necessarily be associated with it. In any case, its definition as an element does not complicate the elaboration of the model.

2. **Decision Activities**

To recognize the decision activities, it is necessary to explore the actions that take place in the system. To these actions we must assign a type of value (binary, integer, or continuous) and that the value of the action is not determined.

The methodology we will use from the next chapter to obtain all the components of the system will allow us to obtain the activities and assign them the appropriate type of value. In our example we identify as decision activities:

Production of sweet butter

It is an action associated with the sweet butter element.

It is necessary to examine what type of value this activity has:

If we propose a binary value (1, sweet butter is produced; 0, no sweet butter is produced), we would not obtain complete information of the activity. As we will study in the chapter on decision activities, the characteristics of the element discard this type of value for the activity.

If we propose an integer value, we would be restricting the set of values to only integer values. However, production is measured in kilos and this is a continuous measure. Therefore, the appropriate value for the activity is the last of the options, a continuous value.

Defined the type of value, it is necessary to ask whether it is a known value, a value to be determined by a function or an indeterminate value. Being an indeterminate value, we can affirm that it is a decision activity in the system.

Production of raw butter

Activity of identical nature to the previous one.

A decision variable is generated from each activity. Therefore, the decision activities associated with producing butter would give rise to two variables:

x_1: Kilos of sweet butter that are produced**;** $x_1 \geq 0$
x_2: Kilos of sweet butter that are produced**;** $x_2 \geq 0$

3. **Calculations**

It is not necessary to perform calculations on the problem. Nor is it necessary to provide clarity to the model since its size is very small.

4. **Specifications**

Extracting the specifications is the task for which a greater clarity in the description of the system is needed.

In the study of the types of specifications that we will carry out in Chap. 6, we will see that a very common specification occurs when some element acts as a resource. In that case, the element must have associated a numerical characteristic that represents availability or capacity, and there must be decision activities that consume that resource.

In our case, both the pasteurization machine and the whipping machine act as resources.

Regarding pasteurization:

Capacity $= 3.5$ h $= 210$ min

Consumption:

x_1 (sweet butter production) consumes 3 min for each unit that the variable takes.
x_2 (raw butter production) consumes 3 min for each unit that the variable takes.

Constraint generated: Consumption $\leq$ Capacity

$3x_1 + 3x_2 \leq 210$

(It is necessary to work with the same unit of measurement, in this case, minutes.)

Regarding whipping, a specification of the same nature and with the same participating activities is applied. Constraint generated: $3x_1 + 6x_2 \leq 360$.

5. **Objective Criterion**

The criterion is based on maximizing profits. The two existing activities provide profits:

– The production of sweet butter, x_1, provides a unit profit of \$10 (profit per unit of the variable).
– The production of raw butter, x_2, provides a unit profit of \$15 (profit per unit of the variable).

Profit expression: $10x_1 + 15x_2$.
The objective function would be Max $10x_1 + 15x_2$.

Complete Model

$$\text{Max } 10x_1 + 15x_2$$

s.t.

$$3x_1 + 3x_2 \leq 210$$

$$3x_1 + 6x_2 \leq 360$$

$$x_1 \geq 0, x_2 \geq 0$$

Illustration 2.2 (Source: Meléndez 2019): Sale of Batches
A department store has 200 shirts and 100 trousers from the previous season. They launch two batch offers, A and B. The Offer A consists of a batch of one shirt and one trousers, which is sold at \$30; Offer B consists of a batch of three shirts and one trousers, which is sold for \$50. They do not want to launch less than 20 batches of Offer A. On the other hand, they also have the option of transferring pants to an outlet for a price of \$18/trousers, agreeing to pay a fee to the outlet of \$80 if they transfer more than 50 units. The objective is to maximize the profit.

1. **Elements**

Analyzing the statement, we can identify as participating elements:

• Shirts: The numerical information associated with this element in the system is:

  – Stocks of Shirts = 200.

As we will justify in the chapter dedicated to Elements, it would be an error in this case to consider each shirt individually as an element:

- Number of shirts in a <u>Batch Offer A</u> = 1

Here is something that we will also analyze in the next chapter – numerical characteristics that can be associated with more than one element:

- Number of shirts in a <u>Batch Offer B</u> = 3

- Trousers: Similar to the previous element. The associated information is:
  - Stocks of trousers = 100
  - Number of trousers in a Batch Offer A = 1
  - Number of trousers in a Batch Offer B = 1
  - The transfer value to the Outlet = $18
  - Maximum number of trousers sold to the outlet without paying fee = 50
  - Fee for transferring trousers to the outlet = $80

- Batch Offer A: The associated information is:
  - Number of shirts = 1

  As we have already mentioned, this feature is also associated with the Shirts element:

- Number of trousers = 1
- Sale price = $30
- The minimum number of batches = 20

- Batch Offer B: It has the same characteristics as Batch Offer A, except the minimum number of batches, but with different values.
- Outlet:
  - Sale price of trousers to the outlet = $18

  Since the system we model is associated with the department stores, it is more logical to use descriptions regarding the system, not with respect to each element. For this reason, we use the name of the attribute "Sale price" and not Purchase price, despite the fact that the Outlet buys trousers.

- Maximum number of trousers sold to the outlet without paying fee = 50
- Fee for transferring trousers to the outlet = $80

2. **Decision Activities**

To recognize the decision activities, it is necessary to explore the actions that occur in the system.

A department store has 200 shirts and 100 trousers from the previous season. They ***launch two batch offers****, A and B. The Offer A consists of a batch of one shirt and one trousers, which* ***is sold*** *at \$30; Offer B consists of a batch of three shirts and one trousers, which* ***is sold*** *for \$50. They do not want to launch less than 20 batches of Offer A. On the other hand, you also have the option of* ***selling trousers to an outlet*** *for a price of \$18/trousers, agreeing* ***to pay a fee*** *to the outlet of \$ 80 if they transfer more than 50 units. The objective is to* ***maximize the profit****.*

The actions of launching batches and selling batches can be considered equivalent in the system. They refer to the same process.

Launch Batches A and B

It is an action associated with both the Batch Offer A element and the Batch Offer B element, independently. This generates two decision activities:

– Launch Batch Offer A
– Launch Batch Offer B

The fact that the action corresponds to a decision activity is in the analysis of the action, with respect to whether it has a known or unknown value, calculated or not calculated. The first should be to examine what kind of value that activity has. If we proposed a binary value, we could not determine the number of Offers A and B launched. If instead we assign an integer value, the semantics of the action should refer to the number of batches that are launched. This is something indeterminate in the system; we do not know the number of batches that we are going to launch, nor can we calculate it from other decision-making activities. This type of value would determine the decision variables of the problem.

Finally, it would not make sense to use a continuous value since the batches are discrete elements.

Therefore, the decision activities associated with launching batches would give rise to two variables:

x_{A}: Number of **Batch Offer A launched**
x_{B}: Number of **Batch Offer B launched**

For this same decision activity, the statement also uses the verb sell. It is common in many descriptions to use different verbs to express the same action.

Transfer Trousers to the Outlet

We are faced with an action related to the element Trousers and the Outlet element jointly. Again, the value that we can associate with this action is an Integer value, and the variable that is generated could be defined as:

x_{PO}: Number of **Trousers transferred** to the **Outlet**.

Pay a Fee

It cannot be considered a decision activity because its value is obtained by calculation with respect to other decision activities.

Maximize the Profit

It is about the objective function. It will correspond to a variable function of the problem. It is therefore not an action that can be considered a decision activity.

3. **Calculations**

As mentioned, there is a cost associated with a calculation, in this case a logical calculation. The value of paying the fee of \$80 will be defined by a logical function or proposition.

For the modelling of a logical calculation, it is necessary, first of all, to define the variable that will collect the result of the calculation. In this case it is a calculation with binary value (= 1 is to pay the fee; = 0 is not to pay the fee).

Let the variable $\alpha = 1$ if we pay the fee; 0 otherwise.

And secondly, it is necessary to state the logical proposition that defines the calculation values:

"If the number of trousers sold to the Outlet is greater than 50, we pay a fee of \$80. In another case, we do not pay anything."

That would be mathematically expressed as:

IF $x_{\mathrm{PO}} > 50$, THEN $\alpha = 1$; IF $x_{\mathrm{PO}} \leq 50$, THEN $\alpha = 0$.

Chapter 5 is devoted to the calculations of a system, and in Chap. 6 we will learn to model logical propositions. Specifically, this proposition could be modelled with a single constraint:

$x_{\mathrm{PO}} \leq 100\alpha$

4. **Specifications**

From the statement and the elements of the problem, we extract the following specifications:

An imposed rule is easily identified: "You do not want to launch less than 20 batches of Offer A."

This specification refers to the activity of launching batches of Offer A. It imposes a lower bound to the value that the variable x_{A} can take:

$x_{\mathrm{A}} \geq 20$

The elements Shirts and Trousers have data of availability or capacity and then must generate specifications that ensure that the consumption of that availability in the system does not exceed the availability data:

Regarding the Shirts:
Capacity = 200
Consumption:
x_{A} (Launch Batch Offer A) consumes one unit (one shirt) for each unit that the variable takes (each batch that is sold).
x_{B} (Launch Batch Offer B) consumes three units for each unit that the variable takes.

Constraint generated: $x_A + 3x_B \leq 200$
Regarding the Trousers:
Capacity = 100
Consumption:
x_A (Launch Batch Offer A) consumes one unit for each unit that the variable takes.
x_B (Launch Batch Offer B) consumes one unit for each unit that the variable takes.
x_{PO} (Sell Trousers to the Outlet) consumes one unit for each unit of the variable.
Constraint generated: $x_A + \mathrm{x}_B + x_{PO} \leq 100$

5. **Objective Criterion**

The criterion is based on maximizing the benefit. Three existing activities provide income and a logical calculation provides cost.

The Offer A batches, x_A, provide a unit income of \$30 (profit for each unit sold).
The Offer B batches, x_B, provide a unit income of \$50.
The transfer of trousers, x_{PO}, provides a unit income of \$18.
Pay the fee: α provides a unit cost of \$80.
Profit expression: $10x_A + 15x_B + 18x_{PO} - 80\alpha$.
The objective function would be Max $10x_A + 15x_B + 18x_{PO} - 80\alpha$.

Complete Model

$$\text{Max } 10x_A + 15x_B + 18x_{PO}$$

s.t.

$$x_{PO} \leq 100\alpha$$

$$x_A \geq 20$$

$$3x_A + x_B \leq 200$$

$$x_A + x_B + x_{PO} \leq 100$$

$$x_A \geq 0, x_B \geq 0, x_{PO} \geq 0 \text{ integers; } \alpha \text{ binary}$$

References

Aracil, J, & Gordillo, F. (1997). Dinámica de Sistemas. V 168 Alianza Universidad Textos. Alianza editorial.

Castillo, E., Conejo, P., & Pedregal, P. (2002). *Formulación y resolución de modelos de programación matemática en ingeniería y ciencia*. Universidad de Castilla La Mancha Ediciones.

García Sabater, J. P. (2015). http://personales.upv.es/jpgarcia/LinkedDocuments/modeladomatematico.pdf. Modelado y Resolución de Problemas de Organización Industrial mediante. *Programación Matemática Lineal. Accessed May, 2020.*

Klement, K. C. (2006). Propositional logic. In J. Fieser, & B. Dowden (Eds.), *Internet encyclopedia of philosophy*

Meléndez, I. (2019). https://www.monografias.com/trabajos96/distribucion-redes-administracion-proyectos/distribucion-redes-administracion-proyectos.shtml#bibliograa. La distribución de redes y la administración de proyectos. Accessed June 2019.

Chapter 3
The Elements of a System

3.1 Introduction

The elements are the individual actors or entities that participate in the system. They are all the inputs that the system has, the resources it uses, and the outputs it produces. They can be of any nature: materials, tools, personnel, places, objects, etc. Also considered as elements are the typologies that could be defined on other elements, and even resources of space or time.

The elements are distinguished by participating in the activities that occur in the system and/or providing information on the specifications and objective of the activities.

The data associated with the elements will be referred to as attributes of the element. The data are the determined values that the system has, which will refer to continuous or integer magnitudes and logical properties that will be collected with binary values. The attributes and their values will influence the quantitative nature of the elements.

A clear and exhaustive description of the system is necessary for the correct identification of the elements. Sometimes this description can identify elements that later have no participation in the optimization problem raised. Despite this, its preliminary identification will not hinder the construction of the model.

The system itself can always be considered as an element of itself. We will use it when there are data or specifications that can only be attributed to the system itself.

The identification of elements is an important phase because an incorrect definition of the elements of a problem will prevent a correct formulation of it. As we will analyze throughout the chapter, the list of elements of a system does not have to be unique. Sometimes, it is possible to configure the list of elements of a system in more than one way. This may lead to loss of efficiency of the model, but its representation will be admissible.

J. M. García Sánchez, *Modelling in Mathematical Programming*, International Series in Operations Research & Management Science 298,
https://doi.org/10.1007/978-3-030-57250-1_3

Table 3.1 List of elements of Illustration 3.1

Elements
Supplier 1
Supplier 2
Supplier 3
Supplier 4
Company (system)
Product
Market

On the other hand, once the elements are identified, we can organize them into sets. Sets will consist of elements of the same nature and function, with identical attributes with respect to its definition. This will facilitate an abbreviated and abstract formulation of the model.

What is an element in a system?

- Any physical or conceptual entity that participates in the system: people, places, machines, tools, parts, or products
- Time periods in which decisions are made or on which specifications are raised
- Any type defined in the system on other elements
- Phases, processes, modes, or states that occur within the system

What is not an element in a system?

- Any measurement concept (kilo, gram, hour, minute, liter, euro, unit, etc.)
- Any action that occurs in the system that is not collected as a process

Let's take a look at some basic examples:

Illustration 3.1
A company has four suppliers of a product in the market. Each supplier has its own price policy, discounts, and delivery times.The company has to decide on the units that it will buy from each supplier to satisfy the demand of the market.

Without giving data on prices, discounts, times, and any other information that the system uses, it is only possible to obtain the participating elements using the description made (Table 3.1).

Product prices, discounts, or delivery times are data from the suppliers.

Illustration 3.2
There is a set of 100 objects, each with a weight, a height, and a width. There is also a set of 25 shelves, each formed by a set of 10 sections. Each section has a width, a height, and a maximum supported weight. It is about assigning objects to shelving sections regarding the width, height, and weight of the sections, maximizing the number of objects to be placed.

Again, without having to give the weight, height, and width data of both objects and sections, the list of elements could be defined as (Table 3.2).

Table 3.2 List of elements of Illustration 3.2

Elements	
Object 1	
Object 2	
. . .	
Object 100	
Shelving 1	Section 1_1
	Section 1_2
	. . .
	Section 1_10
Shelving 2	. . .
. . .	. . .
Shelving 25	Section 25_1
	Section 25_2
	. . .
	Section 25_10
System	

Table 3.3 List of elements of Illustration 3.3

Elements
Parts
M1 machines
M2 machines
M3 machines
Market
Mode 1
Mode 2
Workstation 1
Workstation 2
Day
Company (system)

It is necessary to identify each object, shelf, and section of each shelving as an element (Table 3.3).

Illustration 3.3
There is a system that produces parts. In the market there are three machine models (M1, M2, M3) for production. There are two production modes and two workstations where any mode can be installed in each. The first mode needs to produce parts, two M1 machines and one M2 machine. The second mode needs to produce one M2 machine and one M3 machine. Each mode produces at a rate that translates into number of parts per hour, 45 and 54 respectively. The company wants to plan the production of 3000 parts during the 18 hours a day, aiming to minimize the purchase costs of the machinery.

Note that we have considered the parts as a single element and not each of the 3000 parts to be manufactured. On the other hand, we have considered each workstation individually. This aspect will be explored in greater depth throughout the chapter. The day that has the data of the available hours is included as an element. The hours are considered a unit of measurement of the time available.

3.2 Data of Elements

The data are specific properties or attributes of the element. A property of an element is an own aspect of it that defines it in the system. Elements can have properties whose value is not determined because they have to be defined in the optimization process and properties of determined value that are the data of the element in the system. The data must therefore have a numerical value assigned.

The data are used in the system specifications and in the optimization criteria. Below is an example that differentiates determined and undetermined properties:

Illustration 3.4
There is a system that manufactures containers. The containers have a height of 4 meters and a width of 3 meters. In the system it is necessary to manufacture ten containers and to determine the length of each container that we manufacture (using a maximum length of 10 meters), so that we can accommodate a set of 50 pallets that have a known width, height, and length.

In this system, we would consider the 10 containers and each of the 50 pallets as elements. The properties of each container are its height and width, but the length is not a attribute because its value is not determined. Height and width are data.

The pallets are also elements of the system and their properties are data because they are determined.

Below is a basic example of the data of an element:

Illustration 3.5

...

The product we produce has a profit of \$10/unit. 1000 units are demanded.

...

In this system, the Product element would have two attributes:

Profit = 10 (\$/unit)
Demand = 1000 (units)

Most data express quantities that correspond to continuous or integer magnitudes. However, logical values and links between elements are also incorporated as data. These relationships between elements are represented by binary values (1/0) with a meaning of true/false. This type of attributes is presented when there are types of elements defined explicitly as elements. It is also used to represent the ability of an element to perform a certain action or not, or to identify logical relationships between elements. The types of data according to value are presented in Sect. 3.2.3.

On the other hand, when systems are represented graphically, elements and data must be extracted from the graph. The data in these cases usually represent the graphic relationship between the elements represented. Section 3.6 will examine these aspects.

The importance of identifying the data of the elements is in the subsequent implementation we carry out of the model with some language more than in the design of the mathematical model itself. In spite of this, structuring the information of the system will help us identify specifications and control the use of all the characteristics of the elements of the system.

An attribute has two components:

- A definition scheme: The scheme will have the following information:
  - A name
  - A parameter with which to identify it in the model
  - The belonging
  - The type of value
- A numerical value.

Next, we will develop the concepts of belonging (Sects. 3.2.1 and 3.2.2) and type of value (Sect. 3.2.3). In Sect. 3.2.4 we will define a formal representation, and in Sect. 3.2.5 we will describe the creation of data from existing ones.

3.2.1 Belonging of the Data

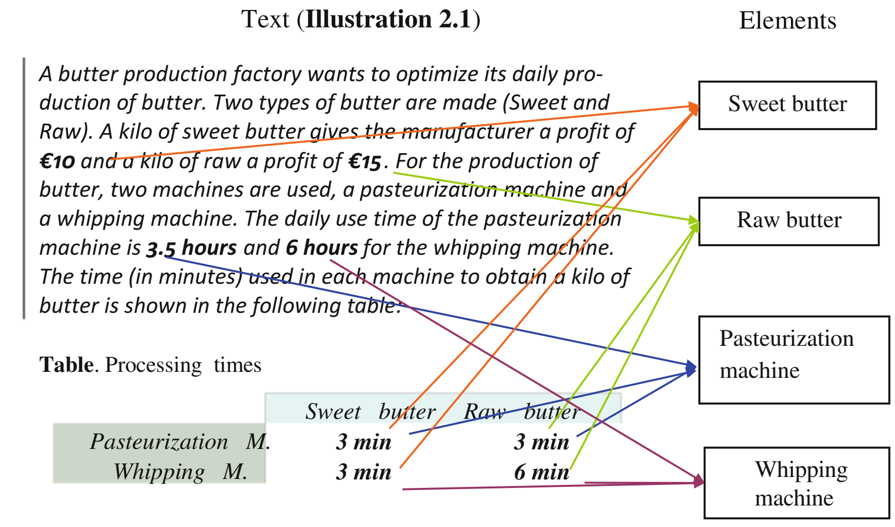
The data refer to the values that the elements of the system have. Regarding the belonging of the data, they can be of two types:

Own Attribute: The attribute and its value are only property of an element. The attribute does not refer to any other element.

Shared Attribute: The attribute relates elements. Its value is obtained from the association between several elements of the system. Logically, the shared data are assignable to all the elements that share it. The value of the attribute is not distributed by a function between the elements that share it, but it is fully attributable to each of the elements that share it.

Let's take a look at the data and their belonging in our header example. Highlighting the values that appear in the text in bold, we can graphically establish the relationship between the four elements of the problem (Fig. 3.1).

If we examine the attributes associated with the pasteurization machine (Table 3.4), usage time is an attribute of its own, not related to any other element. However, the attribute on the time used by sweet butter in the pasteurization machine, 3 min, is data that should also be associated with sweet butter. It is



Text (Illustration 2.1)
Elements
A butter production factory wants to optimize its daily production of butter. Two types of butter are made (Sweet and Raw). A kilo of sweet butter gives the manufacturer a profit of €10 and a kilo of raw a profit of €15. For the production of butter, two machines are used, a pasteurization machine and a whipping machine. The daily use time of the pasteurization machine is 3.5 hours and 6 hours for the whipping machine. The time (in minutes) used in each machine to obtain a kilo of butter is shown in the following table:
Table. Processing times
Sweet butter
Raw butter
Pasteurization M.
3 min
3 min
Whipping M.
3 min
6 min
Sweet butter
Raw butter
Pasteurization machine
Whipping machine

Fig. 3.1 Belonging of attributes of Illustration 2.1

Table 3.4 Attributes of pasteurization machine

Pasteurization machine
Pasteurization machine usage time = 3.5 h/day
Time used by 1 kg of Sweet butter in Pasteurization machine = 3 min
Time used by 1 kg of Raw butter in Pasteurization machine = 3 min

Table 3.5 Attributes of sweet butter

Sweet butter
Profit = $10
Time used by 1 kg of Sweet butter in pasteurization machine = 3 min
Time used by 1 kg of Sweet butter in whipping machine = 3 min

therefore a shared attribute. The same would apply to the time that raw butter uses. The whipping machine would have the same data structure (Table 3.5).

Regarding Sweet butter, the data of time are the same attributes that have been indicated in the machines. The profit is an attribute with its own value. Raw butter has the same data.

In the definition of a system, there are values whose association with the elements can be direct and simple, and in other cases it is more difficult to establish its belonging. This happens when the values are directly assignable to activities and system calculations. Reference is made to values that can take variables or functions of variables of the problem or values that impose variables bounds. In most models, we can obviate the declaration of these values and use them directly in the specifications of the system without leading to a bad design of the model. It must be borne in mind that extracting all the data of the elements of a system is a task that requires thinking more about the implementation than the elaboration of the model. Despite this, all values could be defined as data of some element, even if that element is the system itself.

Let’s illustrate some examples of this type of values. To do this, we are going to incorporate information into the butter production system.

Illustration 3.6
The total production of sweet and raw butter should be a maximum of 80 kg.

The value of a calculation is being established, which is the total of sweet and raw butter. In this case, the maximum production would correspond to an attribute of the system.

Illustration 3.7
The production of raw butter is prohibited if more than 100 kg of sweet butter is produced.

The 100 kg data refers to a value of the production of raw butter. Production is a system activity and therefore will be associated with a variable. That data of 100 kg is a possible value of the activity. In any case, this value could be defined as an attribute of raw butter.

3.2.2 *Primary Element in a Shared Attribute*

When a shared attribute refers to a value of an intrinsic property of one of the elements that share the attribute, we will say that this element is the primary element in the relationship with the shared attribute. The rest of the elements have a secondary role. The clearest case is when the attribute refers to a quantity value of one of the elements.

On the other hand, there may also be shared attribute that does not refer to quantity values of a single element, but refers to a property shared by all the elements equally, that is, there is no primary element in the relationship. Generally, this occurs whenever the attribute has a binary value. The binary value usually expresses a shared property.

Let us see some examples of the identification of primary elements:

Illustration 3.8: Sale of Batches (Illustration 2.2)
A department store has 200 shirts and 100 trousers from the previous season. They launch two batch offers, A and B. The Offer A consists of a batch of one shirt and one trousers, which is sold at $30; Offer B consists of a batch of three shirts and one trousers, which is sold for $50. They do not want to launch less than 20 batches of Offer A. On the other hand, they also have the option of transferring pants to an outlet for a price of $18/trousers, agreeing to pay a fee to the outlet of $80 if they transfer more than 50 units. The objective is to maximize the profit (Table 3.6).

Table 3.6 Table of Elements of Illustration 3.8

Element	Data		
	Name	Belonging	Primary element
Shirts	Stocks of Shirts = 200	Own	
	Number of shirts in a Batch Offer A = 1	Shared	Shirts
	Number of shirts in a Batch Offer B = 3	Shared	Shirts
Trousers	Stocks of Trousers = 100	Own	
	Number of Trousers in a Batch Offer A = 1	Shared	Trousers
	Number of Trousers in a Batch Offer B = 1 = 1	Shared	Trousers
	The transfer value to the Outlet = $18	Shared	–
	Maximum number of trousers sold to the outlet without paying fee = 50	Shared	Trousers
	Fee for transferring trousers to the outlet = $80	Shared	–
Batch Offer A	Sale price = $30	Own	
	Number of shirts in a Batch Offer A	Shared	Shirts
	Number of Trousers in a Batch Offer A	Shared	Trousers
Batch Offer B	Sale price = $30	Own	
	Number of shirts in a Batch Offer B	Shared	Shirts
	Number of Trousers in a Batch Offer B	Shared	Trousers
Outlet	The transfer value to the Outlet	Shared	–
	Maximum number of trousers sold to the outlet without paying fee	Shared	Trousers
	Fee for transferring trousers to the outlet	Shared	–

This example was already modelled in Chap. 2.

3.2.3 *Type of Value of the Data*

Every attribute has a numerical value associated with it. The type of associated value can be:

1. **Continuous or integer quantity**

This refers to some continuous or integer magnitude. If it is continuous, we can specify the unit of measure of the value. If it is integer, it is measured in number of units.

Example: Usage time of the pasteurization machine = 180 minutes.
Example: There is a manufacturing system of part P of which there is a stock of 28 units. Part P would be an element with an attribute. Stock = 28 units.

The values of a system that express quantities are usually easily extractable from the descriptive texts, although there are exceptions. Sometimes, describing quantities using words instead of numbers can mislead the modeller, especially when we have to distinguish between indeterminate articles (indefinite) or a numeral:

"A product needs one machine A and two B tools":

A product: indeterminate article
One machine A: numeral = 1 ⇒ Attribute
Two B tools: numeral = 2 ⇒ Attribute

In this case, one machine A and two B tools would be attributes.

The way to identify whether it is an indeterminate article or a numeral is in proving if its substitution by the word "each" is viable:

"Each product needs one A machine and two B tools."

In the same way, it is possible to discard numbers that refer to quantities of elements defined in the problem:

"The factory has four machines: M1, M2, M3, and M4."

If each machine is going to be an element, the quantity of four machines could be used as an attribute of the system to quantify the number of machines, although it will rarely be used.

2. **Ordinal**

The attribute has an integer value that refers to an ordinal of a list of options reflected or not as elements. This type of value is always optional in a problem. We can incorporate each option as a binary attribute and even incorporate the options to the list of system elements and convert the ordinal value to a binary value that expresses the link between the elements.

On the other hand, the use of an ordinal value requires a simple association between the element owner of the attribute and the list of elements that the ordinal identifies. If the association is multiple, we would need to incorporate more ordinal data, so it is more structured to store the information as binary. In the case of simple association, the use of ordinal values instead of binary values is usually more efficient, although it depends on each system.

Let's take a look at an illustration that reflects these ideas.

Illustration 3.9
There is a set of 20 pieces. Each piece has a color (white, red, or black) and a weight. There is a set of ten sections where the pieces will be placed. Each section has a supported weight capacity. In a section you cannot mix pieces of different color. It is about using the least number of sections.

The elements identified are each piece and each section of unitary type. Each piece has as attribute the weight and the color. Each section has the capacity of supported weight as attribute.

Table 3.7 Table of data of Illustration 3.9

Element	Data	Type
Piece 1	Weight (continuous)	Own
	Colour (ordinal:1,2,3)	Own
...		
Piece 20	Weight (continuous)	Own
	Colour (ordinal:1,2,3)	Own
Section 1	Supported weight (continuous)	Own
...		
Section 10	Supported weight (continuous)	Own

The color of the pieces can be included in a list where each color is referenced by an ordinal: white (1), red (2), and black (3). The value of the color attribute in each piece will have a value between 1 and 3 (Table 3.7).

This representation would not have been valid if the pieces had more than one color, that is, if there had been a multiple association of each piece with the list of colors. For that type of cases, we will use the binary value. I do not consider the option of creating several ordinal value data, although in some cases it could be viable.

3. **Binary**

Binary values are used to define properties of the element that are answered with a yes or a no.

The binary value is used to represent an attribute ordinal value in a flexible way, allowing for the representation of multiple associations. In this case, it may be convenient to have options of the list identified as elements, although it is not mandatory.

The binary value is the value needed for data in which its value identifies another element or elements. This is solved by establishing a shared attribute between both elements. Therefore, binary value is used in attributes that represent links of belonging or compatibility between elements.

Let's see how the list of elements of Illustration 3.9 would be using binary value, in two steps: first, without defining the binary attribute as an element and, second, making the attribute explicit as an element (Table 3.8).

In Table 3.8 colors are defined as data, but they are not identified as elements of the system. This is feasible when there are no decision activities where the color of the pieces participates or specifications that fall on the colors. In any case, the color of each piece is represented in three binary attributes, so it would allow pieces of various colors.

In Table 3.9 we incorporate the three colors as elements. This means that the color attributes become shared between the piece and the corresponding color.

Next, we introduce a couple more illustrations regarding binary attributes.

Table 3.8 Table of attributes of Illustration 3.9 with binary values not shared

Element	Data	Type
Piece 1	Weight (continuous)	Own
	White_Colour (Binary)	Own
	Red_Colour (Binary)	Own
	Black_Colour (Binary)	Own
...	...	...
Piece 20	Weight (continuous)	Own
	White_Colour (Binary)	Own
	Red_Colour (Binary)	Own
	Black_Colour (Binary)	Own
Section 1	Supported weight (continuous)	Own
...	...	...
Section 10	Supported weight (continuous)	Own

Table 3.9 Table of attributes of Illustration 3.7 with binary values shared

Element	Data	Type
Piece 1	Weight (continuous)	Own
	White_Colour_Piece1 (Binary)	Shared
	Red_Colour_Piece1 (Binary)	Shared
	Black_Colour_Piece1 (Binary)	Shared
...	...	...
Piece 20	Weight (continuous)	Own
	White_Colour_Piece20 (Binary)	Shared
	Red_Colour_Piece20 (Binary)	Shared
	Black_Colour_Piece20 (Binary)	Shared
Section 1	Supported weight (continuous)	Own
...	...	...
Section 10	Supported weight (continuous)	Own
White Color	White_Colour_Piece1 (Binary)	Shared
	...	...
	White_Colour_Piece20 (Binary)	Shared
Red Color	Red_Colour_Piece1 (Binary)	Shared
	...	...
	Red_Colour_Piece20 (Binary)	Shared
Black Color	Black_Colour_Piece1 (Binary)	Shared
	...	...
	Black_Colour_Piece20 (Binary)	Shared

Illustration 3.10

There is a city with five districts. There are two stores V1 and V2. V1 is located in District 3 and V2 in District 5.

Considering each district (Districts 1 to 5) and each store (V1, V2) as elements, there is a single attribute in the text, which is the location of the stores. If we were to give an integer value to the location of V1, we would have to use an ordinal value on

Table 3.10 Binary attribute *Location*

Location	*District 1*	*District 2*	*District 3*	*District 4*	*District 5*
Store 1	0	0	1	0	0
Store 2	0	0	0	0	1

the list of districts that identifies the district where the number 3 is located. For V2 the value would be 5.

Since the districts are elements, the most correct way to identify the attribute is with a shared binary value between the elements V1 and V2 and all the elements that were part of the possible ordinal values (Districts 1 to 5). The Location attribute would be shared by each store (V1, V2) with each district (Districts 1 to 5) assigning a value (1 = is located/0 = is not located). These values can be represented in a table (Table 3.10).

Illustration 3.11

There is a company that manufactures a parts model of the aeronautical industry. The company has designed ten work centers. The manufacturing of parts has three possibilities or modes of production:

- *Mode 1*
- *Mode 2*
- *Mode 3*

Work center 1 can only hold Mode 1, center 2 can only hold Modes 2 and 3, and the other centers can hold any mode.

Setting up a center has a cost of $F.

The production cost of each mode is C1 $/part, C2 $/part, and C3 $/part.

Elements:

- Each work center: Work Center 1 to Work Center 10
- Mode 1
- Mode 2
- Mode 3
- Parts

Attributes of Work Centers:

- Each Center has the following attributes:
- Cost: Own Attribute of continuous value.
- Compatibility Center/Mode: A Center can host a series of modes, 1, 2, or 3.

As some centers can have several modes, this precludes the use of ordinal data:

Compatible modes of Center 1 = 1
Compatible modes of Center 2 = 2, 3
Compatible modes of Center 3 = 1, 2, 3
Compatible modes of Center 4 = 1, 2, 3

Table 3.11 Binary attribute *Compatibility*

Compatibility	Mode 1	Mode 2	Mode 3
Center 1	1	0	0
Center 2	0	1	1
. . .	. . .	. . .	. . .

Table 3.12 Elements of Illustration 3.12

Element	Data	Type
Piece of furniture 1	Price (Continuous)	Own
	Assembled (Binary)	Own
. . .	. . .	. . .
Piece of furniture *n*	Price (Continuous)	Own
	Assembled (Binary)	Own

. . .

The solution is to set the attribute as binary and shared between each Center and each Mode. The values could be represented in a table as shown below (Table 3.11).

Each mode also has the Production Cost as an attribute. As there is a production cost per part, we consider it also attributable to the part element, and therefore it is shared with the part.

The natural way in which a binary attribute appears in a problem is in shared mode between elements, as in the illustrations that we have just shown. However, as we have already mentioned, it is possible to enter binary data that are own to an element because the information to which the attribute refers does not need to be explicit as an element of the system, because it does not participate in the decisions of the problem nor there are not specifications on the concept that represents the attribute. Let's see an example:

Illustration 3.12
There is a series of n pieces of furniture. From each piece of furniture, you know the price and a property about whether it is sold assembled or not.

The property "assembled piece of furniture" could be considered as a typology of the furniture elements and would be transformed into an element. On the other hand, we can leave it exclusively as an attribute of each piece of furniture. In this second case, the list of elements would be the following (Table 3.12).

3.2.4 Representation

We can design a framework in the form of a table to collect all the information associated with the definition of a attribute. In the table we will include the following information:

1. Name: name assigned to the attribute.

Table 3.13 Associated framework

Definition				
Name	Parameter	Type of value	Belonging	Value
Attribute name	Parameter name	C (unit)/I/O/B	W/S	–

Table 3.14 Framework associated with an element

	Data				
Element	Name	Parameter	Type of value	Belonging	Value
Element name	Attribute name	Parameter name	C/I/O/B	W/S	–
	. . .	. . .	. . .	. . .	. . .
. . .	. . .	. . .	. . .	. . .	. . .

2. Parameter: the chosen parameter that we will use in the model to identify the attribute.
3. Type of value: C (Continuous), I (Integer), O (Ordinal), B (Binary). In the case of Continuous, we can specify the unit of measure.
4. Belonging: W (Own element), S (Shared).

Finally, and as already mentioned, in addition to the definition, the attribute has a value that determines it, which we will also include in the table.

Table 3.13 shows the framework.

By inserting that framework in a table with the identification of each element, we could outline the information of the elements of a system (Table 3.14).

As an illustration for the first Table of Elements, we will represent the elements of our first example, the production of butter.

Illustration 3.13

Elements of the production of butter

For this first representation, we will indicate all the data of each element, regardless of whether the shared attribute have already been declared in another element.

As it is observed, there are attributes that allude to the same value. Logically, these are the shared attribute, assignable to more than one element. In the pasteurization machine, "Time used by 1 kg of sweet butter in pasteurization machine" is a property that relates to the pasteurization machine with sweet butter and therefore attributable to both elements. In the following, we will note the definition of a shared attribute in a single element, writing down the parameter in the rest of the elements that share it.

On the other hand, there are data with the same definition and different value ("Usage time" that appears associated with both the pasteurization machine and the whipping machine and "Profit" associated with the two types of butter). These are identical attributes in relation to their definition but each with its own value. The definition of data will be simplified when we group the elements in sets (Table 3.15).

Table 3.15 Table of Elements of the production of butter

Element	Data				
	Name	Param	Type	Belonging	Value
Pasteurization machine	Usage time	TP	C (minutes)	W	180
	Time used by 1 kg of sweet butter in pasteurization machine	TPS	C (minutes)	S	3
	Time used by 1 kg of raw butter in pasteurization machine	TPA	C (minutes)	S	3
Whipping machine	Usage time	TW	C (Minutes)	W	240
	Time used by 1 kg of sweet butter in whipping machine	TWS	C (minutes)	S	3
	Time used by 1 kg of raw butter in whipping machine	TWA	C (minutes)	S	6
Sweet butter	Time used by 1 kg of sweet butter in pasteurization machine	TPS	C (minutes)	S	3
	Time used by 1 kg of sweet butter in whipping machine	TWS	C (minutes)	S	3
	Profit	BS	C (euros/kg)	P	10
Raw butter	Time used by 1 kg of raw butter in pasteurization machine	TPA	C (minutes)	S	3
	Time used by 1 kg of raw butter in whipping machine	TWA	C (minutes)	S	6
	Profit	BA	C (euros/Kg)	P	15

3.2.5 *Inclusion of Calculated Data*

To facilitate the modelling of certain specifications, it is useful to calculate additional data from those that already exist. It is a task that appears when we are going to model the specifications of the problem. Let us see an illustration.

Illustration 3.14
In butter production we want to know as an attribute which butters need more than 8 full minutes to produce a kilo.

In this case we must associate a binary attribute with each butter that responds to this calculation. That attribute would be own to each type of butter (Table 3.16).

Table 3.16 Inclusion of an attribute calculated

Element	Data				
	Name	Param	Type	Belonging	Value
Sweet butter	Time used by 1 kg of sweet butter in pasteurization machine	TPS	C (minutes)	S	3
	Time used by 1 kg of sweet butter in whipping machine	TWS	C (minutes)	S	3
	Profit	BS	C (euros/kg)	W	10
	Production of more than 8 min	PS	B	W	0
Raw butter	Time used by 1 kg of raw butter in pasteurization machine	TPA	C (minutes)	S	3
	Time used by 1 kg of raw butter in whipping machine	TWA	C (minutes)	S	6
	Profit	BA	C (euros/kg)	W	15
	Production of more than 8 min	PA	B	W	1

3.3 The Quantitative Nature of the Elements

In this methodology each element is defined by its quantitative or measurable aspect, which we shall call quantitative nature. The quantitative nature of an element is expressed according to the following typologies:

Individual Element
The element represents an individual entity, whose discrete quantity in the system is the unit, the element itself.

An individual element is an element that the system deals with in a particular way and that in many cases, but not in all, is different from the rest of the elements of the system.

The elements of our base illustration are individual elements. Pasteurization machine or the whipping machine is clearly individual, just considering their quantity. There are one pasteurization machine and one whipping machine. Sweet butter and raw butter are also individual since the discrete amount of sweet butter or raw butter is one, there are no raw or sweet butter units. Logically, both raw and sweet butter will have a continuous amount in the system. Therefore, the individual elements will have a second typology depending on their properties.

Collective Element
The element represents a concept that has or will have a number of discrete units in the system, whether that amount is determined or undetermined. The element represents a collective entity that is quantified discretely. We will name items to each of those units. The items of a collective element are identical, that is, they have the same own data in the system, represented in the collective element.

The collective elements will always have a measurable character in the system, either in the decision activities or in the specifications. In the decision variables of integer value, a collective element must always participate.

According to the determination of the number of items of the collective element, we distinguish between:

Determined collective element

The quantity of the element is defined in the system.

Undetermined collective element

The quantity of the element is not determined in the system, but the system works with quantities of the element.

I. ***Individual Elements***

The individual elements have a second typology based on their data and continuous properties not determined. This second typology must be analyzed when the decision activities of the system are analyzed. Anyway, we can now define the typology, for which we need a series of concepts.

As we have already mentioned, the elements have properties that define the element in the system, and with respect to their determination, they can be:

– Data: Properties of known value.
– Properties of undetermined value: They are properties that allude to some characteristic of the element that is necessary to decide its value. With the decisions taken in the system, their values will be determined. We do not consider properties that can be obtained as a result of a calculation or function.

For the quantitative nature of individual elements, we are interested in their data and indeterminate properties of continuous value. The most common continuous properties in an element refer to the measurement of its length, volume, weight, time, position in space, or its scheduling time, in the case of tasks. As property, any continuous value intrinsic characteristic of the element can be considered.

Identifying these properties is usually simple since the element may have data that refers to that property, such as a cost per unit of measure, a bound for the value of the property, or any other information that makes us see the need to consider that property for the element.

Regarding continuous value data, two circumstances may occur:

– That the determined property cannot be used partially, that is, that we cannot divide the element with respect to that property. The property is always used in the system in a comprehensive way.
– That we can make partial use of that determined property. In turn, partial use may be due to continuous decisions or based on logical or integer decisions. If the attribute is partially used for logical or integer decisions, the attribute will be measured in the specifications but will not be measured in the decisions, and therefore we do not consider it as measurable data.

According to these concepts, we distinguish between:

I.1. Unitary individual element: An individual element is considered unitary if it has not any property or own data in the system capable of being measured in a continuous way at the decisions of the problem. A unitary element does not have any indeterminate property, and if it has any continuous attribute, this attribute is used in a comprehensive way, or its partial use is not based on continuous decisions. If the element has shared measurable data, it is never the primary element in the belonging of the data.

I.2. Measurable individual element: These are elements that have indeterminate continuous properties or data of continuous value that can be used partially through continuous decisions. When there is measurable data shared between elements, the measurable character is imputed to the primary element. An individual element can be measurable by more than one concept.

We can identify indeterminate properties in an element, whose value is not obtained from the decision activities, but simply the value of the property is obtained by a function of other activities of decision of the problem, and the measurable character of that property occurs in the specifications. Likewise, the partial use of an attribute can be made directly in the decisions of the system or in a function with other defined decision activities. When the measurement is not carried out in the decision activities, the measurable character of the element is not important because for the methodology the measurable character acquires meaning in the decision activities. The identification of this second typology in the individual elements can be complex at times and requires the analysis of the decision activities to certify their validity. However, identifying the measurable aspect can help visualize different valid model configurations. There are systems that support more than one configuration of the quantitative nature of its elements, depending on how the functions of the elements in the system are described, although generally only one of them is usually the most efficient. This will be seen in detail in Chap. 7, although examples will also be seen in Chap. 6. Therefore, the quantitative nature of individual elements is an auxiliary tool in this phase of the methodology, although the main quantitative nature, individual or collective element, must be determined at the element identification.

From all this, examples are presented below, and many illustrations will be analyzed throughout the book, as we go deeper into the use of all the components.

II. ***Collective Elements***

The collective elements will always have a measurable character in the system, either in the decision activities or in the specifications. In the decision variables of integer value, a collective element must always participate. The integer data of an individual element should always be shared with the collective element to which the units refer. And that collective element should be primary in that relationship and support the discrete measurable character of the data. Collective data can have continuous data referring to each item, but its use can never be partial, but a comprehensive use of the data in the system.

Table 3.17 Typologies of elements with respect to the quantitative nature

Type of element	Notation
I. Individual	I
I.1. Unitary individual	I_U
I.2. Measurable individual	I_M
II. Collective	C
II.1 Determined collective	C_D
II.2 Undetermined collective	C_I

Table 3.18 Types of elements in the example of butter production

Element	Quantitative nature type
Sweet butter	Measurable individual
Raw butter	Measurable individual
Pasteurization machine	Unitary individual
Whipping machine	Unitary individual

Table 3.17 summarizes the typologies regarding the quantitative nature of an element.

The column notation will be added to the format of the Table of Elements defined in Sect. 3.2.4 as QN (quantitative nature). From now on, the individual elements without capacity to be measurable in decision activities will be represented with IU, the individual elements with capacity in the system to be measured in decisions will be represented with the IM notation and the determined and indeterminate collective elements with CD and CI, respectively.

In our base example, without entering into decision activities even if it is a problem already modelled, the four elements are individual. Sweet butter and Acid butter are two elements with an undetermined amount, since the system does not know how much butter will be produced. It is obvious that they will be measurable elements because of having undetermined the quantity property.

The summary of the type of elements of the butter production problem is presented in Table 3.18.

The machines are also individual and among its information there is a continuous attribute, the usage time, on which a partial use can be made in the system. Its partial use can be obtained through a function with respect to the butter produced, but it is not necessary to determine it by decision activity. Recall the formulation:

Decision activities:

x_1: Kilos of sweet butter that are produced (we measure the sweet butter element)
x_2: Kilos of sweet butter that are produced (we measure the acid butter element)

Specifications:

Measurement of the usage time in the pasteurization machine: It occurs in the specification of time consumption control: $3x_1 + 3x_2 \leq 210$.

Measurement of the time of use in the whipping machine:

Equivalently, measurement of the time of use in the whipping machine: $3x_1 + 6x_2 \leq 360$
Objective function: Max $10x_1 + 15x_2$

In this system we could reach to define an equivalent model in which the system's decisions were to measure the time of use of each machine with each type of butter, although it would be larger. In other words, the machines would become measurable elements in decisions and butter production would become calculations. It is not usual, but as mentioned, there are systems that support more than one configuration of its elements. Let's see how it would be:

Decision activities:

$y_{1\mathrm{S}}$: Assigned time in the pasteurization machine for sweet butter
$y_{1\mathrm{A}}$: Assigned time in the pasteurization machine for acid butter
$y_{2\mathrm{S}}$: Assigned time in the whipping machine for sweet butter
$y_{2\mathrm{A}}$: Assigned time in the whipping machine for acid butter

Specifications:

Control of the consumption of available time:

Pasteurization machine: $y_{1\mathrm{S}} + y_{1\mathrm{A}} \leq 210$
Whipping machine: $y_{2\mathrm{S}} + y_{2\mathrm{A}} \leq 360$

The measurement of the quantity of butter is produced in the specifications through a lower bound calculation (the calculations of a system will be studied in Chap. 5).

Measurement of sweet butter production:

x_1: Kilos of sweet butter that are produced

Constraints defining the calculation:

$$x_1 \leq \frac{y_{1S}}{3}$$
$$x_1 \leq \frac{y_{2S}}{3}$$

x_2: Kilos of acid butter that are produced

Constraints defining the calculation:

$$x_2 \leq \frac{y_{1A}}{3}$$
$$x_2 \leq \frac{y_{2A}}{6}$$

Objective function: Max 10x1 + 15x2

To assign the quantitative nature correctly for individual elements, we need to analyze the decision activities of the problem. In any case, knowing the system perfectly, we can also glimpse the measurable nature of the individual elements. Let us look at Illustrations 3.2 and 3.3 described at the beginning of the chapter, introducing the quantitative nature in the table.

Illustration 3.15: Quantitative Nature of the Elements of Illustration 3.2
There is a set of 100 objects each with a weight, a height, and a width. There is also a set of 25 shelves each formed by a set of 10 sections. Each section has a width, a height, and a maximum supported weight. It is about assigning objects to shelving sections, respecting the width, height, and weight of the sections, maximizing the number of objects to be placed.

All the elements have been defined individually since they are identified in the system with their own values in their data, which makes them different from the rest. Each object has its own weight, height, and width. On the other hand, objects are defined as unitary because the system treats them in an integral way; there is no measurement of their continuous data (Table 3.19).

The shelves do not have continuous data and are different according to the data related to the sections that compose it, so they are defined as individual and unitary.

Each section is different and therefore it is also an individual element. They are also unitary because in the system the measurement of its supported weight, used height, or used width is based on logical decisions; the decision of assigning each object is logical (each object is assigned to each section or not).

Illustration 3.16: Quantitative Nature of the Elements of Illustration 3.3
There is a system that produces parts. In the market there are three models of machines (M1, M2, M3) for production. There are two production modes and two workstations where any mode can be installed in each. The first mode needs to produce parts, for two M1 machines and one M2 machine, and the second mode one M2 machine and one M3 machine. Each mode produces at a rate that translates into number of parts per hour, 45 and 54 respectively. The company wants to plan the production of 3000 parts, during the 18 hours a day, aiming to minimize the purchase costs of the machinery.

In this system there are not only individual elements. The parts are not treated individually and each of them is identical. Therefore, it is a collective and determined element (3000 parts).

On the other hand, there are three models of machines. The three models of machines are different from each other, so there is no kind of collective consideration for them. Each model is referenced in the system by quantities or numerals, so it is also collective elements whose quantity a priori is undetermined in the system. Keep in mind that the system does not work with the three machine models in a conceptual way, but in a quantifiable way, because the description refers to units of each model at all times.

The two modes of production are individual elements that have no own data that can be measured, so their nature is unitary. They have data shared with the models of machines, the number of machines that each mode needs, and the rate of production

Table 3.19 Elements of Illustration 3.15

Element	QN	Data				
		Name	Parameter	Type of value	Belonging	Value
Object 1	I_U	Weight	P_1	C	W	–
		Height	H_1	C	W	–
		Width	W_1	C	W	–
. . .						
Object 100	I_U	Weight	P_{100}	C	W	–
		Height	H_{100}	C	W	–
		Width	W_{100}	C	W	–
Shelves 1	I_U					
. . .						
Shelves 25	I_U					
Section 1	I_U	Height	HS_1	C	W	–
		Width	WS_1	C	W	–
		Supported weight	PS_1	C	W	–
		Shelves	SS_1	O	W	–
. . .						
Section 250	I_U	Height	HS_{250}	C	W	–
		Width	WS_{250}	C	W	–
		Supported weight	PS_{250}	C	W	–
		Shelves	SS_{250}	O	W	–

of parts, shared with the latter. Workstations are treated individually in the system (*any mode can be installed in each*) (Table 3.20).

The shared attributes will be represented in a single element, indicating only the parameter for the rest.

In order to give uniformity to the data and for the grouping process to be simpler, as we will see in Sect. 3.4, we have included as an attribute each ratio of the number of machines with each mode, regardless of whether its value is null.

3.3.1 Collective Element vs Individual Items

As we have already mentioned, the items of a collective element are identical, both in their definition of data and in their values. In addition to this, another important characteristic that the collective element must have, whether determined or indeterminate, is in the very concept of collectivity. Items from collective elements should never be treated individually in the decisions and specifications of a system. If this happens, each item should be considered as an individual element and identified within the typologies of the individual elements. Therefore, the allusions in the description of decisions and specifications of a collective element must be by numerals of the element, never particularizing on any or each of them.

Table 3.20 Elements of Illustration 3.3

		Data				
Element	QN	Name	Parameters	Type	Belonging	Value
Parts	C_D	Production	Q	I	W	3000 parts/day
		Rate_Mode1	R_1	I	S	45 parts/hour
		Rate_Mode2	R_2	I	S	54 parts/hour
M1 machines	C_I	Cost	C_1	C	W	–
		M1_Mode1	M_{11}	I	S	2
		M1_Mode2	M_{12}	I	S	0
M2 machines	C_I	Cost	C_2	C	W	–
		M2_Mode1	M_{21}	I	S	1
		M2_Mode2	M_{22}	I	S	1
M3 machines	C_I	Cost	C_3	C	W	–
		M3_Mode1	M_{31}	I	S	0
		M3_Mode2	M_{32}	I	S	1
Mode 1	I_U	R_1; $M_{11;}$ $M_{21;}$ M_{31}				
Mode 2	I_U	R_2; $M_{12;}$ $M_{22;}$ M_{32}				
Workstation 1	I_U					
Workstation 2	I_U					
Day	I_U	Available time disponible	T	C	W	18 hours

This characteristic acquires greater relevance if the element, a priori collective, is indeterminate and the description alludes in a particular way to each item in the system. In that case, as its quantity is indeterminate, it is necessary to establish an upper bound in the number of items that the system may need and individualize each one of them.

Any optimization problem in which elements defined individually with the same functionality are presented, being identical in their definition of data and values and they are not referenced in a particular way in the system, can be modelled by grouping these elements as items of a single collective element. The grouping always means some loss of relational information, which would be recoverable after the resolution of the model. However, the grouping streamlines the resolution process because we will approach a model with fewer variables, which is in general more efficient. Whenever the conversion of individual elements into items of a collective element is carried out, an attribute must be added that counts the number of grouped items.

In Chap. 7 of the book, we will see examples of complex cases in which these incidents occur, where the influence of the description of the system for the consideration of the element quantitative nature is fundamental.

As an illustration, we are going to analyze two problems: first, the batch sale system already modelled in Chap. 2 as an example of identifying collective elements and, second, a classic of optimization, the knapsack problem, as an example that accentuates the influence of the values of the data to establish the typology of elements.

Illustration 3.17: Sale of Batches (Illustration 3.8) (Meléndez 2019)
A department store has 200 shirts and 100 trousers from the previous season. They launch two batch offers, A and B. The Offer A consists of a batch of one shirt and one trousers, which is sold at $30; Offer B consists of a batch of three shirts and one trousers, which is sold for $50. They do not want to launch less than 20 batches of Offer A. On the other hand, they also have the option of transferring pants to an outlet for a price of $18/trousers, agreeing to pay a fee to the outlet of $80 if they transfer more than 50 units. The objective is to maximize the profit.

In this example there are two concepts, the shirts and the trousers, which can a priori have a double treatment. The shirts could be considered either as a collective element or determined, since it consists of 200 units or items, or we could consider each of those shirts individually as a unitary element. The same would happen with the trousers. This disjunctive is produced by the discrete and determined character of the element.

Let's analyze how the shirts are alluded to in the text:

A department store has ***200 shirts*** *and 100 trousers from the previous season. They launch two batch offers, A and B. Offer A consists of a batch of* ***one shirt*** *and one trousers, which is sold at $30; Offer B consists of a batch of* ***3 shirts****. . ..*

It is observed that in all cases the text alludes to numerals of the collective element shirts, and at no time a shirt is distinguished in a particular way. This means that the element is collective and each shirt must not be set up individually. Something similar happens with trousers.

The system will need to quantify the number of shirt items that it uses to make the batches in order not to exceed the stock it has, which is 200 shirts.

On the other hand, Batch Offer A and Batch Offer B are also elements of the system. Both are also collective elements, since in the system the number of batches that we launch of each type will be decided. Then they are elements whose quantity is measurable and indeterminate, because we do not know a priori the number of units that we will have of each batch. There is no reference to any specific item of the batches, so there is no room for an individual analysis of those items.

In addition to the previous elements, the Outlet also exists as a unitary element. It is unitary because it is an indivisible element and there are no Outlet items, but we have a single Outlet. We could also consider the department store as an element, that is, the system itself.

The rest of the information of the system is referred to data that have the elements (prices, number of shirts and trousers that make up each batch, the profit of each batch, a fee, or the limit number of trousers to apply that cost). We portray all of this information in Table 3.21.

Table 3.21 Table of Elements of Illustration 3.17

Element	QN	Data				
		Name	Parameters	Type of value	Belonging	Value
Shirts	C_D	Stock	S_S	I	W	200
		N° shirts in Batch A	N_{SA}	I	S	1
		N° shirts in Batch B	N_{SB}	I	S	3
Trousers	C_D	Stock	S_P	I	W	100
		N° trousers in Batch A	N_{TA}	I	S	1
		N° trousers in Batch B	N_{TB}	I	S	1
		Sale price to the Outlet	P_O	C ($/unit)	S	18
		Maximum number of trousers sold to the outlet without paying fee	M_{TO}	I	S	50
		Fee for transferring trousers to the outlet	F_{TO}	C ($)	S	80
Batch Offer A	C_I	Sale price	P_A	C ($/unit)	W	30
			N_{SA}; N_{TA}			
Batch Offer B	C_I	Sale price	PB	C ($/unit)	W	50
			$N_{SB;}$ N_{TB}			
Outlet	I_U		P_O; $M_{TO;}$ F_{TO}			

Maximum number of trousers sold to the outlet without paying fee = 50
Fee for transferring trousers to the outlet = $80

Illustration 3.18: Knapsack Problem (Martello and Toth 1990)

Given a knapsack with a volume V and a set of objects, each characterized by a volume and a value, it is about maximizing the total value of the objects introduced in the knapsack.

Specifications: The total volume of the objects introduced cannot be greater than the volume of the knapsack

Objective: Maximize the total value of the objects introduced

We will use two instances of values, which will determine different options for the elements.

Analyzing the statement, we can extract the backpack and objects as elements of the problem. Both the backpack and each object are considered, a priori, as individual. However, depending on the instance, this can vary, because other options can be considered. This will have an impact on the other components of the model, that is, activities, specifications, and objective function, which will be defined according to

Table 3.22 Elements of the knapsack problem (Instance 1)

Element	QN	Data				
		Name	Parameter	Type of value	Belonging	Value
Knapsack	I_U	Volume	VM	C	W	8
Object 1	I_U	Value	p_1	C	W	5
		Volume	v_1	C	W	3
Object 2	I_U	Value	p_2	C	W	4
		Volume	v_2	C	W	6
…						
Object 5	I_U	Value	p_5	C	W	4
		Volume	v_5	C	W	5

the table of problem elements. Despite not having reached the chapters of these components, as the model is simple, we will present the components according to the Table of Elements. The instances are the following:

Instance 1 Five objects			Instance 2 Five objects		
Objects	Value	Volume	Objects	Value	Volume
1	5	3	1	2	2
2	4	6	2	2	2
3	6	5	3	6	5
4	3	1	4	6	5
5	4	5	5	6	5
Knapsack volume = 8			Knapsack volume = 12		

Instance 1

Regarding the first set of values, the six objects have the same attributes, weight and volume, and each object has its own values for them, different from the rest, so there is no consideration of each object as an instance of a collective element. We are obliged to consider them as individual. In addition, the objects are of unitary type because they are treated completely; there is no continuous measurement of their data in the system, even if they are continuous.

The knapsack is an individual element with a continuous attribute, the volume, which cannot be measured in decisions as its partial use is based on logical decisions. The elements table would be as follows (Table 3.22).

- Decision activities:

The activity of the problem is to introduce objects in the knapsack. As we will see in the chapter dedicated to decision activities, this activity must be defined with

Table 3.23 Table of Elements of the knapsack problem (Instance 2)

Element	QN	Data				
		Name	Parameter	Type of value	Belonging	Value
Knapsack	I_U	Volume	VM	C	W	12
Objects of Group 1	C_D	Value	p_1	C	W	2
		Volume	v_1	C	W	2
		Stock	s_1	I	W	2
Objects of Group 2	C_D	Value	p_2	C	W	6
		Volume	v_2	C	W	5
		Stock	s_2	I	W	3

binary value. The activity defines an event with each element of the set of objects and the knapsack, since it is necessary to determine if each object is introduced or not in the knapsack. Therefore, we define as variables:

$$\alpha_1 = \begin{cases} 1 & \text{if we introduce object 1 in the knapsack} \\ 0 & \text{in other case} \end{cases}$$

$$\alpha_2 = \begin{cases} 1 & \text{if we introduce object 2 in the knapsack} \\ 0 & \text{in other case} \end{cases}$$

$$\ldots$$

$$\alpha_5 = \begin{cases} 1 & \text{if we introduce object 5 in the knapsack} \\ 0 & \text{in other case} \end{cases}$$

- Specifications:
  - Knapsack volume: The total volume of the objects introduced cannot be greater than the volume of the knapsack:
    $v_1\alpha_1 + v_2\alpha_2 + v_3\alpha_3 + v_4\alpha_4 + v_5\alpha_5 \leq \text{VM}$
- Objective criterion: Maximize the value of objects introduced in the knapsack:
  Max $p_1\alpha_1 + p_2\alpha_2 + p_3\alpha_3 + p_4\alpha_4 + p_5\alpha_5$

Instance 2

In Instance 2, in addition to the approach taken for Instance 1, it is possible to develop another Table of Elements. There are two groups of objects where the values of the data are identical, and the description of the system does not refer to any object in a particular way. The only specification of the problem refers in particular to the knapsack. The objects are treated collectively in the description. This means that for this second instance, each of these groups of objects could be considered as a collective element, which would adopt the two attributes of the objects plus an additional attribute with the number of items of the group (Table 3.23).

- Decision activities:

The activity of introducing objects in the knapsack is associated in this case with the knapsack and the two groups of objects, giving rise to two events. As the objects are measurable, the activity of introducing objects becomes a quantification activity. We will decide how many objects of each group can be introduced in the knapsack. We will also see in the chapter dedicated to decision activities that when an activity involves collective elements, this activity will also be discrete and measurable. The variables of the problem would be defined as:

x_1 = Number of objects of Group 1 introduced in the knapsack
x_2 = Number of objects of Group 1 introduced in the knapsack

- Specifications:
  - Availability of items:
    $x_1 \leq n_1$
    $x_2 \leq n_2$
  - Knapsack volume:
    $v_1x_1 + v_2x_2 \leq \text{VM}$
- Objective criterion:
  Max $p_1x_1 + p_2x_2$

As we discussed when simplifying unitary elements in a collective element, certain information is lost. The solution of Instance 2 would provide the number of objects that we can introduce from each group, but not the particular objects. However, this information is irrelevant; since all the objects of each group are identical, we could select any of them until the numbers of objects x_1 and x_2 are introduced.

Relationship Between both Configurations
If we express the Instance 2 with the first Table of Elements, we would have the following variables and expressions:

Decision activities:
Binary variables: α_1, α_2, α_3, α_4, α_5
Specifications:
$v_1\alpha_1 + v_2\alpha_2 + v_3\alpha_3 + v_4\alpha_4 + v_5\alpha_5 \leq \text{VM} \Rightarrow$
$\Rightarrow 2\alpha_1 + 2\alpha_2 + 6\alpha_3 + 6\alpha_4 + 6\alpha_5 \leq \text{VM}$
$\Rightarrow 2(\alpha_1 + \alpha_2) + 6(\alpha_3 + \alpha_4 + \alpha_5) \leq \text{VM}$
Objective criterion:
$\text{Max } p_1\alpha_1 + p_2\alpha_2 + p_3\alpha_3 + p_4\alpha_4 + p_5\alpha_5 \Rightarrow \text{Max } 2\alpha_1 + 2\alpha_2 + 5\alpha_3 + 5\alpha_4 + 5\alpha_5$
$\Rightarrow \text{Max } 2(\alpha_1 + \alpha_2) + 5(\alpha_3 + \alpha_4 + \alpha_5)$

Given that:

$x_1 = \alpha_1 + \alpha_2 \Rightarrow 0 \leq x_1 \leq 2 = n_1$
$x_2 = \alpha_3 + \alpha_4 + \alpha_5 \Rightarrow 0 \leq x_2 \leq 3 = n_2$
The expressions of the model remain as:

$\text{Max } 2x_1 + 5x_2$

$2x_1 + 6x_2 \leq \text{VM}$

$x_1 \leq n_1$

$x_2 \leq n_2$

$x_1 \geq 0 \;\; x_2 \geq 0$

which corresponds with the model for the second Table of Elements ($p_1 = 2; p_2 = 5; v_1 = 2; v_2 = 6$).

3.4 Association of Elements in Sets

Certain elements of a system can be grouped into sets when they have the same functionality in the system, are of the same quantitative nature, and have the same definition of data. The grouping into sets introduces the use of subscripts to reference each element of the set.

The advantage of using sets is that the mathematical definition of the model is simplified, both in its presentation and in the implementation for its resolution.

The association brings with it a slight modification of the Table of Elements. To indicate that we allude to a set of elements and to modify our structure, we will add a column between the name and the quantitative nature that indicates the sets, the subscripts used, and the number of elements that make up the set (Table 3.24).

For a matter of space, the values of the attributes for the grouped elements will not be presented in the table, unless there are few elements in the set. The data values of elements that are not included in any set can be maintained.

The Table of Elements of our butter production system could be simplified by associating elements into sets. The two machines have the same attributes. Similarly, for the two types of butters, the same attributes are also defined.

When the elements become part of a set, they end up being referenced by an ordinal rather than by their own identification in the system. Therefore, the association in a set of the two machines means the pasteurization machine is referenced as index 1 in the set of machines and the whipping machine as index 2. The same happens with the two butters: sweet butter becomes the index 1 within the set of butters and raw butter index 2.

The table would be defined as (Table 3.25).

As can be seen, the list of attributes of each set has become two, when actually each element has three attributes. This simplification occurs because the data shared between sets are represented only once in the set. "Time used by 1 kg of butter *j* in machine *i*" is representing the time that each machine uses with each butter in a single row since both the machines and the butters are associated in sets.

Table 3.24 Framework of Table of Elements with sets

			Data				
Elements	Set	QN	Name	Parameters	Type	Belonging	Value
Name of set	Subscripts = 1...number of elements	I/I_U/I_M/ C/C_D/C_I	Name	*A*	C/I/ O/B	W/S	...
			...	...	...	...	...
...							

Table 3.25 Table of Elements associated into sets of Illustration 2.1

			Data				
Elements	Set	QN	Name	Parameters	Type	Belonging	Value
Machines	$i = 1,2$	I_U	Usage time	T_i	C (Min)	W	. . .
			Time used by 1 kg of butter *j* in machine *i*	TM_{ij}	C (Min)	S	. . .
Butters	$j = 1,2$	I_M	Profit	B_j	C ($/kg)	W	. . .
				TM_{ij}			

3.4.1 Assigning or Removing Data to Create Sets

Sometimes it is possible to assign an attribute to an element that initially does not have that characteristic, with the idea of matching the attributes of this element with those of others and thus being able to create a set with all of them. In this way, we can extend the elements of the same set and thereby simplify the Table of Elements and the definition of the model. In that case, we assign it an appropriate value to the data that does not affect the problem. Appropriate value means that a value without utility is imputed, which for the system is equivalent to not having it. Let's see an example that illustrates this possibility.

Illustration 3.19
For the purchase of a product, we have three suppliers in the market (P1, P2, P3). Each supplier has a product sale price and transport costs (Table 1). The third provider offers us a discount of $0.5/unit if we buy more than 100 units.

Table 1 Sale prices and transport cost

	Sale price/unit	Transport cost
P1	85	140
P2	88	159
P3	90	150

We identify P1, P2, and P3 as elements, as well as the product. We define the suppliers as unitary and the product as collective. The Table of Elements would be as follows (Table 3.26).

To associate the three suppliers in a set, we incorporate the Discount and Threshold data to P1 and P2, assigning zero value, since they do not make any discounts. With this we managed to unify the data of the three providers, and we can create the set of suppliers (Table 3.27).

Another possibility that we can admit for the representation of the data is to form sets and incorporate the data that are specific only to some of the elements of the set

Table 3.26 Table of Elements of Illustration 3.19

Elements	QN	Data				
		Name	Parameters	Type	Belonging	Value
P1	I_U	Sale price	Pr1	C ($/unit)	S	85
		Transport cost	$C1$	C ($)	W	140
P2	I_U	Sale price	Pr2	C ($/unit)	S	88
		Transport cost	$C2$	C ($)	W	159
P3	I_U	Sale price	Pr3	C ($/unit)	S	90
		Transport cost	$C3$	C ($)	W	150
		Discount	D	C ($/unit)	S	0,5
		Threshold	Th	I	S	100
Products	C_I		Pr1, Pr2, Pr3, D, Th			

Table 3.27 Elements associated into sets of Illustration 3.19

Elements	Set	QN	Data				
			Name	Param	Type	Belonging	Value
Suppliers	$i = 1...3$	I_U	Sale price	Pr_i	C ($/unit)	S	85;88;90
			Transport cost	C_i	C ($/unit)	W	140;159;150
			Discount	D_i	C ($/unit)	S	0;0;0;5
			Threshold	U_i	I	S	0;0;100
Product	–	C_I		Pr_i, D_i			

Table 3.28 Table of Elements with individual parameters of Illustration 3.19

Elements	Set	QN	Data				
			Name	Param	Type	Belonging	Value
Suppliers	$i = 1...3$	I_U	Sale price	Pr_i	C ($/unit)	S	85;88;90
			Transport cost	C_i	C ($/unit)	W	140;159;150
			Discount	D_3	C ($/unit)	S	0;5
			Threshold	U_3	I	S	100
Product	–	C_I		Pr_i, D_i			

without the abstract reference of the subscript, only pointing to the specific subscript of the element that owns the data, as shown in Table 3.28. This is valid for the representation but can give problems when implementing the model in an optimization library. Another solution for the implementation is not to consider that specific data within the element table. This data will be used directly in the specifications without the use of parameters.

The representation of the attributes is not a relevant aspect in the design of the model, but it is for the implementation of the same.

3.4.2 Shared Data Between Elements of the Same Set

The norm is to use a single subscript for each set. However, we can reference a set with more than one subscript in cases where there are shared data between elements of the same set. Let's see the following example.

Illustration 3.20

There is a set of four municipalities of which the number of inhabitants and the distance in Km between each of them (Tables 1 and 2) are known. We want to install a retail store, and we want to determine the most suitable municipality for it in order to place the store as close as possible to the inhabitants of all the municipalities.

Table 1 Distance between municipalities (Km)

Distances	*Municipality 1*	*Municipality 2*	*Municipality 3*	*Municipality 4*
Municipality 1	*0*	*12*	*26*	*35*
Municipality 2	–	*0*	*18*	*29*
Municipality 3	–	–	*0*	*8.5*
Municipality 4	–	–	–	*0*

Table 2 Number of Inhabitants

	Inhabitants
Municipality 1	*4800*
Municipality 2	*6500*
Municipality 3	*7200*
Municipality 4	*10540*

We identify each municipality and the retail store as elements, all unitary. The Table of Elements would be as follows (Table 3.29).

Since I need to reference the distance attribute with each pair of municipalities, I have to define two subscripts, i and j, to refer twice to the same set in the distance data, D_{ij}. For the other non-shared data, we use a single subscript, in this case the index i.

This example also illustrates something that can happen in a system, elements without any attribute, which is the case of the retail store.

3.4.3 Hierarchical Definition of Sets

When a system has elements that are part of other elements, forming a hierarchical tree, it is possible to identify the child elements using the parent subscript in addition to the child subscript. In this way, the number of elements in the table decreases, and the dependency properties are implicit in the identification of the indexes of an element. The only condition for this is that the child elements of each parent element are identical.

Table 3.29 Table of Elements of Illustration 3.20

Elements	Set	QN	Data				
			Name	Par.	Type	Belonging	Value
Municipality	$i, j = 1..4$	I_U	Distance	D_{ij}	C (Km)	S	...
			Inhabitants	H_i	I	W	...
Retail store	–	I_U					

Table 3.30 No-hierarchical table

Elements	Set	QN	Data				
			Name	Param	Type	Belonging	Value
Boxes	$i = 1..10$	I_U					
Compartments	$j = 1..50$	I_U	Volume	VC_j			
			Box	C_j	O	W	...
Objects	$k = 1..30$	I_U	Volume	v_i	C	W	...

Illustration 3.21

Given a set of ten boxes, each consisting of five compartments. Each compartment has a volume and they are identical in all boxes. It is about assigning a list of 30 individual objects, each with a volume, to the compartments. All objects going to the same compartment must not exceed the volume of the compartment.

Each compartment needs to be considered individually, a total of 50 compartments. However, since the compartments are identical in each box, we could also apply a hierarchical structure in which we use fewer elements. Let's first look at a non-hierarchical configuration of the elements.

Non-hierarchical Table of Elements

In Table 3.30 the compartments are identified with the subscript *j*. The box to which each compartment belongs is identified in the Box attribute, with an ordinal value (we could also have used a shared binary value).

If the decision variables are to assign objects to compartments, we will use a variable with subscripts *k* and *j*: $\alpha_{kj} = 1$ if I assign object *k* to compartment *j*, *0* otherwise.

Hierarchical Table of Elements

In Table 3.31 each compartment, since they are individual, should be referenced by index *j* plus index *i* of the box to which it belongs. The attribute of belonging is implicit in this association.

The decision variables associated with the allocation of objects to compartments would be as follows:

$\alpha_{kji} = 1$ if I assign object *k* to compartment *j*, *0* otherwise.

Table 3.31 Hierarchical table

Elements	Set	QN	Data				
			Name	Param	Type	Belonging	Value
Boxes	$i = 1..10$	$\mathrm{I_U}$					
Compartments	$j = 1..5$	$\mathrm{I_U}$	Volume	VC_j			
Objects	$k = 1..30$	$\mathrm{I_U}$	Volume	v_i	C	W	...

3.5 Data Generating Elements

The identification of elements in a problem does not have to present a single configuration. We already introduced in Sect. 3.2.2 the possibility of representing as an element the quality that defines the attribute in the case of binary attribute. This happens with binary data as well as with continuous or integer attributes. This happens both with binary data and with continuous or integer data, own or shared, although in shared attributes it will happen whenever there is no primary element in the relationship.

The concepts that act as attributes and are not defined as elements are referred to intrinsic properties of the element, such as any continuous magnitude of the element (weight, volume, duration, etc.) or characteristics that do not need to be explicit as an element because there are no decisions or specifications that fall on them in the system. In the case of representing one of these characteristics as an element, the design of the model is not hindered, but it is not necessary.

Let’s see a simple illustration of an own attribute that we make explicit as an element:

Illustration 3.22
There is a city formed by a set of 20 neighborhoods. The number of registered citizens from each neighborhood is known. The distance between the central points of each neighborhood is also known. The aim is to install a citizen information point in a central point of a neighborhood so that the sum of the distances that citizens travel between the central points of each neighborhood and the information point is as small as possible.

The elements that we can extract from the text are:

- The city
- 20 neighborhoods
- Citizens
- Central point of each neighborhood (geographical center of each neighborhood)
- Citizen information point

From these elements we can affirm that:

- The city can be considered as the system itself.
- Neighborhood and central point of the neighborhood can be considered as the same element. It is not necessary to create an element for each neighborhood and

Table 3.32 Table of Elements of Illustration 3.22

			Data				
Elements	Set	QN	Name	Param	Type	Belonging	Value
Neighborhoods	*i*, $j = 1\ldots20$	I_U	N° citizens	N_i	I	W	...
			Distance	D_{ij}	C	S	...
Citizen information point	–	I_U					
City	–	I_U					

Table 3.33 Table of Elements of Illustration 3.22 considering citizens as element

			Data				
Elements	Set	QN	Name	Param	Type	Belonging	Value
Neighborhoods	*i*, $j = 1\ldots20$	I_U	N° citizens	N_i	I	S	...
			Distance	D_{ij}	C	S	...
Citizens	–	C_D		N_i			
Citizen information point	–	I_U					
City	–	I_U					

another element for each central point. The neighborhood includes the central point. Referring to one or the other is indifferent in the system.
- We can do without citizens as an element, since it does not have any characteristic of its own and no decision or specification falls on the citizens. If it was considered as an element, it would have to be defined as collective.

Based on this, the Table of Elements could be defined as follows (Table 3.32).

In the case of having made the citizen explicit as an element, the table would have been as follows (Table 3.33).

The relationship between neighborhoods and citizens has been simplified as follows (Fig. 3.2).

When a shared concept without primary elements is established as an element, this new element would have, as own data, all the shared data existing between the elements that shared the concept. In addition, in the case of attributes shared between sets, the configuration possibilities in the element table are expanded. On the one hand, as we already have defined on grouping into sets, the new elements are referenced with a new index, and it would be necessary to incorporate a shared binary attribute that identifies the elements that each new element links. And on the other hand, we can identify each new element with the indexes of the elements that shared it. Thus, it is not necessary to introduce the relationship as an attribute since it is embedded in the indexes that identify it.

Let's see an illustration of this casuistry, expanding Illustration 3.21.

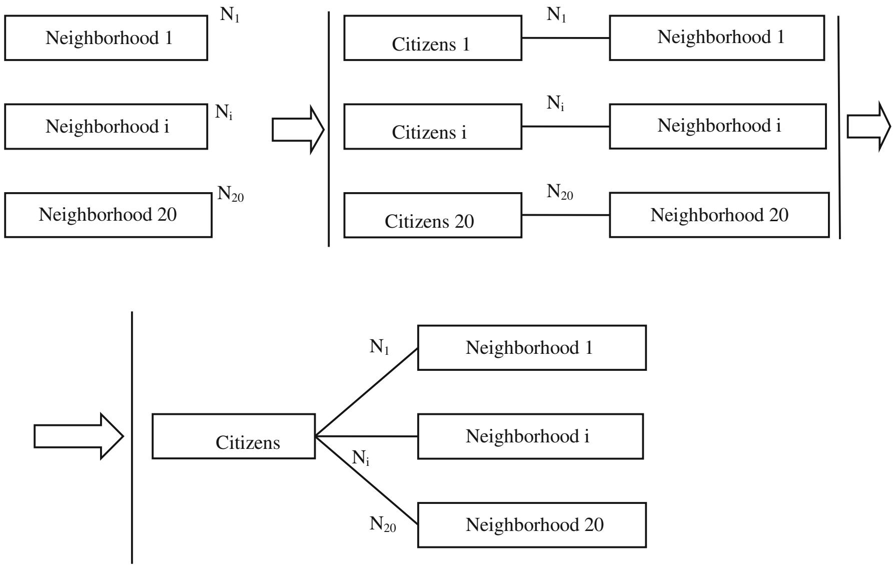


Fig. 3.2 Generation of Citizens as Element

Table 3.34 Table of Elements of Illustration 3.23

Elements	Set	QN	Data				
			Name	Param	Type	Belonging	Value
Municipalities	$i, j = 1..4$	I_U	Distance	D_{ij}	C (Km)	S	. . .
			Refueling	Rij	B	S	. . .
			Citizens	H_a	I	W	. . .
Retail store	–	I_U					

Illustration 3.23

There is a set of four municipalities of which the number of inhabitants and the distance in Km between each of them are known. We want to install a retail store, and we want to determine the most suitable municipality for it in order to place the store as close as possible to the citizens of all the municipalities. We also collect information on whether there is a possibility of refueling between municipalities.

The Table of Elements could be considered in two ways:

Setting 1

We model the table considering only the municipalities and the sales store as elements (Table 3.34).

The citizens are not taken into account as an element as in Illustration 3.20.

Table 3.35 Alternative Table of Elements of Illustration 3.23

			Data				
Elements	Set	QN	Name	Param	Type	Belonging	Value
Municipalities	i, $j = 1\ldots4$	$\mathrm{I_U}$	Connection-Municipality	M_{ik}	B	S	. . .
			Citizens	H_i	I	W	. . .
Connection	$k = 1\ldots6$	$\mathrm{I_U}$	Distance	D_k	C (Km)	W	..
			Refueling	R_k	B	W	. . .
				M_{ik}			
Retail store	–	$\mathrm{I_U}$					

Table 3.36 Table of Elements compatible with LINGO

			Data				
Elements	Set	QN	Name	Param	Type	Belonging	Value
Municipalities	$i, j = 1..4$	$\mathrm{I_U}$	Citizens	H_i	I	W	. . .
Connections	(i, j) = (1,2) (1,3) (1,4) (2,3) (2,4) (3,4)	$\mathrm{I_U}$	Distance	$D_{(ij)}$	C (Km)	W	. . .
			Refueling	$R_{(ij)}$	B	W	. . .
Retail store	–	$\mathrm{I_U}$					

Setting 2

Both distance and refueling are shared attributes that do not have a primary element in the relationship. We make the relationship between municipalities explicit, which we will call *connection*. The data of this relationship, Distance and Refueling, would become data of the new connection element, and we will incorporate a shared attribute for the relationship between municipality and connection (Table 3.35).

Six elements are identified in the connection set: (Municipality 1 – Municipality 2), (Municipality 1 – Municipality 3), (Municipality 1 – Municipality 4), (Municipality 2 – Municipality 3), (Municipality 2 – Municipality 4), and (Municipality 3 – Municipality 4).

Setting 3: New Way to Represent Sets

We configure the shared data creating, as in Setting 2, a new element for the link but incorporating for the identification of each element a subscript for each element that shared the characteristic. Each element of the new set is identified by a tuple of subscripts (Table 3.36).

In this way, the identification data of the municipalities that form each connection is embedded in the subscripts.

3.6 Identification of Data in Graphic Environments. Elements in Graphs

When a system is represented graphically, it is very likely that you have to extract data from the graph, data that relate to the elements of the system. The most popular cases are the optimization problems in graphs.

To collect data of the graphic representation of a system, two steps are established:

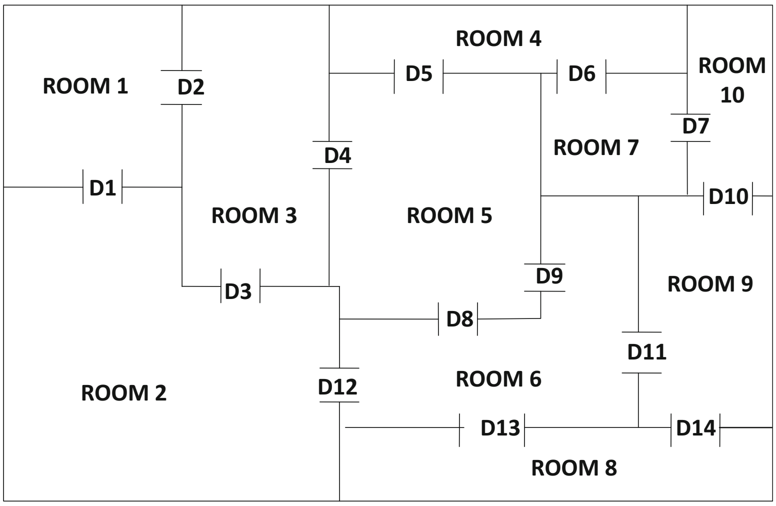
Step 1: Identify the elements of the system, based on the description provided.
Step 2: Identify the elements in the graph and extract the graphic relationships between them as data. Graphic data can mean measurements, coverage, connections, etc.

Connections are a very necessary property in many graphical systems, not only in graph representation. Here is an example of this property.

Illustration 3.24 (Source: Larrañeta et al. 2003)
The following figure shows the floor of a museum with 10 rooms (ROOM 1 to ROOM 10) connected by 14 doors (P1 to P14). Security cameras will be installed to monitor the rooms. The security cameras are of two types:

1. *Model A: these are installed in the doors and provide surveillance to the rooms with that particular door.*
2. *Model B: they are installed in rooms and provide surveillance to the room where they are installed.*

Each type of camera has a different cost (CA and CB, respectively).



Reading the text, the configuration of the list of system elements does not offer much complexity. We have the doors, the rooms, and the Model A and Model B cameras. We discard the nouns installation and surveillance, directly associated with the action of installing and monitoring. We also discard cost because it is an attribute of the camera types and therefore will have a value associated with it. The museum would be the system element, although it has no data, so it can be ignored as an element.

A priori, the list of elements could be subject to different proposals:

Regarding the doors, we could consider the doors as a collective element formed by 14 units, or each door (1 to 14) as an individual element. The same happens with the rooms. It depends on their attributes and the system specifications. From the text, no data are extracted for rooms and doors. However, the graph provides important and necessary information for the modelling of the system and the connection between doors and rooms:

Door 1 (P1) is connected to Room 1 and Room 2.
Door 2 (P2) is connected to Room 1 and Room 3.
...

As discussed in the types of values of an attribute (Sect. 3.2.3), the value of an attribute must not correspond to an ordinal that identifies another element if the association is multiple. The solution is to establish a shared attribute between both elements of binary value. This results in the need to consider the doors and rooms as unitary elements, since each one has its own connections, that is, its own values in the connection data, and there are no continuous attributes.

Regarding the A and B cameras, their number is indeterminate because we do not know how many cameras of each type we are going to have. It is necessary to consider them collective elements, since the activities will measure the number of cameras installed in each door or room, having as a specification that one camera per room or door is installed, at most. In other words, we are looking at an example of two collective elements whose quantity in the decision activities is going to be 0 or 1.

We can set the following configuration for the Table of Elements (Table 3.37).

This configuration of elements is not unique, although it is the most recommended for the model. We could have made another design without considering the doors as an element but an attribute shared between the rooms. This would lead to a more awkward model, since having decision activities on doors is better

Table 3.37 Table of Elements of Illustration 3.24

			Data				
Elements	Set	QN	Name	Param	Type	Belonging	Value
Rooms	$i = 1...10$	I_U	Connection	C_{ij}	B	S	...
Doors	$j = 1...14$	I_U		C_{ij}			
Model A Cam.	–	C_I	Cost	CA	C	W	...
Model B Cam.	–	C_I	Cost	CB	C	W	...

Table 3.38 Alternative table of non-recommended elements of Illustration 3.24

			Data				
Elements	Set	QN	Name	Param	Type	Belonging	Value
Rooms	i, $k = 1\ldots10$	$\mathrm{I_U}$	Door	D_{ik}	B	C	...
Model A Cam.	–	$\mathrm{C_I}$	Cost	CA	C	W	...
Model B Cam.	–	$\mathrm{C_I}$	Cost	CB	C	W	...

considered as elements explicitly rather than having to reference them in the model using the rooms that connect them (Table 3.38).

If $D_{ik} = 1$, then there is a door between room i and room k. This implementation could be the most useful if the system does not consider activities on doors. Since Model A cameras are installed in doors, it is better for the model to make the doors explicit as an element. This configuration has been possible thanks to the fact that there is only one door between the same two rooms.

In the same way as in Illustration 3.23, we can choose a third configuration that identifies each door with the indexes of the rooms it connects, keeping in mind, as in the previous case, that there could not be two doors connecting the same rooms, since in that case both would be identified with the same subscript values.

3.6.1 Representation of Graphs

Graph G (N, A) is formed by a set of elements called nodes ($N = \{1\ldots n\}$) joined by links between pairs of nodes called edges or arcs ($A = \{(i, j)/i{\in}N, j{\in}N\}$) (Balakrishnan 1997).

In Fig. 3.3, graph G is represented by a set of six nodes, $N = \{1,2,3,4,5,6\}$, and a set of seven edges that link pairs of nodes, Edges = {(1,2); (1,3); (2,5); (3,5); (2,4); (4,6); (5,6)}.

The basic configuration of the Table of Elements of a graph considers the nodes of individual type as elements because each node is different from the rest. The edges

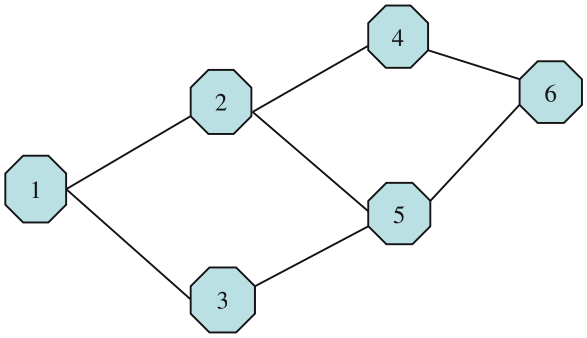


Fig. 3.3 Example of graph

Table 3.39 Basic Table of Elements of a graph

Elements	Set	QN	Data				
			Name	Param	Type	Belonging	Value
NODES	$i,j = 1\ldots6$	I	Connection (edges)	a_{ij}	B	S	...

Table 3.40 Table of Elements of a graph with edges as elements

Elements	Set	QN	Data				
			Name	Par	Type	Belonging	Value
NODES	$i = 1\ldots6$	I	Edge-Node	AN_{ik}	B	S	...
EDGES	$k = 1\ldots7$	I		AN_{ik}			

Table 3.41 Table of Elements of an enlarged graph

Elements	Set	QN	Data				
			Name	Par	Type	Belonging	Value
NODES	$i,j = 1\ldots6$	I	Edge-Node	AN_{ik}	B	S	...
			Connection	a_{ij}	B	S	...
EDGES	$k = 1\ldots7$	I		AN_{ik}			

Table 3.42 Table of Elements in a graph compatible with LINGO

Elements	Set	QN	Data				
			Name	Param	Type	Belonging	Value
NODES	$i,j = 1..6$	I					
EDGES	$(i,j) = (1,2)..(5,6)$	I					

are an attribute of the nodes. Again, the connection between nodes is the attribute that is extracted from the graphic framework (Table 3.39).

From this configuration, we can make the connections between nodes, that is, the edges, explicit as elements (Table 3.40).

We can even keep the connection data within the data of the nodes, in case it is more convenient for representing the system specifications (Table 3.41).

And as in Illustration 3.23, we can reference the edges with two indexes in a LINGO (Cunningham and Schrage 2004) implementation (Table 3.42).

Any implementation is valid, and we will always look for the most suitable one to model the system, keeping in mind the rest of the data associated with the graph and the decision activities that occur in it.

Now, let us suppose each node has a weight and each edge has a cost, as shown in Fig. 3.4.

The inclusion of these data in Tables 3.39, 3.40 and 3.42 would be as follows (Tables 3.43, 3.44 and 3.45).

The elements continue to be unitary because those continuous attributes have no measurable character, so they would be used in the objective function.

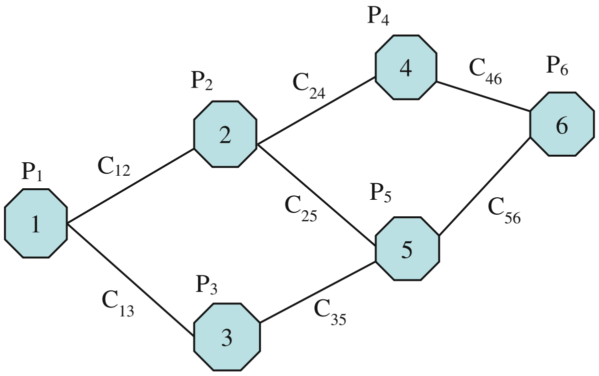


Fig. 3.4 Example of graph with information

Table 3.43 Table of Elements of graph in Fig. 3.4.

			Data				
Elements	Set	QN	Name	Param	Type	Belonging	Value
NODES	$i, j = 1..6$	I	Connection	a_{ij}	B	S	...
			Weight	P_i	C	W	...
			Cost	C_{ij}	C	S	...

Table 3.44 Table of Elements of graph in Fig. 3.4 with edges as elements

			Data				
Elements	Set	QN	Name	Param	Type	Belonging	Value
NODES	$i = 1..6$	I	Edge-Node	AN_{ik}	B	C	...
			Weight	P_i	C	P	...
EDGES	$k = 1..7$	I		AN_{ik}			
			Cost	C_k	C	P	...

Table 3.45 Table of Elements of graph in Fig. 3.4 compatible with LINGO

			Data				
Elements	Set	QN	Name	Param	Type	Belonging	Value
NODES	$i, j = 1..6$	I	Weight	P_i	C	W	...
EDGES	$(i, j) = (1,2)\ldots(5,6)$	I	Cost	C_{ij}	C	W	...

Directed Graphs (Bang-Jensen and Gutin 2000)

When we work with directed graphs, where the edges become arcs, which link a source node with a destination node by establishing an address in the link, the Table of Elements is slightly modified since the arc is constituted by two attributes, the source node and the destination node of the arc. The basic representation of Fig. 3.5 appears in Tables 3.46, 3.47 and 3.48.

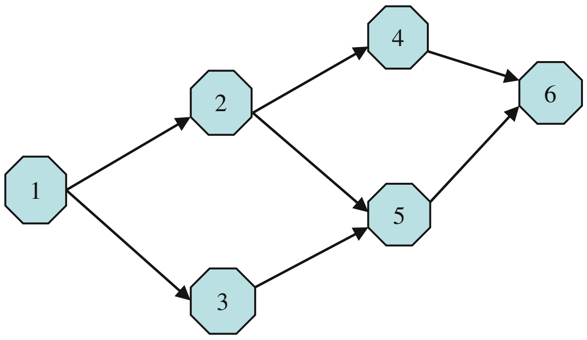


Fig. 3.5 Example of directed graph

Table 3.46 Table of Elements of a directed graph

Elements	Set	QN	Data				
			Name	Param	Type	Belonging	Value
NODES	$i, j = 1..6$	I	Source	O_{ij}	B	S	. . .
			Destination	D_{ij}	B	S	. . .

Table 3.47 Table of Elements of a directed graph with arcs as elements

Elements	Set	QN	Data				
			Name	Param	Type	Belonging	Value
NODES	$i = 1..6$	I	Arc_Source	O_{ik}	B	S	. . .
			Arc_Destination	D_{ik}	B	S	. . .
ARCS	$k = 1..7$	I		O_{ik}			
				D_{ik}			

Table 3.48 Table of Elements in a directed graph compatible with LINGO

Elements	Set	QN	Data				
			Name	Param	Type	Belonging	Value
NODE	$i, j = 1...6$	I					
ARC	(i,j)=(1,2)...(5,6)	I					

In this last representation, the identification of the arc by two indices is also embedding the origin and destination of the arc. The first index is considered the origin and the second the destination.

3.7 The Time Element

The time element is a very common element in a system to be optimized. It appears whenever there are activities or specifications that can be carried out in a set of given time periods (days, weeks, months, etc.).

Do not confuse the use of the time element with systems that use continuous value data whose unit of measure is time (duration or availability in time) nor when there are activities that measure duration. What this means is that there are elements with properties or data whose unit of measure is time.

The default time element is always present in a system, since any activity carried out in a system takes place at some point in time. Therefore, it only makes sense to consider the time element as individual elements when several periods or instants of time are established for carrying out the activities or specifications. They are generally unitary, unless the periods have measurable continuous data.

Illustration 3.25

System in which there is a set of n manufacturing tasks and it is necessary to decide on which day of the week each task is scheduled in the factory. Each task has a duration time.

The tasks would form a set of elements. It is not mentioned that the tasks are divisible; therefore they must be considered unitary. To schedule the tasks, we have identified seven periods, 7 days a week (Table 3.49).

Illustration 3.26

We have a product that we buy from a supplier. The system has to decide how many units of the product to buy in each month of the year.

In this example, the time periods indicated are the 12 months of the year. Each month would act as a unitary element (Table 3.50).

Table 3.49 Elements of Illustration 3.25

			Data				
Elements	Set	QN	Name	Param	Type	Belonging	Value
Tasks	$i = 1\ldots n$	I_U	Duration time	T_i	C	W	. . .
Time	$t = 1\ldots 7$	I_U					

Table 3.50 Table of Elements of Illustration 3.26

			Data				
Elements	Set	QN	Name	Param	Type	Belonging	Value
Product	–	C_I					. . .
Supplier	–	I_U					
Months	$t = 1\ldots 12$	I_U					

Time participates in a system without being an element in the following cases:

- When there are elements, generally tasks, that have, as undetermined property, the instant in which they are scheduled. Time acts as a unit of measurement in decisions (Illustration 3.27).
- When there is time data with a resource utility in the system, usually measurable in specifications. This was the case for butter production machines (Illustration 2.1), which had a time of use. Also Illustration 3.28 shows a measurable availability time in which the proprietary elements are periods of time.
- When we have elements that represent tasks or time periods that can be distributed among other elements, that is, those that can be partially used in decision activities (Illustration 3.29), and also when there are elements representing tasks in which their duration must be determined (Illustration 3.28).

In Illustration 3.28 we present an example with elements that have an undetermined property whose unit of measurement is time. Furthermore, in this example the time element is also presented as a set of finite periods and where each period is an element constituted by an available time property that can be measured both in the specifications and in the decisions. It is an interesting case because it shows many of the possibilities in which time can be presented in a system.

Finally, in Illustration 3.29 we are going to define another system where time does not participate as an element, but there are measurable elements whose unit of measurement in the activities where they participate is time, since partial and continuous use is made of their duration.

Illustration 3.27
Given a factory in which there is a set of n manufacturing tasks. Each task has a duration time. We have to decide when each task is scheduled at the factory.

The tasks would form a set of individual elements with an indeterminate continuous property, its scheduling instant. We could consider the factory as an element, even though it represents the system itself (Table 3.51).

Illustration 3.28
Training Planning (Source: Larrañeta et al. 2003)

A sprinter prepares for his next race. For this event he has to do 3 weeks of training. In his preparation he must spend hours of training in three disciplines:

1. *Speed*
2. *Resistance*
3. *Bodybuilding*

Table 3.51 Table of Elements of Illustration 3.27

			Data				
Elements	Set	QN	Name	Param	Type	Belonging	Value
Tasks	$i = 1..n$	I_M	Duration time	T_i	C	W	...

Table 3.52 Table of Elements of Illustration 3.28

Elements	Set	QN	Data				
			Name	Par	Type	Belonging	Value
Training disciplines	$i = 1,2,3$	I_M	Performance	P_i	C	S	{0.4;0.3;0.3}
			Minimal hours	m_i	C	W	{4,4,4}
Weeks	$t = 1,2,3$	I_U	Hours	H_t	C	W	{35,30,25}

It has been observed that in the performance of the athlete, speed training has an influence of 40% and the others 30%. The characteristics of the training should be the following:

- *The number of hours dedicated to speed training should increase by at least 1 hour per week.*
- *In the third week, to avoid fatigue in the runner, the amount of hours dedicated to the first type of training should not exceed 50% of the total training hours.*
- *Weekly, they should train 35, 30, and 25 hours per week respectively.*
- *Finally, for any training to be effective, you must devote at least 4 hours a week, regardless of which week it is.*

Objective: Maximize the performance of the athlete.

For training, it is specified that they use indeterminate hours. That is, a training in the system is a time, a number of hours, undetermined. Time works as a unit of measurement for an undetermined continuous property of an element. Elements with a property, its quantity, measured in hours and not determined, are being described, so they must be considered as measurable individual elements.

On the other hand, we have the time element, with three periods, the 3 weeks. These three elements have an attribute of available time or hours capacity that must be measured in the specifications. We could also consider the sprinter, but since he will be implicit in all activities, since training always refers to him, we could ignore him as an element, since he will not represent any option in decisions. The table, which grouped the elements in sets, would be as follows (Table 3.52).

In the Table of Elements, we have not included attributes that are not attributable to all the elements of a set or that have a complex assignment, such as the data referred to in the first two specifications. That information will be used directly in the modelling of the specifications.

This problem is an interesting example because it presents another configuration of the elements. We could have considered the time of each week as measurable in decisions and the disciplines as unitary concepts.

In the first configuration (Table 3.51), we measure the hours of each type of training in each week, in the decision activities, and in the second we would measure the hours of each week used in each type of training. The two ways are equivalent, but the first one measures, in the decisions, each training, and the second one measures the time of weeks.

Table 3.53 Table of Elements of Illustration 3.29

Elements	Set	QN	Data				
			Name	Param	Type	Belonging	Value
Tasks	$i = 1...n$	I_M	Duration	T_i	C	W	...
			Affinity	A_{ij}	C	S	...
Operators	$j = 1...m$	I_U		A_{ij}			

Illustration 3.29
Given a system with a set of n tasks. The tasks are performed by a set of m operators. Each task has a duration time. Each operator has an affinity with each task. The system has to decide the time that each operator devotes to each task, so that each task is carried out completely.

The tasks appear as the first set of elements. Since a task can be divided among several workers, the task is measurable because we can make continuous partial use of its duration. The unit of measurement of the division is of time, the time performed of each task by each operator. The second set of elements are the operators, which could be considered as unitary (Table 3.53).

3.8 Element Duplication

In the identification of elements, it is important that the description of the system is clear, and we should not use different ways of expressing the same concept, since this could lead to unnecessarily duplicating elements of the system. In our example for butter production, we could have redefined the text by substituting the pasteurization and whipping machines for the pasteurization and whipping processes, respectively. The text would be as follows:

A butter production factory wants to optimize its daily production of butter. Two types of butter are made (Sweet and Raw). A kilo of sweet butter gives the manufacturer a profit of $10 and a kilo of raw a benefit of $15. For the production of butter, two processes are used, a pasteurization process and a whipping process. The daily use time of the pasteurization process is 3.5 hours and 6 hours for the whipping process.

The time (in minutes) used in each process to obtain a kilo of butter is shown in the following table:

Table 1 Processing times (minutes)

	Sweet butter	Raw butter
Pasteurization process	3	3
Whipping process	3	6

In this case, the processes replace the machines, and they would be considered identical elements to the mentioned machines.

However, if in the text, we had used the following wording:

A butter production factory wants to optimize its daily production of butter. Two types of butter are made (Sweet and Raw). A kilo of sweet butter gives the manufacturer a benefit of $10 and a kilo of raw a benefit of $15. For the production of butter, two processes are used, a pasteurization process and a whipping process. For each process there is a machine. The daily use time of the pasteurization process is 3.5 hours and 6 hours for the whipping process. The times (in minutes) used in each machine to obtain a kilo of butter are collected in the following table:

Table 1 Processing times (minutes)

	Sweet butter	Raw butter
Pasteurization process	3	3
Whipping process	3	6

We would be using both concepts, processes and machines, and we could fall into an error of duplicity of elements. Neither of them contributes anything that differentiates one from the other, so they are the same element in the system.

3.9 Examples

3.9.1 Fire Stations (Source: Larrañeta et al. 2003)

An initial study is planned to install two fire stations in an urban area that currently has none. The approach has been adopted to divide the urban area into five sectors and carry out a preliminary analysis of the repercussions of the possible location of the stations in each of the sectors. The average time, in minutes, of answering a call from a fire station located in a certain sector i for an incident received from each of the sectors j has been estimated in t_{ij}. The average number of calls per day that will take place from each of the five sectors (F_j) has also been estimated. All these values are shown in Table 1. For example, it takes 12 minutes to attend an incident from

Table 1 Frequencies and time between sectors

t_{ij}	Sector 1	Sector 2	Sector 3	Sector 4	Sector 5
Sector 1	5	12	30	20	15
Sector 2	20	4	15	10	25
Sector 3	15	20	6	15	12
Sector 4	25	15	25	4	10
Sector 5	10	25	15	12	5
Frequency	2.5	1.6	2.9	1.8	3.1

Table 3.54 Version 1 of the Table of Elements in Example 3.9.1

			Data				
Elements	Set	QN	Name	Param	Type	Belonging	Value
Sectors	$i, j = 1\ldots5$	I_U	Time	t_{ij}	C	S	...
			Frequency	f_i	C	W	...
Fire stations	$k = 1,2$	I_U					

Table 3.55 Version 2 of the Table of Elements in Example 3.9.1

			Data				
Elements	Set	QN	Name	Param	Type	Belonging	Value
Sectors	$i, j = 1\ldots5$	I_U	Time	t_{ij}	C	S	...
			Frequency	f_i	C	W	...
Stations	–	C_D	N° items	n	I	W	2

sector 5 from a station located in sector 3. The last row shows the average daily frequency of calls made to the fire service.

The Table of Elements of this system can be considered in two ways, depending on the consideration of the fire stations. In the first version, we will consider each fire station as unitary:

Table of Elements: Version 1 (Table 3.54)
As we can see, the stations are identical in the system, as they do not have data. On the other hand, the text does not allude to any of it in a particular way. In the phrase:

"The average time, in minutes, of answering a call from a fire station located in a certain sector i for an incident received from each of the sectors j has been estimated in t_{ij}".

The reference is made to the time data **between sectors**. It could really eliminate the reference to the fire station and have simply put "the average time, in minutes, to answer a call from a certain sector for an incident received from each of the sectors." Therefore, we could have proposed a version considering the station element as collective with two items:

Table of Elements: Version 2 (Table 3.55)
Other nouns in the text such as call or incidence do not need to be defined as elements, although if they were defined, they would be collective elements.

3.9.2 *Food Service (Source: Larrañeta et al. 2003)*

A food service business has contracted four banquets for the next four days, requiring 150 clean tablecloths for the first banquet, 100 for the second, 140 for the third, and 130 for the fourth. Currently, it has 200 tablecloths in the storeroom,

Table 3.56 Table of Elements of Example 3.9.2

Elements	Set	QN	Data				
			Name	Param	Type	Belonging	Value
Banquets	$i = 1\ldots4$	I_U	Tablecloths	m_i	I	S	. . .
			Day	d_{it}	B	S	. . .
Storeroom	–	I_U	Stock	S	I	S	200
Market	–	I_U	Price	p	C	S	12
Basket	–	I_U					
Laundry	–	I_U					
Fast wash	–	I_U	Cost	cF	C	S	6
Slow wash	–	I_U	Cost	cL	C	S	4
Days	$t = 1\ldots4$	I_U		d_{it}			
Tablecloths	–	C_I		$m_{i;}$ S; p; cF; cL			

all of them clean, and they can buy each day what you need in the market, at a cost of 12 u.m/tablecloth.

After the banquets, the tablecloths can go to the laundry basket, or you can send them to be washed in the laundry. The laundry offers the following washing service:

– *Fast: Wash tablecloths for the next day, at a cost of 6 u.m/tablecloth.*
– *Slow: Wash tablecloths for 2 days, at a cost of 4 u.m/tablecloth.*

The washing processes, fast and slow, could have been included in a set. In spite of being part of the laundry, it is necessary to consider it as an element since activities fall on them. The laundry is an element that fades into the background, and it could even be removed from the table (Table 3.56).

3.9.3 Location of TV Cameras (Source: Larrañeta et al. 2003)

CPL has to televise the game of the year. The producers have identified 10 possible locations for the installation of cameras and 25 stadium areas that need to be covered by the cameras. The table below indicates the relationship between both:

Location	Covered area
1	1, 3, 4, 6, 7
2	8, 4, 7, 12
3	2, 5, 9, 11, 13
4	1, 2, 18, 19, 21
5	3, 6, 10, 12, 14
6	8, 14, 15, 16, 17

(continued)

Location	Covered area
7	18, 21, 24, 25
8	2, 10, 16, 23
9	1, 6, 11
10	20, 22, 24, 25

Each area of the stadium must be covered by a camera
Location 9 must have a camera
Areas 1 and 2 require coverage of at least two cameras
The objective is to minimize the number of cameras used

In "Minimal coverage" we have recorded the minimum number of cameras that are required for coverage in each area, being two for areas 1 and 2 and for the rest one camera (Table 3.57).

3.9.4 Trip Planning

There is a system that assigns travellers to buses. We have a group of 180 travellers who have hired the services of the TOURBUS Company for today. There are five trips offered. Each traveller has chosen one of the five trips.

Each traveller also chooses the language (English and Spanish) for explanations. They have three options to choose from:

- *Spanish*
- *English*
- *Both of them (if they speak English and Spanish)*

The buses have a capacity of 60 seats. There are eight buses.

Each bus that is used must be configured with a language and a trip. Since the explanations are given on the bus journey, it is necessary to place each traveller so that the trip and the language of the explanations that are configured on the bus taking them are compatible with their choice.

TOURBUS wants to use as few buses as possible to cover the trips (Table 3.58).

We have configured the buses as individual, even though they are identical. The reason is that the buses are referred in an individual way all times in the specifications ("...Each bus that is used must be configured with a language...") and it is obvious that you have to choose a specific bus for each traveller; therefore, it is

Table 3.57 Table of Elements of Example 3.9.3

Elements	Set	QN	Data				
			Name	Par	Type	Belonging	Value
Areas	$i = 1\ldots25$	I_U	Coverage	C_{ij}	B	S	
			Minimal coverage	m_i	I	S	
Locations	$j = 1\ldots10$	I_U		C_{ij}			
Cameras	–	C_I		m_i			

Table 3.58 Table of Elements of Example 3.9.4

Elements	Set	QN	Data				
			Name	Param	Type	Belonging	Value
Travellers	$i = 1\ldots180$	I_U	Trip choice	E_{ij}	B	S	...
			Language choice	I_{ik}	B	S	...
Trips	$j = 1\ldots5$	I_U		E_{ij}			
Languages	$k = 1{,}2$	I_U		I_{ik}			
Buses	$r = 1\ldots8$	I_U	Capacity	K_r	I	S	60
Seats	–	C_D		K_r			

necessary to identify each bus individually. And in this case they are unitary because they do not have any measurable continuous data. They have an integer capacity attribute, the number of seats. The seats have been set up as element although there are no decisions about them.

On the other hand, we have identified the trips and the languages as elements. Regarding languages, the shared property I_{ik} will be reflected if the traveller has chosen Spanish ($I_{i1} = 1, I_{i2} = 0$), English ($I_{i1} = 0, I_{i2} = 1$), or both ($I_{i1} = 1; I_{i2} = 1$).

Reading the text, travellers are treated in a particular way, both to note the values of their data and in the following specification regarding the assigned bus:

Since the explanations are given on the bus journey, it is necessary to place each traveller so that the trip and the language of the explanations that are configured on the bus taking them are compatible with their choice.

It is easy to realize that there are 15 different groups of travellers where each group would be made up of an identical set of travellers regarding the data of the system. If there are 5 trips and 3 different options to configure the language data, that means that there are 15 different types of travellers (Trip 1 + Spanish, Trip 1 + English, Trip 1 + Any Language, Trip 2 + Spanish, …, Trip 5 + Any Language). This means that we could treat each group as a collective element, since their items are identical, provided that the text does not deal in a particular way with the items. This is feasible by changing the description of the previous specification to an equivalent description:

"Since explanations are given on the bus journey, if we place travellers of a group on a bus, the trip and the language of the explanations that are configured on that bus must be compatible with the group's choice."

Now travellers are referred to collectively, as items of the travellers groups.

"If we locate travellers…" is equivalent to "If we place a number of travellers…".

The reduced table would look like the following (Table 3.59).

This table will suppose a considerable reduction in the number of decision variables of the problem with respect to the original table.

Table 3.59 Reduced Table of Elements of Example 3.9.4

Elements	Set	QN	Data				
			Name	Param	Type	Belonging	Value
Groups of travellers	$i = 1\ldots15$	C_D	Trip choice	E_{ij}	B	S	…
			Language choice	I_{ik}	B	S	…
Trips	$j = 1\ldots5$	I_U		E_{ij}			
Languages	$k = 1{,}2$	I_U		I_{ik}			
Buses	$r = 1\ldots8$	I_U	Capacity	K	I	S	60
Seats	–	C_D		K_r			

Table 3.60 Table of Elements of Example 3.9.5

Elements	Set	QN	Data				
			Name	Param	Type	Belonging	Value
Tasks	$i = 1\ldots n$	I_U	Start time	S_i	C	W	…
			End time	E_i	C	W	…
			Weight	P_i	C	W	…
Machines	$j = 1\ldots m$	I_U					

3.9.5 *Fixed Job Scheduling Problem (Kroon et al. 1995)*

There is a set of n tasks with a given start and end time and a weight. There is also a set of m machines. It deals with selecting tasks to be processed in the machines so that the selected tasks have a maximum weight. A selected task is processed completely on a single machine. A machine cannot perform two tasks overlapped in time.

It is a classic problem within scheduling, specifically a type of problem within the interval scheduling problem (Kolen et al. 2007).

Two sets of elements, tasks and machines, are clearly identified. Each task has its own values in its attributes, and they are treated in a complete way; they are not divisible. It is not allowed to divide it into parts, and those parts can be processed in different machines.

The machines are identical, but they cannot be treated collectively because the specifications description alludes to them individually ("A machine cannot perform two tasks overlapped in time") (Table 3.60).

Since the system states in the specification "A machine cannot perform two tasks overlapping in time" to the concept of overlap between tasks, we can make this characteristic explicit between tasks in a new calculated attribute (Sect. 3.2.5):

We define O_{ik} between tasks i and k, of value 1 if tasks i and k overlap in time and 0 otherwise. The calculation to define the attribute would be:

IF ($I_i <= I_k$ and $I_k < F_i$) *or* ($I_i > I_k$ and $F_k > I_i$), THEN $O_{ik}=1;$ OTHER CASE $O_{ik}=0.$

O_{ik} would be incorporated as a shared attribute between the tasks.

Table 3.61 Table of Elements of Example 3.9.6

			Data				
Elements	Set	QN	Name	Param	Type	Belonging	Value
Health centers	$i = 1\ldots12$	I_U	Distance	D_{ij}	C	S	...
			Capacity	K_i	I	W	...
Citizens	$j = 1\ldots n$	I_U		D_{ij}			
City	–	I_U	Attendance	A	C	W	0,01

3.9.6 Health Centers

There is a city formed by 12 health centers. Due to population changes, it has been decided to reassign Health Centers to citizens, a total of n. We know the address of each citizen, and therefore, the system allows us to know the distance between their home and each Health Center. Each health center has a capacity that is expressed in the number of patients that can be attended per day. It is estimated that 1% of people go daily to health centers. The objective is to minimize the sum of the distances of each citizen from the health center assigned (Table 3.61).

References

Balakrishnan, V. K. (1997). *Graph theory* (1st ed.). New York: McGraw-Hill.

Bang-Jensen, J., & Gutin, G. (2000). *Digraphs: Theory, algorithms and applications*. Springer.

Cunningham, K., & Schrage, L. (2004). The LINGO algebraic modeling language. In J. Kallrath (Ed.), *Modeling languages in mathematical optimization. Applied optimization* (Vol. 88). Boston: Springer.

Kolen, A., Lenstra, J. K., Papadimitriou, C., & Spieksma, F. (2007). Interval scheduling: A survey. *Naval Research Logistics, 54*, 530–543.

Kroon L G, Salomon M and Van Wassenhove L N (1995) Exact and approximation algorithms for the operational fixed interval scheduling problem. European Journal of Operational Research 82: 190-205

Larrañeta, J., Onieva, L., Cortés, P., Muñuzuri, J., & Guadix, J. (2003). Métodos Cuantitativos en Ingeniería de Organización. Editorial Universidad de Sevilla.

Martello, S., & Toth, P. (1990). *Knapsack problems: Algorithms and computer implementations*. Wiley-Interscience.

Meléndez, I. (2019). https://www.monografias.com/trabajos96/distribucion-redes-administracion-proyectos/distribucion-redes-administracion-proyectos.shtml#bibliograa. La distribución de redes y la administración de proyectos.. Accessed June 2019.

Chapter 4
Decision Activities

4.1 Introduction

Decision activities are independent actions of undetermined value that are carried out in the system. They are translated into the decision variables of the problem.

A decision activity is made up of the following components:

- The action that determines the activity: this corresponds to a verb.
- The elements that participate in the activity: elements of the system must participate in the action that corresponds to the decision activity.
- The quantitative meaning of the action: the quantification of the action defines the type of value of the variables. The meaning of an action can be of two types:
  - **Measure**: the result of the action is a value referring to any continuous measure (liters, kilos, time, etc.) or discrete measure (number of units) of an element. It corresponds to continuous or integer variables, respectively. If the measurement of units is bounded superiorly by one, the integer variable can be defined as binary. It is necessary that a measurable element (measurable individual or collective) participates in a measurement activity as object direct of the action.
  - **Logical**: the action corresponds to an activity of choice or selection for an element or group of elements where the response is evaluated with a logical value, True or False. The activity is determined with binary variables (1/0). The "1" corresponds to a positive or true evaluation and the "0" to a negative. All activities where only unitary elements participate will be logical.

The original version of this chapter was revised. The correction to this chapter is available at https://doi.org/10.1007/978-3-030-57250-1_9

J. M. García Sánchez, *Modelling in Mathematical Programming*, International Series in Operations Research & Management Science 298
https://doi.org/10.1007/978-3-030-57250-1_4

The abovementioned factors are summarized in the definition of a decision variable. The definition of a variable must contain the elements that participate, the action, and the unit of measurement if the action is quantified as measure.

However, the identification of decision activities is not carried out in a singular way, but collectively or jointly. This occurs because the same action can be carried out independently by the participation of different groups of elements. The participation of each group defines a decision variable, but all the variables correspond to the same action in the system. Therefore, we will identify decision activities in the form of a set of events, in which an event is a valid association of elements participating in an action. The participation of the elements is unique with respect to the rest of the events. Each event corresponds to an individual decision variable regarding the joint decision activity.

In the definition of a variable, it is easy to identify errors in the design of the activity. You can detect inconsistencies, as a meaning without logic, an activity that has a known value and therefore is not a decision activity of the system, or simply a definition that corresponds to a function or calculation, for the wrong use of the participating elements. The variables generated from the decision activities must be independent. Their values are not obtained from any previous calculation.

Not all variables of a problem are independent. There are also the variables that store calculations. In them, the value is always obtained from the value of other variables, by calculating a linear function or a conditional function. The calculations of a system generate the calculation variables, which will be discussed in Chap. 5.

We shall now illustrate the definition of a decision activity and the analysis of inconsistencies without defining any rules, at least for now, in the definition of activities. We shall start with the first example.

Illustration 4.1: Production of Butter

A butter production factory wants to optimize its daily production of butter. Two types of butter are made (Sweet and Raw). A kilo of sweet butter gives the manufacturer a profit of $10 and a kilo of raw a profit of $15. For the production of butter, two machines are used: a pasteurization machine and a whipping machine. The daily use time of the pasteurization machine is 3.5 hours and 6 hours for the whipping machine. The time (in minutes) consumed by each machine to obtain a kilo of butter is shown in the following table:

Table 1. Butter processing times (in minutes)

	Sweet butter	Raw butter
Pasteurization	3	3
Whipping	3	6

Table of Elements (Table 4.1)

Decision Activities

The only action we can extract from the wording is “manufacture” butter, which is also used with the synonym “produce.”

Table 4.1 Elements of Illustration 4.1

			Data				
Elements	Set	QN	Name	Par	Type	Belonging	Value
Machines	$i = 1\ldots2$	$\mathrm{I_U}$	Usage time	T_i	C (Min)	W	…
			Time consumed by 1 kg of butter j in machine i	TM_{ij}	C (Min)	S	…
Butters	$j = 1\ldots2$	$\mathrm{I_M}$	Benefit	B_j	C ($/kg)	W	…
				TM_{ij}			

Action: PRODUCE.

Participating elements: The set of butters ($j = 1\ldots2$), that is, Sweet Butter and Raw Butter.

Quantification: The action of producing must obtain the quantity of butter produced since the quantity of butter is a property of undetermined value. This determines the measurable character of the butters. The unit of measurement used is the kilo.

Events: the activity generates two events:

Produce ⇒ Butter j=1 (Sweet).

Produce ⇒ Butter j=2 (Raw).

These events give rise to two **decision variables**:

x_1 = Kilos of sweet butter produced.

x_2 = Kilos of raw butter produced.

The information of the decision activities in simplified form would be:

Produce ⇒ Butter $j = 1,2$.

Decision variables: x_j = Kilo of butter type j produced.

If we analyze the definition of a variable, we can see the components (elements that participate in the event, action, and unit of measurement) that identify the activity.

Kilos	of sweet butter	Produced
Unit of measure	**Participating element**	**Action**

Incorrect Definitions

Let us consider some incorrect definitions we could have made of the activity. In Sect. 4.3.1, within the section dedicated to elements participating in a decision activity, and in Sect. 4.4 regarding quantification of the activity, we will present a series of rules that avoid these incorrect definitions.

1. *Suppose we identify a single event, in which the two types of butter participate:*

   Produce ⇒ Sweet Butter, Raw Butter

   A single variable is generated:

   x = Kilos of sweet butter and raw butter produced.

The definition of the components is correct, since it identifies the participating elements, the action, and the unit of measurement. However, the semantics could be understood in two ways:

A. We are assuming that the production of each type must be the same.
B. The activity represents the sum of the production of the two types of butter.

In proposition A, we make the mistake of assuming something that the system does not specify. On the other hand, the definition of a decision variable must not express a specification. The modelling of the specifications of a system has its own space that is carried out after the definition of activities. The decision activities must be defined independently, and then the specifications will impose the values they can take.

In proposition B, we are representing a function in the definition of the variable. We are adding up the production of the two types of butter. Therefore, we are defining a calculation on the correct decision variables, the amount of sweet butter and the amount of raw butter ($x = x_1 + x_2$).

2. *Suppose we incorporate the pasteurization machine into the participating elements.*
   Produce ⇒ Sweet Butter, Raw Butter, Pasteurization Machine
   For this action we define two events:
   Event 1: Produce ⇒ Sweet Butter, Pasteurization Machine.
   Event 2: Produce ⇒ Raw Butter, Pasteurization Machine.
   We analyze one of the events:
   Event 1: Produce ⇒ Sweet Butter, Pasteurization Machine.
   Variable: x_1 = Kilos of sweet butter produced in the pasteurization machine.

The semantics of the definition is correct and reflects all the components of the activity. The incorrectness is in considering the participation of the pasteurization machine in the decision of the activity. The kilos that are produced from sweet butter are equal to the kilos of sweet butter that are processed in the pasteurization machine. The machine does not contribute anything to the decision of how much to produce.

Kilos of sweet butter produced = Kilos of sweet butter produced in the pasteurization machine

Therefore, the participation of the pasteurization machine element in the decision can be suppressed.

When the element is implicit as a participant in all the events, it is not necessary to identify it, although there are occasions when, due to the clarity of the definition, it is maintained. In this case, it was not necessary.

3. *Suppose we identify the type of quantification as whole: the definition of the variables would be:*

   x_1 = N° of units of sweet butter produced.
   x_2 = N° of units of raw butter produced.

Obviously it is wrong because to define integer decision variables, there must be a collective element in the decision activity. If we take the unit as the kilo, we would be restricting the production to an integer number of kilos.

We are making the mistake of associating a discrete quantization with a continuous measurable element. The integer quantification is exclusively associated with the measurement of collective elements.

4. *Suppose we identify the type of quantification as binary: in this case the definition of the variables would be:*

$\alpha_1 = 1$ if we produce Sweet Butter; 0 if we do not produce Sweet Butter.
$\alpha_2 = 1$ if we produce Raw Butter; 0 if we do not produce Raw Butter.

This error is usually quite common in modelling. By defining the activity with this type of value, we do not have information about the quantity that we produce, and we will not be able to express the specifications and the objective of the problem. It is enough to analyze the data associated with the butters that always refer to the unit of measure of their quantity, both the production times in the machines and the profit.

The definition made of the activity, which obviously does not correspond to a decision activity but to a calculation, is a logical calculation. If we look at this in more depth, the values of α_1 and α_2 are obtained from the value of the correct decision variables of the activity (x_1 = kilos of sweet butter produced; x_2 = kilos of raw butter produced) via the following logical proposition:

If $x_1 > 0$ then $\alpha_1 = 1$; if $x_1 = 0$ then $\alpha_1 = 0$.

This would also be the case for α_2.

In the chapters dedicated to logical calculations and to specifications expressed as propositional logic, we will carry out an in-depth study of the use of conditional formulas and their modelling.

4.2 Actions of a System

The actions of a system correspond lexically with verbs (buy, sell, send, produce, install, assign, select, etc.) that may be accompanied by adverbs or prepositions. Incorporating the participating elements, to any action in a system it is possible to assign a type of value, either integer, continuous, or binary.

The actions with their participating elements that give rise to decision activities are actions of indeterminate value and are not always dependent on the value of other decision activities. Actions that do not fulfill these two properties cannot be considered as decision activities, either because they have a value already assigned or because they take values that are obtained from the values of other activities, that is, they always depend on other activities. These actions will result in system specifications.

As actions that can give rise to decision activities, those verbs that denote imposition (impose, limit, restrict, etc.) are excluded, since they are ways of defining specifications but do not define an activity in the system. Also, actions that can be missed out of the text because they act in an explanatory way, i.e., without any capacity to give rise to activities or specifications, are associated with a defined type of binary value. Let us take a look at an example:

Illustration 4.2
There is a system of buying a product from suppliers where you must encourage each supplier to buy at least 20 units.

Encourage is an action, but its value is determined; the wording clearly specifies that "we must encourage." This action could be eliminated from the wording without any problem, being as follows:

System of buying a product from suppliers where each supplier must buy at least 20 units

4.2.1 Actions with Calculated Value

These actions should not be considered as decisions because their value can always be calculated using other variables of the problem. The system allows to obtain the value of the action from the variables of other decision activities and even from other calculations.

If the actions of calculated value have been defined as decision activities, we must never forget the specification of the calculation that defines the action. That is why it is more advisable not to define them as an activity and to define them as a calculation, so as not to forget the restrictions that define it, as will be discussed in Chap. 5.

In terms of representing the calculation, the following will be involved:

Non-conditional Action
The activity value is determined directly by a linear mathematical function on other variables of the problem.

The actions with linear value will be part of what we will call auxiliary calculations.

Illustration 4.3
There is a system of buying and selling a product. To buy we have a set of three suppliers, and the products are subsequently sold on the market at a price of € p/unit. You must sell 50% of what you buy.

Buying is a decision activity. However, selling is not a decision activity because it can be defined as an auxiliary calculation:

$$\text{Selling} \equiv y = \frac{x_1 + x_2 + x_3}{2}$$

With x_1, x_2, x_3 decision variables of the activity of buying product from suppliers.

If the objective function manages costs and benefits, it will take the term py as a benefit.

Conditioned Action or Conditional Value Action

The definition of the action establishes the conditions to determine whether or not it occurs or the value it will take. The value of the action is determined by conditional propositions on other variables of the problem.

The conditioned actions will be part of the logical calculations and can be of two types:

– Conditioned action with a determined value: The action has an associated value. They are associated with binary variables.
– Linear value conditioned action: The value of the action is obtained from a function. They will be defined according to the value of that function.

Let us illustrate the two cases:

Illustration 4.4: Conditional Action of Determined Value

The system of Illustration 4.3 *has an activity to pay a fee of €100 if the units purchased exceed 200 units.*

The action would be Pay [a fee]. The fee would become an element of the system with an attribute of value equal to €100. The action of paying the fee is not independent; it is conditioned to the decision variables of buying. The value of the conditioned action is binary:

$$\alpha = \begin{cases} 1 & \text{if I pay the fee} \\ 0 & \text{otherwise} \end{cases}$$

The conditional proposition that defines its value is the following:

If the purchased units > 200 then I pay the fee, otherwise I do not pay the fee.
Mathematically: *If* $x_1 + x_2 + x_3 > 200$ *then* $\alpha = 1$, *otherwise* $\alpha = 0$.

The value of \$100 will be associated with the variable of the logical calculation α in the system cost function with the term 100α.

Illustration 4.5: Conditional Action of Linear Value

The system of Illustration 4.3 *has an activity of paying a fee of 1% of the total purchased if the units purchased exceed 200 units. The purchase price of the product is c \$/unit.*

The action is again Pay a fee, but in this case the fee does not have a certain value but the result of a linear function on the total purchased. Specifically, we can define the linear function as follows:

Price of the Fee $= 0, 01(x_1 + x_2 + x_3)c$

If we defined the conditioned action as binary, the cost function would have to be non-linear:

$$\alpha = \begin{cases} 1 & \text{if I pay the fee} \\ 0 & \text{otherwise} \end{cases}$$

If $x_1 + x_2 + x_3 > 200$ *then* $\alpha = 1$, *otherwise* $\alpha = 0$.

In the cost function of the system, we would associate the variable α with the function that calculates the fee $0, 01(x_1 + x_2 + x_3)c$, by means of the non-linear expression $0, 01(x_1 + x_2 + x_3)c\alpha$.

Since we always try to avoid non-linear expressions, the correct way would be to define the action of paying the fee with the value of the function that obtains the price of the fee, in this case a continuous value:

The logical calculation would be represented in a continuous variable z:

$z =$ Fee paid.

The conditional proposition that defines the calculation is:

If $x_1 + x_2 + x_3 > 200$ *then* $z = 0, 01(x_1 + x_2 + x_3)c$, *otherwise* $z = 0$.

In this way we will maintain the system with linear expressions.

Although we have entered into the logical calculations of a system, they will be examined in depth in Chap. 5.

4.2.2 Actions with Undetermined Value

Actions of undetermined value give rise to the decision activities of the system and therefore to the decision variables. The decision activities have values independent of the rest of the problem variables. Notwithstanding this, the specifications of the system can condition the values that the decision variables can take up to a point of being able to convert a decision variable into a variable of a calculated action. But a priori, without the specifications of the system, these values are free and are not decided by a calculation with respect to other variables. Let us take a look at a very simple example to illustrate this fact.

Illustration 4.6
In the purchase system of Illustration 4.3, *we define a specification that requires the total units purchased from the product to be 100.*

This imposes a constraint on the model of the form:

$x_1 + x_2 + x_3 = 100$

Table 4.2 Elements of Illustration 4.7

			Data				
Elements	Set	QN	Name	Parameter	Type	Belonging	Value
Stores	$i = 1...N$	$\mathrm{I_U}$					
Raw material	–	$\mathrm{I_M}$					

From this specification, the values of x_1, x_2, and x_3 are no longer completely independent, since $x_1 = 100 - x_2 - x_3$, $x_2 = 100 - x_1 - x_3$, or $x_3 = 100 - x_1 - x_2$. It could be understood that any one of these decision variables is an auxiliary calculation of the remaining two and therefore could have been defined as a calculation instead of as a decision activity. It is inevitable to find situations analogous to this, where the two interpretations fit. My opinion is that it is more structured to define them as decision activities and then represent the specification that relates them. Anyway, either of the two options leads to the same mathematical model.

On the other hand, decision activities can determine values of other decision activities, but for the latter to be defined as calculation rather than as a decision activity, the first decision activities must determine the value of the action for any of the values they take, without exceptions. If only the value of the action is determined for a subset of the values of the decision activities, the action retains its indeterminate nature and therefore prevails as a decision activity. Let's see a significant illustration of this fact:

Illustration 4.7

There is a system in which stores are rented and raw materials are stored within them. The system has a set of N stores that can be rented for storage.

The table of elements could be presented in the following way, taking into account the wording (Table 4.2):

Two actions are clearly identified, rent and store:

Action: Rent [stores].
Participating elements: Stores ($i = 1...N$ $\mathrm{I_U}$).
Quantification: Binary.
Events: Stores $i = 1...N$.
Decision variables:

$$\alpha_i = \begin{cases} 1 & \text{if rent store } i \\ 0 & \text{otherwise} \end{cases}$$

Action: Store [raw material in stores].
Participating element: Raw material $\mathrm{I_M}$; Stores ($i = 1...N$ $\mathrm{I_U}$).
Quantification: Continuous.
Events: Raw material $\Rightarrow$ Stores $i = 1...N$.
Decision variables: x_i: Amount of raw material stored in Store i.

Although not explicitly described, it cannot be stored in a store that you have not rented (the implicit specifications of a system will be discussed in Chap. 6). This establishes a partial determination of values between the two groups of variables:

If you have not rented Store i, you cannot store in store i:
$\Rightarrow$ *If* $\alpha_i = 0$ *then* $x_i = 0$.

One of the values of the Rent activity determines the value of the Store activity. However, for the value $\alpha_i = 1$, a value for x_i cannot be determined, so it does not determine x_i as a calculation.

On the other hand, activity x_i also partially determines the values of α_i, since if I have stored something in store i it is because I have rented the store:

$x_i > 0 \rightarrow \alpha_i = 1$

However, the missing value of x_i, $x_i = 0$, does not determine the value for α_i, because it could have rented the store and not stored any raw material.

Regarding the latter, it is quite reasonable to think, if there are no more elements in the system that must be taken into account to store in the stores or other activities related to the stores, that if we rent a store, with the supposed cost that this entails, it will be to store raw material. This has an important consequence: we can convert the activity of renting into an action of calculated value, and therefore it is no longer an activity decision of the system. In this case, we can contemplate the calculation of all values of α_i:

$x_i > 0 \rightarrow \alpha_i = 1$
$x_i = 0 \rightarrow \alpha_i = 0$

4.3 Participating Elements in a Decision Activity

Obtaining the right list of participating elements of an activity can be a complex task in some problems. In fact, it is the phase with the least ability to be regulated. Despite this, we will try to give some guidelines for a correct selection.

Indicating the elements that participate in an action requires full knowledge of the system and a table of elements that is defined correctly.

Obtaining the elements involved in an activity is based on looking for relationships between the elements of the system and the action. The search is based on asking to the action looking for as answer to the element.

The questions we can ask the action are dependent on the meaning of it. Questions about the value of the activity are logically excluded, like the question how much? Thus, they are generally questions of the type “What?”, “For what?”, “Who?”, “Where?”.

If the action is of logical value, obtaining the elements is usually simpler. You have to look for the elements that we associate with the election.

In general, if for example I have a Send activity, the logical thing is to ask What do I send? It will also seem logical to ask where do I send it from? And where do I send it? Asking the action is simply an informal tool to help identify the participating elements. The key is to have a rounded knowledge of the system and to fully understand the meaning of the activity.

4.3.1 Rules of Participation

Despite being an unregulated task, it is possible to define some work premises in the participating of the elements:

1. **An event of a measurable action can only measure a single element.**

In an event we cannot consider the quantification of more than one element since that would mean establishing a function, and a decision activity is an independent event, not the result of a calculation or function.

Illustration 4.8: Production of Butter
In our basic problem, the action we identified as a decision activity was PRODUCE. What do we produce? Sweet Butter and Raw Butter.

There are two measurable elements so we cannot measure both in the same event. It is necessary to generate two events:

PRODUCE Sweet Butter
PRODUCE Raw Butter

2. **In an action in which we identify the participation of an element that does not function as a direct object to the action, its participation combined or not with other elements should be an alternative or option among a set of alternatives, which will be collected in the events of the activity.**

There must be other elements that are alternatives to perform that action. If an element always participates in all the events of an action in a secondary way, we can exclude it because its participation is implicit. In spite of this, in some cases, to maintain a clear definition of the activity, we can keep it in participation.

Illustration 4.9: Production of Butter
As we have indicated in 4.1, with this standard we deduce that it is not necessary to include any of the machines in the participation, since its use is not an alternative but is obligatory in all the production processes and they are not the direct object of the action. In this case the direct object of the action is butter.

3. **The participation of an element cannot be subject to the participation of other elements.**

If the participation of an element is always produced by the participation of another, its participation should not be contemplated because there is no alternative.

Table 4.3 Elements of Illustration 4.10

Elements	Set	QN	Data				
			Name	Param	Type	Belonging	Value
Machines	$i = 1\ldots6$	$\mathrm{I_U}$	Production rate	R_i	C (parts/hour)	W	. . .
			Modes machines	MM_{ij}	B	S	. . .
Operators	–	$\mathrm{C_D}$	Quantity	Q	I	W	9
Modes	$j = 1\ldots4$	$\mathrm{I_U}$	Modes machines	MM_{ij}			
Parts	–	$\mathrm{C_D}$	Demand	D	I	W	10.000

Illustration 4.10: Production of Parts

There is a parts production system. For the production of parts, we have 6 machines and 9 operators. A part can be produced in 4 ways:

– Using machine 1 and machine 2
– Using machine 1 and machine 3
– Using machine 4
– Using machine 5 and machine 6

Each machine needs to have an operator for its operation and has a production rate in number of parts per hour. This production rate of parts increases by 20% if two operators are assigned to the machine.

Schedule the production of 10,000 parts in order to minimize the total production time. For simplicity and consistency with respect to the level we are currently at in the methodology, we discard the sequence concept and assume that machine 2 starts working when all the parts assigned to mode 1 have been processed in machine 1 (same with machine 3 and with the production in machines 5 and 6).

Table of Elements

The table of elements can be defined as follows (Table 4.3):

Each machine is individual and unitary because they do not have any measurable properties. MM_{ij} is a binary attribute to identify which machines belong to each part production mode. Operators are a single collective element, since their nine items are identical in the problem and there is no specification on each item individually. Modes that only have binary data must be unitary.

Decision Activities

The action of this system is to produce parts. The parts can be produced using four different forms or production modes. Therefore, each mode represents an alternative production and must participate in the action. However, it would be a mistake to consider the participation of the machines in the action of producing, even though the parts are produced in the machines. Considering the modes of production, each mode requires the participation of a subset of machines, those included in the MM_{ij} attribute, so that the participation of the machine elements is subject to production modes and therefore their participation should not be considered.

Table 4.4 Elements of Illustration 4.11

Elements	Set	QN	Data				
			Name	Param	Type	Belonging	Value
Suppliers	$i = 1\ldots N$	I_U	Origin	P_i	B	W	...
T-shirts	$j = 1\ldots 10$	C_D	Demand	DT_j	I	W	...
Trousers	$k = 1\ldots 3$	C_D	Demand	DP_k	I	W	...

Correct Decision Activity
Action: PRODUCE [parts].
Participating elements: Parts (C_D), Modes ($j = 1\ldots4$ I_U).
Quantification: Integer.
Events: Parts $\Rightarrow$ Modes $j = 1\ldots4$.
Decision variables: x_j = Number of parts produced in mode j; $j = 1\ldots4$.

Incorrect Decision Activity
Action: PRODUCE [parts].
Participating elements: Parts (C_D), Modes ($j = 1\ldots4$ I_U), Machines ($i = 1\ldots6$ I_U).
Quantification: Integer.
Events: Parts $\Rightarrow$ Modes $j = 1\ldots4 \Rightarrow$ Machines $i = 1\ldots6/MM_{ij} = 1$.
Decision variables:
x_j = Number of parts produced in mode j with m.achine i; $j = 1\ldots4$.
$i = 1\ldots6/MM_{ij} = 1$.

4. **Participation of a subset of elements within a set.**

From a table of elements associated into sets, the participation of a set of elements in a decision activity does not have to be complete, but we can only choose a subset of elements whose data meet a certain condition.

This consideration also occurs in the own generation of events, where we are not obliged to contemplate all the options of the combination of elements.

Illustration 4.11
Let us suppose a system of purchasing units of our products from a list of N suppliers. We purchase t-shirts of ten different models and trousers of three models. Demand data for each model is known. We have an attribute for the supplier regarding its origin (1: National; 0: Foreign). Foreign suppliers do not supply trousers.

Table of Elements

The table of elements for this description has the following structure (Table 4.4):

Since the text points to the number of t-shirts and trousers demanded, we class these elements as collectives.

Decision Activities

If we establish "Buy T-shirts and Trousers" as a decision activity, but taking into account that trousers are only purchased from national suppliers, the definition of the activity should reflect the options allowed:

Action: PURCHASE [T-shirts AND Trousers FROM Suppliers].
Participating elements: T-shirts ($j = 1\ldots10$ C_D); Trousers ($k = 1\ldots3$ C_D); Suppliers ($i = 1\ldots N$ I_U).
Quantification: Integer.
Events:
T-shirts $j = 1\ldots10 \Rightarrow$ Suppliers $i = 1\ldots N$.
Trousers $k = 1\ldots3 \Rightarrow$ Suppliers $i/P_i = 1$.
Decision variables:
x_{ij} = Number of t-shirts of model j purchased from supplier i; $i = 1\ldots N, j = 1\ldots10$.
y_{ik} = Number of trousers of model j purchased from supplier i; $i/P_i = 1$, $k = 1\ldots3$.

In the same way we could have considered two decision activities regarding the action "Purchase":

Decision Activity 1:
Action: PURCHASE [T-shirts FROM Suppliers].
Participating elements: T-shirts ($j = 1\ldots10$ C_D); Suppliers ($i = 1\ldots N$ I_U).
Quantification: Integer.
Events: T-shirts $j = 1\ldots10 \Rightarrow$ Suppliers $i = 1\ldots N$.
Decision variables:
x_{ij} = Number of t-shirts of model j purchased from supplier i; $i = 1\ldots N, j = 1\ldots10$.

Decision Activity 2:
Action: PURCHASE [Trousers FROM Suppliers].
Participating elements: Trousers ($k = 1\ldots3$ C_D); Suppliers ($i = 1\ldots N$ I_U).
Quantification: Integer.
Events: Trousers $k = 1\ldots3 \Rightarrow$ Suppliers $i/P_i = 1$.
Decision variables:
y_{ik} = Number of trousers of model j purchased from supplier i; $i/P_i = 1$, $k = 1\ldots3$.

5. **Elements with participation subject to conditions.**

When the participation of an element in a decision activity is subject to that element, it fulfills some condition on other variables (decision activities or calculations), something that a priori is not determinable, and its participation in the activity will always be considered, and the condition of the definition in the activity will be excluded.

Subsequently, a specification will control the values of that activity depending on the condition. Let us take a look at an example.

Table 4.5 Elements of Illustration 4.12

			Data				
Elements	Set	QN	Name	Param	Type	Belonging	Value
Centers	$i = 1\ldots3$	I_U					
Vehicles	$j = 1\ldots10$	I_U					
Pallets		C_I					

Illustration 4.12
There is a system in which vehicles are allocated to delivery centers. There are ten vehicles and three delivery centers, each with a given location. For vehicles that are assigned to more than one center, we also have to decide on the number of pallets assigned.

Table of Elements

Although the text is simple and does not incorporate a lot of information regarding data and specifications, with the information that is provided on each set of elements (centers, vehicles, and pallets), we can define centers and vehicles as individual elements and pallets as collective. It specifies that each center has own values in its data. Regarding the vehicles, they are identical but are defined as unitary by the specification stating that "For vehicles that are assigned to more than one center, we also have to decide on the number of pallets assigned," where each vehicle is considered individually (we could have replaced "vehicles" for "each vehicle"). Regarding pallets, we decided to consider them collectives because they are identical items with an indeterminate number in the system.

We define therefore the following table of elements (Table 4.5).

Decision Activities

From the text, it is easily perceived that there are two activities: assign vehicles to centers and assign pallets to vehicles. In this second activity, it is not a priori determined which vehicles will participate, since it depends on the activity of assigning centers to vehicles, since the statement states that "For vehicles that are assigned to more than one center, we also have to decide the number of pallets assigned." As we have said, in these cases it is necessary to consider the participation of all vehicles, without any condition. In the specifications section, it will be necessary to contemplate this condition. Therefore, the activities would be defined as follows:

Decision Activity 1:
Action: ASSIGN [Centers to Vehicles].
Participating elements: Centers ($i = 1\ldots3$ I_U); Vehicles ($j = 1\ldots10$ I_U).
Quantification: Binary.
Events: $i = 1\ldots3 \Rightarrow j = 1\ldots10$.
Decision variables: $\alpha_{ij} = 1$ if we assign center i to vehicle j; 0 otherwise. $i = 1\ldots3$. $j = 1\ldots10$.

Decision Activity 2:
Action: ASSIGN [Pallets to vehicles].
Participating elements: Vehicles ($j = 1...10$ $\mathrm{I_U}$); Pallets $\mathrm{C_{I.}}$
Quantification: Integer.
Events: Pallets $\Rightarrow j = 1...10$.
Decision variables:
x_j = Number of pallets assigned to vehicle j; $j = 1...10$.

4.4 Quantification of the Activity

Every system requires an analysis of its decision activities, an analysis to determine what I need to obtain from the decisions. Among the elements participating in the action, the element that acts as direct object to the action, its capacity to be measurable and the quantitative analysis of the action on the element, must determine the type of variable that is generated in the decision activity.

The direct object is a grammatical issue necessary to understand the quantification of the activity. The direct object is the recipient of the action; it is the thing being acted upon, the receiver of the action. Let us see some examples:

- "Purchase products from suppliers": The products are the direct object of the action, what I purchase. The suppliers are an indirect object.
- "Making butter at the factory": Butter is the direct object of the action, what I make.
- "Placing the object on the shelf": The object is the direct object of the action.

When an action is measurable, the element being measured must necessarily act as direct object. Carrying out a previous analysis of the elements that can be measured helps to establish the quantification of the activity. Let us look at some illustrations on quantifying decision activities.

Illustration 4.13
There is a set of 10 workers and a set of 25 jobs. Each job has an affinity value between 0 and 1 with each operator. It involves assigning jobs to operators so that each operator does at least 2 jobs and maximizes the total affinity of the assignment. Each work must be assigned to a single operator.

Table of Elements

The elements that the description reflects are the jobs and the workers. Affinity is a shared attribute between each job and each worker.

The activity of the system "assign jobs to workers" does not measure either workers or jobs but has a logical meaning or choice between elements.

Each job is unitary for several reasons:

Table 4.6 Elements of Illustration 4.13

Elements	Set	QN	Data				
			Name	Param	Type	Belonging	Value
Jobs	$i = 1\ldots25$	$\mathrm{I_U}$	Affinity	A_{ij}	C	S	...
Workers	$j = 1\ldots10$	$\mathrm{I_U}$		A_{ij}			

– Each job is individual because it has an attribute, affinity on each worker, with an own value not necessarily equal to the rest.
– It is also individual because it refers in an individual way to each job in the specifications (each job must be assigned to a single operator).
– Each job does not have the capacity to be divisible in the system, so it cannot be measured continuously, and therefore will have a unitary character.

The same attribute of affinity also makes us consider the workers as individual elements, and since their data have different values, they are differentiated. Also, the specifications treat each operator in a particular way. On the other hand, workers do not have any property with the capacity to be measurable, so they are also unitary elements.

The table of elements corresponds to the following structure (Table 4.6).

Decision Activities

As we have said, the only action in the system is “assign jobs to workers.”

Action: ASSIGN [Jobs to Workers].
Participating Elements: Jobs $i = 1\ldots25$ $\mathrm{I_U}$; Workers $j = 1\ldots10$ $\mathrm{I_U}$.
Quantification: Binary.
Events: $i = 1\ldots25 \Rightarrow j = 1\ldots10$.
Decision variables:
$\alpha_{ij} = 1$ if I assign Job i to Worker j; 0 otherwise. $i = 1\ldots25, j = 1\ldots10$.

The works are the element that I assign to the operators, which act as an indirect object of the action.

In the example a significant characteristic is revealed, the same action can be expressed using another element as a direct object. We could also have written: Assign Operators to Jobs. In that case, the quantification would not have changed because the workers are also unitary individual elements.

Illustration 4.14
We add the following information to the problem of Butter Production:

The system must also assign three workers from the factory to the production of each type of butter. The company has a staff of 15 workers, and there is a cost for the allocation of each worker to each type of butter.

Table of Elements

Table 4.7 Elements of Illustration 4.14

Elements	Set	QN	Data				
			Name	Param	Type	Belonging	Value
Machines	$i = 1 \ldots 2$	I_U	Usage time	T_i	C (Min)	W	...
			Time consumed by 1 kg of butter j in machine i	TM_{ij}	C (Min)	S	...
Butters	$j = 1 \ldots 2$	I_M	Profit	B_j	C ($)	W	...
				TM_{ij}			
			Worker cost	C_{kj}	C ($)	S	...
			Number of workers	N_j	I	W	3
Workers	$k = 1 \ldots 15$	I_U		C_{kj}			...

The new functions of the system add information to the table, which now has the following structure (Table 4.7).

The workers are incorporated as unitary elements, sharing with each type of butter the cost of assigning each worker to each type of butter.

In addition, each butter also incorporates the number of workers needed, 3, the same amount for each butter. Although the attribute refers to the number of workers, the attribute is own because it cannot be shared with each worker, because each worker can assume only its individual information. If workers were a collective element, it would also be the owner of the attribute.

Decision Activities

"The system also has to assign three workers from the factory to the production of each type of butter."

In addition to producing butter, the system presents a new decision activity, "Assign workers to each type of butter."

As we discussed, no numerical data is included in the definition of an activity, so we ignore the concept of assigning "three" workers to each butter. This information will be used in the specifications. The activity would be configured as follows:

Action: ASSIGN [Workers to Butters].
Participating elements: Workers $k = 1 \ldots 15$; Butters $j = 1{,}2$.
Quantification: The action falls on the workers, so it is a logical activity.
Events: Workers $k = 1 \ldots 15 \Rightarrow$ Butters $j = 1{,}2$.
Decision variables: $\alpha_{kj} = 1$ if I assign butter j to *worker* k; 0 otherwise.

If we try to define the activity as "Assign butters to workers," the butters as direct objects, which are measurable elements, we must realize that the amount we assign to a worker will correspond to the total butter produced or none; therefore we are not measuring the butters, we are defining a logical decision.

To illustrate measurable activities, we propose a classic problem in the world of mathematical optimization and another example in which most of the participating elements are continuous and measurable.

Illustration 4.15

A company has m warehouses where its products are located. Each warehouse A_i ($i = 1 \ldots m$) has a stock of K_i units. There is a set of n Customers ($j = 1 \ldots n$) with a demand D_j of product units. The company must supply the customers' demand of products from the warehouses. The cost of sending a product from each warehouse A_i ($i = 1 \ldots m$) to each customer C_j ($j = 1 \ldots n$) is estimated in c_{ij}.

Table of Elements

The elements that are identified are:

– The company: the system itself, an implicit individual element in all problems.
– The products: element formed by a set of identical items. It will be considered collective since we do not need to consider each product unit individually.
– The m warehouses (A_i, $i = 1 \ldots m$): Each store is necessary to consider it individually since it has an attribute with own value, the number of products. The use of this attribute will be measured in the system, although this is already included in the quantitative nature of the product, so we can consider each warehouse as unitary. When elements are defined abstractly with an index, they are already being defined in a set.
– The n customers (C_j, $j = 1 \ldots n$): Same as the warehouses, it is a set of unitary elements.

Nouns that refer to data:

– Stocks, K_i product units that each warehouse owns.
– The demand D_j of product units that each client owns.
– The cost of sending a product from each warehouse to each customer.

All data that refer to product units are also attributable to the product and therefore are shared with it.

All this is reflected in the following table of elements (Table 4.8).

Decision Activities

Action: SEND [products from warehouses to customers].

Participating elements:

Products (What do I send?) *Direct object.*

Warehouses (Where do I send it from?).

Table 4.8 Elements of Illustration 4.15

			Data				
Elements	Set	QN	Name	Param	Type	Belonging	Value
Warehouses	$i = 1 \ldots m$	I_U	Stock	K_i	I	S	...
			Cost	C_{ij}	C	S	...
Customers	$j = 1 \ldots n$	I_U	Demand	D_j	I	S	...
				C_{ij}			
Products	–	C_D		K_i; D_j; C_{ij}			

Customers (Who do I send it to?).
Quantification: Integer.
Events: Product $\Rightarrow i = 1...m \Rightarrow j = 1...n$.
Decision variables:
x_{ij} = Number of product units sent from warehouse i to customer j.

The text names another action, *supply*. This action can be considered in the text as equivalent to *send*, assuming the same participating elements. If instead we define it as an activity in which only each client and the product participate, we would make the mistake of using an action with determined value as a decision activity. The quantity of products to supply to each customer j is a known value, its demand D_j.

Illustration 4.16
To make two mixtures, M1 and M2, it is necessary to mix four compounds A, B, C, and D. Of the compounds A, B, and C, we need between 20% and 40% of the same in the mixtures. If the content of compound A is higher than compound B in the mixture M1, it is necessary to introduce compound D in an amount equal to 5% of A.

The costs per kilo of A, B, C, and D are, respectively, CA, CB, CC, and CD.

Determine the composition of the most economical mixtures if I must make a total of 25 kg of mixtures.

Table of Elements

The elements that are identified as actors in the problem are the two mixtures and the four compounds. Mixtures must be made, an undetermined amount, by mixing compounds. Since the mixtures have an undetermined quantity, of the compounds, we are also going to use an undetermined quantity, which defines them as measurable. The amount of mixing will be obtained by a function of the compounds, their sum. Therefore, mixtures will not be measured in decisions but in specifications.

The table of elements collects all the information (Table 4.9).

In order to unify sets, the minimum and maximum data has been established for the four compounds, with the following values (Table 4.10).

Decision Activities

A priori, they are identified as actions in the system, *make mixtures and mix compounds in mixtures*. When we refer to introducing compound D, we are referring

Table 4.9 Elements of Illustration 4.16

			Data				
Elements	Set	QN	Name	Param	Type	Belonging	Value
Mixtures	$i = 1...2$	I_U	Minimum	N_{ij}	C (%)	S	...
			Maximum	M_{ij}	C (%)	S	...
Compounds	$j = 1...4$	I_M	Cost	C_j	C ($/kilo)	W	...
				N_{ij}			
				M_{ij}			
System	–	I_U	Total mixtures	T	C (kg)	W	25

Table 4.10 Minimum and maximum values

	M1		M2	
	Minimum	Maximum	Minimum	Maximum
A	20%	40%	20%	40%
B	20%	40%	20%	40%
C	20%	40%	20%	40%
D	0%	100%	0%	100%

to the very action of mixing compound D. Regarding the activity of making mixtures, it is not really a decision activity but an action with calculated linear value. The amount of a mixture made is the sum of the compounds that compose it.

Mixing compounds in mixtures: The direct objects are the compounds that will be the elements that are measured in each event.

Action: MIX [compounds in mixtures].
Participating elements:
Compounds $j = 1 \ldots 4$.
Mixtures $i = 1,2$.
Quantification: Continuous.
Events: $j = 1 \ldots 4 \Rightarrow i = 1,2$.
Decision variables:
x_{ij} = Amount (Kgs) of compound j that are mixed in the mixture i.

4.5 Union of Activities

In some systems, there is the possibility of joining activities that are closely related to each other. It happens when there is a logical activity, which we will call secondary, with some elements that also participate in another (logical or measurable) activity, which we will call primary. The two activities can be combined in a single activity that incorporates the options for choosing the secondary activity to the primary activity, provided that there is a conditional relationship between them. The union is not valid in all cases. The relationship that must exist between both so that the union of activities can be carried out is the following:

x = primary activity *If* $y > 0$ *then* $x = y$ *and* $\alpha = 1$
α = secondary logical activity *If* $y = 0$ *then* $x = 0$ *or* $\alpha = 0$
y = union activity

In the union activity, the action of the primary activity is maintained, assimilating the secondary activity into the definition itself.

It must be said that in the majority of cases, the union processes are inefficient since they multiply the number of decision variables of the problem. Only in cases where they reduce specifications can they make any sense. There may be cases in which the union embeds specifications between primary and secondary activity. In

Table 4.11 Elements of Illustration 4.17

			Data				
Elements	Set	QN	Name	Param	Type	Belonging	Value
Supermarkets	$i = 1...45$	I_U					
Distributors	$j = 1...8$	I_U					
Vehicles	$k = 1...6$	I_U					

general, I discourage this option in the modelling of problems, although it is necessary to incorporate it into the methodology as something that we can find in the formulation of models.

Let's see some illustrations of a union process.

Illustration 4.17

There is a set of supermarkets (i = 1...45), a set of distributors (j = 1...8), and a set of vehicles (k = 1...6) of distribution. The system must assign distributors to vehicles and also assign distributors to supermarkets.

Although data have been omitted, we are going to consider each element as different from the rest and therefore individual and unitary.

Table of Elements (Table 4.11).

Decision Activities

Two activities are identified, on the one hand, to decide the assignment of distributors to supermarkets and, on the other hand, to assign vehicles to distributors.

Decision Activity 1:
Action: Assign [Supermarkets to distributors].
Participating elements: Supermarkets ($i = 1...45$ I_U); Distributors ($j = 1...8$ I_U).
Quantification: Binary.
Events: $i = 1...45 \Rightarrow j = 1...8$.
Decision variables:
$\alpha_{ij} = 1$ if I assign Distributor j to supermarket i; 0 otherwise.
$i = 1...45, j = 1...8$.

Decision Activity 2:
Action: Assign [distributors to vehicles].
Participating elements: Distributors ($j = 1...8$); Vehicles ($k = 1...6$).
Quantification: Binary.
Events: $i = 1...45 \Rightarrow j = 1...8$.
Decision variables:
$\beta_{ij} = 1$ if I assign Distributor j to Vehicle k; 0 otherwise. $j = 1...8$, $k = 1...6$.

Union of Activities
Action: Assign [supermarkets to distributors with vehicles].

Table 4.12 Number of variables in Illustration 4.17

Setup 1	Setup 2: Union
Activity 1: 45*8 = 360 variables Activity 2: 8*6 = 48 variables Total = 408 variables	Activity: 45*8*6 = 2160 variables

Table 4.13 Elements of Illustration 4.18

			Data				
Elements	Set	QN	Name	Param	Type	Belonging	Value
Clients	$i = 1 \ldots 10$	I_U	Route_Client	RC_{ik}	B	S	...
			Demand	D_i	C	S	...
Vehicles	$j = 1 \ldots 5$	I_U					
Routes	$k = 1 \ldots 7$	I_U		RC_{ik}			
Merchandise	–	I_M		D_i			

Participating elements: Supermarkets ($i = 1 \ldots 45$); Distributors ($j = 1 \ldots 8$). Vehicles ($k = 1 \ldots 6$).
Quantification: Binary
Events: $i = 1 \ldots 45 \Rightarrow j = 1 \ldots 8 \Rightarrow k = 1 \ldots 6$.
Decision variables:
$\alpha_{ijk} = 1$ if I assign Distributor j "with" Vehicle k to Supermarket i; 0 otherwise. $i = 1 \ldots 45, j = 1 \ldots 8; k = 1 \ldots 6$.

In the variable definition, the second assignment activity is embedded in the "with" preposition. If we analyze the number of variables generated with each configuration, we will check the low viability of the union (Table 4.12).

Illustration 4.18
There is a distribution merchandise system. There is a set of clients ($i = 1 \ldots 10$) with a merchandise demand D_i, a set of vehicles ($j = 1 \ldots 5$) and a set of routes ($k = 1 \ldots 7$). Each route passes through a subset of clients that are collected in the RC_{ik} *attribute:*

$RC_{ik} = 1$ *if Route k passes through Client i; 0 if not.*

Vehicles must select delivery routes and from these routes serve merchandise to customers.

We do not add more elements to the problem, since for the illustration this description suffices. Logically, you could enter trips, days, and also impose specifications of capacity, time, etc.

Table of Elements (Table 4.13)

Since no information about the merchandise is specified, we assume it is continuous.

Decision Activities

Two activities are identified: select distribution routes to vehicles and serve merchandise to customers. Let us see how they are configured:

Decision Activity 1:
Action: Select [Routes for Vehicles].
Participating elements: Vehicles ($j = 1\ldots5$ $\mathrm{I_U}$); Routes ($k = 1\ldots7$ $\mathrm{I_U}$).
Quantification: Binary.
Events: $j = 1\ldots5 \Rightarrow k = 1\ldots7$.
Decision variables:
$\alpha_{jk} = 1$ if we select Route k for Vehicle j; 0 otherwise. $j = 1\ldots5$, $k = 1\ldots7$.

Decision Activity 2:
Action: Serve [merchandise to customers with vehicles].
Participating elements*: Merchandise ($\mathrm{I_M}$); Clients ($i = 1\ldots10$ $\mathrm{I_U}$); Vehicles ($j = 1\ldots5$ $\mathrm{I_U}$).
Quantification: Continuous measurable.
Events: Merchandise $\Rightarrow i = 1\ldots10 \Rightarrow j = 1\ldots5$.
Decision variables:
x_{ij} = Amount of merchandise served to customer i with the vehicle j.

***:** Obtaining the elements in this action may have some complexity. If we had only placed Merchandise and Clients, we would be defining an action with a determined value, the value of its Demand. If with Merchandise we attend to the question “What do we serve?”, with the clients “Whom do we serve?”, then with the vehicles we answer the question “With what do we serve?”

Union of Activities
Primary activity: Serve.
Secondary activity: Select.
Action: Serve [merchandise to customers with vehicles by routes].
Participating Elements: Merchandise ($\mathrm{I_M}$); Clients ($i = 1\ldots10$ $\mathrm{I_U}$); Vehicles ($j = 1\ldots5$ $\mathrm{I_U}$); Routes ($k = 1\ldots7$ $\mathrm{I_U}$).
Quantification: Continuous measurable.
Events: Merchandise $\Rightarrow i = 1\ldots10 \Rightarrow j = 1\ldots5 \Rightarrow k/\mathrm{RC}_{ik} = 1$.
Decision variables:
x_{ijk} = Amount of merchandise served to customer i with vehicle j by route k.
The secondary activity of Select is replaced with the preposition “by.”

In the generation of events, not all routes must be specified. For each client, only the routes that pass through it must be considered, information collected in the RC_{ik} parameter. This fact, which is not determined by any decision, since it is due to information about the problem, is not controlled in the non-union version of the activities. In that version it would be necessary to include a specification that would control that a vehicle can only serve a customer if it has chosen a route that passes through said client. Let us consider how that specification would be:

Taking the sets of variables.
$\alpha_{jk} = 1$ if we select Route k for Vehicle j; 0 otherwise. $j = 1\ldots5$, $k = 1\ldots7$.
x_{ij} = Amount of merchandise served to customer i with the vehicle j.

We must assume that the quantity of merchandise served to a customer i by a vehicle j must be 0 if that vehicle has chosen a route k that does not pass through customer i ($\text{RC}_{ik} = 0$). Mathematically, it would correspond with the following logical proposition:

$$\forall i, \forall j, \forall k / \text{RC}_{\text{ik}} = 0 : \text{If } \ \alpha_{jk} = 1 \ \text{ Then } \ x_{ij} = 0$$

This specification is not necessary in the union configuration, since the specification is included in the definition of events. We can say that this is an advantage of the union configuration.

4.6 Examples

We are going to obtain the decision activities of the examples presented in examples Sect. 3.9 from the previous chapter. Therefore, we present in each example the description and the table or tables of elements already obtained in that chapter.

4.6.1 *Fire Stations (Example 3.9.1; Source: Larrañeta et al. 2003)*

An initial study is planned to install two fire stations in an urban area that currently has none. The approach has been adopted to divide the urban area into five sectors and carry out a preliminary analysis of the repercussions of the possible location of the stations in each of the sectors. The average time, in minutes, of answering a call from a fire station located in a certain sector i for an incidence received from each of the sectors j has been estimated in t_{ij}. The average number of calls per day that will occur from each of the five sectors (F_j) has also been estimated. All these values are

Table 1 Frequencies and time between sectors

t_{ij}	Sector 1	Sector 2	Sector 3	Sector 4	Sector 5
Sector 1	5	12	30	20	15
Sector 2	20	4	15	10	25
Sector 3	15	20	6	15	12
Sector 4	25	15	25	4	10
Sector 5	10	25	15	12	5
Frequency	2,5	1,6	2,9	1,8	3,1

Table 4.14 Version 4.1 of the Elements in Example 4.6.1

Elements	Set	QN	Data				
			Name	Param	Type	Belonging	Value
Sectors	$i,j = 1\ldots5$	I_U	Time	t_{ij}	C	S	...
			Frequency	f_i	C	W	...
Fire stations	$k = 1,2$	I_U					

Table 4.15 Version 4.2 of the Elements in Example 4.6.1

Elements	Set	QN	Data				
			Name	Param	Type	Belonging	Value
Sectors	$i,j = 1\ldots5$	I_U	Time	t_{ij}	C	S	...
			Frequency	f_i	C	W	...
Fire stations	–	C_D	N° items	N	I	W	2

shown in Table 1. *For example, it takes 12 minutes to go from a station located in sector 3 to an incident from sector 5. The last row shows the average daily frequency of calls made to the fire service.*

Version 4.1
Table of Elements (Table 4.14)

Decision Activities
Action: *Install [fire stations in sectors].*
Participating Elements: Fire stations ($k = 1,2$ I_U); Sectors ($i = 1\ldots5$ I_U).
Quantification: Binary.
Events: $k = 1,2 \Rightarrow i = 1\ldots5$.
Decision variables: $\alpha_{ki} = 1$ if we install fire station k in sector i; 0 otherwise.

Version 4.2
Table of Elements (Table 4.15)

Decision Activities

Action: Install [fire stations in sectors].
Participating elements: Fire stations C_D; Sector ($i = 1\ldots5$ I_U).
Quantification: Integer.
Events: Fire stations $\Rightarrow i = 1\ldots5$.
Decision variables: x_i = Number of fire stations installed in sector i.

Table 4.16 Table of elements in Example 4.6.2

Elements	Set	QN	Data				
			Name	Param	Type	Belonging	Value
Banquets	$i = 1\ldots4$	I_U	Tablecloths	m_i	I	S	...
			Day	d_{it}	B	S	...
Storeroom	–	I_U	Stock	S	I	S	200
Market	–	I_U	Price	p	C	S	12
Basket	–	I_U					
Laundry	–	I_U					
Fast wash	–	IU	Cost	cF	C	S	6
Slow wash	–	IU	Cost	cL	C	S	4
Days	$t = 1\ldots4$	IU		dit			
Tablecloths	–	CI		mi; S; p; cF; cL			

4.6.2 *Food Service (Example 3.9.2; Source: Larrañeta et al. 2003)*

A food service business has contracted four banquets for the next 4 days, requiring 150 clean tablecloths for the first banquet, 100 for the second, 140 for the third, and 130 for the fourth. Currently, it has 200 tablecloths in the storeroom, all of them clean, and they can buy what you need on the market every day at a cost of 12 m.u./ tablecloth.

After the banquets, the tablecloths can go to the laundry basket or send them to wash in the laundry. The laundry offers the following washing service:

– *Fast: Clean tablecloths for the next day, at a cost of 6 m.u/tablecloth.*
– *Slow: Clean tablecloths for 2 days, at a cost of 4 m.u/tablecloth.*

Table of Elements (Table 4.16)

Decision Activities
Action: Buy [tablecloths every day in the market].
Participating elements: Tablecloths C_I; Days ($t = 1\ldots4$) I_U; Market $I_{U.}$
Quantification: Integer.
Events: Tablecloths $\Rightarrow t = 1\ldots4 \Rightarrow$ Market.
Decision variables: x_t = Number of tablecloths bought on day t in the market.

Action: Take [tablecloths after the banquets to the basket].
Participating elements: Tablecloths C_I; Banquets ($i = 1\ldots4$) I_U; Basket $I_{U.}$
Quantification: Integer.
Events: Tablecloths $\Rightarrow i = 1\ldots4 \Rightarrow$ Basket.
Decision variables:
y_i = Number of tablecloths taken to the basket after the banquet i; $i = 1\ldots4$.

Action: Send [tablecloths to the laundry in wash modes after the banquets].
Participating elements: Tablecloths C_I; Banquets ($i = 1\ldots4$) I_U; Laundry I_U; Fast clean; Slow clean.
Quantification: Integer.
Events: Tablecloths $\Rightarrow$ $i = 1\ldots4$ $\Rightarrow$ Laundry $\Rightarrow$ Fast wash.
Tablecloths $\Rightarrow$ $i = 1\ldots4$ $\Rightarrow$ Laundry $\Rightarrow$ Slow wash.
Decision variables:
zF_i = Number of tablecloths sent to wash the laundry in fast wash after banquet i; $i = 1\ldots4$.
zS_i = Number of tablecloths sent to wash the laundry in slow wash after banquet i; $i = 1\ldots4$.

In this last action, it was necessary to include the type of washing process, fast or slow, in the participation. If we had only included laundry, we would not have known how to wash the tablecloths, and therefore, we would not know when they are available. Similarly, the participation of the laundry could have been overlooked, as it is implicit.

In the last two actions, we could have swapped the participation of the banquet with the day, since each banquet i corresponds with each day t:

Action: Take [tablecloths on day t to the basket].
Action: Send [tablecloths to the laundry in wash modes on day t].

In the chapter dedicated to specifications, we will see how this type of problem, where a measurable element is subject to activities over a set of periods, can be represented graphically to facilitate the obtaining of activities, auxiliary calculations, and specifications of equilibrium between the different variables associated with the measurable element.

4.6.3 Location of TV Cameras (Example 3.9.3; Source: Larrañeta et al. 2003)

CPL has to televise the game of the year. The producers have identified 10 possible locations for the installation of cameras and 25 stadium areas that need to be covered by the cameras. The table below indicates the relationship between both:

Location	Covered Area
1	1, 3, 4, 6, 7
2	8, 4, 7, 12
3	2, 5, 9, 11, 13
4	1, 2, 18, 19, 21
5	3, 6, 10, 12, 14
6	8, 14, 15, 16, 17
7	18, 21, 24, 25

(continued)

Table 4.17 Table of elements in Example 4.6.3

			Data				
Elements	Set	QN	Name	Param	Type	Belonging	Value
Areas	$i = 1...25$	$\mathrm{I_U}$	Coverage	C_{ij}	B	S	
			Minimal coverage	m_i	I	S	
Locations	$j = 1...10$	$\mathrm{I_U}$		C_{ij}			
Cameras	–	$\mathrm{C_I}$		m_i			

Location	Covered Area
8	2, 10, 16, 23
9	1, 6, 11
10	20, 22, 24, 25

Each area of the stadium must be covered by a camera
Location 9 must have a camera
Areas 1 and 2 require coverage of at least two cameras.
The objective is to minimize the number of cameras used (Table 4.17).

Decision Activities
Action: Install [cameras in locations].
Participating Elements: Cameras $\mathrm{C_I}$; Locations ($j = 1...10$ $\mathrm{I_U}$).
Quantification: Integer.
Events: Cameras $\Rightarrow j = 1...10$.
Decision variables: x_j = Number of cameras installed in location j.

In this type of system, the decision variables are defined as binaries:

$x_j = 1$ if we install camera in location j; 0 otherwise.

This way the definition of the maximum number of cameras that we can install in a location is included in the variable definition, because it is understood. However, our methodology would define that specification where it should be defined, in the specifications section, and the variable is defined as integer. In the specifications, the maximum number of cameras is established: $x_j \leq 1$. Of course, it is understood that the variable x_j can be defined as binary.

"Cover" would not be a decision activity in the system as it is a determined value action, we have to provide coverage, it is not an option, and therefore we do not have to decide anything, so it is a specification.

4.6.4 Trip Planning (Example 3.9.4)

There is a system that assigns travellers to buses. We have a group of 180 travellers who have hired the services of the BUSTOUR Company for today. There are five trips offered. Each traveller has chosen one of the five excursions.

Table 4.18 Table of elements of Example 4.6.4

Elements	Set	QN	Data				
			Name	Param	Type	Belonging	Value
Travellers	$i = 1\ldots180$	$\mathrm{I_U}$	Trip choice	E_{ij}	B	S	...
			Language choice	I_{ik}	B	S	...
Trips	$j = 1\ldots5$	$\mathrm{I_U}$		E_{ij}			
Languages	$k = 1,2$	$\mathrm{I_U}$		I_{ik}			
Buses	$r = 1\ldots8$	$\mathrm{I_U}$	Capacity	K_r	I	S	60
Seats	–	$\mathrm{C_D}$		K_r			

Each traveller also chooses the language (English and Spanish) for explanations. It has three options of choice:

- *Spanish*
- *English*
- *Both of them (if they speak both English and Spanish)*

The buses have a capacity of 60 people. There are eight buses.

Each bus that is used must be configured with a language and a trip. Since the explanations are given on the bus journey, it is necessary to place each traveller so that the trip and the language of the explanations that are configured on the bus they are travelling on are compatible with its choice.

BUSTOUR wants to use as few buses as possible to cover the trips (Table 4.18).

Decision Activities
Action: Assign [Travellers to buses].
Participating elements: Travellers $i = 1\ldots180$ $\mathrm{I_U}$; Buses ($r = 1\ldots8$) $\mathrm{I_U}$.
Quantification: Binary.
Events: $i = 1\ldots180 \Rightarrow r = 1\ldots8$.
Decision Variables: $\alpha_{ir} = 1$ if I assign traveller i to bus r; 0 otherwise.

Action: Configure [Language in buses].
Participating elements: Languages $k = 1,2$ $\mathrm{I_U}$; Buses ($r = 1\ldots8$) $\mathrm{I_U}$.
Quantification: Binary.
Events: $k = 1,2 \Rightarrow r = 1\ldots8$.
Decision Variables: $\beta_{kr} = 1$ if I configure language k in bus r; 0 otherwise.

Action: Assign [Trips to buses].
Participating elements: Trips $j = 1\ldots5$ $\mathrm{I_U}$; Buses ($r = 1\ldots8$) $\mathrm{I_U}$.
Quantification: Binary.
Events: $j = 1\ldots5 \Rightarrow r = 1\ldots8$.
Decision Variables: $\omega_{jr} = 1$ if assign trip j to bus r; 0 otherwise.

This last activity could also be understood as a logical calculation. The travelers that you assign to a bus condition the trip that you assign to the bus.

Table 4.19 Reduced table of elements of Example 4.6.4

Elements	Set	QN	Data				
			Name	Param	Type	Belonging	Value
Groups of travellers	$i = 1\ldots15$	C_D	Trip choice	E_{ij}	B	S	...
			Language choice	I_{ik}	B	S	...
Trips	$j = 1\ldots5$	I_U		E_{ij}			
Languages	$k = 1{,}2$	I_U		I_{ik}			
Buses	$r = 1\ldots8$	I_U	Capacity	K	I	S	60
Seats	–	C_D		K_r			

$$\forall j, r : \omega_{jr} = 1 \quad \text{IF AND ONLY IF} \sum_{i/E_{ij}=1} \alpha_{ir} > 0$$

If we define it as a decision activity, we cannot forget to define those propositions as specifications. We will see that these specifications are typified within the methodology.

Reduced Table of Elements (Table 4.19)

Regarding the reduced table of elements, the decision activities would be as follows:

Decision Activities
Action: Assign [Travellers to buses].
Participating elements: Travellers $i = 1\ldots15$ C_D; Buses ($r = 1\ldots8$) I_U.
Quantification: Integer.
Events: $i = 1\ldots15 \Rightarrow r = 1\ldots8$.
Decision Variables: x_{ir} = Number of travellers from group i assigned to bus r.
The actions of configuring the language and trip to the buses are identical to those presented for the table of elements 4.18.

Table 4.20 Table of elements of Example 3.9.5

Elements	Set	QN	Data				
			Name	Param	Type	Belonging	Value
Tasks	$i = 1\ldots n$	I_U	Start time	S_i	C	W	...
			End time	E_i	C	W	...
			Weight	P_i	C	W	...
Machines	$j = 1\ldots m$	I_U					

4.6.5 Fixed Job Scheduling Problem (Example 3.9.5; Kroon et al. 1995; Kolen et al. 2007)

There is a set of n tasks with a given start and end time and a weight. There is also a set of m machines. This set selects tasks to be processed in the machines so that the selected tasks have a maximum weight. A selected task is processed completely on a single machine. A machine cannot perform two tasks overlapped in time.

Table of Elements (Table 4.20).

Decision Activities

The wording talks of selecting tasks to be processed in the machines. Since the elements are unitary, the activities are going to be logics, a selection process, in this case selecting tasks to be processed in machines. The action that defines the activity is to process tasks in machines. This activity will result in a selection of tasks, those that have been processed. Therefore, it is not necessary to consider the activity of selecting tasks in addition to the task of processing tasks in machines. Selecting tasks corresponds to a logical calculation that is obtained from the activity of processing (if you have processed a task in a machine, you have selected that task).

Action: Process [task in machines].
Participating elements: Tasks $i = 1...n$ $\mathrm{I_U}$; Machines $j = 1...m$ $\mathrm{I_U}$.
Quantification: Binary.
Events: $i = 1...n \Rightarrow j = 1...m$.
Decision variables: $\alpha_{ij} = 1$ if we process task i in machine j; 0 otherwise.

4.6.6 Health Centers (Example 3.9.6)

There is a city containing 12 health centers. Due to population changes, it has been decided to reassign health centers to citizens, a total of n. We know each citizen address, and therefore, the system allows us to know the distance from their homes to each health center. Each health center has a capacity that is expressed in the number of patients that can be seen per day. It is estimated that 1% of people go

Table 4.21 Table of elements of Example 4.6.6

			Data				
Elements	Set	QN	Name	Param	Type	Belonging	Value
Health centers	$i = 1...12$	$\mathrm{I_U}$	Distance	D_{ij}	C	S	...
			Capacity	K_i	E	W	...
Citizens	$j = 1...n$	$\mathrm{I_U}$		D_{ij}			
City	–	$\mathrm{I_U}$	Attendance	A	C	W	0,01

daily to health centers. The objective is to minimize the sum of the distances between each citizen and the health center assigned.

Table of Elements (Table 4.21)

Decision Activities
Action: Reassign [citizens to health centers].
Participating elements: Citizens $j = 1...n$ I_U; Health Centers $i = 1...12$ I_U;
Quantification: Binary.
Events: $j = 1...n \Rightarrow i = 1...12$.
Decision variables:
$\alpha_{ij} = 1$ if we reassign citizen i to health center j; 0 otherwise.

References

Kolen, A., Lenstra, J. K., Papadimitriou, C., & Spieksma, F. (2007). Interval scheduling: A survey. *Naval Research Logistics, 54*, 530–543.
Kroon, L. G., Salomon, M., & Van Wassenhove, L. N. (1995). Exact and approximation algorithms for the operational fixed interval scheduling problem. *European Journal of Operational Research, 82*, 190–205.
Larrañeta, J., Onieva, L., Cortés, P., Muñuzuri, J., & Guadix, J. (2003). *Métodos Cuantitativos en Ingeniería de Organización*. Sevilla: Editorial Universidad de Sevilla.

Chapter 5
Calculations in a System

5.1 Introduction

The calculations represent the variables that can appear in an optimization problem whose values can be obtained from the value of other variables, decision variables generally, or even previous calculations. They fall on elements of the system, and, like the activities, they have a type of value associated.

Calculations can sometimes be considered as activities, due to their semantics or their importance in the system, but they will never be of primary decision, since their value will be obtained from the value of other variables. This determines the most significant aspect of these variables:

They always carry constraints that define their meaning, that is, they define the calculation.

The calculations in a system can be of three types:

- **Auxiliary Calculations:** Its value is calculated directly through a direct linear function on the variables on which it depends. Its use is optional except when we want to impose that the result of the function is an integer value. They help to work more comfortably with the restrictions of the system. They give rise to the auxiliary variables of the model.
- **Logical Calculations:** They are formulated by means of a logical proposition. Its use is necessary. The value is binary in its great majority, except for the propositions that are defined to collect the value of a variable when a condition is met, where non-binary calculations are used to be able to express the function in lineal way. Already in Chap. 4, we talked about actions that could be presented in a system and that corresponded with logical calculations. The logical proposition that defines the calculation can also be understood as another specification of the system.

- **Lower/Upper Bound Calculations:** They calculate an upper or lower bound of the values of other system variables. Its use is also necessary. The definition of the calculation is carried out in a simple way with the imposition of bounds.

5.2 Auxiliary Calculations

An auxiliary calculation corresponds to a variable that substitutes a function of other variables in order to work with a more simplified model, generally, or to impose integer values on functions.

Let's take a look at a simple illustration:

Illustration 5.1
Let the decision variables of a system be defined as:

x_i*: Number of kilos of product purchased from the supplier* P_i*.* $i = 1...n$

If I want to collect the total purchased in a variable, I define the following calculation:

Auxiliary calculation: Total kilos (sum) of product purchased from suppliers
Applied to: The system
Definition of calculation variable:
x = Total kilos of product purchased;
$x \geq 0$ continuous.
Constraint that defines the calculation: $x = x_1 + x_2 +...+ x_i +...+ x_n$

If I want that total amount to be an integer number of kilos, the auxiliary variable is defined as integer and its use is mandatory:

Definition of calculation variable:
x = Number of kilos of product purchased;
$x \geq 0$ integer.
Constraint that defines the calculation: $x = x_1 + x_2 +...+ x_i +...+ x_n$

Within the auxiliary calculations, we will now highlight a very common calculation, the auxiliary calculation of value selection.

5.2.1 Auxiliary Calculation of Value Selection

It is a calculation that selects the value of an attribute of an element over a set of elements. For the selection of the element, logical decision variables are usually used as inputs, although variants of this calculation can also be considered in cases with integer or continuous variables (see Illustration 5.3). In the case of logical variables,

Table 5.1 Diagram 1 of the selection calculation table

			Data				
Elements	Set	QN	Name	Param	Type	Belonging	Value
Elements	$i = 1\ldots n$	–	Attribute name	A_j	–	–	...
			...				
...							

Table 5.2 Diagram 2 of the selection calculation table

			Data				
Elements	Set	QN	Name	Param	Type	Belonging	Value
Elements1	$i = 1\ldots n$	–	Attribute name	A_{ij}	–	S	...
			...				
Elements2	$j = 1\ldots m$	–		A_{ij}			
			...				
...							

these variables satisfy a quantitative selection rule in which the selection of an element is forced. The quantitative selection rules are a type of specification that will be studied in the specifications chapter (Table 5.1).

Decision variables:
$\alpha_i = 1$ if we select element i; 0 otherwise.
Selection rule: We select one element.
$$\sum_{i=1}^{n} \alpha_i = 1$$
Auxiliary calculation: Attribute value of the selected element
Applied to: ...
Definition of calculation variable: $x =$ Value of the attribute of the selected element
Constraint that defines the calculation: $x = \sum_{i=1}^{n} A_i \alpha_i$

If the selection of more than one element were allowed, the variable x would collect the sum of the values *Ai* of the selected elements, a less common aspect.

This calculation is extendable to more than one set of elements with shared data (Table 5.2).

Decision Variables
$\alpha_{ij} = 1$ if I select pair (i,j); 0 otherwise.
Selection rule: We select one pair.
$$\sum_{i=1}^{n} \sum_{j=1}^{m} \alpha_{ij} = 1$$
Auxiliary calculation: Attribute value A_{ij} of the pair (i, j) selected
Applied to: The system
Definition of calculation variable: $x =$ Value of the attribute of the selected element

Table 5.3 Diagram 3 of the selection calculation table

			Data				
Elements	Set	QN	Name	Param	Type	Belonging	Value
ELEMENTS1	i=1. . .n	–	Attribute name	A_i	–	P	. . .
			. . .				
ELEMENTS2	j=1. . .m	–	. . .				
. . .							

Table 5.4 Table of elements of Illustration 5.2

			Data				
Elements	Set	QN	Name	Param	Type	Belonging	Value
Locations	$j = 1...n$	I_U	Distance	D_{ij}	C	S	. . .
Distribution centers	–	C_D	Quantity	Q	I	W	3
Supermarkets	$i = 1...m$	I_U		D_{ij}			

Constraint that defines the calculation: $x = \sum_{i=1}^{n} \sum_{j=1}^{m} A_{ij}\alpha_{ij}$

It is also possible to perform the calculation for each element of a set on another set, provided that there is a binary variable that relates them (Table 5.3):

Decision Variables
$\alpha_{ij} = 1$ if we select pair (i,j); 0 otherwise.
Selection rule:

$$\forall j : \sum_{i=1}^{n} \alpha_{ij} = 1$$

Auxiliary calculation: Attribute value A_i of element i selected for j
Applied to: Each Element 2 (j=1 . . . m)
Definition of calculation variable:
x_j = Value of the attribute of the selected element for j
Constraint that defines the calculation:

$$x_j = \sum_{i=1}^{n} A_i\alpha_{ij}$$

Let us take a look at a couple of illustrations of this type of auxiliary calculation:

Illustration 5.2
There is a supermarket company that has several locations ($j = 1...n$) to install three product distribution centers.

The Company owns supermarkets to be supplied from the locations with some distribution center, knowing the distance between each supermarket and each location.

It is about minimizing the sum of the distances between each supermarket and the assigned location.

Table of Elements (Table 5.4)

In the configuration carried out, the locations and the supermarkets have been considered as individual, since they have an attribute of distance that differentiates them from the rest and unitary because their data are not measurable. We consider the distribution centres to be a collective element, since they are identical items and we only refer to the number of centres in the text (numerals of the collective element).

Decision Activities

We will now simplify the notation associated with the decision activities, indicating only the action with its participant elements and the defined decision variables. Quantification and events are defined in the declaration of the decision variables.

Action: Install [distribution centres in locations]
Decision variables:
x_j =Number of distribution centres installed in location j. $j = 1...n$

Action: Supply [supermarkets from locations (with some distribution centre)]
"with some distribution center" is a condition for locations. As we saw in section 4.3-rule 4, we consider the participation of all locations without condition.
Decision variables:
$\alpha_{ij} = 1$ if location j supplies to supermarket i; 0 otherwise. $i = 1...m, j = 1...n$.

For the modelling of the objective criterion, it is necessary to minimize the sum of the distances of each supermarket to each assigned location.

The distance value for each supermarket between itself and the assigned location is not reflected in any decision activity and, therefore, we need to perform a calculation to obtain it. Since it is the value of an attribute, it can be obtained by an auxiliary calculation of value selection:

Auxiliary calculation: Attribute value D_{ij} of Location j selected for i
Applied to: Each Supermarket ($i = 1...m$)
Definition of calculation variable:
y_i = Distance D_{ij} of the selected location for supermarket i
Constraint that defines the calculation:

$$\forall i: \quad y_i = \sum_{j=1}^{n} D_{ij}\alpha_{ij}$$

Prior to the calculation, there was a quantitative selection rule regarding the number of locations that are assigned to a supermarket, one in this case:

$$\forall i: \quad \sum_{j=1}^{n} \alpha_{ij} = 1$$

and for that reason, we have been able to develop the formula of the auxiliary calculation.
Objective function:

$$\text{Min} \sum_{i=1}^{m} y_i$$

Table 5.5 Table of elements of Illustration 5.3

Elements	Set	QN	Data				
			Name	Param	Type	Belonging	Value
Workers	$i = 1\ldots n$	$\mathrm{I_U}$	City_Worker	CW_{ik}	B	S	…
Work Centers	$j = 1\ldots m$	$\mathrm{I_U}$	Workers	W_j	I	W	…
			City_Center	CC_{jk}	B	S	…
Cities	$k, k' = 1\ldots s$	$\mathrm{I_U}$	Distance	$D_{kk}{}'$	C	S	…
				CW_{ik}; CC_{jk}			

Illustration 5.3
There is a set of n workers. Each worker lives in a specific city of the province. There are m work centers in the province located each in a specific city of the province. Each center has a demand for workers. The company wants to reallocate workers to centers in a way that minimizes the total distance traveled by workers. The distances between each pair of cities are known. There are a total of s cities.

Table of Elements (Table 5.5)

Decision Activities
Action: Reallocate [workers to work centers]
Decision variables:
$\alpha_{ij} = 1$ if worker i is reallocated to center j; 0 otherwise. $i = 1\ldots n, j = 1\ldots m$.

For the modeling of the objective function of the system, it is necessary to minimize the sum of the distances traveled by the workers. This value will be the distance from the city to which the worker belongs, to the city in which the assigned center is located. This value can be obtained by an auxiliary calculation of value selection:

Auxiliary calculation: Distance traveled by the worker i.
Applied to: Each worker ($i = 1\ldots n$)
Definition of calculation variable:
y_i = Distance value $D_{kk}{}'$ for worker i belonging to city k with assigned center j belonging to city k'
Constraint that defines the calculation:

$$\forall i: \quad y_i = \sum_{k=1}^{s} \sum_{k'=1}^{s} \sum_{j=1}^{m} D_{kk'} \mathrm{CT}_{ik} \mathrm{CC}_{jk'} \alpha_{ij}$$

Objective function:

$$\text{Min} \sum_{i=1}^{n} y_i$$

In this example we could present a table of elements in which we grouped the workers of each city into a collective element, since they are identical in the system and we can describe the system without referring to them individually in the specifications. This is studied in detail in Chap. 7, now we propose the version

Table 5.6 Table of grouped elements of Illustration 5.3

			Data				
Elements	Set	QN	Name	Param	Type	Belonging	Value
Workers groups	$i = 1\ldots L$	C$_D$	City_Group	CW_{ik}	B	S	…
			Number of workers	N_i	I	W	
Work centers	$j = 1\ldots m$	I$_U$	Workers	W_j	I	W	…
			City_Center	CC_{jk}	B	S	…
Cities	$k, k'=1\ldots s$	I$_U$	Distance	$D_{kk'}$	C	S	…
				CW_{ik}; CC_{jk}			

without going into details. Let's assume that L are the identical groups of workers that can be formed:

Table of Elements (Table 5.6)

Decision Activities
Action: Reallocate [worker groups to work centers]
Decision variables:
x_{ij} = Number of workers of group i reallocated to center j; $i = 1\ldots L, j = 1\ldots m$.

The objective now is the selection of distance values between the city where the group of workers is and all the centers assigned. The value of the total distance traveled by a group will be the sum of the distance to a center multiplied by the number of workers assigned to that center. This auxiliary calculation is similar to the one outlined above.

Auxiliary calculation: Distance traveled by the workers group i
Applied to: Each worker group ($i = 1 \ldots L$)
Definition of calculation variable:
y_i = Sum of the distances $D_{kk'}$ for group i belonging to city k with each center assigned j belonging to city k' multiplied by the number of workers assigned to that center j
Constraint that defines the calculation:

$$\forall i: \quad y_i = \sum_{k=1}^{s} \sum_{k'=1}^{s} \sum_{j=1}^{m} D_{kk'} CT_{ik} CC_{jk'} x_{ij}$$

Objective function: Min $\sum_{i=1}^{n} y_i$

As much for this version with collective elements as for the version with individual elements, it is possible a configuration that would simplify the auxiliary calculations. It would suffice to exclude the elements cities from the system and to incorporate the distance data between cities as a distance between the worker and the center. Since I know the city where the worker is and the city where the center is located, you can calculate that attribute from the distance data between cities:

Table 5.7 Simplified table of elements of Illustration 5.3

			Data				
Elements	Set	QN	Name	Param	Type	Belonging	Value
Workers	$i = 1\ldots n$	$\mathrm{I_U}$	Distance	D_{ij}	C	S	...
Work centers	$j = 1\ldots m$	$\mathrm{I_U}$	Workers	W_j	I	W	...
				D_{ij}			...

Table 5.8 Values of a logical calculation

			Values	
Logical calculation	Binary	True	1	
		False	0	
	Non-binary	True	x	*Value collected in variable x. The calculation takes as value, in case True, the one that corresponds to the variable x*
		False	0	

The distance between each worker i and each center j would be obtained as:

$$\forall i,j: \quad D_{ij} = \sum_{k=1}^{s} \sum_{k'=1}^{s} \sum_{j=1}^{m} D_{kk'} CT_{ik} CC_{jk'}$$

And that attribute would be included in the table of elements that would be as shown in Table 5.7 for the Individual version (Table 5.5).

5.3 Logical Calculations

Logical calculations appear in systems where modelling has a more complex character. Examples can be found Williams (2013) based in modeling techniques proposed in Mitra et al. (1994) and Williams (1995).

Systems without logical calculations are usually easily to model. Logical calculations are represented in mandatory use variables that define their values in a logical statement, so their value is usually binary, although non-binary calculations can also be defined where the calculated value must collect a variable value to avoid non-linear expressions. In either case they define logical values (Table 5.8):

The main difficulty in the modelling of a logical calculation is the definition of the logical proposition or propositions that condition the value of the calculation, and not so much the modelling, which will be governed by simple rules. The modelling of logical propositions will be discussed in detail in the next chapter, since logical propositions are a type of specification within this methodology. In this chapter, we will only focus on the definition of the logical propositions that determine the values of the calculation.

The propositional logic (Mendelson 1997) is necessary for the definition of all the logical calculations, as well as for the specifications that are formulated by means of a logical proposition without the need to define a logical calculation. Any logical proposition can be understood as a system specification, including those that define a calculation. Therefore, we will define the basic concepts of propositional logic in order to understand the definition of logical propositions.

5.3.1 Logical Propositions and Logical Operators

In mathematical programming, we can define logical proposition as a mathematical semantic content that, applied to a solution, is possible to assign a truth value (true or false).

Depending on their structure or internal complexity, the proposals can be classified into:

Atomic or Simple Propositions
They refer to a single content of truth; in mathematical terms, linear formulation corresponds to a linear equation or inequality that can be defined with the following format:

Left side	Sign	Right side (independent term)
Lineal function	$<$; $\leq$; $=$; $\geq$; $>$	Numeric value

Illustration 5.4: Example of Simple Propositions
$\alpha = 1;$
$x_1 + x_2 + x_3 \leq 100;$
$x_1 - x_2 < 50;$
$x \geq y \Rightarrow x - y \geq 0;$
$x \geq 10;$
$x > 0;$
$\sum_{i/d_i<10}^{n} \alpha_i = 1;$
$\sum_{i=1}^{n} \alpha_i \geq 1;$

Any constraint of a problem corresponds to a simple proposition on which we force the solutions to have a truth value. In another case, it would be an inadmissible solution. However, when an atomic proposition is part of a composite proposition, its fulfillment is not obligatory, only the fulfillment of the compound proposition is required.

Table 5.9 Logical operators (given ϕ and ψ logical propositions)

Operator	Symbol	Semantic
Negation	$\neg$	NOT (ϕ)
Disjunction	$\vee$	ϕ OR ψ
Conjunction	$\wedge$; &	ϕ AND ψ
Conditional	$\rightarrow$	IF ϕ THEN ψ
Biconditional	$\leftrightarrow$	ϕ IF AND ONLY IF ψ
Exclusive disjunction	$\oplus$	EITHER ϕ OR ψ

Table 5.10 Equivalents between operators

Proposition	Equivalent propositions
$\phi \leftrightarrow \psi$	$\phi \rightarrow \psi$ $\psi \rightarrow \phi$
$\phi \rightarrow \psi$	$\neg\, \psi \rightarrow \neg\, \phi$

Molecular or Compound Propositions

Propositions constituted by one or more atomic propositions joined by logical operators.

Illustration 5.5: Examples of compound propositions

$\alpha = 1$ OR $\beta = 1$;
NOT ($x_1 + x_2 \geq 10$)
IF $x_1 + x_2 + x_3 \leq 100$ THEN $\alpha = 1$;
EITHER $x_1 - x_2 < 50$ OR $y_1 < 50$;
IF $x_1 \leq 10$ OR $y_1 \leq 20$ THEN $x_2 \geq 15$ AND $y_2 \geq 15$;

Table 5.9 describes the logical operators with which we can express any logical proposition. We will analyze them in detail in Chap. 6.

There are more logical operators than those shown, but they are not necessary because they are reduced to the use of the previous ones. Even for the previous ones, there are equivalences that will be useful for the definition of logical calculations. We refer to equivalences with the operator $\leftrightarrow$ (ϕ IF AND ONLY IF ψ) and the operator $\rightarrow$ (IF ϕ THEN ψ).

Illustration 5.6

Let x be an integer variable and α a binary variable.

$$x \geq 10 \text{ IF AND ONLY IF} \alpha = 1 \tag{5.1}$$

For the first equivalence of Table 5.10, the proposition (5.1) is equivalent to (5.2) and (5.3).

$$1. - \text{IF} x \geq 10 \text{ THEN} \alpha = 1 \tag{5.2}$$

$$2. - \text{IF}\alpha = 1 \text{ THEN} x \geq 10 \tag{5.3}$$

Proposition (5.3) could also be transformed by the second equivalence of Table 5.10 in:

$$\begin{aligned} &\text{IF}\alpha = 1 \text{ THEN} x \geq 10 \Rightarrow \text{IF NOT } (x \geq 10) \text{ THEN NOT } (\alpha = 1) \Rightarrow \\ &\Rightarrow \text{IF} x < 10 \text{ THEN} \alpha = 0 \end{aligned} \tag{5.4}$$

5.3.2 Identification and Definition of a Logical Calculation

Logical calculations represent conditioned actions in a system and conditioned characteristics of elements and are also used in an intermediate way in the modelling of compound propositions to collect the result of integer or continuous simple propositions. They can fall on the elements individually or collectively, involving several of them. Let us see how they can appear in a system:

Identification
As a Conditioned Action

They have already been discussed in the chapter dedicated to decision activities. Conditional or conditioned value actions are those that are discarded as decision activities because their values are determined by other decision activities. The logical calculations of conditioned actions are easy to detect because the action and conditions are always defined jointly. However, if the modeller defines the action as a decision activity because it does not perceive its dependence, it must take into account as a specification the logical propositions that generate the conditions of the action and that define the values of the variable. They can be of two types:

- Determined value conditional actions: They give rise to binary logical calculations.
- Conditional linear value actions: They give rise to non-binary logical calculations. It is the case in which the logical calculation will take the value of another variable of the problem. They could really be defined as binary, but this would result in non-linear expressions in the model, so integer or continuous variables are used to define them.

In any case, conditioned actions need, first, to define the variable and, second, to express the specification or specifications of propositional logic that define their values.

As a Conditioned Characteristic of an Element

In the system it is necessary to know if a certain element or several elements have fulfilled any condition that is obtained by a function of variables. Semantically it can be expressed through a qualification of the element or simply expressing that a variable or function of variables falls into a range of values. As before, we need to define the variable that will collect the result and the specification that defines the calculation. However, this case is more difficult to identify in the problem wording than in the case of conditioned actions. We need to always reflect on whether what is needed is contained in some decision variable or in some attribute and, if not, to express that characteristic as a logical calculation. In general, this category presents binary calculations mainly, although conditioned linear value characteristics can also occur.

Typically, the logical variable is used later quantitatively in some other system specification or in the objective criterion.

In an Auxiliary Way in the Modelling of Compound Logical Propositions

This case presents a compound logical proposition formed with variables already existing in the problem. When modelling the proposition is undertaken, the definition of logical variables is necessary to collect the result of some atomic propositions that are within the global proposition or that are formed in the process. In this case they are always binary calculations. The case will be seen in detail in the next chapter devoted to the modelling of specifications, when we analyze the modelling of the specifications of propositional logic.

Although we have defined three cases, the first two cases are very much related and according to the way of describing the system, the calculation can be considered as case 1 or as case 2.

Definition
Logical calculations are represented as logical variables. A logical calculation will correspond to a variable that is always necessary in the model. The definition of a logical calculation involves the conditional operator or the biconditional operator, regardless of whether it can be enunciated with other operators. The format that we will use to define the logical calculations will be the following:

Binary logical calculation: Semantics of calculation
Applied to: Elements on which it falls.
Definition of logical variable:

$$\beta = \begin{cases} 1 & \text{if the condition is met} \\ 0 & \text{otherwise} \end{cases}$$

Logical proposition: $\beta = 1$ IF AND ONLY IF *Condition*

Non-binary logical calculation: Semantics of calculation
Applied to: Elements on which it falls.
Definition of logical variable:

y = value of x when the condition is met
Logical propositions:
IF *Condition* THEN $y = x$

IF NOT(*Condition*) THEN $y = 0$

Next let us see a first example of identifying and defining logical calculations, case 1 or case 2, or identifying whether a composite proposal is presented with already existing variables and whether for its later modelling we will need the definition of some logical calculation (case 3).

Illustration 5.7
There is a system of buying a product from four suppliers.
They are defined as decision activities:

x_1 = *number of product units purchased from supplier 1.*
x_2 = *number of product units purchased from supplier 2.*
x_3 = *number of product units purchased from supplier 3.*
x_4 = *number of product units purchased from supplier 4.*

Specifications and actions that we could establish:

1. *Pay a fee of $50 if we exceed the 100 total units purchased.*
2. *Have a discount of 2% on the total of units purchased if I buy at least 1000 from supplier 1.*
3. *If I buy more than 30 units from supplier 2, I must buy at least 15 from supplier 3.*
4. *Do not buy from the first and third supplier at the same time*
5. *Buy a total of 2000 units or pay a fee of $100*
6. *The number of suppliers used (from which I buy) is equal to 2.*
7. *Buy at least 100 units from a supplier.*
8. *Minimise the number of suppliers from which I bought a maximum of 50 units*

1. *Pay a fee of $50 if we exceed the 100 total units purchased.*

We are facing a conditioned action of determined value (case 1.1): Pay a fee of $50.

Binary logical calculation: Pay a fee of € 50
Applied to: The system
Definition of logical variable:

$$\beta_1 = \begin{cases} 1 & \text{if we pay the fee of } \$50 \\ 0 & \text{otherwise} \end{cases}$$

Logical proposition:

$$\beta_1 = 1 \quad \text{IF AND ONLY IF} \quad \sum_{i=1}^{4} x_i > 100$$

2. *Have a discount of 2% on the total of units purchased if I buy at least 1000 from supplier 1*

Conditional action of linear value (case 1.2)

Non-binary logical calculation:

Have a discount of 2% on the total purchased $\left(\sum_{i=1}^{4} x_i\right)$

Applied to: The system
Definition of logical variable: y_2 = discount obtained
Logical propositions:

$$\text{IF } x_1 \geq 1000 \quad \text{THEN} \quad y_2 = \ 0,02\sum_{i=1}^{4} x_i$$

$$\text{IF } x_1 < 1000 \quad \text{THEN} \quad y_2 = 0$$

With the continuous variable y_2 we have managed to express linearly the value of the discount obtained. If we had opted for a binary calculation, the result would have been the following:

Binary logical calculation:

Have a discount of 2% on the total purchased $\left(\sum_{i=1}^{4} x_i\right)$

Applied to: The system
Definition of logical variable:

$$\beta_2 = \begin{cases} 1 \text{ if I obtain the discount of } 2\% \\ 0 \quad \text{otherwise} \end{cases}$$

Logical proposition: $\beta_2 = 1$ IF AND ONLY IF $x_1 \geq 1000$

To obtain the discount value we would have to use the next function: $\beta_2 0,02\sum_{i=1}^{4} x_i$ which is not lineal.

3. *If I buy more than 30 units from supplier 2, I must buy at least 15 from supplier 3.*

It is a direct logical proposition about system activities (Case 3). We do not need to incorporate any new variables to state the proposition. We will need binary logical calculations to model each simple non-binary proposition that appears in the compound statement.

Logical proposition: IF $x_2 > 30$ THEN $x_3 \geq 15$

4. *Do not buy from the first and third supplier at the same time*

Compound logical proposition (Case 3)

Logical proposition: NOT($x_1 > 0$ AND $x_3 > 0$)

5. *Buy a total of 2000 units or pay a fee of \$100*

A conditional action of determined value is defined (Pay a fee of \$100) without using a conditional expression. Actually, the phrase is equivalent to saying: you pay a fee of \$100 if you do not buy a total of 2000 units.

Binary logical calculation: Pay a fee of \$100
Applied to: The system
Definition of logical variable:

$$\beta_5 = \begin{cases} 1 \text{ si I pay a fee of } \$100 \\ 0 \text{ otherwise} \end{cases}$$

Logical proposition: EITHER $\sum_{i=1}^{4} x_i = 2000$ OR $\beta_5 = 1$

It can also be expressed with the operator $\leftrightarrow$:

$$\beta_5 = 1 \quad \text{IF AND ONLY IF} \;\; \text{NOT}\left(\sum_{i=1}^{4} x_i = 2000\right)$$

6. *The number of suppliers used (from which I buy) is equal to 2.*

This is a specification that uses a logical calculation of type 2. A qualification of the suppliers appears: Supplier used. This qualification is obviously not an attribute and is not included in any decision activity, so it is necessary to define it as a calculation.

Binary logical calculation: Supplier used
Applied to: Suppliers $i = 1 \ldots 4$
Definition of logical variable:

$$\delta_i = \begin{cases} 1 \text{ if supplier } i \text{ is used} \\ 0 \text{ otherwise} \end{cases}$$

Logical proposition: $\delta_i = 1$ IF AND ONLY IF $x_i > 0$

The specification states that only two providers must be used: $\sum_{i=1}^{4} \delta_i = 2$

7. *Buy at least 100 units from a supplier.*

As in point 6, we again establish a quantitative specification on suppliers that comply with a quality (purchase of at least 100). We must find out if we bought at least 100 from each supplier and then impose a specification on those calculations:

Binary logical calculation: Supplier with at least 100 units supplied
Applied to: Suppliers $i = 1 \ldots 4$
Definition of logical variable:

$$\omega_i = \begin{cases} 1 \text{ if I buy at least 100 units to the supplier } i \\ 0 \text{ otherwise} \end{cases}$$

Logical proposition: $\omega_i = 1$ IF AND ONLY IF $x_i \geq 100$

Specification requested: $\sum_{i=1}^{4} \omega_i \geq 1$

8. *Minimize the number of suppliers from which I bought a maximum of 50 units*

This is similar to the previous one but in this case the calculations are going to be used in the objective function.

Binary logical calculation: Supplier with a maximum of 50 units supplied
Applied to: Suppliers $i = 1 \ldots 4$
Definition of logical variable:

$$\lambda_i = \begin{cases} 1 \text{ if buy from supplier } i \text{ a maximum of 50 units} \\ 0 \text{ otherwise} \end{cases}$$

Logical proposition: $\lambda_i = 1 \quad \text{IF AND ONLY IF} \;\; x_i \leq 50$

Objective function: Min $\sum_{i=1}^{4} \lambda_i$

5.3.3 Reduction of the Definition of a Logical Calculation

In general, the definition of a logical calculation requires the definition of each of the values that calculation can take. In the case of binary variables, the calculation must be defined for the value 1 and the value 0, as seen, for example, in Illustration 5.7 – point 1, regarding the calculation of paying a fee of \$50, which was defined as:

$$\beta_1 = 1 \quad \text{IF AND ONLY IF} \quad \sum_{i=1}^{4} x_i > 100 \tag{5.5}$$

Operator "IF AND ONLY IF" defines both value 1 and value 0 of the variable β_1 in (5.5), since this proposition according to Table 5.10 is equivalent to these two propositions:

$$\begin{aligned} &1.- \quad \text{IF} \quad \beta_1 = 1 \text{ THEN } \sum_{i=1}^{4} x_i > 100 \\ &\Rightarrow \text{IF} \quad \text{NOT}\left(\sum_{i=1}^{4} x_i > 100\right) \text{ THEN NOT}(\beta_1 = 1) \quad \Rightarrow \\ &\Rightarrow \text{IF} \quad \sum_{i=1}^{4} x_i \leq 100 \text{ THEN } \beta_1 = 0 \end{aligned} \tag{5.6}$$

$$2.- \quad \text{IF} \quad \sum_{i=1}^{4} x_i > 100 \text{ THEN } \beta_1 = 1 \tag{5.7}$$

However, in some systems, it is not necessary to define all the values of the calculation variable due to the impact of the variable on the optimization problem. When a variable implies a cost in the objective function of the problem, it is not necessary to define value 0 (5.6) in the calculation proposition; we can give freedom

Table 5.11 Table of elements of Illustration 5.8

Elements	Set	QN	Data				
			Name	Param	Type	Belonging	Value
Customers	$i = 1\ldots10$	$\mathrm{I_U}$					
Product	–	$\mathrm{C_D}$					

to the variable since, when the problem is solved, it will never consider as an optimal solution a solution in which the variable is worth 1 and can be 0. An example occurs in the previous illustration, where $\beta_1 = 1$ meant that a cost was incurred in the system.

If we eliminate the definition of $\beta_1 = 0$ (5.6), in a solution where $\sum_{i=1}^{4} x_i \le 100$, the system will be free to assign β_1 the value 1 or 0 for that solution, but it will never impose paying the fee ($\beta_1 = 1$) since that would go against the objective function.

This also happens in other environments where, although the calculation variable does not have a cost associated, its definition can be reduced because other characteristics that occur in the system make it possible, although it must be clear that it will never be incorrect to define all the values of a calculation.

Let's take a look at an illustration where you can reduce the definition of a calculation without a cost associated to the calculation:

Illustration 5.8
Suppose we must sell units of a product to a set of 10 customers.

Table of Elements (Table 5.11)

Decision Activities

Action: Sell [Product to customers]
Decision variables:
x_i = Number of units of the product sold to the customer i; $i = 1\ldots10$

Specifications

Suppose that the system establishes that you must sell more than 100 units to at least one customer.

To model this specification, we need to define a logical calculation: from the activity of selling, find out the quality of selling more than 100 units for each customer, and then assume that this happens at least once. Therefore, we must define the following logical calculation:

Binary logical calculation: Customer buying more than 100 units
Applied to: Customer $i = 1\ldots10$
Definition of logical variable:

$$\beta_i = \begin{cases} 1 \text{ if I sell more than 100 units to customer } i \\ 0 \text{ otherwise} \end{cases}$$

Logical proposition: $\beta_i = 1$ IF AND ONLY IF $x_i > 100$

$$\text{Specification requested} : \sum_{i=1}^{10} \beta_i \geq 1 \tag{5.8}$$

Thanks to this specification and although the variables β_i do not entail cost, the definition of the variables β_i can be simplified:

According to (5.8) in any solution at least one variable β_i will be worth 1, so we can eliminate value 1 from the definition and only use the definition of the value 0.

$$\forall i : \quad \text{IF} \quad x_i \leq 100 \;\; \text{THEN} \;\; \beta_i = 0 \tag{5.9}$$

When the entry of the proposition (5.9) does not comply, that is, when x_i>100, the variable β_i will be free to be worth 1 or 0, and since we assume that at least one variable is worth 1, we are ensuring that at least one customer complies with the formula x_i>100. This can be seen more clearly if we turn the proposition around, (5.9) can be defined as:

$$\forall i : \quad \text{IF} \quad \beta_i = 1 \quad \text{THEN} \;\; x_i > 100 \tag{5.10}$$

Therefore, if $\beta_i = 1$, that means that x_i>100. There may be customers with $\beta_i =$ 0, and they may have bought more than 100 units, but that lack of precision is due to simplification. What is guaranteed is that at least one customer has more than 100, which is what the specification established.

With this type of simplification, always be careful not to use the same calculation for other specifications that require a precise value. Defining all the calculation values guarantees that there will never be an error.

In the following we will use the reference *Ref.* S_V every time we reduce values in the definition of a logical calculation.

5.4 Lower/Upper Bound Calculations

They are used to calculate upper or lower bounds on a set of variables. Its main function is in the modelling of equilibrium objective functions and specifications, although they can also have other uses, such as obtaining the entire part of a continuous variable.

The variables that function as bounds suppose a saving in the use of logical calculations. With bound calculations the size of the model is reduced and the efficiency in the resolution is improved.

We could use logical calculations to obtain the maximum or minimum value between the values of a set of variables, but the process would generate many constraints and variables. Bound calculations do not necessarily obtain the maximum or minimum of the set of variables, although it can be forced with the use of the objective function to achieve those values. The modeller must analyze if, with the bound variables, it can represent a specification that alludes to the maximum or minimum of a set. If that possibility does not exist, logical calculations would have to be incorporated to obtain them.

In any case, the bound calculations are always defined as follows:

If there is a set of variables x_i (i=1…n) that can define decision activities or calculations made in the system, the standard definition of a lower/upper bound calculation would be:

Upper/lower bound calculation: Upper/Lower bound of the set x_i
Applied to: …
Variable: x_{UP} = Upper bound of x_i (i=1…n) / x_{LOW} = Lower bound of x_i (i=1…n)
Constraints defining the calculation: $\forall i : x_i \leq x_{UP}$/ $\forall\ i : x_i \geq x_{LOW}$

From these two variables, we could model quantitative impositions such as the following:

Assume that the difference between the values of x_i is not greater than 2:
$x_{UP} - x_{LOW} \leq 2$
Assume that the minimum has a value of at least 10:
$x_{LOW} \geq 10$

Its usefulness is also in the objective functions of equilibrium, in which the bound calculations represent the cost variables of the function. Thus, objectives of equilibrium can be defined:

Maximize the minimum: Max x_{LOW}
Minimize the maximum: Min x_{UP}
Minimize the difference between the maximum and the minimum:
Min $x_{UP} - x_{LOW}$

With respect to other utilities of the upper/lower bound calculations, there is the obtaining of integer parts of continuous values. For that, the key is to define the calculation with integer value:

Let x be a continuous variable. To get its integer part ($[x]$), we define a lower bound calculation as follows:

Lower bound calculation: Lower bound of x
Applied to: …
Variable: x_{LOW} = Lower bound of x; x_{LOW} Integer
Constraints defining the calculation: $x \geq x_{LOW}$

To ensure that x_{LOW} takes the nearest integer to the value of x, we assume that x_{LOW} also acts as upper bound to the value of removing an amount infinitesimally close to 1 to x:

$x_{LOW} \geq x - (1 - \xi)$; ξ infinitesimal value

If we had wished to obtain the integer part by excess of $x(\lceil x \rceil)$, the calculation would have been:

Upper bound calculation: Upper bound of x
Applied to: . . .
Variable: x_{UP} = Upper bound of x; x_{UP} Integer
Constraints defining the calculation: $x \leq x_{UP}$

And that same variable also acting as the lower bound of $x + (1-;\xi)$:

$x_{UP} \leq x + (1 - \xi)$

5.4.1 Bounds on Undetermined Variables

When the lower/upper bound calculation is made on a set of variables where each variable depends on a condition to be part of the bounds calculation, the constraints that define the calculation are established by a logical proposition:

There is a set of variables x_i $(i = 1...n)$ representing activities or calculations:

Upper/lower bound calculation: Upper/Lower bound of the set x_i
Applied to: . . .
Variable: x_{UP} = Upper bound of x_i $(i = 1...n)/x_{LOW}$ = Lower bound of x_i $(i = 1...n)$
Constraints defining the calculation: Constraints resulting from the modelling of the following logical statement:
$\forall i$: IF {condition} THEN $x_{SUP} \geq x_i/x_{INF} \leq x_i$

Illustration 5.9
We plan to allocate flights that start or end in Barcelona on any given day to pilots of an airline. The company works with domestic flights that are always doubled, that is, a flight from Barcelona to another city always has a flight from that city to Barcelona, within the same day. For this reason, for the planning the company always assigns the two flights of a double flight to the same pilot, and for that same reason, it is not necessary to consider the individual flights but only the double flights.

The information of a double flight is:

- *Start time*
- *End time*

In the assignment, it is established as a rule that a pilot cannot carry out daily work of more than 12 hours. The schedule of the working day is computed as the difference between the latest end time minus the earliest start time of the double flights assigned that day.

Logically, we have expressed a very simplified allocation system, solely for the purpose of showing the use of bound calculations on undetermined variables.

Table 5.12 Table of elements of Illustration 5.9

			Data				
Elements	Set	QN	Name	Param	Type	Belonging	Value
Pilots	$j = 1...m$	I_U					
Double flights	$i = 1...n$	I_U	start time	S_i	C	W	–
			end time	E_i	C	W	–

Let us take a look at the table of elements and activities.

Table of Elements (Table 5.12)

The pilots are identical but are treated individually in the system ("*a pilot cannot carry out daily work of more than 12 hours*").

Decision Activities

Action: Assign [Pilots to double flights]

Decision variables: $\alpha_{ij} = 1$ if I assign double flight i to pilot j. j=1...m; i=1...n;

Specifications

The only specification defined explicitly in the statement is the one for which we will need the definition of bound calculations:

A pilot cannot carry out daily work of more than 12 hours

The specification falls on each pilot.

To obtain the working interval, we will use an upper bound calculation and a lower bound one, for each pilot. Bound calculations must be made at the beginning of the flights assigned to that pilot (lower bound calculation) and the end times of the flights assigned to the pilot (upper bound calculation).

The reference variables for the calculations would be auxiliary variables (section 5.4) that we would define as:

Auxiliary calculation: Start time of pilots on flights
Applied to: Pilots $j = 1...m \Rightarrow$ Flights i=1...n
Definition of calculation variable: x_{ij} = Start time of pilot j on flight i
Constraint that defines the calculation: $\forall i, j: \quad x_{ij} = S_i\alpha_{ij}$

Auxiliary calculation: End time of pilots on flights
Applied to: Pilots j=1...$m \Rightarrow$ Flights i=1...n
Definition of calculation variable: y_{ij} = End time of pilot j on flight i
Constraint that defines the calculation: $\forall i, j: \quad y_{ij} = E_i\alpha_{ij}$
The lower/upper bound calculations would be defined as:
Lower bound calculation: Lower bound of the set x_{ij}
Applied to: Each pilot $j = 1...m$
Variables: x_j^{INF} = Lower bound of the set x_{ij} (i=1...n)
Constraints defining the calculation: It would be a mistake to consider a direct bound calculation on all auxiliary calculations x_{ij}, since for those flights that the

pilot does not perform the value of x_{ij} is 0, and therefore the value 0 would act as a lower bound, wrongly. For that reason, we are only going to consider the pair (*i,j*) where $\alpha_{ij} = 1$, that is to say, where flights have been assigned to the pilot *j*.

$\forall j, \forall i : \text{IF } \alpha_{ij} = 1 \text{ THEN } x_j^{INF} \leq x_{ij}$

Without having used auxiliary variables x_{ij}, the definition would be as follows:

$\forall j, \forall i : \text{IF } \alpha_{ij} = 1 \text{ THEN } x_j^{INF} \leq I_i \alpha_{ij}$

On the other hand, for the calculation of the upper bound, it is not necessary to establish the condition of non-negativity that we have proposed in the calculation of the lower bound. As it is a calculation of upper bound, the values 0 do not affect the calculation:

Upper bound calculation: Upper bound of the set x_{ij}
Applied to: Each pilot $j = 1...m$
Variables*:* y_j^{SUP} = Upper bound of the set x_{ij} (i=1...n)
Constraints defining the calculation: $\forall j, \forall i : y_j^{SUP} \geq y_{ij}$

Without having used auxiliary variables y_{ij}: $\forall j, \forall i : y_j^{SUP} \geq F_i \alpha_{ij}$

Finally, we can express the specification of a maximum of 12 h per day as:

$\forall j : y_j^{SUP} - x_j^{INF} \leq 12$

References

Mendelson, E. (1997). *Introduction to mathematical logic* (4th ed.) Chapman and May.

Mitra, G., Lucas, C., & Moody, S. (1994). Tools for reformulating logical forms into zero-one mixed integer programs. *European Journal of Operational Research, 72*, 262–276.

Williams, H. P. (1995). Logic applied to integer programming and integer programming applied to logic. *European Journal of Operational Research, 81*, 605–616.

Williams, H. P. (2013). *Model building in mathematical programming* (5th ed.). Wiley. ISBN: 978-1-118-44333-0.

Chapter 6
Modelling and Types of Specifications

6.1 Introduction

The specifications of an optimization problem include all the constraints existing within it. The constraints have been catalogued generally according to their meaning, as Williams (2013) have already pointed out, although proposing a classification of constraints seems to be a rather complex task. The only classification that I consider 100% valid would be the one that refers to the sign of the constraints. There are constraints of three signs: $\leq$, $\geq$, and $=$. However, with this classification, we would not pay attention to the meaning of the specification. And on the other hand, not all specifications correspond to a single constraint. There are specifications, as we will see, that are modelled using more than one constraint.

Therefore, in this methodology, we will propose a classification in two levels. The upper level is determined by the way in which the specification is stated, whereby we would have two types of specifications:

A. Specification stated as a simple proposition
B. Specification stated as a compound proposition

In Sect. 5.3, dedicated to logical calculations, we state the concept of proposition and its typologies. It is important to bear in mind that a proposition in mathematical programming is defined as a mathematical semantic content that, when applied to a solution, can be assigned a truth value (true or false).

Remember also that there are two kinds of propositions:

– Simple: also called atomic, they express a statement that cannot be divided into other propositions because they do not employ any logical operator.

The original version of this chapter was revised. The correction to this chapter is available at https://doi.org/10.1007/978-3-030-57250-1_9

J. M. García Sánchez, *Modelling in Mathematical Programming*, International Series in Operations Research & Management Science 298,
https://doi.org/10.1007/978-3-030-57250-1_6

– Compound: propositions composed of logical operators and one or more simple propositions. In the previous chapter, several examples of the formulation of compound propositions were already shown. In this chapter we will study its modelling.

Specifications Stated as Simple Propositions
The specifications that are formulated as simple propositions give rise to most of the constraints of optimization problems. Simple propositions can also be classified, here with greater difficulty and not exclusively, depending on the meaning or objective of the constraints or the intervening variables. In this methodology, we will define the following types of specifications stated as simple propositions:

Quantitative specification of selection

A specification based on the variables involved. Whenever logical activities are presented on sets of individual elements or binary logical calculations, it is necessary to analyze the existence of this type of specification, since in many cases they are *not explicitly described in the description of the system because they are understood.*
They can be of three types:

– Upper bound
– Lower bound
– Equality

Capacity specifications
Based on the semantics of element data, it is a specification catalogued by some authors. It is based on availability data of elements that have a resource character in the system. The two most common formats in which it is presented in a system are:

– Specification focused on the consumption of capacity: The capacity of the resource can be partially consumed.
– Specification focused on the contribution of capacity: The capacity of the resource is used to satisfy a fixed demand for capacity. In this case, the capacity is used in its entirety.

Supply of a demand
Based also on data semantics, it is the opposite of the capacity specification. It occurs when an element has an attribute that represents a quantity demanded that is necessary to cover or supply.

Bound imposition specifications

These are the most easily recognizable specifications in a system. It would be a typology regarding the way of stating the specification. There are two types:

– Imposition of maximums: these impose upper bounds on variables or functions of variables. Within these we could also include the specifications of capacity consumption.
– Imposition of minimums: these impose lower bounds on variables or functions of variables.

Allocation or balance specifications

These impose values on variables or functions of measurable variables. They also impose a balance between the inputs and outputs of a measurable element, collective or individual. Included in this typology are constraints on the distribution of stocks or those that impose equality between variable functions.

It is inevitable that sometimes certain specifications will be considered as belonging to more than one typology.

Specifications Stated as Compound Propositions

The specifications enunciated as compound propositions have already appeared in the chapter dedicated to the logical calculations of a system. They are specifications that are based on a logical language, using logical operators or a connective to define the specification. They are also usually easily identifiable in the system description. As we already mentioned, the modelling of logical propositions could involve the definition of logical calculations. In this chapter, we will see the modelling of any type of compound proposition.

The formulation of logic-based propositions has been studied mainly by Mitra et al. (1994), Williams (1995), and Williams (2009) proposing constraints using binary decision variables, particular propositions as disjunctive constraints but there is not a general method for formulating any compound propositions with any connective.

In this book we will see a general scheme for modelling compound propositions where the basic rules are based on the modelling of propositions already described by those authors.

The structure of this chapter is as follows: first (Sect. 6.2) we will analyze the elements on which the specifications fall. In Sects. 6.3 to 6.7, we will present the modelling of each type of simple specification (A.1 to A.5). Section 6.8 will deal with the modelling of compound propositions. Section 6.9 is devoted to objective functions, since objective functions can be considered as one more specification of the system. In Sect. 6.10, we will analyze the identification of specifications in the description of a system.

6.2 Elements on Which the Specification Falls on

When it comes to modelling a specification, the first task is to identify to which elements the specification is directed. If it is an individual specification, it will be directed to a particular element within a set, to an element that is not part of any set or to the system itself. If the specification is directed to a set of elements, the construction of the specification uses terms such as "*Every*" or "*A*" as an indeterminate article, without referring to one in particular but to any. This means that the specification will have to be mounted for each element of the set. We can even express a specification using the term "*all.*" In this case the norm must also be mounted for each element individually if they are defined individually.

Illustration 6.1

In a system for assigning jobs to machines, with those elements defined as unitary, the specification that assigns each job to a machine could be defined as:

- *Each job must be assigned to a machine.*
- *A job will be assigned to a machine.*
- *All jobs must be assigned to a machine.*

It could also be that the specification does not apply to all elements of the set but only to those that meet a condition subject to the values of their data or their indices. This will be expressed using the mathematical symbol "such that" (/):

$\forall i/C_i{>}10$: . . . . (C_i is an attribute of the elements of the index set i)
$\forall i,j/i{>}j$: . . . .
$\forall i/i{\geq}3$: . . . .

We should never express variables in the clause *such that*. As its value is not determined, we will not know whether or not the specification is applied:

$\forall i/x_i{>}10$: . . . . **Error!**

Whenever we try to define variables in the clause "such that," we must define a conditional logical proposition "IF. . .THEN. . ." that uses these variables as an input condition of the conditional. The logical propositions will be studied in Sect. 6.8. Let's look at a simple example:

$\forall i/x_i > 10 : \alpha_i + \beta_i \leq 1 \Rightarrow\ \forall\, i : \text{IF}\, x_i > 10\, \text{THEN}\, \alpha_i + \beta_i \leq 1$

It may also be common for specifications to be defined by the combination of several sets of elements. Again, the determination of the elements depends on the way in which they are alluded to in the statement of the specification, if in a determined way on some elements or in an indeterminate way, which is why all the combinations that define the specification should be considered. Let us look at an example:

Illustration 6.2: Distribution of Ham

A ham distribution company has designed a set of 20 delivery routes for distribution. The company has a portfolio of 350 clients. Each delivery route goes through a series of known customers. The company has ten vehicles for the distribution stage. Each vehicle has a given capacity or number of Iberian hams that it can transport.

The demand for ham that must be supplied to each customer is known. Each vehicle that delivers ham must choose a single route, since more than one would take too long.

Two vehicles cannot choose the same route.

If a vehicle delivers more than 50 Iberian hams to a customer, it should not deliver ham to any other customer.

We know the delivery cost of each route.

Table of Elements (Table 6.1)

Table 6.1 Table of elements of Illustration 6.2

			Data				
Elements	SET	QN	Name	Param	Type	Belonging	Value
Routes	$i = 1\ldots 20$	I_U	Customer_Route	RC_{ij}	B	S	...
			Cost	C_i	C	W	...
Customers	$j = 1\ldots 350$	I_U	Demand	D_j	I	S	...
				RC_{ij}			
Vehicles	$k = 1\ldots 10$	I_U	Capacity	K_k	I	S	...
Iberian hams	–	C_D		D_i; K_k			

In the Customer_Route attribute, the customers by which each route passes are annotated. It is therefore shared between routes and customers.

Decision Activities

Action: Deliver [Iberian hams to customers with vehicles]
Decision variables:
x_{kj} = Number of Iberian hams delivered with vehicle k to customer j.
$k = 1\ldots 10, j = 1\ldots 350$

Action: Choose [Routes for vehicles]
Decision variables:
α_{ik} =1 if I choose Route i for Vehicle k; 0 otherwise. $k = 1\ldots 10$, $i = 1\ldots 20$

Let us analyze two of the specifications of the system:

Specification 1: *Two vehicles cannot choose the same route*

The first question would be: which two vehicles? They are not determined, so we should consider the combination of every two vehicles from the ten vehicles. Obviously, since vehicles are individuals, this value (two vehicles) cannot represent a numeral of a collective element.

The second question would be: which route? And again, the answer is indeterminate, so we should consider them all.

Therefore, the specification is presented for each pair of vehicles and each route. For vehicles, we can define a second subscript k'. The specification is defined by a logical statement:

$$\text{Logical proposition} : \forall k, k', i : \text{NOT } (\alpha_{ik} = 1 \text{ AND } \alpha_{ik'} = 1) \tag{6.1}$$

This specification could also be written as follows: The number of vehicles that can choose each route must be at most 1:

$$\forall i : \sum_{k=1}^{10} \alpha_{ik} \leq 1 \tag{6.2}$$

Specification 2: If a vehicle delivers more than 50 hams to a customer, it should not deliver ham to any other customer:

"a vehicle": indeterminate, any vehicle
"a customer": indeterminate, any customer

"any other customer": the rest of customers is also indeterminate, any customer.

$$\text{Logical proposition}: \forall j, j'/j' <> j, k : \text{IF}\, x_{kj} > 50 \text{ THEN}\, x_{kj'} = 0 \tag{6.3}$$

The specification could also have been defined with the following equivalent statement: If a vehicle delivers more than 50 Iberian hams to a customer, the sum of Iberian hams to the rest of customers will be zero:

$$\text{Logical proposition}: \forall j, k : \text{IF}\, x_{kj} > 50 \text{ THEN} \sum_{j' \neq j} x_{kj'} = 0 \tag{6.4}$$

Both in (6.2) and in (6.4) the number of specifications is reduced, which a priori may contribute to improving the resolution times of the model.

6.3 Quantitative Specifications of Selection

Specifications are very common in modelling. They are relevant when we work with variables of logical decision and can also be enunciated with respect to a set of binary logical calculations.

This specification expresses a quantitative condition with respect to the set of binary variables. It is to establish the amount of choices that are going to be given or that can be given as a maximum or minimum.

Format: Sum of Selection options SIGN Value
General Expression:
$\alpha_1 + \alpha_2 + \ldots + \alpha_n \approx Value$
Regarding the SIGN ($\approx$):

$=$Exact imposition or imposition of equality

$\leq$ Imposition of upper bound

$\geq$Imposition of lower bound

$Value$ =Quantification of the selection

The sum of binary variables expresses the set (or sets) of elements that are selection options. The most frequent case of this type of specification occurs between more than one set of individual elements, normally of unitary nature, which are combined in the events of the decision activity, although it is also possible to find it in a single set of elements.

As we will see in Sect. 6.10, this type of specification may not appear explicitly in the description of the system, so we will always have to do an analysis exercise of the quantitative selection rules when we have systems with logical activities.

Let us consider some examples of the specification in different scenarios:

Illustration 6.3: Case of a Set of Unitary Elements

Let us suppose there is a series of activities that make reference to the use of a unitary set of machines:

Elements: Machines $i = 1 \ldots n$ Unitary
Decision activities:
Use (Machines $i = 1 \ldots n$) $\Rightarrow \alpha_i = 1$ if I use machine i; 0 otherwise

Quantitative specifications of selection: In this system, specifications such as the following could be considered:

– *You must use (select) at least one machine:* $\sum_{i=1}^{n} \alpha_i \geq 1$

– *You must use three machines:* $\sum_{i=1}^{n} \alpha_i = 3$

Illustration 6.4: Case of Two Sets of Elements

Let us suppose there is a series of operators that must be assigned to a series of machines:

Elements: Machines $i{=}1 \ldots n$ Unitary; Operators $j{=}1 \ldots m$ Unitary
Decision activities:
Assign (Operators to Machines) $\rightarrow \alpha_{ij}{=}1$ if I assign operator j to machine i; 0 otherwise.
Quantitative specifications of selection:
Maximum two operators per machine: This specification falls on each machine of the system:

$$\forall i : \sum_{j=1}^{m} \alpha_{ij} \leq 2$$

An operator must be assigned only one machine: The specification refers to any operator, since it refers indeterminately to an operator. On the other hand, with machines it refers to the quantity of a machine.

$$\forall j : \sum_{i=1}^{n} \alpha_{ij} = 1$$

If we assume a power attribute (P) for the machines, we could find a specification that does not consider, as a selection option, all the elements of the set of machines:

A machine with power greater than 10000w will be assigned to operator 3:

$$j = 3 : \sum_{i/P>10000} \alpha_{ij} = 1$$

Illustration 6.5: Case of a Set with Choices About the Set Itself

Let us suppose there is a series of elements that must be assigned among them. Each element must be associated with another element of the same set:

Elements: Elements $i=1\ldots n$ Unitary;
Decision activity:

Assign [Elements to Elements] $\rightarrow$ $\alpha_{ij}=1$ if I assign element i to element j; 0 otherwise.

Quantitative specifications of selection:

- Each element is associated with a different element of itself.
  $\forall i: \sum_{j/j\neq i} \alpha_{ij} = 1$
- Since i and j are indices of the same set and is a bi-univocal correspondence, the association of i with j is the same as that of j with i, therefore: $\forall i,j: \alpha_{ij} = \alpha_{ji}$

Illustration 6.6: Case of Two Sets That Share the Selection

Let us suppose that we have divided elements of the same functionality in the system into two sets due to data, but both share the same logical decision activity on which a quantitative selection rule is applied.

Elements:
Individuals $i=1\ldots n$ Unitary;
Groups_Type1 $j=1\ldots m_1$ Unitary;
Groups_Type2 $k=1\ldots m_2$ Unitary;
Decision activity:
Assign [Individuals to Groups_Type1 and Groups_Type2] $\rightarrow$
$\rightarrow$Events: Individuals to Groups_Type1; Individuals to Groups_Type2 $\rightarrow$
$\rightarrow$ $\alpha_{ij}=1$ if I assign Individual i to Group_Type1 j; 0 otherwise.
$\rightarrow$ $\beta_{ik}=1$ if I assign Individual i to Group_Type2 k; 0 otherwise.
Quantitative specifications of selection:
One individual in a group at most:

$$\forall i: \sum_{j=1}^{m_1} \alpha_{ij} + \sum_{k=1}^{m_2} \beta_{ik} \leq 1$$

6.4 Capacity Specifications

Capacity specifications are ones that appear in systems where there are elements acting as resources. For this, they must have an attribute of capacity or availability, either intrinsic to the element or shared with a measurable element, collective or individual. They usually appear in different ways that are synthesized in the following general format:

Format: *Capacity Consumption or Demand for Capacity* $\leq$ *Capacity Contribution*

Expression: Both the consumption or demand for capacity and the contribution of capacity can be fixed and/or variable. Capacity contribution and consumption must be expressed in the same unit of measure. In the same specification we can have both a fixed and a variable consumption or contribution at the same time.

Capacity consumption or demand		Capacity contribution	
Variable	Fixed	Variable	Fixed

Variable Capacity Consumption

This corresponds to the classic consumption format in a capacity specification.

$$\text{General expression of consumption}: c_1x_1 + c_2x_2 + \ldots + c_nx_n \tag{6.5}$$

The capacity consumption is made up of a series of variables represented as x_1, x_2, ..., x_n. Each activity performs a consumption that is given by the expression c_ix_i where c_i is the unit consumption of activity x_i (the amount of capacity consumed by each unit of value that variable x_i takes).

Fixed Capacity Consumption or Demand

This corresponds to an attribute of the element that demands that capacity. It appears in systems where a set of elements provide capacity of a measurable element that has a stock that corresponds to the fixed capacity demand. Decisions focus on the capacity contribution, not on the amount that is consumed or demanded, which is fixed.

Fixed Capacity Contribution

An element provides a given capacity, collected as an attribute. Decisions are focused on how much of that existing capacity is consumed. The specification falls on the proprietary element of that fixed capacity.

Variable Capacity Contribution

This usually appears in specifications where a fixed capacity is demanded. The general expression corresponds to an expression similar to (6.5):

General expression of contribution: $a_1y_1 + a_2y_2 + \ldots + a_my_m$

The variables y_j, $j = 1\ldots m$, represent activities that provide capacity, with a_j being the unit contribution.

There is a set of elements that provide capacity to an element that demands capacity. Decisions are focused on the contribution of capacity, not on the quantity that is consumed or demanded, which is constant. The system does not collect the amount of capacity consumed on each element that contributes capacity, but it collects the amount of capacity contributed.

This case is very similar to the next type of specification, the specification of demand contribution. In any case, they are treated here independently because semantically they are different. In the specification of demand contribution, a

Table 6.2 Most common formats in a capacity specification (Table 6.2)

Cases	Format	Expression
1	*Variable capacity consumption ≤ fixed capacity contribution*	$c_1x_1 + c_2x_2 + \ldots + c_nx_n \leq K$
2	*Variable capacity consumption ≤ fixed and variable capacity contribution*	$c_1x_1 + c_2x_2 + \ldots + c_nx_n \leq K + a_1y_1 + a_2y_2 + \ldots + a_my_m$
3	*Fixed capacity consumption or demand ≤ variable capacity contribution*	$E \leq a_1y_1 + a_2y_2 + \ldots + a_my_m$

quantity of a measurable element is demanded since it is not possessed, and some elements can contribute to it. Here, we already have a quantity of a measurable element and capacity for this is requested. Basically, it is a semantic differentiation.

The most common formats of capacity specifications are collected in Table 6.2. Let us analyze each case.

6.4.1 Case 1: Variable Capacity Consumption and Fixed Contribution

The relationship is established between the capacity consumption of an element and the fixed amount of capacity that the element has in the system.

Format: *Consumption ≤ Capacity*
General Expression: $c_1x_1 + c_2x_2 + \ldots + c_nx_n \leq K$

This type of specification is usually not presented explicitly in the system description, as we will see in Sect. 6.10, but the modeller must detect them from the data of the table of elements.

Illustration 6.7: Production of Butter
We have already analyzed this problem amply. The specifications that are presented are only of capacity.

Table of Elements (Table 6.3)

Table 6.3 Table of elements

			Data				
Elements	Set	QN	Name	Param	Type	Belong	Value
Machines	$i = 1\ldots2$	I_M	Usage time	T_i	C (Min)	P	{6,3.5}
			Time consumed by 1 kg of butter j in machine i	TM_{ij}	C (Min)	C	{3m;3m; 3m;6m}
Butters	$j = 1\ldots2$	I_M	Profit	B_j	C ($)	P	...
				TM_{ij}			

Decision Activities

Action: Produce Butters
Decision variables: x_j = Amount of butter type j produced; j = 1,2;

Capacity Specifications

The Whipping Machine ($i = 1$) has a usage time of $T_1 = 6$ hours which is measurable.

Variables x_1 and x_2 consume TM_{11}=3 min. and TM_{12}= 3 min.

Expressing the specification in minutes: $3x_1 + 3x_2 \leq 360$

$$\text{Parametric}: \sum_{j=1}^{2} \text{TM}_{1j} x_j \leq T_1 \tag{6.6}$$

– The Pasteurization Machine ($i = 2$) also has a T_2 usage time:

$$\sum_{j=1}^{2} \text{TM}_{2j} x_j \leq T_2 \tag{6.7}$$

And the parameterized expression that collects (6.6) and (6.7) would be:

$$\forall i: \quad \sum_{j=1}^{2} \text{TM}_{ij} x_j \leq T_i$$

Although in this type of specification it is usually more common for consumption to be carried out by measurable activities, logical activities can also participate as consumption, as in the following illustration.

Illustration 6.8: Management of a Warehouse

Management of a warehouse in which it is necessary to place a set of Pieces (i = 1..n) in a set of Shelves (j = 1 … m). Each piece has a weight and a volume. Each shelf has a load capacity and a volume.

Capacity specifications:

Do not load a shelf with more weight than it can support.

Do not load a shelf with a total volume higher than its own.

Table of Elements (Table 6.4)

Decision Activities

Action: Assign Pieces to Shelves
Decision variables: α_{ij} = 1 if I assign piece i to shelves j; 0 otherwise.

Table 6.4 Table of elements of illustration 6.8

Elements	Set	QN	Data				
			Name	Param	Type	Belonging	Value
Pieces	$i = 1 \ldots n$	I_U	Weight	p_i	C	W	...
			Volume	v_i	C	W	...
Shelves	$j = 1 \ldots m$	I_U	Load	PE_j	C	W	...
			Volume	VE_j	C	W	...

Capacity Specifications

Do not load shelves with more weight than it can support: specification applied to each of the shelves

$$\forall j: \sum_{i=1}^{n} p_i \alpha_{ij} \le PE_j$$

Do not load a shelf with a total volume higher than its own: also applied to each of the shelves:

$$\forall j: \sum_{i=1}^{n} v_i \alpha_{ij} \le VE_j$$

6.4.2 Case 2: Variable Consumption with Fixed and Variable Capacity Contribution

Although it is less common, we can also find activities to increase capacity in addition to consumption. Let us take a look at the following illustration.

Illustration 6.9: Production of Pills

600 Kg of a certain drug is available to make large and small pills. We also have the possibility of buying more drugs at a price of \$0.05/gram.

The big pills require 40 g and the small ones 30 g.

Each large pill provides a profit of \$2 and a small one of \$1.

The manufacturing capacity is 2000 pills.

Table of Elements (Table 6.5)

Decision Activities

Action: Produce [pills]
Decision variables: x_i = Number of pills produced of type i

Action: Buy [drug]
Decision variables: z **= Amount of drug bought**

Table 6.5 Table of elements of Illustration 6.9

Elements	Set	QN	Data				
			Name	Param	Type	Belonging	Value
Drug	–	I_M	Availability	E	C	W	600 Kg
			Price	p	C	W	\$0.05/ gr
			Amount in pills	c_i	C	S	{40,30}
Pills	$i = 1{,}2$	C_I		c_i			
			Profit	b_i	C	W	{\$2, \$1}
Factory (System)	–	I_U	*Manufacturing capacity*	K	I	W	2000

Capacity Specifications

(Case 2) Drug capacity: relapses on the drug. There is a capacity contribution collected in the variable z.

$$\sum_{i=1}^{n} c_i x_i \leq E + z$$

(Case 1) Manufacturing capacity: applied in the system.

$$\sum_{i=1}^{n} x_i \leq K$$

6.4.3 Case 3: Fixed Capacity Demand and Variable Capacity Contribution

There are systems where elements with capacity data and therefore acting as a resource do not individually define a capacity consumption specification. The reason is that the problem can be modelled without there being activities that measure (discretely or continuously) the consumption of that resource. The capacity attribute is not measured because it is assumed that it has either not been used or used in its entirety. The system has an element that demands a capacity, and it is necessary that other elements provide that capacity. Sometimes this specification occurs because there is a capacity attribute that is assigned to each item of a collective element. Since the instances are not treated individually but collectively, the capacity attribute cannot be used for a capacity consumption specification, since this would be applied individually on each item. The capacity attribute is used as contribution. There are also cases where the capacity attribute is held by an individual element that acts as unitary.

The format of the specification would be:

Format: *Fixed capacity demand* $\leq$ variable *capacity contribution*

Let us first look at an example that illustrates the case of capacity data imputable to each item of a collective element (illustration 6.10), and a second illustration where the capacity attribute is possessed by individual elements. In this second illustration we will propose the two possible versions: measuring capacity through consumption and not measuring the capacity of each individual element, by using it logically in a capacity contribution specification (illustration 6.11).

Illustration 6.10: Celebration Hall

There is a celebration hall where a wedding will take place. The hall has tables of two sizes. There are tables of 8 seats and tables of 12 seats, 14 and 11 of them, respectively.

150 guests attend the wedding. The company must decide which tables to use so that the guests can sit down, minimizing the number of tables used.

Table of Elements (Table 6.6)

Since each type of table is formed by an identical set of items and the text does not refer individually to them, the tables are configured as collective. The same happens with the guests. We make explicit the seats as element although there are no decisions about the seats but so there is a demand for seats for the guests.

Regarding the tables, there are two sets of data on capacity: availability of tables and the number of seats. The availability is applied to the element globally or collectively, while the number of seats is capacity attribute regarding each item of the collective element, so that attribute will not be associated with a specification of capacity consumption.

Decision Activities

The system imposes the action of using tables as a decision activity. We do not need to decide where to locate the guests; we simply select tables and establish the specification of placing 150 guests, based on a certain value action corresponding to a specification of capacity demand.

Table 6.6 Table of elements of Illustration 6.10

Elements	Set	QN	Data				
			Name	Param	Type	Belonging	Value
Tables	$i = 1,2$	C_D	Availability	N_i	I	W	{14,11}
			Seats	K_i	I	S	{8,12}
Seats	–	C_D		K_i			
Guests	–	C_D	Number	NG	I	W	150

Action: Use tables
Decision activities: x_i = Number of tables i used

Specifications

To seat the 150 guests: We can understand the specification as the guests demanding a capacity of 150 seats. The tables act as a resource for these places. Each type of table contributes a certain number of places:

Fixed capacity demand ≤ Variable capacity contribution

$$150 \leq 8x_1 + 12x_2$$

The capacities of the tables are being used as a contribution. The capacity refers to each item of the table.

Table availability: This attribute does give rise to a capacity consumption specification for each type of table

$$\forall i: \quad x_i \leq N_i$$

Objective Criterion

$$\text{Min} \quad x_1 + x_2$$

Illustration 6.11: Containers
There is a set of n containers each with a capacity of volume and a cost. There is a liquid product with an amount of C. The objective consists in the selection of containers at a minimum cost so that we can store the quantity C of the product.

Version 1: By Capacity Consumption
In this first version, we analyze the system as a system to store liquid in containers. Therefore, the capacity data will be measured in the specifications. The liquid is a measurable element by its own continuous quantity used partially.

Table of Elements (Table 6.7)

Decision Activities

Table 6.7 Table of elements of Illustration 6.11 – Version 1

			Data				
Elements	Set	QN	Name	Param	Type	Belonging	Value
Containers	$i = 1\ldots n$	I_U	Cost	c_i	C	W	...
			Capacity	K_i	C	W	...
Liquid product	–	I_M	Amount	Q	C	W	...

Action: Store liquid product in containers
Decision variables: x_i = amount of liquid product introduced in container i

We measure the liquid because it is the direct object of the action, "store liquid."

Specifications

Capacity consumption: In this scenario a capacity consumption specification is to be proposed for each container, to ensure that the quantity introduced does not exceed the capacity of the container:

$$\forall i : x_i \leq K_i \tag{6.8}$$

Balance specification: Although we have not yet seen the balance specifications, in the system, a balance specification is generated with the liquid availability data. The amount of liquid distributed by all the containers corresponds to the quantity Q.

$$\sum_{i=1}^{n} x_i = Q \tag{6.9}$$

Objective Criterion

Since we minimize the cost of the used or *selected* containers, it is necessary to calculate that qualifier for the containers, which becomes a logical calculation on the containers (if you store in a container, it is because you have selected it). If we had considered it as a decision activity, that is, on the one hand store and on the other select, the implicit relationship between the two activities cannot be ignored as a specification: If I store then I select.

Binary logical calculation: Selected container
Applied to: Containers *i=1...n*
Definition of logical variable: $\alpha_i = 1$ if I select container i; 0 otherwise
Logical proposition: $\forall i : {}_i = 1$ IF AND ONLY IF $x_i > 0 \Rightarrow$ Ref. S_V (The selection of a container has a cost so it is useless to select containers if you do not put anything in them [Sect. 5.3.3])

$$\Rightarrow \forall i : \text{IF} \quad x_i > 0 \quad \text{THEN} \quad \alpha_i = 1 \tag{6.10}$$

In spite of not having presented the modelling of logical propositions yet, we are going to write the constraints that are generated from modelling (6.10), since it will be necessary for the simplification process between versions. The constraints generated are:

$$\forall i : \quad x_i \le K_i \alpha_i \tag{6.11}$$

where K_i acts as the upper bound of what can be stored in each container.

The objective function would be:

$$\text{Min} \quad \sum_{i=1}^{n} c_i \alpha_i \tag{6.12}$$

Version 2: By capacity contribution

In this second version, we will not introduce liquid into the containers, and therefore we will not measure at the specifications the capacity of the containers. The only thing we are going to do as a decision activity is select containers, making sure that the liquid fits. The liquid becomes unitary because we stop making a partial use of its quantity in the decision activities.

Table of Elements (Table 6.8)

Decision Activities

Action: Select [Containers]

Decision variables: $\alpha_i = 1$ if I select container i; 0 otherwise.

Specifications

1. *Capacity contribution*: The liquid element demands a capacity for its quantity. Each container i has a capacity contribution of value K_i.

$$Q \le \sum_{i=1}^{n} K_i \alpha_i \tag{6.13}$$

Objective Criterion

Minimize costs:

Table 6.8 Table of elements of Illustration 6.11 – Version 2

			Data				
Elements	Set	QN	Name	Param	Type	Belonging	Value
Containers	$i = 1\ldots n$	I_U	Cost	c_i	C	W	...
			Capacity	K_i	C	W	...
Liquid product	–	I_U	Amount	Q	C	W	...

$$\text{Min} \quad \sum_{i=1}^{n} c_i\alpha_i \tag{6.14}$$

Relation Between Versions

Objective functions (6.12) and (6.14) are identical. Version 1 is equivalent to version 2 but logically larger. If we perform two simple operations, we can get to version 2 from version 1:

On the one hand, we have in Version 1:

- Modelling of the logical proposition (6.11): $\forall i: \quad x_i \leq K_i\alpha_i$
- Capacity specification (6.8): $\forall i: x_i \leq K_i$

It is only necessary to use (6.11). If I select the container (α_i=1) I cannot store more than its capacity, and if I do not select the container (α_i=0), then $x_i \leq 0 \Rightarrow x_i = 0$.

On the other hand, if we add the n constraints of (6.11), we obtain:

$$\sum_{i=1}^{n} x_i \leq \sum_{i=1}^{n} K_i\alpha_i \tag{6.15}$$

Substituting (6.9) with (6.15):

$\sum_{i=1}^{n} x_i = Q \leq \sum_{i=1}^{n} K_i\alpha_i$

This corresponds with (6.13), the only constraint that the first version had. In this way, it is guaranteed that at least quantity Q can be stored, and the use of decision variable x_i is not necessary.

6.5 Supply of a Demand

This is a specification which is analogous to the capacity contribution specification. It expresses a relationship between the supply of a measurable element, collective or individual, and the demand requested of that measurable element.

Format: *Measurable element supply* $\geq$ (=) *Measurable element demand*
General Expression:
The supply is considered variable in the specification. Demand is considered both fixed and variable, although the most common case is fixed.

$a_1x_1 + a_2x_2 + \ldots + a_nx_n \geq D + d_1y_1 + d_2y_2 + \ldots + d_my_m \quad or$

$a_1x_1 + a_2x_2 + \ldots + a_nx_n = D + d_1y_1 + d_2y_2 + \ldots + d_my_m$

The supply is made up of a series of variables represented as x_1, x_2,. . .x_n

Each activity makes a contribution determined by the value of the expression: $a_i x_i$ where a_i is the unit contribution of the variable x_i (the amount supplied by each unit of the value that takes the variable x_i).

The quantity demanded is defined as D. The variables y_j represent activities that may exist in the system to increase demand (d_j would be the unit increase in demand for each activity y_j). The supply and the demand must be expressed in the same unit of measure.

Regarding the sign of the constraint, we can find cases where only $\geq$ is admissible, others where only $=$ is admissible, and others where both signs are valid. The validity is governed by the following rules:

– The expression with sign $\geq$ is correct when the supply involves a cost in the system or there are unit contributions other than 1.
– The expression with $=$ is correct as long as the unit contributions are all equal to 1.

If the unit contributions are different from 1, imposing equality could make the problem inadmissible because we would not obtain values of x_i that give equality, or we would exclude solutions that could be better for the objective function. Let us analyze this validity with an illustration:

Illustration 6.12: Buying from Providers

There is a simplified system of buying a product from two providers. It is necessary to buy 1000 units of the product. The purchase cost is €3 for both providers. (We can ignore other system features).

Table of Elements (Table 6.9)

Decision Activities

Action: Buy product from providers

Decision variables: x_i = Product units bought from provider i

Demand Supply Specification

There is a fixed demand $D = 1000$ units of the product for which we have the variable x_i as input. The unit contribution is 1. For each unit of product purchased, a unit of demand is provided.

In this case, the two signs would be valid:

$x_1 + x_2 \geq 1000$ Buying has a cost.

Table 6.9 Table of elements of Illustration 6.12

			Data				
Elements	Set	QN	Name	Param	Type	Belonging	Value
Providers	$i = 1,2$	I_U	Cost	p_i	C	S	{\$3, \$3}
Product	–	C_D		p_i			
			Demand	D	I	W	1000

Table 6.10 Table of elements of Illustration 6.12 modified

			Data				
Elements	Set	QN	Name	Param	Type	Belonging	Value
Provider 1	–	I_U	Product Cost	*p*	C	S	€3
Provider 2	–	I_U	Lot Cost	Pl	C	S	€22
Lot	–	C_I		Pl			
			Product units	*N*	I	S	8
Product	–	C_D		*p; N*			
			Demand	*D*	I	W	1000

$x_1 + x_2 = 1000$ Unitary contributions are 1.

Now we can modify the statement and incorporate the following information:

"Provider 2 does not serve single units but serves lots of 8 units at a price of $22."

This modifies the table of elements and the decision activity as follows:

Table of Elements (Table 6.10)

Decision Activities
Action: Buy product from provider 1 and lots from provider 2
Events: Product $\Rightarrow$ provider 1; Lot $\Rightarrow$ provider 2
Decision variables:
x = Product units bought from provider 1
y = Lots bought from provider 2

Demand Supply Specification

The two decision variables, x and y, act as input, but in this case, the unit contribution of the variable y is 8 (8 units of product are contributed for each lot).

$x + 8y \geq 1000$: Correct expression. Buying has a cost and the contributions are not all the units.

$x + 8y = 1000$: Incorrect expression. Solutions are excluded.

Obviously, if the system explicitly specifies that we must provide exactly the amount of demand, then we are obliged to use the equality sign.

6.6 Bound Imposition Specifications

Bound imposition specifications are usually the simplest to identify and model in a system. They establish lower or upper bounds for measurable activities or for variable functions (calculations). There are two types:

Upper bound: Imposition of maximum value
Lower bound: Imposition of minimum value

Both the capacity specifications and the demand contribution with sign $\geq$ can be understood as a particular case of bound specification. However, we have taken them out of this category because of the meaning they express. Bound specifications do not have to represent a concept of capacity consumption or demand supply but are impositions without defined semantics and of an explicit nature (they must be explicitly defined in the statement).

Illustration 6.13

For a system that generates the following decision variables:

x_i *= Number of product units purchased from provider Pi; i = 1 . . . n*

The following bound specifications are defined:

1. Do not buy more than 50 units from provider 1: $x_2 \leq 50$.
2. Buy at least 10 units from provider 2: $x_1 \geq 10$.
3. Buy more than 20 units from provider 3: $x_1 > 20 \Rightarrow x_1 \geq 21$.

Illustration 6.14

System: A set of operators (O1, On) that work in the production of a set of substances (S1, Sm). The system measures the amount of substance produced by each operator. The number of kilos produced by operator 1 must be at least 1 kilo more than the kilos of operator 2.

Table of Elements (Table 6.11)

Decision Activities

Action: Produce substances using operators

Decision variables: x_{ij} = Kilos of Substance j produced by operator i

Specifications

Imposition: The number of kilos produced by operator 1 must be at least 1 kilo more than the kilos of operator 2:

Number of kilos produced by operator 1: Not included in the decision variables since it is an auxiliary calculation (y_1):

$$y_1 = \sum_{j=1}^{m} x_{1j}$$

Number of kilos produced by operator 2:

$$y_2 = \sum_{j=1}^{m} x_{2j}$$

Table 6.11 Elements of Illustration 6.14

Name	Set	QN
Operators	$i = 1\ldots n$	I_U
Substances	$j = 1\ldots m$	I_M

Bound specification: a lower limit is imposed on the kilos produced by the operator 1.

$y_1 \geq y_2 + 1$

6.7 Allocation, Balance, or Equilibrium Specifications

Allocation, balance, or equilibrium specifications are all of the specifications that define:

- An exact assignment to or imposition of a value on an integer or continuous variable or on variables functions.
- Balance or equilibrium between variables or functions of variables. This includes auxiliary calculations.

Their controversy is that they can cause equivalences between auxiliary calculation and decision activity.

Illustration 6.15
For a system that generates the following decision variables:

x_i = *Number of product units purchased from provider Pi; i = 1 . . . n*

A. *Buy 10 units from provider 1:*
$x_1 = 10$
B. *The sum of quantities purchased from provider 1 and 2 must be 50:*
$x_1 + x_2 = 50$
C. *Buy a total of 500 units:*
$\sum_{i=1}^{n} x_i = 500$
D. *Buy the same from provider 1 as from 3:*
$x_1 = x_3$

As we have said, this type of specification makes certain decision variables cease to exist because they are given a certain value (Case A) or because they become auxiliary calculations (cases B, C, D). However, if we differentiate the description of elements and activities from the specifications, we can consider them as decision variables. In any case, the system does not vary, but its structure can simply be represented in several ways.

This type of specification has greater relevance when a relationship of flow balance is expressed. This happens when a directed graph G (N, A) formed by nodes (N) and arcs (A) is established as a scenario, where a concept that corresponds to a measurable element of the system flows through the graph.

The adjective of directed graphs (Bang-Jensen and Gutin, 2000) is important since the systems that are represented as non-directed graphs (formed by nodes and edges) have a completely different meaning. An undirected graph always has a

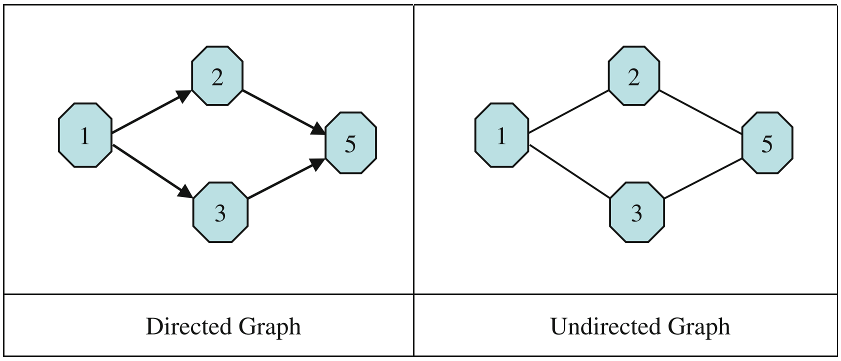


Fig. 6.1 Directed and undirected graphs

Table 6.12 Table of elements of a system with a directed graph

Elements	Set	QN	Data				
			Name	Param	Type	Belonging	Value
Nodes	$i = 1 \ldots n$	I	Origin node	NO_{ik}	B	S	. . .
			Destination node	ND_{ik}	B	S	. . .
			Flow injection	I_i	C	S	. . .
			Flow demand	D_i	C	S	. . .
			Node Data	. . .	. . .	. . .	. . .
Arcs	$k = 1 \ldots m$	I		NO_{ik}; ND_{ik}			
			Arc Data	. . .	. . .	. . .	. . .
Flow	–	$\mathrm{C/I_M}$		I_i; D_i			
Graph	–	$\mathrm{I_U}$					

meaning of relationship or association between elements that are presented as nodes. These relationships are represented by its edges. Both nodes and edges can have associated data (Fig. 6.1).

A directed graph instead has a different meaning. The nodes are elements of the system, and the arcs are connections between nodes to enable the circulation of an added concept, the flow, in most cases. There are some scenarios where the direction has some meaning of relationship between the nodes, as dependency or offspring meanings. When there is flow, the flow is an element of the problem whose activity is to circulate through the arcs of the graph. This flow must be injected into the graph by the nodes and must also be extracted from the graph through the nodes.

The constraints of flow balance mean that the flow to arrive at a node must be equal to the flow that leaves it.

In Chap. 3 (Tables 3.46, 3.47, and 3.48), we saw three ways of representing the table of elements of a directed graph. We use Table 3.47, incorporating the flow concept (Table 6.12):

As a directed graph, the data of injection or flow input and demand or flow output that each node can possess have been incorporated. A node will inject flow, demand flow, or do neither of these. On the other hand, we can also find systems where injecting or demanding flow is not an attribute but a decision activity. In that case, those data would disappear.

In the system, the sum of the flow injected must always be equal to the sum of the flow demanded.

In *Node Data* and *Arc Data*, all those specific data of the system associated with nodes and arcs would be represented, respectively.

Regarding the decision activities, each system will have specific activities, but there is always the flow circulation activity in common:

Action: Circulate [flow through the arcs]
Participating elements: Flow C/I_M; Arcs $k = 1\ldots m$;
Quantification: Continuous
Events: Flow $\Rightarrow k{=}1\ldots m$
Decision variables: x_k = Flow that circulates through the arc k.

In general, the constraint of equilibrium or flow balance is expressed as follows:

Format: At each node of the graph: Sum of the Input Flow = Sum of the Output Flow
General expression:

$$\forall i: \sum_{k/\mathrm{ND}_{ik}=1} x_k + I_i = \sum_{k/\mathrm{NO}_{ik}=1} x_k + D_i$$

The flow x_k of all the incoming arcs in i ($k/\mathrm{ND}_{ik} = 1$) plus the flow I_i that injects the node is recorded as input flow.

In output flow, the flow x_k of all the outgoing arcs of i ($k/\mathrm{NO}_{ik} = 1$) plus the D_i flow demanded by the node is recorded.

We are going to differentiate two types of scenarios in which these restrictions are presented, and therefore we require the use of directed graphs:

- **Explicit Case**: the system itself is a directed graph on which an optimization problem is raised (shortest path, maximum flow, minimum cost flow, etc.).
- **Implicit Case**: this is the most interesting case. The system is not described as a graph. However, part of its activity can be represented by a directed graph. The condition for this is that there must be a **measurable element**, individual or collective, subject to activities in which unitary elements also participate. This acquires a greater meaning if a set of **time periods** participates in the system as elements. The complexity in the implicit case lies in the creation of the graph, although we will give a series of guidelines for its construction in Sect. 6.7.2.

6.7.1 Explicit Case

Within the problems associated with directed graphs, we can see one of the most known and applied, which is the shortest path problem. Let us look at an illustration of it:

Illustration 6.16: Shortest Path *Problem (Dijkstra 1959)*
There is a graph G (N, A) where each arc has a cost, obtaining the shortest path from a source node to a destination node. We model the problem for the graph of the figure, using Node 1 as the source node and Node 9 as destination.

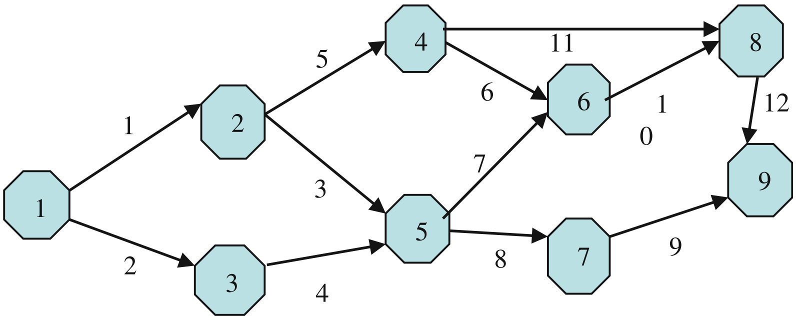


Fig. 1 Directed graph G(N,A)

Nodes and arcs have been labelled with a number.

Table of Elements

The problem uses a flow unit that will flow from node 1 to node 9. Therefore, node 1 injects a flow unit that requires node 9. Although we inject an integer amount of flow, it is not necessary to consider the flow as collective (discrete measurable), since in operative research it is demonstrated that by considering it as a continuous

Table 6.13 Table of elements of illustration 6.16

			Data				
Elements	Set	QN	Name	Param	Type	Belong	Value
Nodes	$i = 1\ldots9$	I_U	Origin node	NO_{ik}	B	S	…
			Destination node	ND_{ik}	B	S	…
			Flow injection	I_i	C	S	{1,…,0,0}
			Flow demand	D_i	C	S	{0,…,0,1}
Arcs	$k = 1\ldots12$	I_U		NO_{ik}; ND_{ik}			
			Cost	c_k	C	W	…
Flow	–	I_M		I_i; D_i			
Graph	–	I_U					

measurable, due to the property of unimodularity of its coefficients matrix, variables will always be integers in the optimal solution (Table 6.13).

Decision Activities

Action: Circulate flow through the arcs
Decision variables: x_k = Flow that circulates through the arc k.

Specifications

Flow balance:

$$\forall i: \sum_{k/\mathrm{ND}_{ik}=1} x_k + I_i = \sum_{k/\mathrm{NO}_{ik}=1} x_k + D_i$$

Objective Function

$$\text{Min} \sum_{k=1}^{12} c_k x_k$$

The shortest path problem does not need any additional specification. The specification of flow balance not only guarantees the conservation of the flow but also its continuity; therefore, the set of arcs of the solution guarantees a path.

On the other hand, there are some problems associated with undirected graphs that have been modelled, transforming the graph into directed and introducing a flow concept in it. Examples include the Minimum spanning tree problem, MST (Graham and Hell 1985), or the Steiner problem (Hwang et al. 1992), as well as variants thereof. Let us consider the case of the MST problem.

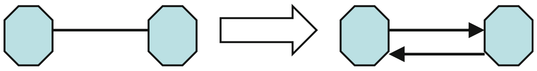
Illustration 6.17: MST Problem

There is an undirected graph G (N, A), where each edge has a cost. We try to obtain the minimum cost spanning tree, that is, obtain a subgraph of G that connects all the nodes at a minimum cost.

To model this problem with the use of a directed graph, we proceed as follows:
Each edge is transformed into two arcs:

Table 6.14 Table of elements of Illustration 6.17

			Data				
Elements	Set	QN	Name	Param	Type	Belonging	Value
Nodes	$i = 1\ldots n$	$\mathrm{I_U}$	Origin node	NO_{ik}	B	C	...
			Destination node	ND_{ik}	B	C	...
			Flow injection	I_i	C	C	...
			Flow demand	D_i	C	C	...
Arcs	$k = 1\ldots 2m$	$\mathrm{I_U}$		NO_{ik}; ND_{ik}			
			Cost	c_k	C	P	...
Flow	–	$\mathrm{I_M}$		I_i; D_i			
Graph	–	$\mathrm{I_U}$					



$n-1$ flow units are included in the problem (n = number of nodes) that will send (inject) a node, labelled as root node, to the rest of nodes of the graph. Therefore, node i will have an injection $I_i = n-1$, with $D_i = 0$, and the $n-1$ remaining nodes $I_i = 0$ and $D_i = 1$.

In the objective function, we consider the cost of the arcs through which the flow has circulated, so that the cost is incurred if the flow has circulated, regardless of the amount of flow that has circulated. The cost of each arc is the cost of the associated edge.

Table of Elements (Table 6.14)

It is not necessary to use a binary attribute that identifies the root node, since it can be identified by the injection of $n-1$ flow units.

Decision Activities

Action: Circulate flow through the arcs.
Decision variables: x_k = Flow that circulates through the arc k. $k = 1\ldots2m$.

Specifications

Flow balance:

$$\forall i : \sum_{k/\mathrm{ND}_{ik}=1} x_k + I_i = \sum_{k/\mathrm{NO}_{ik}=1} x_k + D_i$$

Objective Criterion

For the objective function, it is necessary to define a logical calculation on each arc to know whether or not the flow has circulated:

Binary logical calculation: Circulate flow through an arc.
Applied to: Arcs *k=1...2m.*
Definition of logical variable:
$\forall k$: $\alpha_k = 1$ if the flow circulates through arc k; 0 otherwise.
Logical proposition: $\forall k : \alpha_k = 1$ IF AND ONLY IF $x_k > 0 \Rightarrow$ Ref. $S_V \Rightarrow \forall k$: IF $x_k > 0$ THEN $\alpha_k = 1$.
The expression of costs would be as follows:

$$\text{Min} \sum_{k=1}^{2m} c_k \alpha_k$$

6.7.2 *Implicit Case*

The implicit case is an optional support tool for the identification of equilibrium specifications between variables of the system. It occurs in systems where a measurable element is subjected to activities in which other no measurable individual elements participate. These activities suppose injections or demands of the measurable element and even the transfer of quantities between individual elements. This becomes even more relevant if the time element participates in the system, that is, if there is a set of periods in which the activities occur. The activities contribute, demand, or simply move units of the element over time.

In a graph, only the movement of a measurable element can be represented. If there is more than one measurable element in a system, a graph must be made for each of them.

If there are no periods of time, there is only one implicit period in which the activities take place, as already mentioned in Chap. 3. However, what does need to happen to represent the problem as a graph is that other individual elements must participate in the measurement activities.

As we have said, it is not mandatory to design a directed graph to model these systems, but it is convenient. The graph will contain the activities of the system with respect to that measurable element and the relationships between them.

The construction procedure of the graph is the following:

Nodes
We will use a node for each individual element that intervenes in the flow of units of the measurable element in each period of time, except for the time element. By default, we can use all individual elements as nodes in each period of time and afterwards eliminate those that are not connected in the final graph. There are systems where an element only participates in a certain period, and therefore its use does not make sense in other periods.

The nodes can inject or demand flow. If they are known values, they correspond to data of those elements. The sum of the flow injected must always be equal to the demanded flow. It is also possible that the node injects or demands flow, but the amount is not determined. In this case, injecting or demanding flow corresponds to decision activities or calculations. By annotating this in the graph, we will represent the injection with a negative superscript (−) and the demand with a positive superscript (+).

Arcs
In the arc activities, simple calculations and data of the measurable element are represented.

We must analyze:

- The movements of units between elements.
- The elements that maintain units over time: since the nodes cannot store units, in order to respect the principle of flow balance, the units not subject to any activity in a node must also circulate over time to the node that represents the same

element in the next period. With this we achieve a circulation between nodes that is equivalent to the storage of units of the measurable element over time in that element.

Finally, once the graph has been designed, it will be necessary to explore which arcs and which auxiliary calculations define decision activities.

The design of the graph does not have to adapt to a single configuration. Depending on the interpretation of the activities over time, different designs can be generated.

From the constructed graph, a flow balance constraint is proposed on each node. This type of graph can always be refined and simplified, since nodes that have an input arc and an output arc can be discarded for the balance.

Table 6.15 Elements of Illustration 6.18

			Data				
Elements	Set	QN	Name	Param	Type	Belonging	Value
Warehouses	$i = 1 \ldots m$	I_U	Stock	K_i	I	S	...
			Cost	C_{ij}	C	S	...
Customers	$j = 1 \ldots n$	I_U	Demand	D_j	I	S	...
			Cost	C_{ij}			
Product	–	C_D		$K_i; D_j; C_{ij}$			

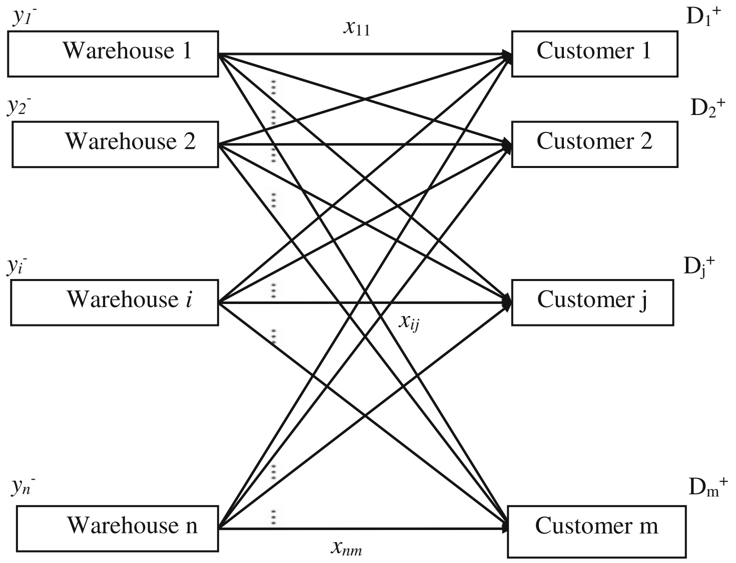


Fig. 6.2 Implicit directed graph of Illustration 6.18

This type of graph has always been used in some optimization problems, to turn them into problems associated with directed graphs. A clear example is the transport problem, which has been modelled as a minimum cost flow problem:

Illustration 6.18: Transportation Problem (Öztürk et al. 2015)
A company has m warehouses where its products are located. Each warehouse i (i = 1...m) has a stock of Ki units. There is a set of n customers (j = 1...n) with a demand Dj of product units. The company has to supply the product demand of the customers from the warehouses. The cost of sending a product from each warehouse i (i = 1...m) to each customer j is estimated in c_{ij}.

Table of Elements (Table 6.15)

We are facing a system that does not have periods of time. The measurable element is the product. We designed a graph (Fig. 6.2) with the *m* warehouses and the *n* customers:

The graph will collect the admissible movements of the product units (flow), which is produced from each warehouse to each customer. Nodes associated with warehouses inject an undetermined amount of flow (y_i for warehouse i). The nodes associated with customers demand a certain amount of flow, their product demand D_j. The arcs between each warehouse and each customer include the problem decision activities (x_{ij} = product units that are sent from warehouse i to customer j).

The equations of flow balance generated by the graph are:

$\forall i: \quad y_i = \sum_{j=1}^{m} x_{ij}$ The flow injection could be defined as an auxiliary calculation.

$\forall j: \quad \sum_{i=1}^{n} x_{ij} = D_j$ This corresponds with a specification of demand supply (unit contributions allow the use of the equality sign).

To finish formulating the transport problem, we would have to incorporate the capacity consumption specification at each warehouse:

$$\forall i: \quad \sum_{j=1}^{m} x_{ij} \leq K_i$$

Objective Function

Minimize associated costs. Each decision variable has a unit cost:

$$\text{Min} \quad \sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij} x_{ij}$$

Let us now take a look at an illustration of the implicit case in a system that considers more than a period of time. Production planning problems are included in this type of case:

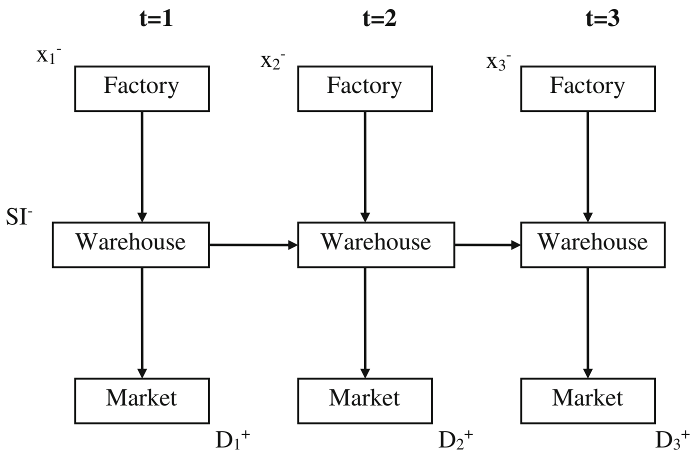


Fig. 6.3 Implicit directed graph of Illustration 6.19

Table 6.16 Elements of Illustration 6.19

Elements	Set	QN	Data				
			Name	Param	Type	Belonging	Value
Factory	–	I_U					
Warehouse	–	I_U	Initial stock	SI	I	S	5
Market	–	I_U	Demand	D_t	I	S	{30,12,26}
Product	–	C_I		SI; D_t			
Months	$t = 1...3$	I_U		D_t			

Illustration 6.19: Production Planning of a Product (Larrañeta et al. 1995)
System for the production planning of a factory that produces a product for which there is a market demand for the next three months of 30, 12, and 26 units. A warehouse is available to store the manufactured units. Initially, there is a stock in the warehouse of five units.

The system must determine the quantities produced in each period as well as the quantities stored (Fig. 6.3).

Table of Elements (Table 6.16)

The three individual elements participate in the three periods.

Since the warehouse can keep units from one period to the next, we join those nodes with arcs. Labelling flow movements (Fig. 6.4):

The nodes with an entry or injection and an exit or demand can simplify the labelling, as it is evident that for each node factory and each node market, it is fulfilled by flow balance (Fig. 6.5):

$$F_i = x_i \quad i = 1, 2, 3$$
$$E_i = D_i \quad i = 1, 2, 3$$

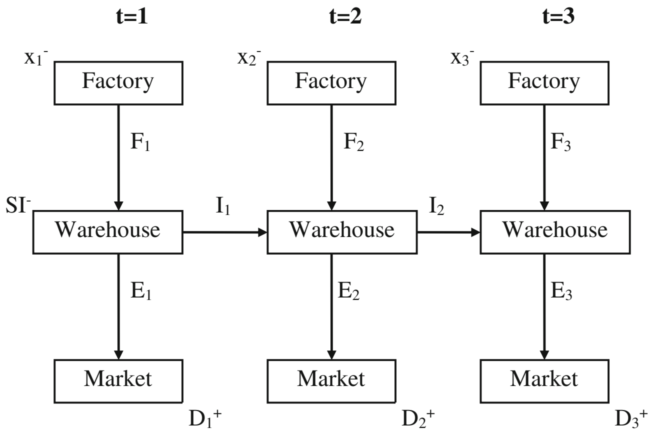


Fig. 6.4 Labelled graph of Illustration 6.19

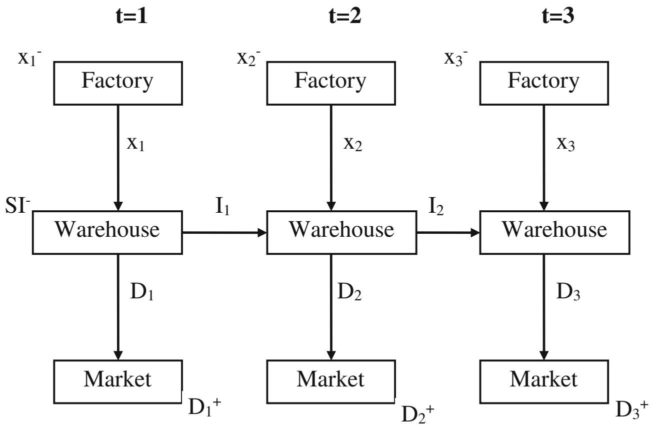


Fig. 6.5 Simplified labelled graph of Illustration 6.19

Therefore, we generate only the equations in the warehouse nodes:

Warehouse $t=1$: $x_1 + SI = D_1 + I_1$
Warehouse $t=2$: $x_2 + I_1 = D_2 + I_2$
Warehouse $t=3$: $x_3 + I_2 = D_3$
Generically: $\forall t: \quad I_{t-1} + x_t = D_t + I_t \quad (I_0 = SI)$

The flow balance equations define in themselves both the demand supply specification of the market element in each period and the auxiliary calculation of the

Table 6.17 Elements of Illustration 6.20

			DATA				
Elements	Set	QN	Name	Param	Type	Belonging	Value
Banquets	$i = 1\ldots3$	I_U	Tablecloths	m_i	I	S	...
			Day	d_{it}	B	S	...
Storeroom	–	I_U	Stock	S	I	S	200
Market	–	I_U	Price	p	C	S	12
Basket	–	I_U					
Laundry	–	I_U					
Fast wash	–	I_U	Cost	cF	C	S	6
Slow wash	–	I_U	Cost	cL	C	S	4
Days	$t = 1\ldots3$	I_U		d_{it}			
Tablecloths	–	C_I		m_i; S; p; cF; cL			

quantity stored in each period (the action of storing is not really a decision activity but an auxiliary calculation):

Picking up the excess of units of what stays in the warehouse:

$$I_1 = (\text{SI} + x_1) - D_1$$
$$t = 2: \quad I_1 + x_2 \geq D_2$$
$$I_2 = (I_1 + x_2) - D_2$$
$$t = 3: \quad I_2 + x_3 \geq D_3$$
$$I_3 = (I_2 + x_3) - D_3$$
$$\forall t: \quad I_{t-1} + x_t \geq D_t \Rightarrow \forall t: \quad I_{t-1} + x_t - I_t = D_t$$

Finally, to reinforce the graph design, let us consider an example with more content:

Illustration 6.20: Food Service (Illustration 3.9.2)

A food service business has contracted three banquets for the next 3 days, requiring 150 clean tablecloths for the first banquet, 100 for the second, 140 for the third, and 130 for the fourth. Currently, it has 200 tablecloths in the storeroom, all of them clean, and they can buy what you need on the market every day at a cost of 12 m.u/ tablecloth.

After the banquets, the tablecloths can go to the laundry basket or be sent to the laundry to be washed. The laundry offers the following washing services:

– *Fast: Clean tablecloths for the next day, at a cost of 6 m.u/tablecloth.*
– *Slow: Clean tablecloths in 2 days, at a cost of 4 m.u/tablecloth.*

Table of Elements (Table 6.17)

Next, we present a graph design. In the graph, we have excluded the possibility of washing slowly from the second and three periods and washing quickly from the fourth one.

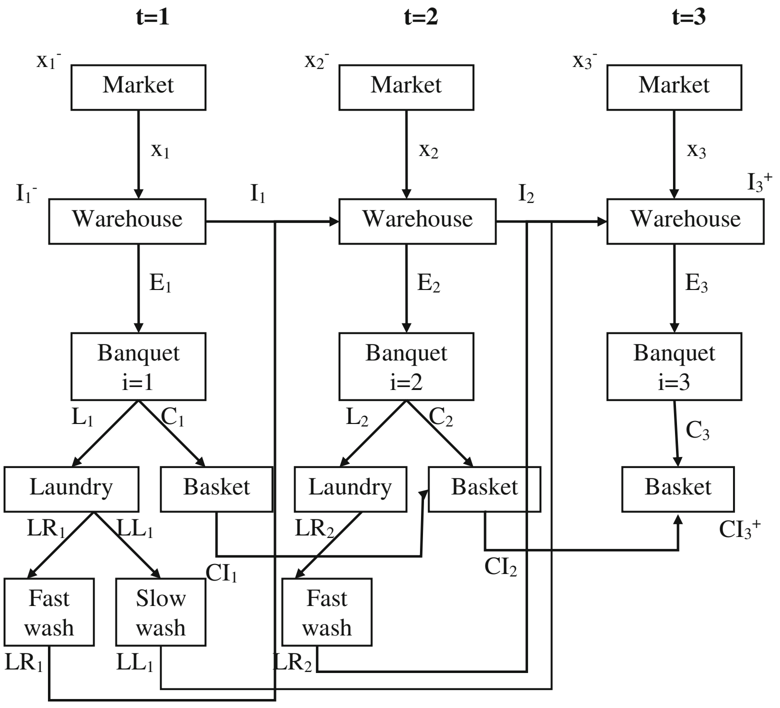


Fig. 6.6 Implicit directed graph of Illustration 6.20

On the other hand, since the injected units must be extracted from the graph, the warehouse and the laundry basket are taken as nodes that demand flow units in the last period of undetermined value.

For the flow balance we can exclude the market because it has a flow injection and a single exit arc. Similarly, the nodes that represent the fast wash and slow wash element have a single input and output and it is not necessary to propose the balance equation.

The labelling of arcs has been as follows:

x_t: Quantity of tablecloths purchased on day t
I_t: Quantity of tablecloths in store (clean tablecloths)
E_t: Quantity of tablecloths brought to the banquet $i = t$ ($E_t = D_t$)
L_t: Quantity of tablecloths sent to be washed on day t
C_t: Quantity of tablecloths taken to laundry basket on day t
CI_t: Quantity of tablecloths in basket
LR_t: Quantity of tablecloths washed quickly on day t
LL_t: Quantity of tablecloths washed slowly on day t (Fig. 6.6)

It would have been possible to use two elements to represent the tablecloth element: clean tablecloth and dirty tablecloth. This configuration would also have been correct in the system, but it was not necessary to make the distinction since in

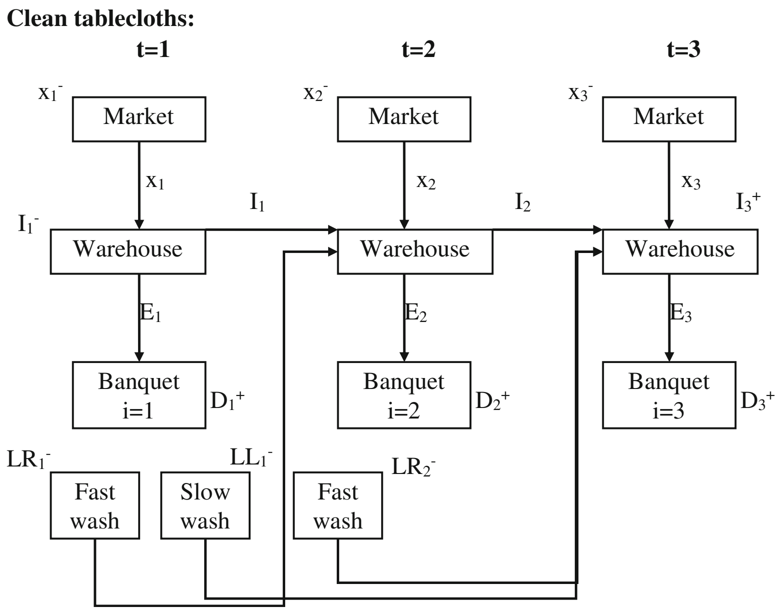


Fig. 6.7 Implicit directed graph for clean tablecloths

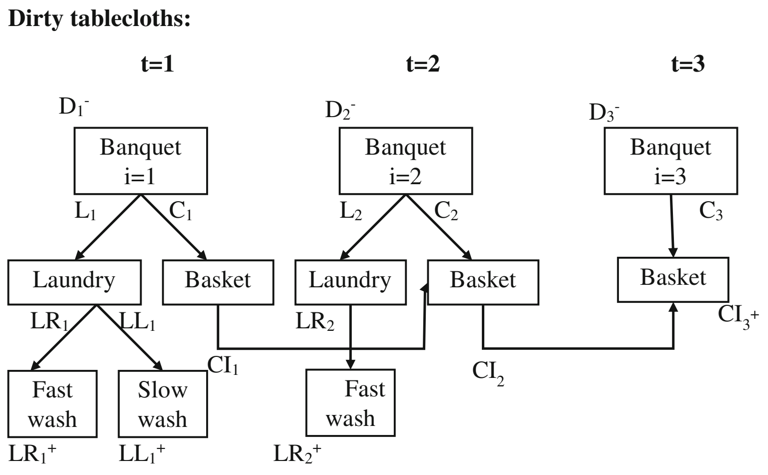


Fig. 6.8 Implicit directed graph for dirty tablecloths

the decision activities the concepts are not intermingled (only clean tablecloths are bought, only dirty tablecloths are washed, etc.). The system can work with a single concept.

However, if we had made the distinction in the table of elements, we should have designed two graphs, one for each measurable element. The flow of each graph should be related later. The design would be as follows:

Clean tablecloths (Fig. 6.7).

Dirty tablecloths (Fig. 6.8).

The tablecloth demand D_t becomes a demand for flow in the graph of clean tablecloths and injection of flow in the graph of dirty tablecloths. With the tablecloths to be washed (LR and LL), the opposite happens.

6.8 Modelling of Propositional Logic Specifications

At the beginning of the chapter, we assigned specifications to the nature of propositions, which may be simple or compound. The simple propositions are all the typologies that we have just studied. Compound propositions are those propositions that use logical operators or connectives (If . . . then; If and only if; Not; Or; And; Either . . . or) to relate simple propositions. In this section, we focus on the modelling of compound propositions. Compound propositions are a key aspect in the formulation of optimization problems of a certain depth.

Since compound propositions are the basis of propositional logic, we shall consider the propositional logic specification as that which is formulated as a compound proposition. When using operators, we are always representing a compound proposition.

We already saw in Chap. 5 that logical calculations were defined by compound propositions, and therefore these propositions can be considered as a propositional logic specification.

Let us look at some examples of specifications and logical calculations that give rise to compound propositions:

Illustration 6.21

For a product purchasing system with five suppliers, the following decision variables are generated:

xi = units of the product purchased from the supplier i. i = 1 . . . 5.

Specifications that we could define:

- *Logical proposition 1. – We cannot buy units from supplier 1 and supplier 2*:
  "NOT (x_1>0 AND x_2>0)"
- *Logical proposition 2. – If you buy more than 10 units from supplier 1 you cannot buy more than 5 units from supplier 3*:
  "IF x_1>10 THEN $x_3 \leq 5$"
- Logical proposition 3. – You must buy 25 units from only one of the suppliers:
  "EITHER x_1=25 OR x_2=25 OR x_3=25 OR x_4=25 OR x_5=25"

- Logical proposition 4. – If you buy more than 10 units from supplier 4 or supplier 5, you must buy 15 units from supplier 1:
  "IF x_4>10 OR x_5>10 THEN x_1=15"
- *Logical proposition 5. – If the system needs a logical calculation to know from which suppliers we have purchased units*:

  Binary logical calculation: Supplier provides units
  Applied to: Each supplier *i=1…5*
  Variables: $\alpha_i = 1$ if we buy units from supplier i; 0 otherwise. i=1…5
  Logical proposition: $\forall i: \quad \alpha_i = 1 \quad \text{IF AND ONLY IF} \;\; x_i > 0$

The difficulty of modelling logical propositions lies not so much in obtaining the constraints that define it, which as we will see in the following sections is based on the application of some rules but on correctly stating the proposition.

We have already defined the concepts related to the propositional logic in Chap. 5, when we present the logical calculations. Now we present some of these concepts with the aim of structuring their modelling. For modelling, we propose a general scheme where some rules are based on the modelling of propositions already described by authors. We emphasize as a reference the modelling of propositions described by Williams (2013).

6.8.1 Simple Propositions and Logical Operators

Atomic or simple propositions are those that are defined without the use of any logical operator. The format in a lineal formulation would be:

Left part	Sign	Right part
X (Lineal function)	$< ; \leq ; = ; \geq ; >; \neq$	Independent term (Numeric value)

In mathematical programming, we will distinguish three types of simple propositions:

- Binary simple proposition: The lineal function from the left part only takes binary values (1; 0).
- Integer simple proposition: The lineal function from the left part only takes integer values.
- Continuous simple proposition: The lineal function from the left part takes continuous values.

This distinction is necessary for the modelling of compound propositions.

In mathematical programming, any valid or admissible solution must satisfy a truth result (T). Therefore, any restriction such as those defined in Sections 6.3, 6.4, 6.5, 6.6 and 6.7 would correspond to a simple proposition, with an admissible solution of the problem being one that satisfies a truth result when applied to the

Table 6.18 Truth tables of logical operators

Operator	Symbol	Semantic	Truth table
Negation	$\neg$	NOT (ϕ)	$\begin{array}{c\|c} \phi & \neg\phi \\ \hline \mathrm{F} & \mathrm{T} \\ \mathrm{T} & \mathrm{F} \end{array}$
Disjunction	$\vee$	ϕ OR ψ	$\begin{array}{cc\|c} \phi & \psi & \phi\vee\psi \\ \hline \mathrm{F} & \mathrm{F} & \mathrm{F} \\ \mathrm{F} & \mathrm{T} & \mathrm{T} \\ \mathrm{T} & \mathrm{F} & \mathrm{T} \\ \mathrm{T} & \mathrm{T} & \mathrm{T} \end{array}$
Conjunction	$\wedge$; &	ϕ AND ψ	$\begin{array}{cc\|c} \phi & \psi & \phi\wedge\psi \\ \hline \mathrm{F} & \mathrm{F} & \mathrm{F} \\ \mathrm{F} & \mathrm{T} & \mathrm{F} \\ \mathrm{T} & \mathrm{F} & \mathrm{F} \\ \mathrm{T} & \mathrm{T} & \mathrm{T} \end{array}$
Conditional	$\rightarrow$	IF ϕ THEN ψ	$\begin{array}{cc\|c} \phi & \psi & \phi\rightarrow\psi \\ \hline \mathrm{F} & \mathrm{F} & \mathrm{T} \\ \mathrm{F} & \mathrm{T} & \mathrm{T} \\ \mathrm{T} & \mathrm{F} & \mathrm{F} \\ \mathrm{T} & \mathrm{T} & \mathrm{T} \end{array}$
Biconditional	$\leftrightarrow$	ϕ IF AND ONLY IF ψ	$\begin{array}{cc\|c} \phi & \psi & \phi\leftrightarrow\psi \\ \hline \mathrm{F} & \mathrm{F} & \mathrm{T} \\ \mathrm{F} & \mathrm{T} & \mathrm{F} \\ \mathrm{T} & \mathrm{F} & \mathrm{F} \\ \mathrm{T} & \mathrm{T} & \mathrm{T} \end{array}$
Exclusive disjunction	$\bigoplus$	EITHER ϕ OR ψ	$\begin{array}{cc\|c} \phi & \psi & \phi\oplus\psi \\ \hline \mathrm{F} & \mathrm{F} & \mathrm{F} \\ \mathrm{F} & \mathrm{T} & \mathrm{T} \\ \mathrm{T} & \mathrm{F} & \mathrm{T} \\ \mathrm{T} & \mathrm{T} & \mathrm{F} \end{array}$

Table 6.19 Equivalences between operators

Proposition	Equivalent proposition	Reference
$\phi \leftrightarrow \psi$	$\phi \rightarrow \psi$ $\psi \rightarrow \phi$	f_1
$\phi \bigoplus \psi$	$(\phi \wedge \neg(\psi)) \vee (\neg(\phi) \wedge \psi)$	f_2
$\phi \rightarrow \psi$	$\neg\psi \rightarrow \neg\phi$	f_3

simple proposition. When faced with composite propositions, where we use logical operators, the allowable solutions must satisfy the truth results of the operator's truth table. Let us see the truth tables of each operator:

ϕ y ψ are shown as logical propositions (Table 6.18).

Already in Chap. 5 devoted to logical calculations, we presented equivalences between some operators. Table 6.19 collects those equivalences in addition to another with the operator Exclusive disjunction ($\bigoplus$). The operator Biconditional ($\leftrightarrow$) as well as the operator Exclusive disjunction could be ignored thanks to these equivalences. However, for convenience, we will take a look at the modelling of those operators as well. From Table 6.19, we will label with references all the transformations or formulations that can be used in the modelling of propositions, in order to be able to reference the origin of the transformation in the text.

6.8.2 Reduction of Signs

For integer or continuous simple propositions, the group of signs is convenient to reduce it to the set ($\leq$; $=$; $\geq$), except for those to which the negation operator applies. In that case, for simplicity, reduction is not necessary.

Calling X the linear function of the proposition and V the independent term or numerical value of the right part of the simple proposition, the transformation of integer/continuous propositions is the following:

ξ is a small enough value to avoid in continuous propositions that the value less than V that the linear function could take is greater than $(V - \xi)$, in the case of $X < V$. The same applies for $X > V$.

We do not consider the case of binary atomic propositions in the reduction of signs, since the forms in which they can be presented are reduced to:

α as a binary linear function: $\alpha = 1$; $\alpha = 0$;

For convenience in modelling, in the case $\alpha = 0$, we can change the value to 1 using the following equivalence:

$\alpha = 0 \Rightarrow 1 - \alpha = 1$; $(1 - \alpha)$ is still a binary expression. [Reference f_7]

6.8.3 Modelling Operators Individually

First, we are going to analyze the modelling of connectives or logical operators individually, that is, we only consider compound propositions that do not have more than one different operator.

Table 6.20 Reduction of signs in simple propositions

Simple proposition	Reduction		
Sign	$X \in Z$	$X \in \mathfrak{R}$	Reference
$X < V$	$X \leq \lceil V \rceil - 1$	$X \leq V - \xi$	f_4
$X > V$	$X \geq \lfloor V \rfloor + 1$	$X \geq V + \xi$	f_5
$X \neq V$	$X < V$ OR $x > V$ $\Rightarrow$ Ref. f_4 *y* f_5 $\Rightarrow$ $(X \leq \lceil V \rceil - 1)$ OR $(X \geq \lfloor V \rfloor + 1)$	$X < V$ OR $X > V$ $\Rightarrow$ Ref. f_4 *y* f_5 $\Rightarrow$ $(X \leq V - \xi)$ OR $(X \geq V + \xi)$	f_6

Table 6.21 Increment and decrement parameters

	$V \in Z$		$V \notin Z$	
	$X \in Z$	$X \in \mathfrak{R}$	$X \in Z$	$X \in \mathfrak{R}$
$\mathbf{R^I}$	1	ξ	$\lceil V \rceil - V$	ξ
$\mathbf{R^D}$	1	ξ	$V - \lfloor V \rfloor$	ξ

To express the constraints resulting from modelling operators, we will distinguish between the type of value of the linear function (binary, integer, or continuous) and the sign ($\leq$; $=$; $\geq$) for the case of integer or continuous simple propositions.

We will denote with the variables X or Y the function of the left part of an integer or continuous simple proposition. The binary propositions will be expressed with a Greek letter (α, β, ω, δ, etc.).

In the modelling of operators, the integer or continuous simple propositions will only be differentiated in the increment or decrement parameters of the independent term V, as in the case of the reduction of signs (Table 6.20). Therefore, to simplify the notation, we are going to call the increment parameter R^I and the decrement parameter R^D. They will be defined as (Table 6.21):

ξ it will be a small enough value.

In the development of the modelling of some compound propositions, the definition of binary logical calculations is necessary, as we expressed in Section 5.2.3 of the previous chapter. These logical calculations serve to collect the result of simple propositions that are within the compound proposition. They will be collected in binary variables denoted by ω or by δ^1, δ^2, δ^3, and δ^4, when necessary. With these calculations we are going to ignore this semantic and mathematical definition. The proposition that defines them mathematically is integrated within the formulation of the operator.

On the other hand, for any integer or continuous atomic proposition, it will be necessary to obtain an upper bound and a lower bound of the linear function. The upper bound is a value that is never surmountable by the function. Equivalently, the lower bound is a value that can never be exceeded inferiorly by the linear function. Any value of dimension will be valid in the modelling, although adjusting the upper bound to the maximum of the linear function and the lower one to the minimum reduces the space of solutions and usually offers better behavior in the resolution. If

Table 6.22 Negation operator modelling

Sign	Model	Reference
NOT ($X < V$)	$X \geq V$	f_8
NOT ($X \leq V$)	$X > V \Rightarrow f_5 \Rightarrow X \geq V + R^I$	f_9
NOT ($X > V$)	$X \leq V$	f_{10}
NOT ($X \geq V$)	$X < V \Rightarrow f_4 \Rightarrow X \leq V - R^D$	f_{11}
NOT ($X = V$)	EITHER ($X < V$) OR ($X > V$) $\Rightarrow$* ($X < V$) OR ($X > V$) $\Rightarrow$($X \leq V - R^D$) OR ($X \geq V + R^I$)	f_{12}

*In that case, the exclusive disjunction coincides with the inclusive disjunction since the two propositions can never be fulfilled at the same time. The modelling would not have ended in case f_{12} since the connective OR would have to be modelled (Sect. 6.8.3.4)

Table 6.23 Nomenclature

Input Proposition	ϕ
Output Proposition	ψ
Binary functions	$\alpha; \beta$
Integer/continuous functions	$X; Y$
Independent terms of integer/continuous functions	$V; V_1; V_2$

Table 6.24 Conditional operator modelling with binary input proposition

IF ϕ THEN ψ					
ϕ		ψ			
Type	Sign	Type	Sign	Model	Ref.
Binary	$\alpha = 1$	Binary	$\beta = 1$	$\alpha \leq \beta$	f_{13}
Binary	$\alpha = 1$	Integer/continuous	$X \leq V$	$X \leq V + (UB_X - V)(1-\alpha)$	f_{14}
Binary	$\alpha = 1$	Integer/continuous	$X \geq V$	$X \geq V\alpha + LB_X(1-\alpha)$	f_{15}
Binary	$\alpha = 1$	Integer/continuous	$X = V$	$X \leq V + (UB_X - V)(1-\alpha)$ $X \geq V\alpha + LB_X(1-\alpha)$	f_{16}

X is our integer or continuous function, we will denote its dimensions with the following parameters:

Upper bound of X: UB_X
Lower bound of X: LB_X

Table 6.25 Conditional operator modelling with integer/continuous input proposition

	IF ϕ THEN ψ						
	ϕ			ψ			
Ref	Type	Sign	Model	Type	Sign	Model	Ref.
f_{17}	Int/ Cont	$X \le V_1$	$X \le V_1 + (UB_X - V_1)(1-\omega)$ $X \ge (V_1 + R^I)(1-\omega) + LB_X\,\omega$	Int/ Cont	$Y \le V_2$	$Y \le V_2 + (UB_Y - V_2)(1-\omega)$	f_{20}
f_{18}	Int/ Cont	$X \ge V_1$	$X \ge V_1\omega + LB_X(1-\omega)$ $X \le (V_1 - R^D)(1-\omega) + UB_X\,\omega$	Int/ Cont	$Y \ge V_2$	$Y \ge V_2\omega + LB_Y(1-\omega)$	f_{21}
f_{19}	Int/ Cont	$X = V_1$	$X \le V_1 + (UB_X - V)(1-\omega)$ $X \ge V_1\omega + LB_X(1-\omega)$ $X \le (V_1 - R^D)\delta_1 + UB_X\omega + UB_X\delta_2$ $X \ge (V_1 + R^I)\delta_2 + LB_X\omega + LB_X\delta_1$ $\delta_1 + \delta_2 = 1 - \omega$	Int/ Cont	$Y = V_2$	$Y \le V_2 + (UB_Y - V_2)(1-\omega)$ $Y \ge V_2\omega + LB_Y(1-\omega)$	f_{22}

6.8.3.1 Negation Operator (NOT; $\neg$)

Negation operator modelling does not require any complex modelling exercise; it is just based on representing the opposite proposition. We show the case of whole or continuous propositions; for binary propositions, the application of the connective negation is something evident (Table 6.22).

6.8.3.2 Conditional Operator (IF … THEN … ; $\rightarrow$)

The modelling of the conditional operator will be separated into two tables. In Table 6.24, we will present the modelling of the connective when the proposition of input of the condition is binary. In Table 6.25, we will deal with the modelling options when the input and output propositions are integer or continuous.

The nomenclature used in both Tables 6.24 and 6.25 is shown in Table 6.23.

Whenever possible, we should avoid the signs of equality in simple propositions. If the independent term corresponds to a lower bound of X, we can replace it with the sign $\le$ ([Reference f_{LB}]). Similarly, if it corresponds to an upper bound, we can work with the sign $\ge$ ([Reference f_{UB}]).

Any combination of types of propositions not contemplated in the two previous tables can easily be deduced with the use of the equivalences Ref. f_3 and Ref. f_7.

The tables could have been further reduced, since we can change the sign $\ge$ to the sign $\le$ simply by multiplying the proposition by -1. Even equality corresponds to two propositions of sign $\ge$ and $\le$ with the connective AND, but I prefer this representation to facilitate the obtaining of the mathematical formulation without having to change the propositions too much.

Let us take a look at some illustrations:

Illustration 6.22

We have x_1, x_2, x_3 integer variables ≥ 0.

We also have α_1 and α_2 binary variables.

IF α_1=1 THEN x_1+x_2 $\leq$ 10 $\Rightarrow$ Ref. f_{14} [$\alpha = \alpha_1$; $X = x_1+x_2$; $V=10$] $\Rightarrow$
$\Rightarrow x_1 + x_2 \leq 10 + (UB_{x_1+x_2} - 10)(1 - \alpha_1)$
IF $\alpha_1 = 0$ THEN $\alpha_2 = 0$ $\Rightarrow$ Ref. f_7 $\Rightarrow$ IF $1-\alpha_1 = 1$ THEN $1-\alpha_2 = 1$
$\Rightarrow$ Ref. f_{13} $\Rightarrow$ $1 - \alpha_1 \leq 1 - \alpha_2$
IF $\alpha_2 = 0$ THEN $x_1 > 10$ $\Rightarrow$ Ref. f_5 $\Rightarrow$ IF $\alpha_2 = 0$ THEN $x_1 \geq 10$ $\Rightarrow$ Ref. f_7 $\Rightarrow$
$\Rightarrow$ IF $1-\alpha_2 = 1$ THEN $x_1 \geq 11$ $\Rightarrow$ Ref. f_{15} $\Rightarrow$ $x_1 \geq 11\alpha_2 + LB_{x_1}(1 - \alpha_2)$

$\Rightarrow[LB_{x_1} = 0]$

$\Rightarrow x_1 \geq 11\alpha_2$

IF $x_1 > 5$ THEN $x_2 \leq 3$ $\Rightarrow$ Ref. f_5 $\Rightarrow$ IF $x_1 \geq 6$ THEN $x_2 \leq 3$ $\Rightarrow$ Ref. f_{18} ; f_{20} $\Rightarrow$

$$\begin{array}{llll} & x_1 \geq 6\omega + LB_{x_1}(1 - \omega) & & x_1 \geq 6\omega \\ \Rightarrow & x_1 \leq 5(1 - \omega) + UB_{x_1}\omega & \Rightarrow [LB_{x_1} = 0] \Rightarrow & x_1 \leq 5(1 - \omega) + UB_{x_1}\omega \\ & x_2 \leq 3 + (UB_{x_2} - 3)(1 - \omega) & & x_2 \leq 3 + (UB_{x_2} - 3)(1 - \omega) \end{array}$$

IF $x_1 + x_3 \geq 10$ THEN $\alpha_1 = 1$ $\Rightarrow$ Ref. f_3
$\Rightarrow$ IF NOT ($\alpha_1 = 1$) THEN NOT ($x_1 + x_3 \geq 10$) $\Rightarrow$
$\Rightarrow$ IF $\alpha_1 = 0$ THEN NOT ($x_1 + x_3 \geq 10$) $\Rightarrow$ Ref. f_{11} $\Rightarrow$ IF $\alpha_1 = 0$ THEN $x_1 + x_3 \leq 9$ $\Rightarrow$
Ref. f_7 $\Rightarrow$ IF $1-\alpha_1 = 1$ THEN $x_1+x_3 \leq 9$ $\Rightarrow$ Ref. f_{14} $\Rightarrow$
$\Rightarrow x_1 + x_3 \leq 9 + (UB_{x_1+x_3} - 9)(1 - (1 - \alpha_1)) \Rightarrow x_1 + x_3 \leq 9 + (UB_{x_1+x_3} - 9)\alpha_1$

Table 6.26 Biconditional connective modelling with binary propositions

ϕ IF AND ONLY IFψ					
ϕ		ψ			
Type	Sign	Type	Sign	Model	Ref.
Binary	$\alpha = 1$	Binary	$\beta = 1$	$\alpha = \beta$	f_{23}
Binary	$\alpha = 1$	Integer/continuous	$X \leq V$	$X \leq V + (UB_X - V)(1-\alpha)$ $X \geq (V+R^I)(1-\alpha) + LB_X\,\alpha$	f_{24}
Binary	$\alpha = 1$	Integer/continuous	$X \geq V$	$X \geq V\alpha + LB_X\,(1-\alpha)$ $X \leq (V - R^D)(1-\alpha) + UB_X\,\alpha$	f_{25}
Binary	$\alpha = 1$	Integer/continuous	$X = V$	$X \leq V + (UB_X - V)(1-\alpha)$ $X \geq V\alpha + LB_X\,(1-\alpha)$ $X \leq (V-R^D)\delta_1 + UB_X\,\alpha + UB_X\delta_2$ $X \geq (V+R^I)\delta_2 + LB_X\,\alpha + LB_X\delta_1$ $\delta_1+\delta_2=1-\alpha$	f_{26}

Table 6.27 Biconditional connective modelling with integer/continuous propositions

	ϕ IF AND ONLY IF ψ						
	ϕ			ψ			
Ref.	Type	Sign	Model	Type	Sign	Model	Ref.
f_{27}	Int/ Cont	$X \leq V_1$	$X \leq V_1 + (UB_X - V_1)(1-\omega)$ $X \geq (V_1 + R^I)(1-\omega) + LB_X\,\omega$	Int/ Cont	$Y \leq V_2$	$Y \leq V_2 + (UB_Y - V_2)(1-\omega)$ $Y \geq (V_2 + R^I)(1-\omega) + LB_Y\,\omega$	f_{30}
f_{28}	Int/ Cont	$X \geq V_1$	$X \geq V_1\omega + LB_X(1-\omega)$ $X \leq (V_1 - R^D)(1-\omega) + UB_X\,\omega$	Int/ Cont	$Y \geq V_2$	$Y \geq V_2\omega + LB_Y(1-\omega)$ $Y \leq (V_2 - R^D)(1-\omega) + UB_Y\omega$	f_{31}
f_{29}	Int/ Cont	$X = V_1$	$X \leq V_1 + (UB_X - V)(1-\omega)$ $X \geq V_1\omega + LB_X(1-\omega)$ $X \leq (V_1 - R^D)\delta_1 + UB_X\omega + UB_X\delta_2$ $X \geq (V_1 + R^I)\delta_2 + LB_X\omega + LB_X\delta_1$ $\delta_1 + \delta_2 = 1 - \omega$	Int/ Cont	$Y = V_2$	$Y \leq V_2 + (UB_Y - V_2)(1-\omega)$ $Y \geq V_2\omega + LB_Y(1-\omega)$ $Y \leq (V_2 - R^D)\delta_3 + UB_Y\omega + UB_Y\delta_4$ $Y \geq (V_2 + R^I)\delta_4 + LB_Y\omega + LB_Y\delta_3$ $\delta_3 + \delta_4 = 1 - \omega$	f_{32}

IF $x1 - x2 = 5$ THEN $x3 \geq 1 \Rightarrow$ Ref. f19; f21 $\Rightarrow$

$$\Rightarrow \begin{aligned} & x_1 - x_2 \leq 5 + (UB_{x_1-x_2} - 5)(1-\omega) \\ & x_1 - x_2 \geq 5\omega + LB_{x_1-x_2}(1-\omega) \\ & x_1 - x_2 \leq 4\delta_1 + UB_{x_1-x_2}\omega + UB_{x_1-x_2}\delta_2 \\ & x_1 - x_2 \geq 6\delta_2 + LB_{x_1-x_2}\omega + LB_{x_1-x_2}\delta_1 \\ & \delta_1 + \delta_2 = 1 - \omega \\ & x_3 \geq 1\omega + LB_{x_3}(1-\omega) \end{aligned}$$

6.8.3.3 Biconditional Operator (IF AND ONLY IF; $\leftrightarrow$)

Following the same format as for the conditional, the modelling tables are the following (Tables 6.26 and 6.27):

6.8.3.4 Disjunction Operator (OR; $\vee$)

If there are two or more atomic propositions joined with the operator OR:

ϕ_i, i=1,2,. . ., n: ϕ_1 OR ϕ_2 OR . . . OR ϕ_n

Modelling follows two steps:

1. We define a logical calculation (ω_i) for each atomic proposition ϕ_i that is integer or continuous (not binary), to know when the proposition is fulfilled. However, it

is not necessary to control the two output values of the calculation, which would have been formulated as $\phi_i \leftrightarrow \omega_i = 1$. To simplify the modelling, we just need to pick up the value of ω_i when the proposition is not fulfilled:

$\forall i/\phi_i \in Z \vee \phi_i \in \mathfrak{R}$: IF NOT (ϕ_i)THEN $\omega_i = 0$

By equivalence f_3, we can also define it as:

$\forall i/\phi_i \in Z \vee \phi_i \in \mathfrak{R}$: IF $\omega_i = 1$ THEN ϕ_i [Reference f_{33}]

2. A quantitative selection specification is incorporated for the defined ω_i and the binary propositions ($i/\phi_i \in \{0,1\}$: $\alpha_i = 1$*):

$$\sum_{i/\phi_i \in \mathfrak{R} \vee \phi_i \in Z} \omega_i + \sum_{i/\phi_i \in \{0,1\}} \alpha_i \geq 1 \text{ [Reference } f_{34}]$$

where it is required that at least one proposition be fulfilled.

*: If the binary proposition were defined with value 0, by equivalence f_7, we transform it into value 1.

Illustration 6.23

We have x_1, x_2 continuous variables ≥ 0.

We also have α_1 and α_2 binary variables.

Proposition: $x_1 \leq 10 \ \vee \ x_2 \geq 4 \vee \ \alpha_1 = 1 \vee \ \alpha_2 = 0 \Rightarrow$ Ref. $f_{14} \Rightarrow$

$\Rightarrow x_1 \leq 10 \ \vee \ x_2 \geq 4 \vee \ \alpha_1 = 1 \vee \ (1 - \alpha_2) = 1$

Model:

1. Logical calculations [Ref. f_{33}]:

IF $\omega_1 = 1$ THEN $x_1 \leq 10 \Rightarrow$ Ref. $f_{14} \Rightarrow x_1 \leq 10 + (\mathrm{UB}_{x_1} - 10)(1 - \omega_1)$

IF $\omega_2 = 1$ THEN $x_2 \geq 4 \Rightarrow$ Ref. $f_{15} \Rightarrow x_2 \leq 4\omega_2 + \mathrm{LB}_{x_2}(1 - \omega_2) \Rightarrow \mathrm{LB}_{x_2} = 0$
$\Rightarrow x_2 \leq 4\omega_2$

2. Quantitative selection specification [Ref. f_{34}]:

$\omega_1 + \omega_2 + \alpha_1 + (1 - \alpha_2) \geq 1$

6.8.3.5 Conjunction Operator (AND; ∧)

When we have a compound proposition where only the disjunction operator appears, it is not necessary to perform any modelling processes. Each atomic proposition corresponds to a restriction in the model.

Instead, the conjunction operator within compound proposals with more operators needs a modelling process, which we will see in Sect. 6.8.4.

6.8.3.6 Exclusive Disjunction Operator (EITHER … OR…;$\bigoplus$)

If there are two or more atomic propositions joined with the operator $\bigoplus$:

ϕ_i , i =1,2,. . ., n: $\phi_1 \bigoplus \phi_2 \bigoplus \ldots \bigoplus \phi_n$

Modelling follows two steps:

1. Similar to step 1) of the connective DISJUNCTION (OR), but in this case the logical calculation must be defined as:
   $\forall i/\phi_i \in Z \vee \phi_i \in \mathfrak{R}: \ \phi_i \leftrightarrow \omega_i = 1$ [Reference f$_{35}$]
2. A quantitative selection specification is incorporated:
   $\sum_{i/\phi_i \in \mathfrak{R} \vee \phi_i \in Z} \omega_i + \sum_{i/\phi_i \in \{0,1\}} \alpha_i = 1$ [Reference f$_{36}$]

This means that one and only one atomic proposition can be fulfilled.

6.8.4 Modelling Compound Propositions with Various Operators

Compound propositions can join several atomic propositions using different operators. Examples can be the following:

Illustration 6.24: Compound Propositions with Several Operators
EITHER (($x_1 \geq 20$ AND $y_1 \leq 10$) OR $\alpha = 1$
(($x_1 \geq 20$ AND $y_1 \leq 10$) OR NOT ($x_3 \geq 20$))
IF (($x_1 \geq 20$ OR $y_1 \leq 10$) THEN NOT ($\alpha = 1$ AND $\beta = 1$)
(($x_1 \geq 20$ AND $y_1 \leq 10$) IF AND ONLY IF ($\alpha = 1$ OR $\beta = 1$)
. . . .

The modelling process of a compound proposition with several operators is done from the lowest level in the structure of the proposition to the highest level. The level is determined by the priority of the operators, according to the structure of parentheses. The lower the level, the higher the execution priority of the operator.

The process will always end with a proposition that has only one type of operator and that will be modelled as defined in Sect. 6.8.3.

We call ψ the proposition that is part of the original compound proposition and in which only one operator type appears. The modelling process of ψ depending on the operator is as follows.

6.8.4.1 Negation Operator (NOT;¬):

The result of the negation operator modelling replaces ψ with $\neg\psi$ in ϕ, but does not incorporate additional constraints into the model.

Illustration 6.25
ϕ: EITHER ($x_1 \geq 8$ AND $x_2 \leq 10$) OR NOT($y \geq 10$)
ψ= NOT($y \geq 10$)
Model of ψ: ⇒Ref. f$_{11}$ ⇒ ($y \leq 9$) [we consider y as integer]
Result: EITHER ($x_1 \geq 8$ AND $x_2 \leq 10$) OR ($y \leq 9$)

6.8.4.2 Disjunction Operator (OR;$\vee$) and Exclusive Disjunction (EITHER… OR…;$\oplus$):

The step 1) of the exclusive disjunction operator described for the cases in which the operator appears individually is modelled (Sect. 6.8.3.6.). This is:

$\psi = (\psi_1 \vee \psi_2 \vee \ldots \vee \psi_i \vee \ldots)$
or
$\psi = (\psi_1 \oplus \psi_2 \oplus \ldots \oplus \psi_i \oplus \ldots)$
$\forall_i / \psi_i \in Z \vee \psi_i \in \mathfrak{R}: \; \psi_i \leftrightarrow \omega_i = 1$ [Reference f$_{35}$]

The constraint resulting from step 2) of the modelling process (Sect. 6.8.3.4 for disjunction operator and Sect. 6.8.3.6 for exclusive disjunction operator) is not incorporated as a constraint to the model, but instead replaces ψ in ϕ. With this we reduce operators of the original proposition ϕ.

Only for some compound propositions, the following expression for the OR operator may also be valid:

$\forall_i / \psi_i \in Z \vee \psi_i \in \mathfrak{R}: \; \psi_i \leftarrow \omega_i = 1$

Illustration 6.26
ϕ: EITHER ($x_1 \geq 8$ OR $x_2 \leq 10$) OR ($y \leq 9$) [x_1, x_2 integers]
$\psi = (x_1 \geq 8$ OR $x_2 \leq 10)$
Model of ψ:

$$\begin{aligned} &\Rightarrow f_{35} \Rightarrow \omega_1 = 1 \text{IF AND ONLY IF} x_1 \geq 8 \Rightarrow \\ &\Rightarrow f_{25} \Rightarrow x_1 \geq 8\omega_1 + \text{LB}x_1(1-\omega_1) \end{aligned} \tag{6.16}$$

$$\Rightarrow x_1 \leq 7(1-\omega_1) + \text{UB}x_1\omega_1 \tag{6.17}$$

$$\begin{aligned} &\Rightarrow f_{35} \Rightarrow \omega_2 = 1 \text{IF AND ONLY IF} x_2 \leq 10 \\ &\Rightarrow f_{24} \Rightarrow x_2 \leq 10 + (\text{UB}x_2 - 10)(1-\omega_2) \end{aligned} \tag{6.18}$$

$$\Rightarrow x_2 \geq 11(1-\omega_2) + \text{LB}x_2\omega_2 \tag{6.19}$$

(6.16), (6.17), (6.18), and (6.19) are constraints that are incorporated into the model.

Result: $\Rightarrow f_{34} \Rightarrow$ EITHER ($\omega_1 + \omega_2 \geq 1$) OR ($y \leq 9$)

The modelling for this could be carried out as described in Sect. 6.8.3.6.

6.8.4.3 Conjunction Operator (AND; $\wedge$)

This operator had not been used individually for the obvious reasons that there was no need for any modelling exercise. However, within a proposal with more operators, it operates in a similar way to the OR and EITHER OR operators:

$\psi = (\psi_1 \wedge \psi_2 \wedge \ldots \wedge \psi_\mathrm{i} \wedge \ldots \wedge \psi_n)$
$\forall i/\psi_i \in Z \vee \psi_i \in \mathfrak{R}: \ \psi_i \leftrightarrow \omega_i = 1$ [Reference f_{35}]

2. An expression ϕ_0 is created with the following format:

$\phi_{0:} \sum\limits_{i/\psi_i \in \mathfrak{R} \vee \psi_i \in Z} \omega_i + \sum\limits_{i/\psi_i \in \{0,1\}} \alpha_i \geq n$ [Reference f_{37}]

Replacing ψ with ϕ_0 in ϕ.

Illustration 6.27
ϕ: EITHER ($x_1 \geq 8$ AND $x_2 \leq 10$ AND $\beta = 0$) **OR** ($y \leq 9$) [x_1, x_2 integers; β binary]
ψ= ($x_1 \geq 8$ AND $x_2 \leq 10$ AND $\beta = 0$)
Model of ψ:

$$\Rightarrow f_{35} \Rightarrow \omega_1 = 1 \leftrightarrow x_1 \geq 8 \Rightarrow f_{25} \Rightarrow x_1 \geq 8\omega_1 + \mathrm{LB}x_1(1-\omega_1) \tag{6.20}$$

$$\Rightarrow x_1 \leq 7(1-\omega_1) + \mathrm{UB}x_1\omega_1 \tag{6.21}$$

$$\Rightarrow \omega_2 = 1 \leftrightarrow x_2 \leq 10 \Rightarrow f_{24} \Rightarrow x_2 \leq 10 + (\mathrm{UB}x_2 - 10)(1-\omega_2) \tag{6.22}$$

$$\rightarrow x_2 \geq 11(1-\omega_2) + \mathrm{LB}x_2\omega_2 \tag{6.23}$$

(6.20), (6.21), (6.22), and (6.23) are constraints that are incorporated into the model.

ϕ_0: $\omega_1 + \omega_2 + (1-\beta) \geq 3$
Result: $\Rightarrow f_{37} \Rightarrow$ EITHER ($\omega_1 + \omega_2 + (1-\beta) \geq 3$) OR ($y \leq 9$)

The modelling for this could again be carried out as described in Sect. 6.8.3.6.

6.8.4.4 Conditional and Biconditional Operators

The constraints resulting from operator modelling replace ψ.
Let $\pi_{1\ldots}\ \pi_r$ be constraints resulting from operator modelling
Proposition (π_1 AND π_2 AND … AND π_r) replaces ψ in ϕ.

Illustration 6.28
ϕ: EITHER $x_1 \geq 8$ OR (IF $x_1 + x_2 \geq 8$ THEN $\beta = 1$)
ψ = (IF $x_1 + x_2 \geq 8$ THEN $\beta = 1$)
Model of ψ:

$$\begin{aligned} &\Rightarrow f_3 \Rightarrow \mathrm{IF}\beta = 0\,\mathrm{THEN}\,x_1 + x_2 < 8 \\ &\Rightarrow f_7; f_4 \Rightarrow \mathrm{IF}\,1-\beta = 1\,\mathrm{THEN}\,x_1 + x_2 \leq 7 \\ &\Rightarrow f_{14} \Rightarrow x_1 + x_2 \leq 7 + (\mathrm{UB}_{x1+x2} - 7)(1-\omega_1) \end{aligned} \tag{6.24}$$

(6.24) replaces ψ in ϕ:
EITHER $x_1 \geq 8$ OR $x_1 + x_2 \leq 7 + (\mathrm{UB}_{x1\,+\,x2} - 7)(1-\omega_1)$

The modelling for this could again be carried out as described in Sect. 6.8.3.6.

A couple of illustrations to express the complete process.

Illustration 6.29

$$\phi : \text{IF } (x_1 \geq 20 \text{ OR } y_1 \leq 10) \text{ THEN NOT } (\alpha = 1 \text{ OR } \beta = 0)$$

$$[x_1, y_1 \text{ integers}, \alpha \text{ and } \beta \text{ binaries}]$$

$$\Rightarrow \text{f}_{7;}\ \text{f}_{34} \Rightarrow \text{IF } (x_1 \geq 20 \text{ OR } y_1 \leq 10) \text{ THEN NOT } (\alpha + (1-\beta) \geq 1)$$

$$\Rightarrow \text{f}_{11} \Rightarrow \text{IF } (x_1 \geq 20 \text{ OR } y_1 \leq 10) \text{ THEN } \alpha + (1-\beta) \leq 0$$

$$\Rightarrow \text{f}_{35} \Rightarrow \omega_1 = 1 \text{ IF AND ONLY IF } x_1 \geq 20$$

$$\Rightarrow \text{f}_{25} \Rightarrow x_1 \geq 20\omega_1 + \text{LB}x_1(1-\omega_1) \tag{6.25}$$

$$\Rightarrow x_1 \leq 19(1-\omega_1) + UBx_1\omega_1 \tag{6.26}$$

$$\Rightarrow \text{f}_{35} \Rightarrow \omega_2 = 1 \text{ IF AND ONLY IF } y_1 \leq 10$$

$$\Rightarrow \text{f}_{24} \Rightarrow y_1 \leq 10 + (\text{UB}y_1 - 10)(1-\omega_2) \tag{6.27}$$

$$\Rightarrow y_1 \geq 11(1-\omega_2) + LBy_1\omega_2 \tag{6.28}$$

$$\Rightarrow \text{f}_{34} \quad \Rightarrow \text{IF } \omega_1 + \omega_2 \geq 1 \text{ THEN } \alpha + (1-\beta) \leq 0 \Rightarrow$$

$$\Rightarrow \text{IF } \omega_1 + \omega_2 \geq 1 \text{ THEN } \alpha - \beta \leq -1$$

$$\Rightarrow \text{f}_{18} \Rightarrow \omega_1 + \omega_2 \geq \omega \tag{6.29}$$

$$\Rightarrow \omega_1 + \omega_2 \leq 2\omega \tag{6.30}$$

$$\Rightarrow \text{f}_{20} \Rightarrow \alpha - \beta \leq -1 + 2(1-\omega) \tag{6.31}$$

Therefore, the starting proposition is modelled with a total of seven constraints (6.25–6.31).

Illustration 6.30

$$\phi : \text{EITHER } (x_1 \geq 1 \text{ IF AND ONLY IF } \alpha = 1) \text{ OR } (x_2 \geq 1 \text{ AND } x_3 \geq 1)$$

$$(x_1, x_2, x_3 \geq 0 \text{ integers}, \alpha \text{ binary})$$

$$\Rightarrow \text{f}_{35} \Rightarrow \omega_1 = 1 \ \text{ IF AND ONLY IF } \ x_2 \geq 1$$

$$\Rightarrow \text{f}_{25} \Rightarrow x_2 \geq \omega_1 \tag{6.32}$$

$$\Rightarrow x_2 \leq \ \text{UB}x_2\ \omega_1 \tag{6.33}$$

$$\Rightarrow \text{f}_{35} \Rightarrow \omega_2 = 1 \text{ IF AND ONLY IF } x_3 \geq 1$$

$$\Rightarrow \ \text{f}_{25} \Rightarrow x_3 \geq \omega_2 \tag{6.34}$$

$$\Rightarrow \ \mathrm{f}_{25} \Rightarrow x_3 \leq \ \mathrm{LB}x_3\omega_2 \tag{6.35}$$

$$\Rightarrow \mathrm{f}_{37} \Rightarrow \text{EITHER } (x_1 \geq 1 \text{ IF AND ONLY IF } \alpha = 1) \text{ OR } (\omega_1 + \omega_2 \geq 2)$$

$$\Rightarrow \mathrm{Ref.f}_{25} \ \Rightarrow x_1 \geq \alpha$$

$$\Rightarrow \mathrm{Ref.f}_{25} \ \Rightarrow x_1 \leq \ \mathrm{UB}_{x1}\ \alpha$$

$$\Rightarrow \phi : \text{EITHER } (x_1 \geq \alpha \ \text{ AND } x_1 \leq \mathrm{UB}_{x1}\ \alpha) \text{ OR } (\omega_1 + \omega_2 \geq 2)$$

$$\Rightarrow \mathrm{f}_{35} \ \Rightarrow \omega_3 = 1 \text{ IF AND ONLY IF } x_1 \geq \alpha$$

$$\Rightarrow \mathrm{f}_{25} \Rightarrow x_1 - \alpha \ \geq -(1 - \omega_3) \tag{6.36}$$

$$\Rightarrow x_1 - \alpha \ \leq -(1 - \omega_3) + \mathrm{UB}_{\mathrm{x}1}\omega_3 \tag{6.37}$$

$$\Rightarrow \mathrm{f}_{35} \ \Rightarrow \omega_4 = 1 \text{ IF AND ONLY IF } x_1 \leq \mathrm{UB}_{x1}\ \alpha$$

$$\Rightarrow \mathrm{f}_{24} \Rightarrow x_1 - \mathrm{UB}_{x1}\alpha \leq \mathrm{UB}_{x1}(1 - \omega_4) \tag{6.38}$$

$$\Rightarrow x_1 - \mathrm{UB}_{x1}\alpha \ \geq (1 - \omega_5) - \mathrm{UB}_{x1}\omega_4 \tag{6.39}$$

$$\Rightarrow \mathrm{f}_{37} \ \Rightarrow \text{EITHER } (\omega_3 + \omega_4 \geq 2) \text{ OR } (\omega_1 + \omega_2 \geq 2)$$

$$\Rightarrow \mathrm{f}_{35} \ \Rightarrow \omega_5 = 1 \text{ IF AND ONLY IF } \omega_3 + \omega_4 \geq 2$$

$$\Rightarrow \mathrm{f}_{25} \Rightarrow \omega_3 + \omega_4 \geq 2\omega_5 \tag{6.40}$$

$$\Rightarrow \ \mathrm{f}_{25} \Rightarrow \omega_3 + \omega_4 \leq (1 - \omega_5) + 2\omega_5 \tag{6.41}$$

$$\Rightarrow \mathrm{f}_{35} \ \Rightarrow \omega_6 = 1 \text{ IF AND ONLY IF } \omega_1 + \omega_2 \geq 2$$

$$\Rightarrow \ \mathrm{f}_{25} \Rightarrow \omega_1 + \omega_2 \geq 2\omega_6 \tag{6.42}$$

$$\Rightarrow \ \mathrm{f}_{25} \Rightarrow \omega_1 + \omega_2 \leq (1 - \omega_6) + 2\omega_6 \tag{6.43}$$

$$\Rightarrow \mathrm{f}_{36} \Rightarrow \omega_5 + \omega_6 = 1 \tag{6.44}$$

(6.32) to (6.44) are incorporated as constraints to the model.

Regardless of this methodology, which is sufficient for the modelling of any proposition, we can also make use of the distributive law between propositions in order to present the concatenation of propositions in a different way:

If we have ϕ, ψ, and σ propositions, the distributive laws between expressions are defined as:

$\phi \vee (\psi \wedge \sigma) \equiv (\phi \vee \psi) \wedge (\phi \vee \sigma)$ [Reference f_{38}]
$\phi \wedge (\psi \vee \sigma) \equiv (\phi \wedge \psi) \vee (\phi \wedge \sigma)$ [Reference f_{39}]

It is also possible to divide conditional propositions when they are at the highest level of the compound proposition:

We have ϕ_1, ϕ_2, . . ., ϕ_n, ϕ_m propositions:

IF ϕ_1 v ϕ_2 v . . . v ϕ_n THEN $\phi_m \Rightarrow$
$\Rightarrow$ IF ϕ_1 THEN ϕ_m
$\Rightarrow$ IF ϕ_2 THEN ϕ_m [Reference f_{40}]

$\Rightarrow \ldots$
$\Rightarrow$ IF ϕ_n THEN ϕ_m
IF ϕ_m THEN $\phi_2 \wedge \phi_2 \wedge \ldots \wedge \phi_n \Rightarrow$
$\Rightarrow$ IF ϕ_m THEN ϕ_1
$\Rightarrow$ IF ϕ_m THEN ϕ_2 [Reference f_{41}]
$\Rightarrow \ldots$
$\Rightarrow$ IF ϕ_m THEN ϕ_n

6.8.5 Data as Propositions

Sometimes and whenever the specification refers to one or more sets of elements, we can propose propositions where the wording includes, among its atomic propositions, conditions on element data values. Let us take a look at some simple examples in the following illustration.

Illustration 6.31
There is a system for allocating distribution hubs to supermarkets. We have n hubs and m supermarkets. The distance between hubs and supermarkets and the demand of each supermarket is known. The system has the following specifications:

1. *If the distance between a supermarket and a hub exceeds 50Km, the supermarket cannot be assigned to the hub.*
2. *If a supermarket has a demand higher than 1000 kgs, it will be assigned two hubs.*
3. *If hub 2 is assigned a supermarket, the supermarket should be less than 1 km away.*
4. *If a supermarket assigned to a hub exceeds the distance of 30 km, the hub will be limited to a maximum of ten supermarkets.*

Table of Elements (Table 6.28)

Decision Activities
Action: Allocate hubs to supermarkets
Decision variables: $\alpha_{ij} = 1$ if I allocate Hub i to Supermarket j; 0 otherwise.
Specifications

Table 6.28 Elements of Illustration 6.31

			Data				
Elements	Set	QN	Name	Param	Type	Belonging	Value
Hubs	$i = 1\ldots n$	I_U	Distance	D_{ij}	C (km)	S	...
Supermarkets	$j = 1\ldots m$	I_U		D_{ij}			
			Demand	M_j	C (Kgs)	W	...

The four specifications of the system are enunciated as conditional logical propositions. Let us take a look at this statement:

1. If the distance between a supermarket and a hub exceeds 50 km, the supermarket cannot be assigned to the hub.

   If we apply it to any supermarket and any hub:

$\forall i,j$: IF $D_{ij} > 50$ THEN $\alpha_{ij}=0$

2. If the supermarket has a demand higher than 1000 kgs, it will be assigned two hubs.

   If we apply it to any supermarket:

$$\forall j : \text{IF } M_j > 1000 \text{ THEN } \sum_{i=1}^{n} \alpha_{ij} = 2$$

3. If hub 2 is allocated to a supermarket, the supermarket must be less than 1 km away.

   If we apply it to any supermarket:

$\forall j$: IF $\alpha_{2j}=1$ THEN $D_{2j}< 1$

4. If a supermarket assigned to a hub exceeds the distance of 30 km, the hub will be limited to a maximum of 10 supermarkets.

   If we apply it to any supermarket and any hub:

$$\forall i,j : \text{IF } \alpha_{ij} = 1 \text{ AND } D_{ij} > 1000 \text{ THEN } \sum_{j=1}^{m} \alpha_{ij} \leq 10$$

This casuistry does not imply an additional modelling exercise and the rules previously seen should not be followed. It is only necessary to extract the atomic propositions associated with data of the global proposition and include it as a condition of the elements on which the specification falls.

Let us call the data propositions with the term P_{At}, whether they are one or several data joined by operators.

We first distinguish the case of propositions with operators individually:

Let ϕ be a proposition of variables (Table 6.29).

Table 6.29 Model of propositions with data

Proposition	Modelling	Reference
$\phi \vee P_{At}$	∀element/NOT(P_{At}): ϕ	f_{42}
$\phi \wedge P_{At}$	∀element/P_{At}: ϕ	f_{43}
$\phi \oplus P_{At}$	∀element/NOT(P_{At}): ϕ ∀element/P_{At}: NOT (ϕ)	f_{44}
IF P_{At} THEN ϕ	∀element/P_{At}: ϕ	f_{45}
P_{At} IF AND ONLY IF ϕ	∀element/P_{At}: NOT (ϕ) ∀element/NOT (P_{At}): ϕ	f_{46}

In the case of several operators, references f_{38}, f_{39}, f_{40}, and f_{41} must be used and operate as follows:

1. If the conditional operator or the biconditional operator does not exist in the upper level:

   1.1 If necessary, the distributive law (Ref. f_{38} and f_{39}) is applied until obtaining a union of propositions of the form:

   (ϕ_1 OR P_{At1}) AND (ϕ_2 OR P_{At2}) AND ...

   1.2. For each compound proposition united with the Operator AND, the following specification is created:

   Ref. $f_{42} \Rightarrow \forall$ element/NOT(P_{At1}): ϕ_1
2. If the conditional operator exists in the upper level:

   2.1. If necessary, apply Ref. f38 and f39 until obtaining a proposal of the form:

IF ϕ_1 OR P_{At1} OR (ϕ_2 AND P_{At2}) OR (ϕ_3 OR P_{At3}) OR ... THEN ψ
Ref. $f_{40} \Rightarrow$ IF ϕ_1 THEN $\psi \Rightarrow$ ...

IF PAt1THEN $\psi \Rightarrow$ Ref. f45 $\Rightarrow\forall$ element/PAt1: ψ
IF (ϕ2 AND PAt2) THEN $\psi \Rightarrow\forall$ element/ PAt2: IF ϕ2 THEN ψ
IF (ϕ3 OR PAt3) THEN $\psi \Rightarrow\forall$ element/ NOT(PAt3): IF ϕ THEN ψ
$\Rightarrow \forall$ element/P_{At3}: ψ
...

Next, we model the propositions of Illustration 6.31.

Illustration 6.32: Modelling the propositions of 6.31

(1)
$\forall i,j$: IF $D_{ij} > 50$ THEN $\alpha_{ij}=0$
$\Rightarrow f_{45} \Rightarrow \forall i,j/D_{ij} > 50$: $\alpha_{ij}=0$
(2)

$$\forall j : \text{IF } M_j > 1000 \quad \text{THEN} \quad \sum_{i=1}^{n} \alpha_{ij} = 2$$

$$\Rightarrow f_{45} \Rightarrow \forall j/M_j > 1000\text{: } \sum_{i=1}^{n} \alpha_{ij} = 2$$

(3)
$\forall j$: IF $\alpha 2j=1$ THEN $D2j< 1$
$\Rightarrow f_3 \Rightarrow \forall j$: IF NOT ($D2j< 1$) THEN NOT($\alpha 2j=1$) $\Rightarrow$ IF $D2j\geq 1$ THEN $\alpha 2j= 0$
$\Rightarrow f_{45} \Rightarrow \forall j/D2j\geq 1$: $\alpha 2j=0$
(4)

$$\forall i,j : \text{IF } \alpha_{ij} = 1 \text{ AND } D_{ij} > 1000 \quad \text{THEN} \quad \sum_{k=1}^{m} \alpha_{ik} \leq 10$$

$$\Rightarrow \forall i,j : \text{IF } \left(\forall i, j/D_{ij} > 1000 : \alpha_{ij} = 1\right) \quad \text{THEN} \quad \sum_{k=1}^{m} \alpha_{ik} \leq 10$$

$$\Rightarrow \forall i,j/D_{ij} > 1000 : \text{IF } \alpha_{ij} = 1 \text{ THEN} \quad \sum_{k=1}^{m} \alpha_{ik} \leq 10$$

$$\Rightarrow \text{Ref. } f_{14} \Rightarrow \forall i,j/D_{ij} > 1000 : \sum_{k=1}^{m} \alpha_{ik} \leq 10 + (m - 10)(1 - \alpha_{ij})$$

6.8.6 Logical Propositions That Express Possibility

When the statement of a system refers to possibilities not subject to conditions, no specification is really being established unless some additional imposition is expressed (e.g., "you can buy at most 10 units"). In most cases, possibilities only serve to establish associations between elements to form activities. The verb we use when talking about possibilities is the verb "can." Let us look at an example:

"Provider A can supply units of product 1": The statement does not generate any constraint. It is established that provider A participates in the supply of product 1 action.

"Provider B can supply more than 50 units of product 2": The statement does not create any constraint. It is established that provider B participates in the supply of product 2 action.

If any limitation is included in the statement, then it may be necessary to establish a specification:

"Provider B can ***only*** supply product 1."

In those cases, it is necessary to model the specification by expressing the statement of impossibility, about what cannot be done: If the supplier can only supply product 1, then it cannot supply product 2.

If the possibility statement is part of a logical proposition because it has a condition or is described with any logical connective, then it is necessary to model that proposition in all cases. For the modelling of this type of logical proposition, it is necessary to rephrase the proposition to express it in negative. It is about converting the proposition of possibility into a proposition of impediment. Let us see some illustrations:

1. "*Provider A can supply units of product 1 if provider B does not supply units of that product*": When there is a simple proposition within the compound proposition that expresses possibility, we rephrase the statement to express it as an impediment:

   "Provider A ***cannot*** *supply units of product 1 if supplier B* ~~does not supply~~ *supplies units of that product"*
2. "Provider B can supply more than 50 units of product 2 if provider A supplies more than 10 units of product 1":

   "Provider B ***cannot*** *supply more than 50 units of product 2 if provider A supplies* ~~more than 10~~ *less than 11 units of product 1"*

The logical process is simple. The verb "can" expresses possibility. The opposite, "cannot," expresses that there is no possibility, but both are not disjunctive. Being

Table 6.30 Elements of Illustration 6.33

Elements	Set	QN	Data				
			Name	Param	Type	Belonging	Value
Tasks	$i = 1 \ldots 15$	I_M	Duration	D_i	C	W	...
Operators	$j = 1 \ldots 4$	I_U					

able to perform an action includes doing it and not doing it. Not being able to do it expresses only the option of not doing it. That is why it is necessary to model the expression that imposes a specification, which is the negative.

Let's take a look at an illustration based on a mathematical environment.

Illustration 6.33
There is a system of assigning workers to tasks. We have 15 tasks and 4 operators. The tasks have a duration time. Two specifications are established in the assignment:

– *An operator can carry out more than two tasks if he partially performs a task*
– *The working time of an operator may be more than 10 hours in the case of doing more than 3 tasks.*

Based on the description it is clear that the tasks are divisible in the system, because they can be partially carried out by several operators, so they have a measurable character. It is a statement that lacks a description of other norms and an objective; in this case we will only focus on the two specifications indicated.

Table of Elements (Table 6.30)

Decision Activities
Action: Assign [tasks to Operators]
Decision variables: x_{ij} = Amount of time of Task i assigned to Operator j.
Specifications

1. *An operator can carry out more than two tasks if he partially performs a task*

The specification refers to each operator j=1...4.

First, it is necessary to express the calculation of the number of tasks performed by an operator as an auxiliary calculation that will use a logical calculation to know if an operator has been assigned to each task:

Binary logical calculation: Operator assigned to task
Applied to: Each Operator j=1...4 and each task i=1...15
Variables: $\alpha_{ij} = 1$ if operator j is assigned to task i; 0 otherwise. i=1...15; j=1...4
Logical proposition: $\forall i, \forall j: \quad \alpha_{ij} = 1$ IF AND ONLY IF $x_{ij} > 0$

The number of tasks performed by each operator can be expressed by an auxiliary calculation:

Auxiliary calculation: Number of tasks performed by each operator

Applied to: Each operator j=1...4
Variables: y_j = number of operator tasks j
Constraints that define the calculation:

$$\forall j: \quad y_j = \sum_{i=1}^{15} \alpha_{ij}$$

Second, we also have to create a logical calculation to know if an operator has partially carried out a task:

Binary logical calculation: Operator partially performs a task
Applied to: Each Operator j=1...4 and each task i=1...15
Variables:
$\beta_{ij} = 1$ if the operator j partially performs task i; 0 otherwise. i=1...15; j=1...4
Logical proposition:
$\forall i, \forall j: \quad \beta_{ij} = 1$ IF AND ONLY IF $x_{ij} > 0$ AND $x_{ij} < D_i$

We could also create an auxiliary calculation for collecting the total number of partially performed tasks:

Auxiliary calculation: Number of tasks partially performed by an operator
Applied to: Each Operator j=1...4
Variables: z_j = number of partial tasks of the operator j
Constraints that define the calculation:

$$\forall j: \quad z_j = \sum_{i=1}^{15} \beta_{ij}$$

We return to the starting specification:

"*An operator can carry out more than two tasks if he partially completes a task*"

And we express it in negative:

"An operator cannot perform more than two tasks if he does not partially do any task"⇒ "If an operator does not partially do any task, he cannot perform more than two tasks"

Mathematically:

$\forall j: \quad$ IF $z_j = 0$ THEN $y_j \leq 2$

2. *The working time of an operator may be more than 10 hours in the case of doing more than three tasks.*

Working time can be collected in an auxiliary calculation:

Auxiliary calculation: Working time of an operator
Applied to: Each Operator j=1...4

Variables: w_j = Working time of operator j
Constraints that define the calculation:

$$\forall j: \quad w_j = \sum_{i=1}^{15} x_{ij}$$

We express the specification as an impediment:

"The working time of an operator cannot exceed 10 hours in the case of performing no more than three tasks" $\Rightarrow$
$\Rightarrow$ "If you perform at most three tasks, the working time of an operator cannot exceed 10 hours"

Mathematically:

$$\forall j: \quad \text{IF } y_j \leq 3 \quad \text{THEN } w_j \leq 10$$

6.9 Objective Criterion

The objective function is the criterion that guides the search for solutions. Defining an objective function in the system results in the complete definition of an optimization problem. As we discussed in the introductory chapter, the illustrations will focus on problems with a single objective function. However, the typologies and modelling of the functions that we will explain below can also serve to develop multiobjective problems or simply to create a function that integrates diverse weighted functions.

Once a criterion has been defined, all the costs, positive or negative (profits), of the activities and calculations that participate in that function will be expressed in the objective function. Therefore, the objective function can be used a priori to identify decision activities or calculations, since any action that entails a cost will correspond to a decision activity or calculation.

The normal or most usual situation in a system is that the unit cost of an activity represented in a variable does not vary whatever the value of the variable. For example, let x be the decision activity associated with buying units from a supplier and let c be the cost of a unit. Typically, the cost c is the cost associated with the purchase of units, regardless of the value of x. The total cost of that activity will be cx. This will happen as long as the variable is binary, since it only takes 2 values (Value 0, no cost; Value 1, cost c). Therefore, the objective function is usually the simplest specification of the system in most cases. It is enough to identify which variables have cost data with respect to the proposed objective.

However, there are problems in which for a generic variable x, integer or continuous, the cost that is applied may depend on the value that variable x takes. The cases that may arise are:

1. The cost of variable x depends on the range of values on which the variable falls

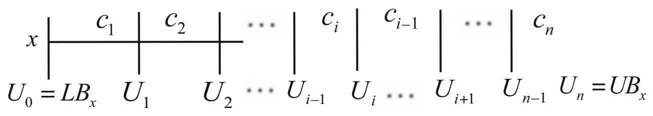


Fig. 6.9 Value intervals

2. The cost of variable x depends on the value that another variable takes
3. The cost depends on the deviation of the variable with respect to reference threshold.

We explain and illustrate below the modelling of each of the cases. Some of the ideas have been based on the modelling presented by Sarker and Newton (2007).

6.9.1 Cost According to Interval of Values

With a variable x, we define a set of n intervals $(U_{i-1}, U_i]$ i=1…n, $U_i \geq 0$ i=0…n, and a cost c_i associated with each interval (Fig. 6.9).

Since we need to know on what interval $(U_{i-1}, U_i]$, i=1 … n the variable x has fallen, it is necessary to define logical calculations, but first, to unify the modelling process, and it is also necessary to define the closed intervals for each cost value. The first interval corresponds to $[U_0, U_1]$. From the second interval, the first value is determined by U_{i-1}+A ($A = 1$ if x is integer, $A = \xi$ if x is continuous). To unify the intervals, we define n intervals $[U_{i-1}+B, U_i]$, where $B = 0$ if i=1, B=A if i> 1.

Binary logical calculation: x belongs to the interval $[U_{i-1}+\mathrm{B}, U_i]$
Applied to: Each interval i=*1…n*
Variables:

$$\alpha_i = \begin{cases} 1 & \text{if } x \in [U_{i-1} + B, U_i] \\ 0 & \text{otherwise} \end{cases}$$

Logical proposition:
$\forall i : \alpha_i = 1$ IF AND ONLY IF $x \geq U_{i-1} + B$ AND $x \leq U_i$
Model:
$\Rightarrow$f$_{35}$ $\Rightarrow$ $\forall i : \omega_{i1} = 1$ IF AND ONLY IF $x \geq U_{i-1} + B$

$$\Rightarrow \mathrm{f}_{25} \Rightarrow x \geq (U_{i-1} + \mathrm{B})\omega_{i1} + \mathrm{LB}_x\,(1 - \omega_{i1}) \tag{6.45}$$

$$\Rightarrow \mathrm{f}_{25} \Rightarrow x \leq (U_{i-1} + B - 1)(1 - \omega_{i1}) + \mathrm{UB}_x\,\omega_{i1} \tag{6.46}$$

$$\Rightarrow \mathrm{f}_{35} \Rightarrow \forall i : \omega_{i2} = 1 \quad \text{IF AND ONLY IF } x \leq U_i$$

$$\Rightarrow \quad \mathrm{f}_{24} \Rightarrow x \leq U_i + (\mathrm{UB}_x - U_i)(1 - \omega_{i2}) \tag{6.47}$$

$$\Rightarrow \quad \mathrm{f}_{24} \Rightarrow x \geq (U_{i+1})(1-\omega_{\mathrm{i}2}) + \mathrm{LB}_x\omega_{i2} \tag{6.48}$$

$$\Rightarrow \mathrm{f}_{37} \Rightarrow \forall i : \alpha_i = 1 \ \text{ IF AND ONLY IF } \ \omega_{i1} + \omega_{i1} \geq 2$$

$$\Rightarrow \mathrm{f}_{25} \Rightarrow \forall i : \omega_{i1} + \omega_{i1} \geq 2\alpha_i \tag{6.49}$$

$$\Rightarrow \mathrm{f}_{25} \Rightarrow \forall i : \omega_{i1} + \omega_{i1} \leq 1 + \alpha_i \tag{6.50}$$

In addition, we need a non-binary logical calculation that collects the value of x in each interval in order to maintain the linearity when expressing the cost in the objective function. Only with the variables α_i,, the definition of the cost of x depending on the interval would be defined as:

$$\sum_{i=1}^{n} c_i \alpha_i x$$

So, we would have a non-linear expression. To avoid this, we must pick up the value of x, according to the interval on which it falls. The calculation would be:

Non-binary logical calculation: Collect the value of x in each interval $[U_{i-1}+B, U_i]$
Applied to: Each interval $i=1...n$
Variables:

$$x_i = \begin{cases} x & \text{if } \alpha_{\mathrm{i}} = 1 \\ 0 & \text{if } \alpha_{\mathrm{i}} = 0 \end{cases}$$

Logical propositions: $\forall i$: IF $\alpha_i = 1$ THEN $x_i = x$
$\forall i$: IF $\alpha_i = 0$ THEN $x_i = 0$
Model:

$$\forall i : \mathrm{IF}\alpha_i = 1 \text{ THEN } \ x_i = x \Rightarrow \mathrm{f}_{16} \Rightarrow \forall i : x \leq x_i + \mathrm{UB}_x(1-\alpha_i) \tag{6.51}$$

$$\Rightarrow \forall i : x \geq x_i \tag{6.52}$$

$$\forall : \mathrm{IF}\alpha_i = 0 \text{ THEN } \ x_i = 0 \Rightarrow \mathrm{f}_7 \Rightarrow \forall i : \text{IF } 1-\alpha_i = 1 \text{ THEN } \ x_i = 0$$

$$\Rightarrow \mathrm{f}_{16} \Rightarrow \forall i : x_i \leq U_i\alpha_i \tag{6.53}$$

$$\Rightarrow \mathrm{f}_{16} \Rightarrow \forall i : x_i \geq 0 \tag{6.54}$$

The expression of the objective function that collects the cost of the variable x would be defined as:

$$\sum_{i=1}^{n} c_i x_i$$

Simplification of Constraints

Proposition $\forall i : \alpha_i = 1$ IF AND ONLY IF $x \geq U_{i-1} + B$ AND $x \leq U_i$ can be simplified by Ref. S_V taking advantage of the characteristics of the variables. The variable x cannot fall onto more than one interval, without imposing it as a condition. That means that it would suffice to impose the relationship between α_i and x:

$$\text{Only variable } \alpha_i \text{ will take value } 1: \sum_{i=1}^{n} \alpha_i = 1 \tag{6.55}$$

$$\forall i: \text{IF } \alpha_i = 1 \quad \text{THEN} \quad x \geq U_{i-1} + B \;\; \text{AND} \;\; x \leq U_i$$

Model:

$$\Rightarrow \mathrm{f}_{41} \Rightarrow \forall i: \text{IF } \alpha_i = 1 \quad \text{THEN } x \geq U_{i-1} + B$$
$$\forall i: \text{IF } \alpha_i = 1 \quad \text{THEN } x \leq U_i$$

$$\Rightarrow \mathrm{f}_{15} \Rightarrow \forall i: x \geq (U_{i-1} + B) \;\; \alpha_i \tag{6.56}$$

$$\Rightarrow \mathrm{f}_{14} \Rightarrow \forall i: x \leq U_i \alpha_i + UB_x (1 - \alpha_i) \tag{6.57}$$

In addition to (6.55), (6.56), and (6.57), we should also include (6.51), (6.52), (6.53), and (6.54) to complete the modelling, although even (6.51) and (6.52) can also be substituted for a single equality function:

$$x = \sum_{i=1}^{n} x_i \tag{6.58}$$

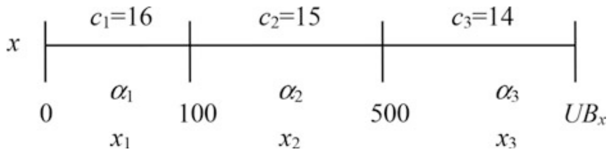
Since with the proposition "IF $\alpha_i = 0$ THEN $x_i = 0$", all x_i take value 0 minus one, that index i for which $\alpha_i = 1$. By imposing (6.58), the x_i that does not take value 0 will automatically take the value of x.

Illustration 6.34

In a purchase system, we have a supplier that offers the following prices for the product:

- *\$16 if we buy a maximum of 100 pcs.*
- *\$15 if we buy more than 100 pcs.*
- *\$14 if we buy more than 500 pcs.*

If x is the variable associated with the activity of purchasing units of the product from that supplier, the cost of x is determined by the interval in which x falls:



The constraints generated by determining the cost would be:

$$x_1 \leq 100\alpha_1$$

By (6.53) $\Rightarrow x_2 \leq 500\alpha_2$

$$x_3 \leq UB_x\alpha_3$$

By (6.55) $\Rightarrow \alpha_1 + \alpha_2 + \alpha_3 = 1$

By (6.56) and (6.57)$\Rightarrow$ $\left|\begin{array}{l} x \geq 1\alpha_1 \\ x \leq 100\alpha_1 + UB_x(1-\alpha_1) \\ x \geq 101\alpha_2 \\ x \leq 500\alpha_2 + UB_x(1-\alpha_2) \\ x \geq 501\alpha_3 \\ x \leq UB_x \end{array}\right|$

By (6.58) $\Rightarrow x = x_1 + x_2 + x_3$

In the objective function, we would introduce the cost terms:

Min $\ldots + 16x_1 + 15x_2 + 14x_3 + \ldots$

6.9.2 Cost According to the Value of Another Variable

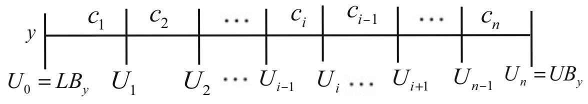
Although the variable that determines the cost is integer or continuous, it will be reduced to depend on the value of one or more binary variables. The modelling is almost identical to the previous case.

Let y be the variable that determines the cost of x, y integer or continuous. Generally, the cost of x will depend on the range of values in which y falls (Fig. 6.10).

To know in which interval $[U_{i-1}$ + B, $U_i]$ $i = 1\ldots$n the value of y has fallen, we can just define a logical calculation for each interval in the following way:

Binary logical calculation: y belongs to the interval $[U_{i-1}$+B, $U_i]$
Applied to: Each interval *i=1...n*
Variables:

Fig. 6.10 Value intervals for variable y



$$\alpha_i = \begin{cases} 1 & \text{if } y \in [U_{i-1} + B, U_i] \\ 0 & \text{otherwise} \end{cases}$$

Logical propositions: As in the reduction of Sect. 6.9.1, it is not necessary to define the two values of α_i, but only impose that:
Only a α_i will take value 1:

$$\sum_{i=1}^{n} \alpha_i = 1 \tag{6.59}$$

$\Rightarrow$ Ref. $S_V \Rightarrow \forall i$: IF $\alpha_i = 1$ THEN $y \geq U_{i-1} + B$ AND $y \leq U_i$
Model:

$$\Rightarrow \text{f}_{41} \Rightarrow \forall i : \text{IF } \alpha_i = 1 \quad \text{THEN } \; y \geq U_{i-1} + B$$

$$\forall i : \text{IF } \alpha_i = 1 \quad \text{THEN } \; y \leq U_i$$

$$\Rightarrow \text{f}_{15} \Rightarrow \forall i : y \geq (U_{i-1} + B) \;\; \alpha_i \tag{6.60}$$

$$\Rightarrow \text{f}_{14} \Rightarrow \forall i : y \leq U_i \alpha_i + UB_y(1 - \alpha_i) \tag{6.61}$$

Variables α_i will determine the cost of x:
With α_1=1, the cost of $x = c_1$
With α_2=1, the cost of $x = c_2$
. . .

If the starting variable y had been binary, the previous process would not be necessary, we would simply continue from this moment, since:

With $y = 1$ ($\alpha_1 = y$), the cost of $x = c_1$
With $y = 0 \Rightarrow \text{f}_7 \Rightarrow 1{-}y = 1$ (α_2 =1$-y$), the cost of $x = c_2$

To model this process, it is sufficient to collect the value of x in one variable for each possible cost value:

Non-binary logical calculation: Collect the value of x according to α_i
Applied to: Each interval *i=1. . .n*
Variables:

$$x_i = \begin{cases} x & \text{if } \alpha_\text{i} = 1 \\ 0 & \text{if } \alpha_\text{i} = 0 \end{cases}$$

Logical propositions:
IF $\alpha_i = 1$ THEN $x_i = x$
IF $\alpha_i = 0$ THEN $x_i = 0$
Model: (identical to that expressed in 6.9.1) Resulting expressions: (6.51), (6.52), (6.53), and (6.54). Similarly, (6.51) and (6.52) can be reduced to (6.58).

The expression of the objective function that collects the cost of variable x would be defined as:

$$\sum_{i=1}^{n} c_i x_i$$

Illustration 6.35

In a purchase system, we have a product whose purchase cost depends on whether we have signed a contract with the supplier. That contract implies a cost C. The price of the product unit is \$$c_1$ with the signing of the contract and \$$c_2$ without the contract.

The system would have two decision activities:

– Buy product from the supplier
– Sign contract with supplier

This would generate the variables:

$x =$ Product units purchased from the supplier.
$\beta = 1$ If I sign a contract with the supplier; 0 otherwise.
The cost of x depends on the value taken by the variable β
$\beta = 1 \Rightarrow$ Cost of $x = c_1$
$\beta = 0 \Rightarrow 1 - \beta = 1 \Rightarrow$ Cost of $x = c_2$
Constraints generated are:

By (6.53): $\begin{array}{l} x_1 \leq \mathrm{UB}_x \beta \\ x_2 \leq \mathrm{UB}_x (1 - \beta) \end{array}$

By (6.58): $x = x_1 + x_2$

In the objective function we would include the cost of signing the contract and the cost of purchasing units:

Min $\quad \ldots + C\beta + c_1 x_1 + c_2 x_2 + \ldots$

6.9.3 Costs Depending on the Deviation of the Variable

In some situations, a reference value can be imposed on the values of a variable, so that we can be interested in the approach of the variable to that reference value or we are interested in distancing from it. The distance from this reference is not imposed as a specification in the problem, but the variable has freedom, penalizing or rewarding the deviation on the reference value in the objective function.

The bonuses or penalties imposed affect the units deviated from the reference value.

The following table summarizes all the possibilities that may arise (Table 6.31):

Table 6.31 Cases of deviation

Deviation	Repercussion	Section
Excess	Penalty	6.9.3.1
	Bonus	6.9.3.2
Default	Penalty	6.9.3.3
	Bonus	6.9.3.4

6.9.3.1 Penalty by Excess

Given

U: Reference value (attribute or variable)
p: unit penalty (attribute)
Affected variable: x
Auxiliary variables:
x_d: Deviated units by default of x over U
x_e: Deviated units by excess of x over U

These variables come from defining a free auxiliary variable y, collecting the difference between U and x: $x + y = U$. In order to use variables ≥ 0, we perform the change of variables: $y = x_d - x_e$, $x_d \geq 0$ and $x_e \geq 0$, and the resulting constraint would be:

$x + x_d - x_e = U$

Although that expression would allow values to be given simultaneously to x_d and x_e, this would never occur even in the best case of the problem since the excess implies a cost, and therefore it is important that x_e be as low as possible.

In the objective function, the cost term $p\ x_e$ is included.

Illustration 6.36
Within a sales system, we have a customer to whom we offer the following prices for the product:

- *\$16 for the first 100 units purchased*
- *\$15 for units that exceed 100 units.*

The affected variable would be the quantity sold of product units to the customer, which we call x.

The reference value is $U = 100$
The company suffers a penalty for excess, $p = \$1$ (\$16−\$15)
Having defined x_d and x_e, this means that:

$x + x_d - x_e = 100$

In the objective function we incorporate the terms of the profit of the sale x and the penalty:

Max $\ldots + 16x - 1x_e$

Being a function of maximizing, the penalty has a minus sign because it is a cost.

6.9.3.2 Bonus by Excess

Given

U: Reference value (attribute or variable)

b: unit bonus (attribute)

Affected variable: x

Auxiliary variables:

x_d: Deviated units by default of x over U

x_e: Deviated units by excess of x over U

Binary variables from logical calculations:

α_d: x_d>0 IF AND ONLY IF $\alpha_d = 1$

α_e: x_e>0 IF AND ONLY IF $\alpha_e = 1$

Constraints:

In this case it would not be worth imposing only $x + x_d - x_e = U$, since it interests the greatest possible value of x_e, so that x_d and x_e would grow to infinity simultaneously. Therefore, the logical calculations are defined to know if x_d and x_e have become positive and then it is imposed that both cannot be made positive simultaneously.

Simplifying the definition of the logical calculations to:

α_d: IF x_d>0 THEN $\alpha_d = 1$
α_e: IF x_e>0 THEN $\alpha_e = 1$

The resulting constraints of this process are:

$$x + x_d - x_e = U$$
$$x_d \leq U * \alpha_d$$
$$x_e \leq (UB_x - U*)\alpha_e$$
$$\alpha_d + \alpha_e \leq 1$$

* If the reference value U is a variable, we have to use another upper bound for x_d and x_e in the modeling process.

In the objective function, the profit term bx_e is incorporated and x would enter in the function with its base cost.

Illustration 6.37

Within a purchasing system, we have a supplier that offers the following prices for the product:

- *\$16 for the first 100 units purchased*
- *\$15 for units that exceed 100 units.*

The affected variable would be the purchased quantity of product units to the supplier, which we call x.

The reference value is U = 100

The bonus for excess is $b = \$1$ (\$16−\$15)

Having defined x_d, x_e, α_d and α_e, this means that:

$$x + x_d - x_e = 100$$
$$x_d \leq 100\alpha_d$$
$$x_e \leq (\mathrm{UB}_x - 100)\alpha_e$$
$$\alpha_d + \alpha_e \leq 1$$

In the objective function we incorporate the terms of the cost of x and the bonus

$$\text{Min} \quad \ldots + 16x - 1x_e$$

that in a minimizing function would have a minus sign because it is a profit.

6.9.3.3 Penalty by Default

Equivalent to Sect. 6.9.3.1.
Given

U: Reference value (attribute or variable)
p: Unit penalty (attribute)
Affected variable: x
Auxiliary variables:
x_d: Deviated units by default of x over U
x_e: Deviated units by excess of x over U

Constraint: $x + x_d - x_e = U$

The objective function incorporates the cost term px_d

Illustration 6.38
In a system of production and sale of product units we have signed with a customer to supply 1000 units per month, so that if we do not meet that supply we have a penalty of $P for each unit not delivered.

There is a default penalty with a reference number of $U = 1000$ units.

The affected variable would be the quantity supplied to the customer, which we call x.

The default penalty, $p = \$P$

Having defined x_d, x_e this means that:

$$x + x_d - x_e = 1000$$

In the objective function we would incorporate the penalty:

$$\text{Min} \quad \ldots + Px_d + \ldots$$

6.9.3.4 Bonus by Default

Equivalent to Sect. 6.9.3.2.

$$x + x_d - x_e = U$$
$$x_d \le U\alpha_d$$
$$x_e \le (UB_x - U)\alpha_e$$
$$\alpha_d + \alpha_e \le 1$$

If the reference value U is a variable, we have to use another upper bound for x_d and x_e in the modeling process.

In the objective function, the benefit term bx_d is incorporated.

Illustration 6.39
After the Kyoto protocol, the state proposes bonuses on the gas emissions of our company in the case of not exceeding the A kg/year, meliorating with \$A for each Kg/year deviated by default.

6.10 Identification of Specifications

The specifications of a system include the standards and operation regulations declared within it.

From the statement or description of the system, the first task for modelling involves the identifying of specifications. Extracting the specifications of a system consists of identifying all the declared norms, both those that are presented explicitly in the statement and those that are assumed from the nature of the elements and activities and do not have an explicit description.

Norms that appear explicitly in the description are easy to identify and one only has to look for verbs of imposition or logical propositions. Those that are found implicitly in a system, without there being the need to declare them, are specifications that are based on data of elements, quantitative selection rules, logical conditions between activities, impositions of flow balance, or bounds of measurable activities:

- Based on data: attributes that express a continuous or integer magnitude of intrinsic capacity, availability, or demand on a collective or measurable element always have a specification associated with capacity consumption, capacity contribution, demand or balance, depending on the system operation. These specifications may not be defined as such, only the attribute. The same happens with relational data between elements, for example, of incompatibility of some action, that probably define constraints regarding decision activities, but they are not made explicit because they are defined with the attribute itself.
- Quantitative selection rules: many systems assume without making explicit the norms that define the quantitative selection specifications for certain logic decision activities between sets of elements. Therefore, it will be necessary to analyze if there are selection rules on each of the elements that participate in the activity (Sect. 6.3).

- Logical conditions between activities: sometimes definitions of decision activities have an implicit relationship between them defined by a logical proposition. This does not mean that any activity represents a calculation, but some of the values of one variable condition the value of another.

When a variable defines a calculation, all its values are obtained from the values of other variables, or they are negligible values in the system. The logical conditions between activities also occur when we identify a logical calculation as a decision activity. By not defining it as a calculation, we ignore the logical proposition that defines it. This logical proposition cannot be ignored from the model, and it would be represented as a specification.

- Bounds of discrete measurable activities: they appear in measurable decision activities in which there is an upper bound of measurement of the activity, individually generally, but also jointly with other measurable activities, without this information being explicitly included in the statement.
- Flow balance constraints: in the system, equilibrium relationships between activities and calculations are established, when there are measurable elements and generally over a set of time periods, and these restrictions are assumed in the operation of the system without these relationships being explicit.

The modeller must analyze the following aspects in the identification of specifications:

- The data of elements that can refer to capacity, availability, or demand, fundamentally
- The selection rules in decision activities
- The decision activities identified in the system in case there is any implicit relationship between them or if any of them were really a logical calculation
- Balance relationships between variables
- The upper bound of discrete measurable activities

The description of a system should always avoid wrong interpretations, so it is desirable that the number of implicit specifications be as low as possible, with all the details indicated in the statement, although some are obvious.

Let's look at some examples of identifying specifications in systems.

Illustration 6.40: Assigning objects to positions (Romero and Romeijn 2005)
There is a set of n objects and m positions, m> n. Each object has a weight. Each position has a maximum weight supported. It is about assigning objects to positions. There is a cost involved in assigning each object to each position. It is about minimizing the cost of the assignment.

Table of Elements (Table 6.32)

Decision Activities

Action: Assign objects to positions
Decision variables:

Table 6.32 Elements of Illustration 6.40

Elements	Set	QN	Data				
			Name	Param	Type	Belonging	Value
Objects	$i = 1 \ldots n$	I_U	Weight	p_i	C	W	...
			Cost	c_{ij}	C	S	...
Positions	$j = 1 \ldots m$	I_U	Max_Weight	M_j	C	W	...
				c_{ij}			

$\alpha_{ij} = 1$ if I assign Object i to Position j; 0 otherwise. $i=1\ldots n$, $j=1\ldots m$

Specifications

The statement does not present any explicit specification. All specifications are given implicitly in the description:

Specifications I1. Based on data: each position has a capacity attribute, the maximum weight supported; therefore, it will be necessary to define a consumption specification in this case on each position.

Constraints:

$$\forall j : \sum_{i=1}^{n} p_i \alpha_{ij} \le M_j$$

Specifications I2. Quantitative selection rules: the activities of the system are logic; therefore, it will be necessary to analyze which are the quantitative norms in the selection (Table 6.33):

The most logical analysis is to assume that an object occupies exactly one position. If it could occupy more than one position, it should have been specified in the statement. And that amount is mandatory and does not act as a higher level, since you must place all objects. Therefore, there is an implicit selection rule for each object. Regarding the positions, there is no rule.

Logically with any position, we can always impose as an upper bound all objects and as a lower bound no objects, but those specifications would not be necessary and therefore are not defined.

Table 6.33 Selection diagram

Elements selecting	Selectable elements	Type of Norm	Quantity	Constraints
Object $i = 1\ldots n$	Positions	Upper bound	–	
		Lower bound	–	
		Equality	1	$\forall i : \sum_{j=1}^{m} \alpha_{ij} = 1$
Position $j = 1\ldots m$	Objects	Upper bound	–	
		Lower bound	–	
		Equality	–	

Specifications I3. Logical conditions between activities: there are no relationships between activities, since there is only one activity.

Specifications I4. Bounds of discrete measurable activities: there are no discrete measurable activities.

Specifications I5. Flow balance constraints: they do not exist.

Illustration 6.41: Ham distribution

A ham distribution company has designed a set of 20 delivery routes for distribution. The company has a portfolio of 350 customers. Each delivery route goes through a series of known customers.

The company has ten vehicles for the distribution. Each vehicle has a given capacity or number of Iberian hams that it can transport.

The demand for ham is known from each customer and must be attended to. Each vehicle that delivers Iberian hams must choose a single route, because more than one would take too long. We know the delivery cost of each route.

Table of Elements (Table 6.34)

In the Route_Customer attribute, customers of each route are annotated. It is therefore shared between routes and customers.

Decision Activities

Action: Deliver Iberian hams with vehicles to customers.

Decision variables:

x_{kj} = Number of Iberian hams delivered with vehicle k to customer j.
k=1...10, j=1...350

Action: Choose routes for vehicles.

Decision variables:

α_{ik} =1 if I choose Route i for Vehicle k; 0 otherwise. k=1...10, i=1...20

Explicit Specifications

This statement does present explicit impositions:

– *"The demand for ham from each customer is known and* ***must be attended to****"*: It is an imposition of demand contribution. The specification refers to satisfying a

Table 6.34 Elements of Illustration 6.41

			Data				
Elements	Set	QN	Name	Param	Type	Belonging	Value
Routes	$i = 1...20$	I_U	Route_Customer	RC_{ij}	B	S	...
			Cost	C_i	C	W	...
Customers	$j = 1...350$	I_U	Demand	D_j	I	S	...
				RC_{ij}			
Vehicles	$k = 1...10$	I_U	Capacity	K_k	I	S	...
Iberian hams	–	C_D		D_i; K_k			

Table 6.35 Selection diagram of decision activity "Choose routes for vehicles"

Elements selecting	Selectable elements	Type of norm	Quantity	Constraints
Vehicle $k = 1\ldots10$	Routes	Upper bound	1	$\forall k : \sum_{i=1}^{20} \alpha_{ik} \leq 1$
		Lower bound	–	
		Equality	–	
Route $i = 1\ldots20$	Vehicles	Upper bound	–	
		Lower bound	–	
		Equality	–	

demand, so if that imposition had not been explicitly specified, it would have been logical to identify it implicitly.

Constraints:

$$\forall j : \sum_{k=1}^{10} x_{kj} = D_j$$

Unitary contributions validate the equality sign

– *"Each vehicle that delivers Iberian hams* ***must choose a single route****"*: Explicitly, a selection rule is being presented for each vehicle in the activity of choosing a route (Table 6.35):

Note that it would be a mistake to assume that the route selection rule for each vehicle would be equal to 1, since we would assign a route to each vehicle. The specification states that a route is chosen for each vehicle that distributes, so we cannot consider that everyone will perform the delivery.

In addition, the phrase "each vehicle that delivers" brings light to an implicit specification existing in the problem, a logical condition between activities (I3).
Implicit Specifications

- **Specifications I1. Based on data:** each vehicle has a capacity attribute, the number of Iberian hams that can be transported; therefore it will be necessary to define a capacity consumption specification for each vehicle. Each customer also has a demand attribute that will give rise to a demand contribution specification, although we have already mentioned that it is given explicitly.

  Constraints: $\forall k : \sum_{j=1}^{350} x_{jk} \leq K_k$

- **Specifications I2. Quantitative selection rules:** the selection rule for each vehicle with respect to routes appears explicitly.
- **Specifications I3. Logical conditions between activities:** as mentioned, between the two decision activities there is a logical condition reflected in the phrase "Each vehicle that delivers ham slices must choose only one route." This phrase refers to the fact that in order for a vehicle to distribute ham, it must have a route assigned to it. If we do not assign a route, it will not distribute Iberian hams.

Logical proposition: If a vehicle delivers ham, then it must have been assigned a route.

(If a vehicle delivers ham to a customer then it must have been assigned a route)

Logical proposition with mathematical formulation:

$$\forall k,j\text{: IF } x_{kj} > 0 \text{ THEN } \sum_{i=1}^{20} \alpha_{ik} = 1$$

In addition to this, it is necessary to contemplate that it is not worth assigning any route, but only one that passes by the customer to whom it delivers:

Logical proposition: If a vehicle delivers Iberian hams to a customer, then the vehicle must have assigned a route that passes by that customer.

Logical proposition with mathematical formulation:

$$\forall k,j\text{: IF } x_{kj} > 0 \text{ THEN } \sum_{i/RC_{ij}=1} \alpha_{ik} = 1$$

This second logical proposition encompasses the previous one, since if it is fulfilled the previous one is fulfilled, so we can omit the first one.

- **Specifications I4. Bounds of discrete measurable activities:** Measurable activities do not have a given upper bound.
- **Specifications I5. Flow balance constraints:** The measurable activities participate in specifications that have already been reflected.

Illustration 6.42: Supermarket Allocation

There is a supermarket company that has several locations (j =1...6) to install a maximum of 3 product distribution centers.

The cost of installing a center in each location is established in CI_j m.u.

The Company has 30 supermarkets to be supplied from locations with distribution centers.

In addition, the following rules must apply in the system:

– *Each location with a distribution center can supply a maximum of ten supermarkets.*
– *For legal requirements, if the company installs a center in location 3 and another in location 5, it cannot install any in location 6.*

Objective Function

Minimize the cost of the problem taking into account that if the number of supermarkets assigned to a location is less than 8, it is penalized with a cost of F m.u.

Table of Elements (Table 6.36)

In the configuration carried out, the locations have been considered as unitary, since they are different, apart from the fact that they are referred to in a particular way. Distribution centers and supermarkets are considered collectives, since they are identical items, determined in the case of supermarkets and indeterminate in the case

Table 6.36 Elements of Illustration 6.42

Elements	Set	QN	Data				
			Name	Param	Type	Belonging	Value
Locations	$j = 1...6$	I_U	Cost	CI_j	C	W	...
			Max_Supermarkets	MS	I	S	5
Distribution centers	–	C_I	Maximum quantity	M	I	W	3
Supermarkets	–	C_D	Quantity	S	I	W	30
				MS			

of distribution centers. And on the other hand, we can avoid referring to those instances in a particular way in the statement, so we only refer to numerals of the collective element.

Decision Activities
Action: Install distribution centers in locations.
Decision variables:
x_j = Number of distribution centers installed in location j. $j = 1...6$

Action: Supply supermarkets from locations.
Decision variables:
y_j =Number of supermarkets supplied from location j; $j = 1...30$
Explicit Specifications

The statement explicitly presents the following specifications:

E1. *"install a maximum of 3 product distribution centers":*
E2. *"30 supermarkets to be supplied from locations with distribution centers"*
E3. *"Each location with a distribution center can supply a maximum of 10 supermarkets"*
E4. *"If the company installs a center in location 3 and another in location 5, it cannot install any in location 6"*

The first three correspond to specifications that are based on data, although they are explicitly described. The fourth specification corresponds to a logical proposition.

Constraints:

E1. $\sum_{j=1}^{6} x_j \leq 3$

E2. $\sum_{j=1}^{6} y_j = 30$

E3. $\forall j : y_j \leq 10$
E4. IF y_3=1 and y_5=1 THEN y_6=0

Implicit Specifications

- **Specifications I1. Based on data:** as mentioned, the specifications that could be based on data have been made explicit.
- **Specifications I2. Quantitative selection rules:** there are no decision activities selected.
- **Specifications I3. Logical conditions between activities:** between the two activities there is a conditional relationship, which is mentioned in the following sentence:

"30 supermarkets to be supplied from locations with distribution centers"

We have considered as explicit the norm of supplying a total of 30 supermarkets, but in this sentence, we also allude to the conditional relationship between the two activities: in order for a location to supply supermarkets, it must have installed a distribution center.

Mathematically:

We are going to express it negatively because we are dealing with a proposition of possibility:

"A location can supply supermarkets if it has a distribution center"
"If a location does not have a center installed, then it cannot supply any supermarket"

$\forall j$: IF $x_j = 0$ THEN $y_j = 0$

- **Specifications I4. Bounds of discrete measurable activities:** in the problem we have two measurable activities. The first of the activities, installing centers in locations, carries an implicit type I4 specification, since the most sensible thing to do is to assume that a distribution center will be installed in a physical location at most. It would be strange to think that there may be more or other specifications.
  $\forall j$: $x_j \leq 1$

Regarding the second activity, supplying supermarkets, its measurement is not limited by a specific value.

- **Specifications I5. Flow balance constraints:** they do not exist.

References

Bang-Jensen, J., & Gutin, G. (2000). *Digraphs: Theory, algorithms and applications*. Berlin: Springer.

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik, 1*, 269–271.

Graham, R. L., & Hell, P. (1985). On the history of the minimum spanning tree problem. *Annals of the History of Computing, 7*(1), 43–57.

Hwang, F. K., Richards, D. S., & Winter, P. (1992). *The Steiner tree problem* (Annals of discrete mathematics. 53). North-Holland: Elsevier.

Larrañeta, J., Onieva, L., & Lozano, S. (1995). *Métodos Modernos de Gestión de la Producción*. Madrid: Alianza Editorial.

Mitra, G., Lucas, C., & Moody, S. (1994). Tools for reformulating logical forms into zero-one mixed integer programs. *European Journal of Operational Research, 72*, 262–276.

Öztürk, Ö., Gazibey, Y., & Gerdan, O. (2015). The triple test algorithm to get feasible solution for transportation problems. *International Journal of Numerical Methods and Applications, 13*, 37–50.

Romero, D., & Romeijn, H. E. (2005). The generalized assignment Problem and Extensions. In D.-Z. Du & P. M. Pardalos (Eds.), *Handbook of combinatorial optimization* (Vol. 5, pp. 259–311). Boston: Springer Kluwer Academic Publishers.

Sarker, R. A., & Newton, C. S. (2007). *Optimization modelling. A practical approach*. New York: CRC Press.

Williams, H. P. (1995). Logic applied to integer programming and integer programming applied to logic. *European Journal of Operational Research, 81*, 605–616.

Williams, H. P. (2009). *Logic and integer programming* (pp. 71–103). New York: Springer.

Williams, H. P. (2013). *Model building in mathematical programming* (5th ed.). Wiley.. ISBN: 978-1-118-44333-0

Chapter 7 The Quantitative Nature of the Elements

7.1 Introduction

We have already seen throughout the book that the nature of the elements is used as a tool to help define decision activities. It is probably an unnecessary tool for an experienced modeller, but it may be useful for people who start modelling in mathematical programming. The factors that determine the quantitative nature of the element are its data and its properties, as well as the treatment it receives in the description of the problem.

Figure 7.1 defines the decision scheme of the quantitative nature of an element.

The most important nuances that can be considered regarding the definition of the quantitative nature of the elements can be summed up in six cases. Let us see each of them:

- Case 1. Individual element not measurable defined as measurable

An individual element that has a continuous quantity attribute may appear a priori to be measurable due to that continuous attribute and may even determine decision activities where the quantity of the element is measured. However, we can be faced with a system where that attribute of quantity is not measured because it is always used completely or globally as a contribution and not partially. Remember that this methodology does not consider an attribute measurable whose use is logical. Therefore, suitable in these cases is to define the element as unitary (it would be like always working with the continent instead of with the content inside it). However, if we defined the element as measurable individual, we could also model the problem, though not very efficiently.

- Case 2. Measurable Element with both measurable and logical decisions

This occurs when regarding an element, measurable individual or collective, decisions are made about its content and its continent, being the direct object to

J. M. García Sánchez, *Modelling in Mathematical Programming*, International Series in Operations Research & Management Science 298,
https://doi.org/10.1007/978-3-030-57250-1_7

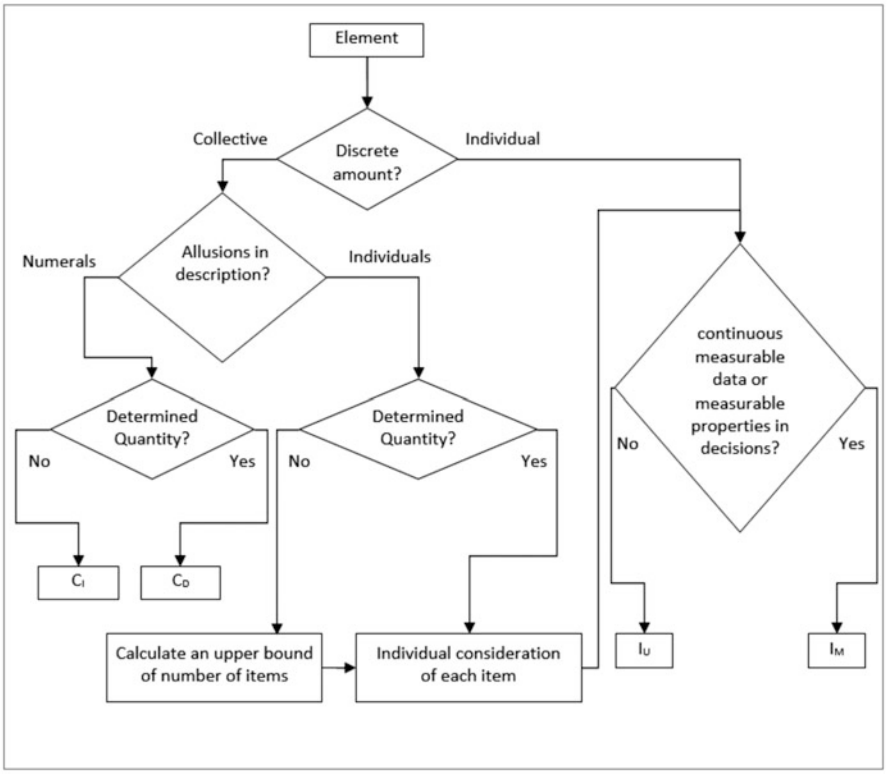

Element
Discrete amount?
Collective
Individual
Allusions in description?
Numerals
Individuals
Determined Quantity?
No
Yes
Determined Quantity?
No
Yes
continuous measurable data or measurable properties in decisions?
No
Yes
C_I
C_D
Calculate an upper bound of number of items
Individual consideration of each item
I_U
I_M

Fig. 7.1 Decision diagram of the nature of an element

the action. Decisions about its content are measurable decisions, while decisions about the global element are logical decisions. We will see how that measurable character can ensure that all decisions can be defined as measurable. We will also propose a configuration on these elements that separates the continent from the content, thus obtaining that the two types of decisions on the same element cannot be considered.

- Case 3. Individual elements with capacity to be grouped in collective elements

In the configuration of the elements of a system, we can find situations where we can have individual elements with the capacity to define themselves as collective, through the modification of the description or through the identification of subgroups of identical elements. In general, the use of collective elements instead of individual elements produces a model with a smaller size and greater efficiency. Therefore, it is good to look for that possibility if the problem does not arise in the first instance. The cases that we can consider are the following:

– Case 3.1. Redefinition of the system

This case occurrs when we have a set of individual elements with the same functionality in the system and identical with respect to their data and values, and we can avoid referring to them individually by redefining the description of the system. In this way we manage to group the elements as a single collective element, which will collect all the data of these elements plus an additional attribute of existence.

With the redefinition of the specifications, we will always convert the particular allusions to the items in references to numerals of a collective element. This process can be simple, with a small change in the description of the specifications or may involve a change of greater scope, which is achieved when you have a rounded knowledge of the problem and extensive experience in the field of operational research. Therefore, we distinguish the two cases:

Case 3.1.1. Redefinition of the system with simple changes
Case 3.1.2. Redefinition of the system with complex changes

– Case 3.2. Grouping into subsets

We have a set of individual elements with the same functionality in the system (same data definition), but their values are not identical in the totality of the elements, although you can define identical subsets. Again, whenever there is no individual reference to each item or those allusions can be eliminated, as in case 3.1, we can create a collective element for each subset.

– Case 3.3. Small changes in the data values

We have a set of individual elements with the same data definition, but the values of the data differ slightly so that the grouping of collective elements cannot be carried out. The solution is to assume small errors in the data through changes in the data values to allow the grouping into collective elements.

• Case 4. Items of indeterminate collective elements that need to be defined individually

This case differs from the previous ones in the sense that it does not express an improvement process but a process necessary for modelling the system. It occurs when the description of a system implies an element as an indeterminate collective, but the statement treats the possible items of the element individually. In that case it is necessary to transform each item of that collective element into an individual element. Since the quantity of the collective element is indeterminate, it will be necessary to calculate an upper bound of the number of items that it could have, and that dimension will be the number of individual elements created.

To illustrate all these cases, we will present each casuistry in sections throughout the chapter.

7.2 Individual Element Not Measurable Defined as Measurable

We are going to illustrate the double treatment of an individual element that has a continuous attribute that is not used partially: in a first version, the correct version, as a unitary element and the second as a measurable element by mistakenly considering this continuous attribute as measurable. It will be shown the suitability of the first version with respect to the second one.

Illustration 7.1
A factory owns a set of 120 sacks of rice grain. Each sack has its own quantity of kilos of rice. It is desired to assign the rice to two production sections where it is processed.

Each sack can only be assigned to a single section, and the full rice content is discharged.

Each section has a capacity. The time to process rice in each section is T1 sec/kilo and T2 sec/kilo. It is desired to maximize the production of rice within the 8 hours of the day.

The following elements are extracted from the statement:

– The factory (the system).
– The two sections.
– Regarding time, the system focuses on the optimization of a day. There are 8 h within the day. The day would be the individual element in which the activity is situated and that we will make explicit by assigning the attribute of 8 h. In the same way, this attribute could have been attributed to the system without declaring the day element. Moreover, the time of 8 h is applicable as the working time of each section, so we are going to consider it an attribute of the sections.
– Regarding rice sacks, they are distinguished from each other so that each sack must be considered as an individual element. The amount of rice in each sack is already discussed as a measurable attribute and yet the phrase "Each sack can only be assigned to a single section and the full rice content is discharged" indicates that the item is used in full in the system and therefore is not measurable.

This is going to be the element that distinguishes the two versions. In the first version, we will define the sacks as unitary elements, while in the second we will define them as measurable elements.

Version 1
Table of Elements (Table 7.1)

Decision Activities

Action: Assign rice sacks to sections (equivalent to processing sacks in sections).
Decision variables:
$\alpha_{ij} = 1$ if sack j is assigned to Section i; 0 otherwise. $i = 1{,}2; j = 1\ldots120$.

Table 7.1 Elements of Illustration 7.1 – Version 1

Elements	Set	QN	Data				
			Name	Param	Type	Belong	Value
Sections	$i = 1...2$	I_U	Capacity	K_i	C (kilos)	W	–
			Process time	T_i	C (sec/kilo)	W	–
			Working time	TT_i	C (hours)	W	8
Rice sacks	$j = 1...120$	I_U	Rice quantity	A_j	C (kilos)	W	–

Table 7.2 Selection rules of decision activity "Assign Sacks to sections"

Activity	Elements selecting	Selectable elements	Selection	Constraints
Assign	Sack $j = 1...120$	Sections	≤ 1	E1
	Section $i = 1,2$	Sacks	–	

Implicit Specifications

I1. Based on data:

– The capacity data of each section:

$$\forall i : \sum_{j=1}^{120} A_j \alpha_{ij} \leq K_i$$

The capacity consumption is carried out by the activity α_{ij}. The unit consumption corresponds to the quantity of rice that the sack has, since it is completely processed in the section.

– The working time attribute in the day affects each section and works as capacity attribute:

$$\forall i : \sum_{j=1}^{120} T_i A_j \alpha_{ij} \leq \mathrm{TT}_i$$

The unit consumption corresponds to the quantity of kilos of rice that the sack has, multiplied by the time it takes to use a kilogram in the section.

I2. Quantitative selection rules: Let us analyze the quantitative norms of activity α_{ij} (Table 7.2).

Each sack goes at the most to a section (E1, as appears explicitly). We are not required to use them all. No quantitative rule is established for each section.

I3. Logical conditions between activities: they do not exist.
I4. Bounds of discrete measurable activities: they do not exist.
I5. Flow balance constraints: they do not exist.

Explicit Specifications

E1. "Each bag can only be assigned to a single section."

Table 7.3 Elements of Illustration 7.1 – Version 2

			Data				
Elements	Set	QN	Name	Param	Type	Belonging	Value
Sections	$i = 1\ldots2$	$\mathrm{I_U}$	Capacity	K_i	C (kilos)	W	–
			Process time	T_i	C (sec/kilo)	W	–
			Working time	TT_i	C (hours)	W	8
Rice sacks	$j = 1\ldots120$	$\mathrm{I_M}$	Rice quantity	A_j	C (kilos)	W	–

The selection rule already analyzed:

$$\forall j : \sum_{i=1}^{2} \alpha_{ij} \leq 1$$

Objective Criterion

Maximize processed kilo of rice:

$$\mathrm{Max} \quad \sum_{i=1}^{2} \sum_{j=1}^{120} A_j \alpha_{ij}$$

Version 2

We started to consider the sacks of rice as measurable (Table 7.3).

Decision Activities

Action: Assign Rice sacks to Sections (equivalent to processing rice sacks in sections).

Decision Variables:

x_{ij} = Amount of rice of sack j assigned to Section i. $i = 1{,}2; j = 1\ldots120$.

Implicit Specifications

I1. Based on data:

– The capacity data of each section:

$$\forall i : \sum_{j=1}^{120} x_{ij} \leq K_i$$

– The working time attribute in the day affects each section and works as capacity attribute:

$$\forall i : \sum_{j=1}^{120} T_i x_{ij} \leq \mathrm{TT}_i$$

Unitary consumption corresponds to the time a kilo uses in the section.

– The availability of rice in each sack implicitly implies a capacity specification:

$$\forall j : \sum_{i=1}^{2} x_{ij} \leq A_j$$

I2. Quantitative selection rules: they do not exist.

I3. Logical conditions between activities: they do not exist.
I4. Bounds of discrete measurable activities: they do not exist.
I5. Flow balance constraints: they do not exist.

Explicit Specifications

E1: "Each sack of rice can only be assigned to a single section and the entire rice content is discharged."

It actually expresses two specifications:

"Each sack of rice can only be assigned to a single section."
Logical Proposition: $\forall j$: EITHER $x_{1j} > 0$ OR $x_{2j} > 0$.
"The full content of rice is discharged."
Logical Proposition: $\forall i, \forall j$: IF $x_{ij} > 0$ THEN $x_{ij} = A_i$.

This second specification means that the first one is not necessary. Assigning a sack of rice to one section assigns all the rice, and therefore no more rice can be assigned to any other section.

Objective Criterion

Maximize processed kilos of rice:

$$\text{Max} \quad \sum_{i=1}^{2} \sum_{j=1}^{120} x_{ij}$$

7.3 Measurable Element with Both Measurable and Logical Decisions

It is difficult to find systems where a measurable element in decision activities also has logical decisions. In many cases this situation is due to the fact that the optimization problem encompasses several problems that could be independent. The normal situation is that decisions are measurable and logical calculations are obtained from those decisions. Anyway, when this situation occurs, we will say that the logical decisions are made on the continent of the element and the measurable decisions on the content. Therefore, we will propose the separation of the continent and the content in the table of elements.

We present two illustrations. The first refers to a measurable individual element and the second to a collective one. In the first one, we are going to slightly modify the statement in Illustration 7.1 to assign a mandatory measurable attribute to rice sacks.

Illustration 7.2: Measurable Individual Element
A factory owns a set of 120 sacks of rice grain. Each sack has its own quantity of kilos of rice. It is desired to assign the rice to two production sections where it is

Table 7.4 Elements of Illustration 7.2

Elements	Set	QN	Data				
			Name	Param	Type	Belong	V.
Sections	$i = 1\ldots2$	I_U	Capacity	K_i	C (kilo)	W	–
			Process time	T_i	C (sec/kilo)	S	–
			Working time	TT_i	C (hours)	W	8
Sacks	$j = 1\ldots120$	I_U	Rice quantity	A_j	C (kilos)	S	–
Rice	–	I_M		A_j; K_i; T_i			

processed. Each sack can only be assigned to a single section. Subsequently, the amount of rice to be processed from each sack is free.

Each section has a capacity. The time to process rice in each section is T1 sec/kilo and T2 sec/kilo. It is desired to maximize the production of rice in the 8 hours of the day.

The quantity of rice in each bag becomes a measurable attribute in the decision activities, since the amount of rice that is processed must be decided. In addition to deciding the quantity of rice that is processed, in the system there is a logical decision to assign sacks to the sections. That is, on the one hand sacks are assigned to sections, and on the other the amount of rice processed must be decided.

The table of elements would be the one described in Table 7.3. However, in the cases in which the possibility of using the element can be seen both in a measurable and logical way, it may be more convenient to configure the elements where we separate the content element from the continent element, making the continent unitary and the content measurable, as reflected in Table 7.4.

Sacks of rice become unitary. The measurable amount of rice is shared with the rice element, which acts as primary and supports the measurable nature.

As mentioned, the decision activities would be defined as:

Decision Activities

Action: Assign Sacks to Sections.
Decision variables: $\alpha_{ij} = 1$ if I assign sack j to section i; 0 otherwise.
Action: Process Rice from the Sacks.
Decision variables: $x_j =$ Amount of rice processed from sack j.

It is not necessary to include the participation of the sections in the processing activity, since we already have the activity of assigning sacks to sections, but their inclusion may favor the subsequent modelling of the specifications.

Action: Process Rice from Sacks in Sections.
Decision variables: $x_{ij} =$ Amount of rice from sack j processed in section i.

It could be that we had assigned sack j to section i and we would not have processed any rice from the sack. Of course, this fact is strange in itself. The logical thing would be to think that if you assign the sack to a section, it is because you are going to process rice from it in that section; otherwise, it would be normal not to assign it. If we situate ourselves in this last scenario, we would always include the sections as participants in the processing activity and the allocation of sacks to sections would cease to be a decision activity and become a logical calculation. With this, there would be no logical activity on the measurable element:

Action: Process rice from the sacks in sections.
Decision variables: x_{ij} = Amount of rice from sack j processed in section i.

Binary logical calculation: Assign sack j to section i.
Applied to: Each sack $j = 1\ldots120$; each section $i = 1,2$.
Variables: α_{ij}=1 if I assign sack j to section i; 0 otherwise.
Logical propositions:
$\forall j, i : \alpha_{ij} = 1$ IF AND ONLY IF $x_{ij} > 0$

We will now illustrate the same case but with respect to a collective element, which will have decisions about its content and about the continent.

Illustration 7.3: Collective element
A parts machining factory produces 15 different part models ($i = 1\ldots15$). For manufacturing, it has 10 machines ($j = 1\ldots10$). Each machine uses a time of T_{ij} minutes to produce a part of model i.

The weekly planning of the production of parts is P_i units of each model i, although in the system only the daily selection of 5 models is allowed. In the week there are 6 days of production and a capacity of 12 hours of production each day.

It is about assigning the production of parts to the machines, keeping in mind that in any machine you cannot produce more than two models of different parts on any given day.

The objective is to use the least possible time in the production of the parts.

The main elements in the problem are each part model, which are formed by a number of determined units to produce. Each part model would be a collective element because it consists of a set of identical items and without individual reference to each item in the system. That would be the content. The decisions in the system fall on the content, that is, on the number of parts that will be made each day in each machine, although in the description it can also be understood that there is a decision on which models to make each day, since there is a restriction on a maximum of five models per day. According to this, we should also reference the continent, which would be the model of each part. It would be a unitary individual element.

In spite of this, another valid perspective in the problem is to consider only decisions about the content. The decisions about the continent are converted into a logical calculation regarding the content (if I have produced parts of a model 1 day, I have selected that model that day). If I do not opt for the logical calculation option, I

Table 7.5 Elements of Illustration 7.3 – Implementation 7.1

Elements	Set	QN	Data				
			Name	Param	Type	Belong	Value
Models	$i = 1\ldots15$	I_U	Production	P_i	I	S	–
			Time	T_{ij}	C (min)	S	–
Parts	–	C_D		P_i			
Machines	$j = 1\ldots10$	I_U		T_{ij}			
Days	$t = 1\ldots6$	I_U	Available time	H_t	C (min)	W	12*60

Table 7.6 Elements of Illustration 7.3 – Implementation 7.2

Elements	Set	QN	Data				
			Name	Param	Type	Belonging	Value
Part models	$i = 1\ldots15$	C_D	Production	P_i	I	W	–
			Time	T_{ij}	C (min)	S	–
Machines	$j = 1\ldots10$	I_U		T_{ij}			
Days	$t = 1\ldots6$	I_U	Available time	H_t	C (min)	W	12*60

would have to express the implicit relationship between the decisions as a specification to select a model and produce parts. Let us see the two implementations:

Implementation 7.1: Decisions on Content and the Continent (Table 7.5)
Decision Activities

Action: Select Models in Days.
Decision variables: $\alpha_{it} = 1$ if I select model i in day t; 0 otherwise.
Action: Produce Parts of Models in Machines in Days.
Decision variables:
x_{ijt} = Number of parts of model i produced in machine j in day t.

Implicit Specification

We indicate the relationship between both decision variables, regardless of whether there are other implicit specifications.

I3. Logical conditions between activities: Between α_{it} and x_{ijt}, the following relationship cannot be ignored:

$\forall i, j, t$: IF $\alpha_{it} = 0$ THEN $x_{ijt} = 0$.

Implementation 7.2: Decisions Made Only About the Content

In this second version, we do not make the continent explicit in the table of elements because the decisions fall on the content exclusively. For this reason, each part model is considered a collective element (Table 7.6).

Decision Activities

Action: Produce parts of models in machines in days.
Decision variables:
x_{ijt} = Number of parts of model i produced in machine j in day t.

Logical Calculation

Binary logical calculation: Select models in days.
Applied to: Each model $i = 1...15$; Each day $t = 1...6$.
Variables: $\alpha_{it}=1$ if I select model i in day t; 0 otherwise.
Logical propositions:

$$\forall i, t : \alpha_{it} = 1 \quad \text{IF AND ONLY IF} \quad \sum_{j=1}^{10} x_{ijt} > 0$$

7.4 Individual Elements with Capacity to be Grouped in Collective Elements: Redefining the System with Simple Changes

To illustrate the change of individual elements to items of a collective element, let us take a look at a system of allocating distribution centers to supermarkets. The same system will be defined in three different ways by making small changes in the description. The three systems are equivalent.

Illustration 7.4: System of Allocating Distribution Centers
There is a supermarket company that has 10 locations (i = 1...10) to install a maximum of 3 product distribution centers. The installation cost in each location is established in \$CI$_i$. In a location one center is installed at most. And each installed distribution center must be in a single location.

The Company has 25 supermarkets (j = 1...25) to be supplied from the distribution centers. Each supermarket must be allocated to a distribution center for its supply. Each location has a K_i capacity that is expressed in the number of supermarkets that it can supply.

By legal requirements, if the company installs a center in location 3 and another in location 5, it cannot install any in location 6.

Objective: Minimize the costs of the problem taking into account that if the number of supermarkets allocated to a center exceeds 10, the center is penalized with a cost of \$F.

Table 7.7 Elements of Illustration 7.4 – Version 1

Elements	Set	QN	Data				
			Name	Param	Type	Belonging	Value
Locations	$i = 1\ldots10$	I_U	Cost	CI_i	I	W	–
			Capacity	K_i	I	W	–
Centers	$j = 1,2,3$	I_U	Penalty	P_j	C	W	F
Supermarkets	$k = 1\ldots25$	I_U					

Version 1

Table of Elements

According to the statement:

– The locations are individual and unitary.
– The supermarkets are identical, but they are referred to individually ("each supermarket must be assigned to a distribution center for its supply"). We consider them unitary.
– The centers are identical as well, but there is a particular reference to them in the statement to express a specification ("each distribution center must be in a single location"). Also, in the objective function, the centers are particularized. We consider the penalty as an attribute of the center, although being a constant of all centers we could have considered it as an attribute of the system (Table 7.7).

Decision Activities

There are two decision activities. On the one hand, install centers in locations and, on the other hand, assign supermarkets to centers.

Action: Install centers in locations.
Decision variables: $\alpha_{ij} = 1$ if center j is installed in location i; 0 otherwise.
Action: Allocate centers to supermarkets.
Decision variables: $\beta_{kj} = 1$ if supermarket k is allocated to center j; 0 otherwise.

Specifications

1. *Implicit Specifications*

I1. Based on data:

Although it could be considered that the specification of capacity is explained in the statement, we will consider it in the implicit section. There is a capacity attribute for the locations, expressing the maximum number of supermarkets that can be supplied from that location (k_i). The activities of the problem do not consume that resource. We need to calculate the supermarkets that are supplied from each location, because those variables will be the ones that consume that capacity. It is a logical calculation:

Binary logical calculation: Supermarket supplied from location.
Applied to: Each supermarket $k = 1...25$ and each location $i = 1...10$.
Variables:$\omega_{ki} = 1$ if supermarket k is supplied from location i.
Logical proposition: The supermarket k is supplied from the location i if the supermarket k is allocated to center j that is installed in location i:
$\forall k, i : \omega_{ki} = 1 \leftrightarrow \ (\beta_{k1} = 1 \quad \text{AND} \quad \alpha_{i1} = 1) \text{ OR } (\beta_{k2} = 1 \quad \text{AND} \quad \alpha_{i2} = 1)$
$\text{OR } \ (\beta_{k3} = 1 \quad \text{AND} \quad \alpha_{i3} = 1)$
Proposition modelling:
$\Rightarrow$ Ref. $f_1 \Rightarrow$

$$\begin{array}{l}\Rightarrow \forall k, i : \text{IF } (\beta_{k1} = 1 \quad \text{AND} \quad \alpha_{i1} = 1) \text{ OR } (\beta_{k2} = 1 \quad \text{AND} \quad \alpha_{i2} = 1) \\ \text{OR } \ (\beta_{k3} = 1 \quad \text{AND} \quad \alpha_{i3} = 1) \text{ THEN } \omega_{ki} = 1\end{array} \tag{7.1}$$

$$\begin{array}{l}\Rightarrow \forall k, i : \text{IF } \omega_{ki} = 1 \text{ THEN } \ (\beta_{k1} = 1 \text{ AND } \quad \alpha_{i1} = 1) \text{ OR } (\beta_{k2} = 1 \text{ AND } \alpha_{i2} = 1) \\ \text{OR } \ (\beta_{k3} = 1 \quad \text{AND} \quad \alpha_{i3} = 1)\end{array} \tag{7.2}$$

Model of (7.1):

$$\begin{array}{ll} & \Rightarrow \forall k, i : \text{IF } (\beta_{k1} = 1 \quad \text{AND} \quad \alpha_{i1} = 1) \text{ THEN } \omega_{ki} = 1 \Rightarrow \text{Ref.f}_3 \Rightarrow \\ \Rightarrow \text{Ref.f}_{40} & \Rightarrow \forall k, i : \text{IF } (\beta_{k2} = 1 \quad \text{AND} \quad \alpha_{i2} = 1) \text{ THEN } \omega_{ki} = 1 \Rightarrow \text{Ref.f}_3 \Rightarrow \\ & \Rightarrow \forall k, i : \text{IF } (\beta_{k3} = 1 \quad \text{AND} \quad \alpha_{i3} = 1) \text{ THEN } \omega_{ki} = 1 \Rightarrow \text{Ref.f}_3 \Rightarrow \end{array}$$

$$\begin{array}{l}\Rightarrow \text{IF NOT}(\omega_{ki} = 1) \text{ THEN NOT}(\beta_{k1} = 1 \quad \text{AND} \quad \alpha_{i1} = 1) \Rightarrow \text{Ref.f}_7 \Rightarrow \\ \Rightarrow \text{IF NOT}(\omega_{ki} = 1) \text{ THEN NOT}(\beta_{k2} = 1 \quad \text{AND} \quad \alpha_{i2} = 1) \Rightarrow \text{Ref.f}_7 \Rightarrow \\ \Rightarrow \text{IF NOT}(\omega_{ki} = 1) \text{ THEN NOT}(\beta_{k3} = 1 \quad \text{AND} \quad \alpha_{i3} = 1) \Rightarrow \text{Ref.f}_7 \Rightarrow \\ \Rightarrow \text{IF } 1 - \omega_{ki} = 1 \text{ THEN NOT}(\beta_{k1} = 1 \quad \text{AND} \quad \alpha_{i1} = 1) \Rightarrow \text{Ref.f}_{37} \Rightarrow \\ \Rightarrow \text{IF } 1 - \omega_{ki} = 1 \text{ THEN NOT}(\beta_{k2} = 1 \quad \text{AND} \quad \alpha_{i2} = 1) \Rightarrow \text{Ref.f}_{37} \Rightarrow \\ \Rightarrow \text{IF } 1 - \omega_{ki} = 1 \text{ THEN NOT}(\beta_{k3} = 1 \quad \text{AND} \quad \alpha_{i3} = 1) \Rightarrow \text{Ref.f}_{37} \Rightarrow \\ \Rightarrow \text{IF } 1 - \omega_{ki} = 1 \text{ THEN NOT}(\beta_{k1} + \quad \alpha_{i1} \geq 2) \Rightarrow \text{Ref.f}_{11} \Rightarrow \text{IF } 1 - \omega_{ki} = 1 \text{ THEN } \beta_{k1} + \quad \alpha_{i1} \leq 1 \\ \Rightarrow \text{IF } 1 - \omega_{ki} = 1 \text{ THEN NOT}(\beta_{k2} + \quad \alpha_{i2} \geq 2) \Rightarrow \text{Ref.f}_{11} \Rightarrow \text{IF } 1 - \omega_{ki} = 1 \text{ THEN } \beta_{k2} + \alpha_{i2} \leq 1 \\ \Rightarrow \text{IF1} - \omega_{ki} = 1 \text{THEN NOT}(\beta_{k3} + \alpha_{i3} \geq 2) \Rightarrow \text{Ref.f}_{11} \Rightarrow \text{IF } 1 - \omega_{ki} = 1 \text{ THEN } \beta_{k3} + \quad \alpha_{i3} \leq 1\end{array}$$

$$\left|\begin{array}{l}\Rightarrow \text{IF } 1 - \omega_{ki} = 1 \text{ THEN } \beta_{k1} + \quad \alpha_{i1} \leq 1 \Rightarrow \text{Ref.f}_{14} \Rightarrow \beta_{k1} + \quad \alpha_{i1} \leq 1 + \omega_{ki} \\ \Rightarrow \text{IF } 1 - \omega_{ki} = 1 \text{ THEN } \beta_{k2} + \quad \alpha_{i2} \leq 1 \Rightarrow \text{Ref.f}_{14} \Rightarrow \beta_{k2} + \quad \alpha_{i2} \leq 1 + \omega_{ki} \\ \Rightarrow \text{IF } 1 - \omega_{ki} = 1 \text{ THEN } \beta_{k3} + \quad \alpha_{i3} \leq 1 \Rightarrow \text{Ref.f}_{14} \Rightarrow \beta_{k3} + \quad \alpha_{i3} \leq 1 + \omega_{ki}\end{array}\right| \tag{7.3}$$

Model of (7.2):

In this calculation, we just need the case in which we use $\omega_{ki} = 1$, case (7.1). We can give freedom to the system when conditions to use $\omega_{ki} = 1$ do not coincide. That means that it is not necessary to impose (7.2). Anyway, if we want ω_{ki} to be worth 0 when the proposal is not fulfilled, for a formal question, we model the case (7.2):

$$\Rightarrow \forall k,i : \text{IF } \omega_{ki} = 1 \text{ THEN } \ (\beta_{k1} = 1 \quad \text{AND} \quad \alpha_{i1} = 1) \\ \text{OR } (\beta_{k2} = 1 \quad \text{AND} \quad \alpha_{i2} = 1) \qquad \text{OR } (\beta_{k3} = 1 \quad \text{AND} \quad \alpha_{i3} = 1)$$

$$\Rightarrow \text{Ref } f_{37} \Rightarrow \forall k,i : \text{IF } \omega_{ki} \\ = 1 \text{ THEN } (\beta_{k1} + \alpha_{i1} \geq 2) \text{ OR } (\beta_{k2} + \alpha_{i2} \geq 2) \text{ OR } (\beta_{k3} + \alpha_{i3} \geq 2)$$

$\Rightarrow$ Ref f_{35} $\Rightarrow$

$$\left.\begin{array}{l} \Rightarrow \forall k,i : \lambda_{ki1} = 1 \text{ IF AND ONLY IF } \ (\beta_{k1} + \alpha_{i1} \geq 2) \\ \Rightarrow \text{Ref } f_{25} \Rightarrow \beta_{k1} + \alpha_{i1} \geq 2\lambda_{ki1}; \beta_{k1} + \alpha_{i1} \leq 1 + \lambda_{ki1} \\ \Rightarrow \forall k,i : \lambda_{ki2} = 1 \text{ IF AND ONLY IF } \ (\beta_{k2} + \alpha_{i2} \geq 2) \\ \Rightarrow \text{Ref } f_{25} \Rightarrow \beta_{k2} + \alpha_{i2} \geq 2\lambda_{ki2}; \beta_{k2} + \alpha_{i2} \leq 1 + \lambda_{ki2} \\ \Rightarrow \forall k,i : \lambda_{ki3} = 1 \text{ IF AND ONLY IF } \ (\beta_{k3} + \alpha_{i3} \geq 2) \\ \Rightarrow \text{Ref } f_{25} \Rightarrow \beta_{k3} + \alpha_{i3} \geq 2\lambda_{ki3}; \beta_{k3} + \alpha_{i3} \leq 1 + \lambda_{ki3} \end{array}\right| \tag{7.4}$$

$$\Rightarrow \forall\ k, i : \text{IF } \omega_{ki} = 1 \text{ THEN } \ \lambda_{ki1} = 1 \text{ OR } \lambda_{ki2} = 1 \text{ OR } \lambda_{ki3} = 1 \\ \Rightarrow \text{Ref } f_{34} \Rightarrow \ \forall\ k, i : \text{IF } \omega_{ki} = 1 \text{ THEN } \ \lambda_{ki1} + \lambda_{ki2} + \lambda_{ki3} \geq 1$$

$$\Rightarrow \text{Ref } f_{15} \Rightarrow \forall k,i : \ \lambda_{ki1} + \lambda_{ki2} + \lambda_{ki3} \geq \omega_{ki} \tag{7.5}$$

The specification would be:

$$\forall i : \ \sum_{k=1}^{25} \omega_{ki} \leq K_i \tag{7.6}$$

I2. Quantitative selection rules:

The two sets of binary variables have associated quantitative selection specifications. Let's analyze each activity and its participating elements (Table 7.8).

The three specifications are explicitly reflected in the statement.

I3. Logical conditions between activities:

Table 7.8 Selection rules in Illustration 7.4 – Version 1

Activity	Elements selecting	Selectable elements	Selection	Constraints
Install	Locations $i = 1\ldots10$	Centers	≤ 1	E1
	Centers $j = 1\ldots3$	Locations	≤ 1	E2
Allocate	Supermarket $k = 1\ldots25$	Centers	$= 1$	E3
	Center $j = 1\ldots3$	Supermarkets	No imposed selection specification	–

The two decision activities have an obvious logical relationship: in order for a center to supply supermarkets, it must have been installed. We express the logical proposition:

“IF a center has been installed THEN it can supply supermarkets.”
As the proposition expresses possibility, we define it negatively (see Sect. 6.8.6):
“IF a center has not been installed THEN It cannot supply supermarkets.”

Mathematically:
The specification falls on any center that has not been installed in any location and will not serve any supermarket of the existing ones. Therefore, it falls on the three sets of elements:

$$\forall j, i, k : \text{IF } \alpha_{ij} = 0 \quad \text{THEN} \quad \beta_{kj} = 0$$

But it can also be expressed by reducing the number of propositions in the following way (we change the way of expressing the specification using numerals):

“IF the number of locations in which a center is installed is null THEN the number of supermarkets to which it will supply will be null”:

$$\forall j : \text{IF } \sum_{i=1}^{10} \alpha_{ij} = 0 \quad \text{THEN} \quad \sum_{k=1}^{25} \beta_{kj} = 0$$

Expression $\sum_{i=1}^{10} \alpha_{ij}$ is binary, thanks to the specification that a center cannot be installed in more than one location. $\sum_{k=1}^{25} \beta_{kj}$is an integer expression.

We express the specification as:

$$\Rightarrow \text{Ref.f}_7 \Rightarrow \forall j : \text{IF } 1 - \sum_{i=1}^{10} \alpha_{ij} = 1 \quad \text{THEN} \quad \sum_{k=1}^{25} \beta_{kj} = 0 \Rightarrow$$

$$\Rightarrow \text{Ref.f}_{\text{LB}} \Rightarrow \forall j : \text{IF } 1 - \sum_{i=1}^{10} \alpha_{ij} = 1 \quad \text{THEN} \quad \sum_{k=1}^{25} \beta_{kj} \leq 0 \Rightarrow$$

$$\Rightarrow \text{Ref.f}_{14} \Rightarrow \forall j: \sum_{k=1}^{25} \beta_{kj} \leq 25 \sum_{i=1}^{10} \alpha_{ij} \tag{7.7}$$

I4. Bounds of discrete measurable activities: they do not exist.
I5. Flow balance constraints: they do not exist.

Explicit Specifications

E1. "In a location one center is installed at the most."

It is a quantitative selection norm of upper bound for $\forall i$ in α_{ij}:

$$\forall i: \sum_{j=1}^{3} \alpha_{ij} \leq 1 \tag{7.8}$$

E2. "Each installed distribution center must be in a single location."

It is also a quantitative selection rule for $\forall j$ in α_{ij}. It would be an upper bound since there is no obligation to install each center.

$$\forall j: \sum_{i=1}^{10} \alpha_{ij} \leq 1 \tag{7.9}$$

E3. "Each supermarket must be allocated to a distribution center for its supply."

Quantitative selection rule for $\forall k$ in the variables β_{kj}

$$\forall k: \sum_{j=1}^{3} \beta_{kj} = 1 \tag{7.10}$$

E4. "If the company installs a center in location 3 and another in location 5, it cannot install any in location 6."

It is a compound logical proposition. The simple propositions of which it is formed are quantitative selection specifications:

– Install a center in location 3:
  $\sum_{j=1}^{3} \alpha_{3j} = 1$
– Install a center in location 5:
  $\sum_{j=1}^{3} \alpha_{5j} = 1$

– Unable to install a center in location 6:

$$\sum_{j=1}^{3} \alpha_{6j} = 0$$

Compound logical proposition:

$$\text{IF } \sum_{j=1}^{3} \alpha_{3j} = 1 \text{ AND } \sum_{j=1}^{3} \alpha_{5j} = 1 \text{ THEN } \sum_{j=1}^{3} \alpha_{6j} = 1$$

where all simple propositions are binary, by the definition of the number of centers in a location. The modelling would be:

$$\Rightarrow \text{Ref.f}_{37} \Rightarrow \text{IF } \sum_{j=1}^{3} \alpha_{3j} + \sum_{j=1}^{3} \alpha_{5j} \geq 2 \text{ THEN } \sum_{j=1}^{3} \alpha_{6j} = 0 \Rightarrow$$

$$\Rightarrow \text{Ref.f}_{3} \Rightarrow \text{IF } \sum_{j=1}^{3} \alpha_{6j} = 1 \text{ THEN } \sum_{j=1}^{3} \alpha_{3j} + \sum_{j=1}^{3} \alpha_{5j} < 2 \Rightarrow$$

$$\Rightarrow \text{Ref.f}_{4} \Rightarrow \text{IF } \sum_{j=1}^{3} \alpha_{6j} = 1 \text{ THEN } \sum_{j=1}^{3} \alpha_{3j} + \sum_{j=1}^{3} \alpha_{5j} \leq 1 \Rightarrow$$

$$\Rightarrow \text{Ref.f}_{14} \Rightarrow \sum_{j=1}^{3} \alpha_{3j} + \sum_{j=1}^{3} \alpha_{5j} \leq 1 + (2-1)\left(1 - \sum_{j=1}^{3} \alpha_{6j}\right) \Rightarrow$$

$$\Rightarrow \sum_{j=1}^{3} \alpha_{3j} + \sum_{j=1}^{3} \alpha_{5j} \leq 2 - \sum_{j=1}^{3} \alpha_{6j} \tag{7.11}$$

Objective Criterion

Minimize Costs

Costs = Installation costs in locations + Additional cost to allocate more than 10 supermarkets to any center

– Installation costs in locations $= \sum_{i=1}^{10} \sum_{j=1}^{3} \text{CI}_i \alpha_{ij}$.
– Additional Cost: it requires a logical calculation to find out if more than 10 supermarkets have been assigned in each center. The calculation falls on each center:

Binary logical calculation: Center with more than 10 supermarkets.
Applied to: Centers $j = 1\ldots3$.
Variables:

$$\delta_j = \begin{cases} 1 \text{ if center } j \text{ has more than 10 supermarkets} \\ 0 \text{ otherwise} \end{cases}$$

Logical propositions:

$$\forall j : \text{IF } \sum_{k=1}^{25} \beta_{kj} > 10 \text{ THEN } \delta_j = 1$$

[Ref. $S_{V:}$ It implies a cost, so I can stop defining the value $\delta_j = 0$].

Proposition modelling:

$\Rightarrow$ Ref.f_3 $\Rightarrow \forall j$: IF $\delta_j = 0$ THEN $\sum_{k=1}^{25} \beta_{kj} \le 10$

$\Rightarrow$ Ref.f_7 $\Rightarrow \forall j$: IF $1 - \delta_j = 1$ THEN $\sum_{k=1}^{25} \beta_{kj} \le 10$

$\Rightarrow$ Ref.f_{14} $\Rightarrow \forall j$: $\sum_{k=1}^{25} \beta_{kj} \le 10 + (25 - 10)\big(1 - (1 - \delta_j)\big)$

$$\Rightarrow \forall j : \sum_{k=1}^{25} \beta_{kj} \le 10 + 15\delta_j \tag{7.12}$$

O.F.: Min $\sum_{i=1}^{10} \sum_{j=1}^{3} \mathrm{CI}_i \alpha_{ij} + \sum_{j=1}^{3} F\delta_j$

Version 2

By analyzing the table of elements of Version 1, where the centers are identical elements, we will try to eliminate the specifications that allude to each center in an individual way, to be able to treat it as a collective element without changing the system. Let us take a look at the references to the centers in the statement:

– “Install a maximum of 3 product distribution centers”: Here, the centers are referred to with a numeral, so there is no change at all.
– “In a location one center is installed at the most”: Here we can also interpret a center as a numeral. It refers to the number of centers that can be installed in one location.
– “Each installed distribution center must be in a single location”: Here you make an individual reference about each center. Therefore, it is necessary to modify this reference. What we are going to do is eliminate that specification. Through specifying that the number of centers installed in locations cannot be more than 3 and that in one location a center is installed at most, we assume that the number of centers installed will correspond to the sum of the centers installed in the locations.
– “If the company installs a center in location 3 and another in location 5, it cannot install any in location 6”: here again the reference to the center can be considered as a numeral. Written in another way: “if the company installs 1 center in location 3 and 1 center in location 5, the number of centers in 6 must be 0.”
– “If the number of supermarkets allocated to a center exceeds 10, the center is penalized . . ..”

Here, each center is referred to in a particular way, since the affirmation is made for each center. This statement makes use of one of the decision activities: allocate supermarkets to centers.

Table 7.9 Elements of Illustration 7.4 – Version 2

			Data				
Elements	Set	QN	Name	Param	Type	Belong	Value
Locations	$i = 1 \ldots 10$	I_U	Cost	CI_i	I	W	8
			Capacity	K_i	I	W	–
			Penalty	P_i	C	W	F
Centers	–	C_I	Max number	N	I	W	3
Supermarkets	$k = 1 \ldots 25$	I_U					

The only way to eliminate this allusion is to modify that decision activity. This way of describing systems is quite common when there are collective elements whose measurement on an element is at most one. This element ends up taking the place of the individual element with which it is related, although it is the latter on which the activities really fall. The idea is to embed the role of the centers in the locations. The centers act only as an indicator that the location can supply supermarkets. If we have installed a center in the location we can supply, otherwise we cannot. We are going to make the locations assume the role of center regarding the allocation of supermarkets. In that way, we would avoid the particular allusion to centers. The statement will then be as follows:

"There is a supermarket company that has 10 locations ($i = 1 \ldots 10$) to install a maximum of 3 product distribution centers. The installation cost in each location is established in \$$CI_i$. In a location one center is installed at the most. ~~And each installed distribution center must be in a single location.~~

The Company has 25 supermarkets ($j = 1 \ldots 25$) to be supplied from the locations with distribution centers. Each supermarket must be allocated to a location with a distribution center for its supply. Each location has a K_i capacity that is expressed in the number of supermarkets that it can supply.

By legal requirements, if the company installs a center in location 3 and another in location 5, it cannot install any in location 6.

Objective: Minimize the costs of the problem taking into account that if the number of supermarkets allocated to a ~~center~~ location exceeds 10, the ~~center~~ location is penalized with a cost of \$F."

Table of Elements (Table 7.9)

Decision Activities
Action: Install centers in locations.
Decision variables: x_i = number of centers installed in location *i*.
Action: Allocate supermarkets to locations.
Decision variables:

$$\beta_{ki} = 1 \text{ if supermarket } k \text{ is allocated to location } i; \text{ 0 otherwise.}$$

Specifications

We are going to review the specifications indicated in Version 1.

Table 7.10 Selection rules in Illustration 7.4 – Version 2

Activity	Elements selecting	Selectable elements	Selection	Constraints
Allocate	Supermarket $k = 1 \ldots 25$	Locations	$= 1$	E3
	Location $i = 1 \ldots 10$	Supermarkets	$\leq K_i$	I1

1. Implicit Specifications

I1. Based on data: In this version the modelling associated with the capacity of the locations is simplified, since it is the activity represented in β_{ki} that consumes that capacity. The logical calculation ω_{ki} defined in Version 1 is equivalent to β_{ki}

$$\forall i: \sum_{k=1}^{25} \beta_{ki} \leq K_i \tag{7.13}$$

I2. Quantitative selection rules:

Those associated with centers with locations and centers with supermarkets disappear, and supermarkets with locations appear (Table 7.10).

They are specifications that are expressed explicitly (E3) or already defined as capacity specification (I1).

I3. Logical conditions between activities: The two decision activities maintain the same logical relationship as in Version 1, but it is formulated with the following expression:

IF a center has been installed in a location, THEN that location "can" supply supermarkets.

In a negative way:

IF the number of centers installed in a location is zero, THEN that location cannot supply any supermarket.

Mathematically:

The specification falls on any location and any supermarket: $\forall i, k$: IF $x_i = 0$ THEN $\beta_{ki} = 0$.

And again, the number of propositions could be reduced if we join all the propositions for i:

$$\forall i : \text{IF } x_i = 0 \quad \text{THEN} \quad \sum_{k=1}^{25} \beta_{ki} = 0$$

Modelling: $\Rightarrow$ Ref. $f_7 \Rightarrow \forall i$: IF $1 - x_i = 1$ THEN $\sum_{k=1}^{25} \beta_{ki} = 0 \Rightarrow$

$\Rightarrow$ Ref.$f_{LB} \Rightarrow \forall i$: IF $1 - x_i = 1$ THEN $\sum_{k=1}^{25} \beta_{ki} \leq 0$

$$\Rightarrow \text{Ref.f}_{14} \Rightarrow \forall i: \sum_{k=1}^{25} \beta_{ki} \leq 25x_i \tag{7.14}$$

I4. Bounds of discrete measurable activities: explicitly in E1.
I5. Flow balance constraints: they do not exist.

2. Explicit Specifications

E1. "In a location one center is installed at most."

Now it is an upper bound of a measurable activity.

$$\forall i: x_i \leq 1 \tag{7.15}$$

E2. "Install a maximum of 3 product distribution centers."

$$\sum_{i=1}^{10} x_i \leq 3 \tag{7.16}$$

E3. "Each supermarket must be allocated to a location with a distribution center for its supply."

It is a quantitative norm of selection for the β_{ki} variables.

$$\forall k: \sum_{i=1}^{10} \beta_{ki} = 1 \tag{7.17}$$

The rule states that the location you choose must have a center installed, that is, $x_i = 1$. As already explained in Sect. 6.3, this type of condition is excluded from the quantitative rule, where I will consider all the options for the standard. It must be in an additional specification where the selection of an element that does not meet that condition is limited or prevented. That is already reflected and modelled in the implicit specification number 13.

E4. "If the company installs a center in location 3 and another in location 5, it cannot install any in location 6."

Now the simple propositions that make up this compound proposition express assignment specifications:

Compound logical proposition: IF $x_3 = 1$ AND $x_5 = 1$ THEN $x_6 = 0$.

All the functions of simple propositions can be considered as binary, since $\forall i: x_i \leq 1$:

$\Rightarrow$Ref. $\text{f}_{37} \Rightarrow$ IF $x_3 + x_5 \geq 2$ THEN $x_6 = 0$

$\Rightarrow$Ref. $f_3 \Rightarrow$ IF $x_6 = 1$ THEN $x_3 + x_5 < 2$
$\Rightarrow$Ref. $f_4 \Rightarrow$ IF $x_6 = 1$ THEN $x_3 + x_5 \leq 1$

$$\Rightarrow \text{Ref.f}_{14} \Rightarrow x_3 + x_5 \leq 2 - x_6 \tag{7.18}$$

Objective Criterion

Minimize Costs
Costs = Installation costs in locations + Additional cost to allocate more than 10 supermarkets to any location.

Installation costs of centers in locations = $\sum_{i=1}^{10} \text{CI}_i x_i$.
Additional cost: Logical calculation to find out if more than 10 supermarkets have been assigned in each location:

Binary logical calculation: Location with more than 10 supermarkets.
Applied to: Locations $i = 1\ldots10$.
Variables:

$$\delta_i = \begin{cases} 1 \text{ if location } i \text{ has more than 10 supermarkets} \\ 0 \text{ otherwise} \end{cases}$$

Logical proposition: $\forall i$: IF $\sum_{k=1}^{25} \beta_{ki} > 10$ THEN $\delta_i = 1$ [Ref. S_V].

Proposition Modelling: $\Rightarrow$ Ref. $f_3 \Rightarrow \forall i$: IF $\delta_i = 0$ THEN $\sum_{k=1}^{25} \beta_{ki} \leq 10$.

$\Rightarrow$ Ref.$f_7 \Rightarrow \forall i$: IF $1 - \delta_i = 1$ THEN $\sum_{k=1}^{25} \beta_{ki} \leq 10$

$$\Rightarrow \text{Ref.f}_{14} \Rightarrow \forall i : \sum_{k=1}^{25} \beta_{ki} \leq 10 + 15\delta_i \tag{7.19}$$

O.F: Min $\sum_{i=1}^{10} \text{CI}_i x_i + F\delta_i$

Version 3

In this third version, we are going to consider supermarkets as a collective element. As we have observed in the tables of elements of the previous versions, the 25 supermarkets are identical. Therefore, we must try to modify the description of the system where they are alluded to in a particular way, without changing the system.

To do this, the allocation of each supermarket will be made globally, requiring 25 supermarkets to be assigned to the locations with centers. The statement would now look like this:

Table 7.11 Elements of Illustration 7.4 – Version 3

			Data				
Elements	Set	QN	Name	Param	Type	Belonging	Value
Locations	$i = 1\ldots10$	I_U	Cost	CI_i	I	W	8
			Capacity	K_i	I	S	–
			Penalty	P_i	C	W	F
Centers	–	C_I	Max number	N	I	W	3
Supermarkets	–	C_D	Quantity	NS	I	W	25
				K_i			

"There is a supermarket company that has 10 locations ($i = 1\ldots10$) to install a maximum of 3 product distribution centers. The installation cost in each location is established in $\$CI_i$. In a location one center is installed at the most.

The Company has 25 supermarkets ($j = 1\ldots25$) to be supplied from the locations with distribution centers (the Company must assign 25 supermarkets to the locations with installed centers). ~~Each supermarket must be allocated to a location with distribution center for its supply.~~ Each location has a K_i capacity that is expressed in the number of supermarkets that it can supply.

By legal requirements, if the company installs a center in location 3 and another in location 5, it cannot install any in location 6.

Objective: Minimize the costs of the problem taking into account that if the number of supermarkets allocated to a location exceeds 10, the location is penalized with a cost of \$F."

The configuration of the model would now be as follows:

Table of Elements (Table 7.11)

The capacity attribute of the locations is shared with the supermarket element, since it refers to the number of supermarkets that can supply.

Decision Activities
Action: Install centers in locations.
Decision variables: x_i = number of centers installed in location i.

Action: Allocate supermarkets to locations.
Decision variables: y_i = Number of supermarkets allocated to location i.
Specifications

1. Implicit Specification.

I1. Based on data: In this version, the capacity consumption activity of the locations is y_i:

$$\forall i: \quad y_i \leq K_i \tag{7.20}$$

Supermarket inventories are explicitly specified (E3) in the statement.

I2. Quantitative selection rules: they do not exist.

I3. Logical conditions between activities: Between x_i and y_i, there is the same logical condition as in previous versions: If you do not install any center in a location, you cannot supply supermarkets:

The specification falls on any location:

$\forall i$: IF $x_i = 0$ THEN $y_i = 0$
Modelling: $\Rightarrow$ Ref. $\mathrm{f_{LB}} \Rightarrow \forall i$: IF $x_i \leq 0$ THEN $y_i \leq 0$

$$\begin{array}{ll} & \forall i: x_i \leq 0 + (1-0)(1-\omega_i) \\ \Rightarrow \mathrm{Ref.f_{17};f_{20}} & \Rightarrow \forall i: x_i \geq (0+1)(1-\omega_i) + 0\omega_i \\ & \forall i: y_i \leq 0 + (K_i - 0)(1-\omega_i) \\ \forall i: x_i \leq 1-\omega_i & \\ \Rightarrow \forall i: x_i \geq 1-\omega_i & \Rightarrow \begin{array}{l} \forall i: x_i = 1-\omega_i \\ \forall i: y_i \leq K_i(1-\omega_i) \end{array} \Rightarrow \forall i: y_i \leq K_i x_i \Big| \\ \forall i: y_i \leq K_i(1-\omega_i) & \end{array} \tag{7.21}$$

I4. Bounds of discrete measurable activities: explicitly in I1 and E1.

I5. Flow balance constraints: they do not exist.

2. Explicit Specifications

E1. "In a location one center is installed at the most."

Identical to the one expressed in Version 2.

$$\forall i: x_i \leq 1 \tag{7.22}$$

We can consider x_i as binary variable both here as in version 2.

E2. "Install a maximum of 3 product distribution centers."

Identical to the one expressed in Version 2.

$$\sum_{i=1}^{10} x_i \leq 3 \tag{7.23}$$

E3. "The Company must assign 25 supermarkets to the locations with installed centers."

In this version, what was a quantitative norm of selection becomes an assignment or equilibrium specification.

$$\sum_{i=1}^{10} y_i = 25 \tag{7.24}$$

E4. "If the company installs a center in location 3 and another in location 5, it cannot install any in location 6"

Same as in Version 2.

$$x_3 + x_5 \leq 2 - x_6 \tag{7.25}$$

Objective Criterion

Regarding Version 2, the defined logical calculation varies slightly:

Binary logical calculation: Location with more than 10 supermarkets
Applied to: Locations $i = 1 \ldots 10$
Variables:

$$\delta_i = \begin{cases} 1 \text{ if location } i \text{ has more than 10 supermarkets} \\ 0 \text{ otherwise} \end{cases}$$

Logical proposition: $\forall i$: IF $y_i > 10$ THEN $\delta_i = 1$ [Ref. S_V]
Proposition Modelling:
$\Rightarrow$Ref. $f_3 \Rightarrow \forall i$: IF $\delta_i = 0$ THEN $y_i \leq 10$
$\Rightarrow$Ref. $f_7 \Rightarrow \forall i$: IF $1 - \delta_i = 1$ THEN $y_i \leq 10$
$\Rightarrow$Ref.$f_{14} \Rightarrow \forall i$: $y_i \leq 10 + 15\delta_i$ (7.26)

O.F.: Min $\sum_{i=1}^{10} (\mathrm{CI}_i x_i + F\delta_i)$

The following table summarizes the analysis of the three versions regarding their size (Table 7.12).

Version 1 Constraints: (7.3); (7.6) to (7.12).
Version 2 Constraints: (7.13) to (7.19) except (7.15).
Version 3 Constraints: (7.20) to (7.26) except (7.22).
(7.15) and (7.22) do not count when considering x_i as binary.

It can be observed that the optimal size is to make both centers and supermarkets collectives. Making collectives only the centers, the number of variables is slightly reduced. This is because the number of centers is not significant, only 3, which does not lead to a considerable decrease in size.

Table 7.12 Comparison of version sizes

	Version 1	Version 2	Version 3
Number of binary variables	358	270	10
Number of integer variables	0	0	10
Number of constraints	805	57	53

7.5 Individual Elements with Capacity to Be Grouped in Collective Elements: Redefining the System Description with Complex Changes

When a system can be defined as an alternative to an initial description, with significant differences, we are facing systems that require a broad knowledge in the field of operational research and experience in the area to which the problem corresponds, to allow us to develop the specifications differently.

In this illustration, we are going to deal with an interval scheduling problem, the fixed jobs scheduling problem (Arkin and Silverberg 1987). In the first description, the problem specifications prevent the union of items as a collective element, despite being identical. However, through understanding the problem, we can make a substantial change in the specifications to stop referring to each item in particular.

Illustration 7.5: Fixed Job Scheduling Problem

There is a set of n jobs characterized by a starting time, a duration, and a weight. There is also a set of m machines to process the jobs. It is about maximizing the total weight of the processed jobs taking into account that a machine cannot process two jobs that overlap in time. A job, if processed, is processed by a single machine.

Unitary Version

After reading the problem, we can define jobs and machines as elements. Each job has three data of own values, starting time, duration, and weight. Therefore, we consider them individual and also unitary because none of the data are measurable. When processing all the jobs in a single machine, we cannot divide the elements, so we work with it completely.

On the other hand, the machines do not present any data, so they can be considered identical; nor do they possess any property that must be measured in the decision activities. Therefore, we consider each machine as a unitary element a priori or the *m* machines as items of a collective element in the system. This will depend on the specifications of the problem. When analyzing the text, we can see that there is a specification that states that "a machine cannot process two jobs that overlap in time." When we say "a machine," since we use the indeterminate article, we refer to any of the machines in a particular way. We could have rewritten the text using "any": any machine cannot process two jobs that overlap in time. Therefore, machines are referred to in a particular way, and it is necessary to consider each machine as unitary.

Table of Elements (Table 7.13)

Table 7.13 Elements of Illustration 7.5 – Individual Version

			Data				
Elements	Set	QN	Name	Param	Type	Belonging	Value
Jobs	$i = 1...n$	I_U	Starting time	s_i	C	W	–
			Duration	d_i	C	W	–
			Weight	p_i	C	W	–
Machines	$j = 1...m$	I_U					

Table 7.14 Selection rules in Illustration 7.5

Activity	Elements selecting	Selectable elements	Selection	Constraints
Process	Jobs $i = 1...n$	Machines	≤ 1	E1
	Machines $j = 1...m$	Jobs	–	

Decision Activities

Action: Process jobs in machines.
Decision variables: $\alpha_{ij} = 1$ if I process job i in machine j; 0 otherwise.

Specifications

1. Implicit Specifications

I1. Based on data: they do not exist.
I2. Quantitative selection rules: the only rule for the variables α_{ij} is described explicitly: A job, if processed, is processed by a single machine (Table 7.14).

I3. Logical conditions between activities: they do not exist.
I4. Bounds of discrete measurable activities: they do not exist.
I5. Flow balance constraints: they do not exist.

2. Explicit Specifications

From the statement the following specifications are extracted:

E1. "A job, if processed, is processed by a single machine."

$$\forall i : \quad \sum_{j=1}^{m} \alpha_{ij} \leq 1 \tag{7.27}$$

E2. "A machine cannot process two jobs that overlap in time."

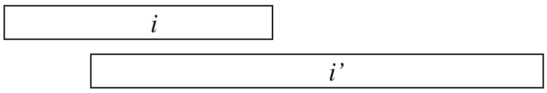


Fig. 7.2 Overlapping jobs

A machine cannot process any pair of jobs that overlap in time. It is a logical proposition with the operator negation that falls on each machine and on each pair of jobs that overlap in time.

The pairs of overlapping jobs can be calculated before the modelling of the specification, since it is information obtained from the data of the jobs. Graphically, two jobs i and i' are overlapped if they are processed together at some point in time (Fig. 7.2).

Mathematically, the overlap occurs when the values of the start and duration data of i and i' comply with:

$$\forall i, i' / i \neq i' : \ s_i \leq s_{i'} \ \ \& \ \ s_i + d_i > s_{i'}$$

The specification would then be as follows:

$$\forall j, \forall i, i' / i \neq i', s_i \leq s_{i'} \ \ \& \ \ s_i + d_i > s_{i'} : \mathrm{NOT}\big(\alpha_{ij} = 1 \quad \mathrm{AND} \ \ \alpha_{i'j} = 1\big)$$

Modelling:

$$\begin{aligned} &\Rightarrow \mathrm{Ref.f}_{37} \Rightarrow \forall j, \forall i, i' / i \neq i', s_i \leq s_{i'} \ \ \& \ \ s_i + d_i > s_{i'} : \mathrm{NOT}\big(\alpha_{ij} \ + \ \alpha_{i'j} \geq 2\big) \\ &\Rightarrow \mathrm{Ref.f}_{11} \Rightarrow \forall j, \forall i, i' / i \neq i', s_i \leq s_{i'} \ \ \& \ \ s_i + d_i > s_{i'} : \alpha_{ij} \ + \ \alpha_{i'j} \leq 1 \end{aligned} \tag{7.28}$$

Objective Criterion

$$\mathrm{Max} \ \sum_{i=1}^{n} \sum_{j=1}^{m} p_i \alpha_{ij}$$

Collective Version

Let's modify the statement accordingly: If the modeller has some experience in this kind of problem, they will know that the overlap specification is equivalent to saying that, at any moment of the horizon in which the jobs are processed, the number of jobs that are processed must not exceed the number of existing machines (Kroon et al. 1995).

Let us look at the following example:

Suppose there is a system with six jobs that are represented in a time diagram (Fig. 7.3). The system has two machines.

If in any of the moments that make up the plan we say that no more jobs are processed than existing machines, any solution can be assigned to the machines afterwards without any overlapping conflict, because every time I start a job I will have the security of knowing that there is a machine free, since if it were not so, the

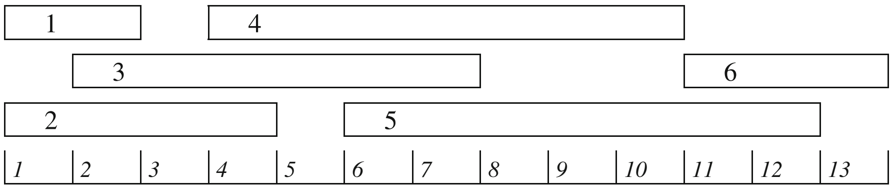


Fig. 7.3 Example

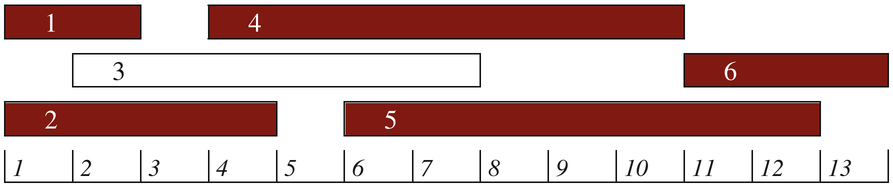


Fig. 7.4 A solution to the problem

selection would not comply with the specification. For our example, a possible solution would be to select the jobs 1,2,4,5,6, discarding job 3 (Fig. 7.4).

Moreover, it is not necessary to impose this specification at any time in the horizon, which would require defining all the instants as an element, but only imposing the overlap specification at times when new overlaps can occur, which are only the starting time of each job.

The solution for this version of the problem will provide the selected jobs, but not the specific assignment of which machine processes each job. This task should be obtained after the resolution, through a simple assignment procedure.

On the other hand, the second specification of the previous version claimed that a job is processed by a single machine at the most. This specification ceases to exist in this version, since the processing of the job in the machines is obtained afterwards and it is in the assignment procedure where that characteristic is imposed.

For all this, the problem would have been stated and modelled in the following way in the collective version:

There is a set of n jobs characterized by a starting time, a duration, and a weight. There is also a set of m machines to process the jobs. For every starting time of a job, the number of jobs that are being processed must not exceed the number of existing machines. It is about maximizing the total weight of the processed jobs ~~*taking into account that a machine cannot process two jobs that overlap in time. A job, if processed, is processed by a single machine*~~.

Table of Elements (Table 7.15)

Decision Activities

Action: Process jobs in machines.

Table 7.15 Elements of Illustration 7.5 – Collective Version

			Data				
Elements	Set	QN	Name	Param	Type	Belonging	Value
Jobs	$i = 1...50$	I_U	Starting time	s_i	C	W	–
			Duration	d_i	C	W	–
			Weight	p_i	C	W	–
Machines	–	C_D	Quantity	m	I	W	–

Decision variables: $\alpha_i = 1$ if I process job i in the machines; 0 otherwise.

The jobs are the direct complement of the action. The machines are implicit in the processing and could be excluded from the definition.

Specifications

1. Implicit Specifications

I1. Based on data: they do not exist.
I2. Quantitative selection rules: they do not exist.
I3. Logical conditions between activities: they do not exist.
I4. Bounds of discrete measurable activities: they do not exist.
I5. Flow balance constraints: they do not exist.

2. Explicit Specifications

A specification is extracted from the statement:

E1. "For every starting time of a job, the number of jobs that are being processed must not exceed the number of existing machines."

It is not necessary to enter the starting times of the jobs in the table of elements as a period element, although it could have been done in that way as well. The reason is that these elements as such are embedded in each job element, and therefore, to allude to them is to refer to each job.

Therefore, the specification falls on each job i. It is a specification of upper bound for the sum of the variables α_k corresponding to jobs k that are processed in the instant of the starting time of job i. These variables will return the total of machines that process those jobs k. This sum assumes that the number of m machines is not exceeded.

$$\forall i : \sum_{k/s_k \leq s_i \ \& \ s_k + d_k > s_i} \alpha_k \leq m \tag{7.29}$$

Objective Criterion

Maximize the total weight of the processed jobs

Table 7.16 Comparison of versions of Illustration 7.5

	Individual Version	Collective Version
Number of binary variables	$n\times m$	n
Number of integer variables	0	0
Number of constraints	$n + (m\times R)$	n

O.F.: Max $\sum_{i=1}^{50} p_i\alpha_i$

Let us take a look at a comparison of both models. We consider a general scenario with n jobs and m machines. For the individual version, we consider R as the number of overlapping job pairs $[0 \leq R \leq (n^2-n)]$ (Table 7.16).

Individual version constraints: (7.27) and (7.28).

Collective version constraints: (7.29).

The number of variables and constraints in the collective version is significantly reduced.

7.6 Individual Elements with Capacity to be Grouped in Collective Elements: Through Grouping into Subsets

The grouping into subgroups of items that have the same data values usually leads to models that are much smaller in size and generally more efficient. We are going to raise the grouping in this section without modifying data values, since it is not necessary. In the illustration, guests of a wedding are grouped according to their characteristics. You will see two versions of modelling, one that does not group and considers each element as individual and the version that groups collective elements.

Illustration 7.6: Assigning Tables to Wedding Guests (Lewis and Carroll 2016)
There is a celebration hall that should seat the guests of a wedding. The living room has tables of two sizes. Tables of 8 seats (18 tables) and tables of 12 seats (12 tables).

200 guests attend the wedding, including seniors, young people, and children. The guests are also classified as either the groom's guest or the bride's guest. Therefore, each guest has the information of age category and origin.

It has been decided that guests who are children must be placed at tables just for them. However, older people and young people can mix at the tables, but what is intended is to have the least number of tables where older and younger people mix (i.e., if at a table they seat young and old people, that table is removed in the objective function).

On the other hand, if a table is used, it must have at least six people seated.

Regarding the groom's guests and the bride's guests, we also intend to mix as little as possible at the wedding. In this way, in the objective function we will minimize the tables where both the mix of older and younger people will be

Table 7.17 Elements of Illustration 7.6 – Unitary Version

Elements	Set	QN	Data				
			Name	Param	Type	Belonging	Value
Tables	$i = 1\ldots30$	I_U	Capacity	K_i	I	S	–
Seats	–	C_D		K_i			
Guests	$k = 1\ldots200$	I_U	Origin	P_{kp}	B	S	–
			Age category	C_{kc}	B	S	–
Origins	$p = 1,2$ 1: Bride 2: Groom	I_U		P_{kp}			
Age categories	$c = 1,2,3$ 1: Children 2: Young 3: Senior	I_U		C_{kc}			

produced, as groom's guests and bride's guests, but if in a table there are both circumstances, that table only penalizes once, not twice.

For the resolution, we will propose two versions: first of all, a unitary individual version, where each guest is a unitary individual element of the system. In the second version, we will group the guests into collective elements according to their typology, since the statement allows it.

Regarding the tables, both in one version and in another, it is necessary to consider them as individual, although the 18 tables of 8 people are identical, as well as the 12 tables of 12 people. The reason is that they are alluded to in an individual way on several occasions in the problem: the text implies the need to know, with respect to each table, whether or not it penalizes the objective function ("i.e., if both young and old people sit at a table, that table penalizes the objective function"). There is also a specification to sit at least six people at each table that is used. "On the other hand, if you use a table, you must have at least 6 people sitting at it." In addition to all this, each table is individually measurable because the attribute of number of guests acts as capacity attribute.

Individual Version

In this first version, the model is built considering each guest as unitary, since guests are not identical, each one has its age category and its origin as not measurable data.

The table of elements would be as follows:

Table of Elements (Table 7.17)

In the definition of elements, we have made the Origin explicit (groom's guest or bride's guest) and the age category of the guests. We could also have separated the tables according to their size, but to obtain a more simplified model, we have included them all in the same set.

Decision Activities

Table 7.18 Selection rules in Illustration 7.6

Activity	Elements selecting	Selectable elements	Selection	Constraints
Seat	Guests $i = 1...200$	Tables	$= 1$	$\forall k: \sum_{i=1}^{30} \alpha_{ik} = 1$ (7.31)
	Tables $j = 1...30$	Guests	$\leq K_i$	I1

Action: Sitting guests at tables.
Decision variables:
α_{ik}: 1 if I seat guest k at table i; 0 otherwise. $i = 1...30$; $k = 1...200$;
Specifications

1. *Implicit Specifications*

   I1. Based on data: There are specifications of capacity consumption for the tables. Each table has a capacity or number of guests that can be seated.

$$\forall i: \quad \sum_{k=1}^{200} \alpha_{ik} \leq K_i \tag{7.30}$$

   I2. Quantitative selection rules: It is necessary to analyze the activity of the problem (Table 7.18).

   I3. Logical conditions between activities: they do not exist.
   I4. Bounds of discrete measurable activities: they do not exist.
   I5. Flow balance constraints: they do not exist.

2. *Explicit Specifications*

   E1. "Guests who are children have to be placed at tables just for them."

It would be modelled as a logical proposition in which if I seat a child at any table, the guests at that table are all children:

$$\forall i, k/C_{k1} = 1 : \text{IF } \alpha_{ik} = 1 \text{ THEN } \sum_{k'=1}^{200} \alpha_{ik'} = \sum_{k'/C_{ik'}=1} \alpha_{ik'} \Rightarrow$$

$$\Rightarrow \forall i, k/C_{k1} = 1 : \text{IF } \alpha_{ik} = 1 \text{ THEN } \sum_{k'=1}^{200} \alpha_{ik'} - \sum_{k'/C_{ik'}=1} \alpha_{ik'} = 0 \Rightarrow$$

$$\Rightarrow \text{Ref } f_{LB} \Rightarrow \forall i, k/C_{k1} = 1 : \text{IF } \alpha_{ik} = 1 \text{ THEN } \sum_{k'=1}^{200} \alpha_{ik'} - \sum_{k'/C_{ik'}=1} \alpha_{ik'} \leq 0$$

$$\Rightarrow \text{Ref f}_{14} \Rightarrow \forall i, k/C_{k1} = 1 : \sum_{k'=1}^{200} \alpha_{ik'} - \sum_{k'/C_{ik'}=1} \alpha_{ik'} \le K_i(1-\alpha_{ik}) \qquad (7.32)$$

This expression is equivalent to defining that: "If a child is at a table, there cannot be young and senior guests at that table":

$$\forall i, k/C_{k1} = 1 : \text{IF } \alpha_{ik} = 1 \ \text{ THEN } \sum_{k'/C_{k'1}=0} \alpha_{ik'} = 0$$
$$\Rightarrow \text{Ref.f}_{\text{LB}} \Rightarrow \forall i, k/C_{k1} = 1 : \text{IF } \alpha_{ik} = 1 \ \text{ THEN } \sum_{k'/C_{k'1}=0} \alpha_{ik'} \le 0$$

$$\Rightarrow \text{Ref.f}_{37} \Rightarrow \forall i, k/C_{k1} = 1 : \sum_{k'/C_{k'1}=0} \alpha_{ik'} \le K_i(1-\alpha_{ik}) \qquad (7.33)$$

This proposition could also have been modelled through the statement that children can only sit with children, so we could have expressed it in negative as follows: "a child cannot be sitting at a table with a non-child guest."

Therefore, the proposition falls on any table, any child and any non-child guest:

$$\forall i, k/C_{k1} = 1, k'/C_{k'1} = 0 : \text{NOT}(\alpha_{ik} = 1 \ \text{ AND } \ \alpha_{ik'} = 1) \Rightarrow$$
$$\Rightarrow \text{Ref.f}_{37} \Rightarrow \forall i, k/C_{k1} = 1, k'/C_{k'1} = 0 : \text{NOT}(\alpha_{ik} + \ \alpha_{ik'} \ge 2)$$

$$\Rightarrow \text{Ref.f}_{11} \Rightarrow \forall i, k/C_{k1} = 1, k'/C_{k'1} = 0 : \alpha_{ik} + \ \alpha_{ik'} \le 1 \qquad (7.34)$$

It is a simpler restriction, but we need many more to impose the same specification.

E2. "If a table is used, it must have at least 6 people seated at it."

It is also a logical proposition that uses a logical calculation not defined until now as an input to the condition: "a table is used" or "used table."

Since the logical calculation is defined as input in the proposition of the specification, we can choose to propose the proposition without defining that logical calculation, since the modelling of the proposition itself will help to define the calculation. We propose it in the following way:

"If there are guests seated at a table, there must be at least 6."

$$\forall i : \text{IF } \sum_{k=1}^{200} \alpha_{ik} > 0 \ \text{ THEN } \sum_{k=1}^{200} \alpha_{ik} \ge 6$$
$$\Rightarrow \text{Ref.f}_5 \Rightarrow \forall i : \text{IF } \sum_{k=1}^{200} \alpha_{ik} \ge 1 \quad \text{THEN } \sum_{k=1}^{200} \alpha_{ik} \ge 6$$

$$\Rightarrow \text{Ref.f}_{18,}\text{f}_{21} \Rightarrow \forall i : \sum_{k=1}^{200} \alpha_{ik} \geq \omega_i \tag{7.35}$$

$$\Rightarrow \forall i : \sum_{k=1}^{200} \alpha_{ik} \leq k_i \omega_i \tag{7.36}$$

$$\Rightarrow \forall i : \sum_{k=1}^{200} \alpha_{ik} \geq 6\omega_i \tag{7.37}$$

ω_i is the logical calculation that responds to whether a table has been used or not. The third group 7.37 makes the group 7.35 unnecessary.

Objective Criterion

Minimize mixed tables.

To establish the expression of the objective function, we need to find out if each table has been mixed, that is, if it penalizes in the objective function.

The mix can occur by mixing guests of age category 2 and 3, or by mixing guests from different origins. Everything can be collected in a logical calculation per table:

Binary logical calculation: Mixed table.
Applied to: Tables $i = 1\ldots30$.
Variables:

$$\delta_i = \begin{cases} 1 \text{ if table } i \text{ is mixed} \\ 0 \ \text{ otherwise} \end{cases}$$

Logical proposition: We can raise it with multiple conditionals to determine value 1 of δ_i (value 0 is not necessary to determine it because value 1 supposes cost [Ref. Sv]) or by means of a smaller proposition:

Multiple conditionals:

$$\forall i : \text{IF } (\alpha_{i1} = 1 \text{ AND } \alpha_{i2} = 1 \text{ AND } (C_{12} \neq C_{22} \text{ OR } C_{13} \neq C_{23} \text{ OR } P_{11} \neq P_{21})) \text{ OR}$$
$$(\alpha_{i1} = 1 \text{ AND } \alpha_{i3} = 1 \text{ AND } (C_{12} \neq C_{32} \text{ OR } C_{13} \neq C_{33} \text{ OR } P_{11} \neq P_{31}) \text{ OR} \ldots$$
$$\text{THEN } \quad \delta_i = 1$$

$$\Rightarrow \text{Ref.f}_{40} \Rightarrow \begin{array}{l} \forall i, k, k' : \text{IF } \alpha_{ik} = 1 \text{ AND } \alpha_{ik'} = 1 \\ \text{AND } (C_{k2} \neq C_{k'2} \text{ OR } C_{k3} \neq C_{k'3} \text{ OR } \text{P}_{k1} \neq P_{k'1}) \text{ THEN } \quad \delta_i = 1 \end{array}$$

In a smaller proposition:

$$\forall i : \text{IF} \left(\sum_{k/C_{k2}=1} \alpha_{ik} > 0 \text{ AND } \sum_{k/C_{k3}=1} \alpha_{ik} > 0 \right)$$

$$\text{OR}\left(\sum_{k/P_{k1}=1}\alpha_{ik}>0\ \text{ AND }\ \sum_{k/P_{k2}=1}\alpha_{ik}>0\right)\ \text{ THEN }\quad \delta_i=1 \tag{7.38}$$

Proposition modelling: (We use Proposition 7.38)

$$\Rightarrow \text{Ref.f}_5 \Rightarrow\Rightarrow \begin{array}{l}\forall i:\text{IF}\left(\sum\limits_{k/C_{k2}=1}\alpha_{ik}\geq 1\ \text{ AND }\ \sum\limits_{k/C_{k3}=1}\alpha_{ik}\geq 1\right)\\ \text{OR}\left(\sum\limits_{k/P_{k1}=1}\alpha_{ik}\geq 1\ \text{ AND }\ \sum\limits_{k/P_{k2}=1}\alpha_{ik}\geq 1\right)\ \text{ THEN }\quad \delta_i=1\end{array}$$

$$\text{Ref.f}_{35}\Rightarrow \boxed{\begin{array}{l}\forall i:\lambda_{i1}=1\quad \text{IF AND ONLY IF}\ \sum\limits_{k/C_{k2}=1}\alpha_{ik}\geq 1\\ \forall i:\lambda_{i2}=1\quad \text{IF AND ONLY IF}\ \sum\limits_{k/C_{k3}=1}\alpha_{ik}\geq 1\\ \forall i:\lambda_{i3}=1\quad \text{IF AND ONLY IF}\ \sum\limits_{k/P_{k1}=1}\alpha_{ik}\geq 1\\ \forall i:\lambda_{i4}=1\quad \text{IF AND ONLY IF}\ \sum\limits_{k/P_{k2}=1}\alpha_{ik}\geq 1\end{array}}$$

$$\Rightarrow \text{Ref.f}_{25}\Rightarrow \boxed{\begin{array}{l}\forall i:\sum\limits_{k/C_{k2}=1}\alpha_{ik}\geq\lambda_{i1}\\ \forall i:\sum\limits_{k/C_{k2}=1}\alpha_{ik}\leq K_i\lambda_{i1}\\ \forall i:\sum\limits_{k/C_{k3}=1}\alpha_{ik}\geq\lambda_{i2}\\ \forall i:\sum\limits_{k/C_{k3}=1}\alpha_{ik}\leq K_i\lambda_{i2}\end{array}} \tag{7.39}$$

$$\begin{array}{l}\forall i:\sum\limits_{k/P_{k1}=1}\alpha_{ik}\geq\lambda_{i3}\\ \forall i:\sum\limits_{k/P_{k1}=1}\alpha_{ik}\leq K_i\lambda_{i3}\\ \forall i:\sum\limits_{k/P_{k2}=1}\alpha_{ik}\geq\lambda_{i4}\\ \forall i:\sum\limits_{k/P_{k2}=1}\alpha_{ik}\leq K_i\lambda_{i4}\end{array} \tag{7.40}$$

$$\text{Ref. f}_{37}\Rightarrow\ \forall i:\text{IF }(\lambda_{i1}+\lambda_{i2}\geq 2)\ \text{ OR }(\lambda_{i3}+\lambda_{i4}\geq 2)\ \text{ THEN }\quad \delta_i=1$$

$$\text{Ref.f}_{35}\Rightarrow \begin{array}{l}\forall i:(\lambda_{i1}+\lambda_{i2}\geq 2)\ \text{ IF AND ONLY IF }\ \pi_{i1}=1\\ \forall i:(\lambda_{i3}+\lambda_{i4}\geq 2)\ \text{ IF AND ONLY IF }\ \pi_{i2}=1\end{array}$$

$$\Rightarrow \text{Ref.f}_{25} \Rightarrow \boxed{\begin{array}{l} \forall i: \lambda_{i1} + \lambda_{i2} \geq 2\pi_{i1} \\ \forall i: \lambda_{i1} + \lambda_{i2} \leq (1 - \pi_{i1}) + 2\pi_{i1} \\ \forall i: \lambda_{i3} + \lambda_{i4} \geq 2\pi_{i2} \\ \forall i: \lambda_{i3} + \lambda_{i4} \leq (1 - \pi_{i2}) + 2\pi_{i2} \end{array}} \tag{7.41}$$

$\Rightarrow$Ref. $f_{34} \Rightarrow\ \forall\, i$: IF $\pi_{i1} + \pi_{i2} \geq 1$ THEN $\delta_i = 1$
$\Rightarrow$Ref. $f_3 \Rightarrow\ \forall\, i$: IF $\delta_i = 0$ THEN $\pi_{i1} + \pi_{i2} \leq 0$

$$\text{Ref.f}_7 \Rightarrow \forall i: \text{IF } 1 - \delta_i = 1 \text{ THEN } \pi_{i1} + \pi_{i2} \leq 0$$
$$\Rightarrow \boxed{\forall i: \pi_{i1} + \pi_{i2} \leq 2\delta_i} \tag{7.42}$$

O.F.: Min $\sum_{i=1}^{30} \delta_i$

Collective Version

It is easy to organize guests into six different groups (Table 7.19).

Within each group, there is a set of identical items that do not have any specification or calculation on each individually in the system.

Obviously, as I know the guests, the values N_1–N_6 are known. For the table of elements, we use a set with those six groups of guests.

Table of Elements (Table 7.20)

Decision Activities

Action: Sitting guests at tables.
Decision variables: x_{ik} = Number of guests of group k sitting at table i.

Specifications

1. Implicit Specifications

I1. Based on data:

I1.1. Table capacity:

$$\forall i: \quad \sum_{k=1}^{6} x_{ik} \leq K_i \tag{7.43}$$

I1.2. The new data of existence of each group of guests gives rise to an assignment specification based on an action of determined value, since it is necessary to seat the invited Nk of each group:

Table 7.19 Subgroups of guests in Illustration 7.6

Guests					
Groom's guest			Bride's guest		
Children	Young	Seniors	Children	Young	Seniors
$1\ldots N_1$	$1\ldots N_2$	$1\ldots N_3$	$1\ldots N_4$	$1\ldots N_5$	$1\ldots N_6$

Table 7.20 Elements of Illustration 7.6 – Collective Version

			Data				
Elements	Set	QN	Name	Param	Type	Belonging	Value
Tables	$i = 1\ldots30$	I_U	Capacity	K_i	I	S	–
Seats	–	C_D		K_i			
Groups of guests	$k = 1\ldots6$	C_D	Number of guests	N_k	I	W	–
			Origin	P_{kp}	B	S	–
			Age category	C_{kc}	B	S	–
Origins	$p = 1,2$	I_U		P_{kp}			
Age categories	$c = 1,2,3$	I_U		C_{kc}			

$$\forall k: \quad \sum_{i=1}^{30} x_{ik} = N_k \tag{7.44}$$

I2. Quantitative selection rules: they do not exist.
I3. Logical conditions between activities: they do not exist.
I4. Bounds of discrete measurable activities: they do not exist.
I5. Flow balance constraints: they do not exist.

2. Explicit Specifications

E1. "Guests who are children have to be placed at tables just for them."

This logical proposition is stated with guest numerals: There cannot be a positive number of non-child guests at a table with a positive number of children.

Therefore, the proposition falls on any table, the two groups of children and the four groups that are not children:

$$\forall i, k/C_{k1} = 1, k'/C_{k'1} = 0 : \text{NOT}(x_{ik} > 0 \ \text{ AND } \ x_{ik'} > 0) \Rightarrow$$
$$\Rightarrow \text{Ref.f}_5 \Rightarrow \forall i, k/C_{k1} = 1, k'/C_{k'1} = 0 : \text{NOT}(x_{ik} \geq 1 \ \text{ AND } \ x_{ik'} \geq 1)$$
$$\Rightarrow \text{Ref.f}_{35} \Rightarrow \begin{array}{l} \forall i, k/C_{k1} = 1 : x_{ik} \geq 1 \ \text{ IF AND ONLY IF } \ \beta_{ik} = 1 \\ \forall i, k'/C_{k'1} = 0 : x_{ik'} \geq 1 \ \text{ IF AND ONLY IF } \ \delta_{ik'} = 1 \end{array}$$

$$\Rightarrow \left. \begin{array}{l} \forall i,k/C_{k1}=1: x_{ik} \geq \ \beta_{ik} \\ \forall i,k/C_{k1}=1: x_{ik} \leq K_i\beta_{ik} \\ \forall i,k'/C_{k'1}=0: x_{ik'} \geq \ \delta_{ik'} \\ \forall i,k'/C_{k'1}=0: x_{ik'} \leq \ K_i\delta_{ik'} \end{array} \right| \qquad (7.45)$$

$$\Rightarrow \text{Ref.f}_{37} \Rightarrow \forall i,k/C_{k1}=1,k'/C_{k'1}=0: \text{NOT}(\beta_{ik}+\ \delta_{ik'} \geq 2)$$

$$\Rightarrow \text{Ref.f}_{11} \Rightarrow \forall i,k/C_{k1}=1,k'/C_{k'1}=0: \beta_{ik}+\ \delta_{ik'} \leq 1 \qquad (7.46)$$

E2. "If a table is used, it must have at least 6 people seated at it."

As stated in the unitary version:

"If there are guests seated at a table, there must be at least 6."

$$\forall i: \text{IF} \ \sum_{k=1}^{6} x_{ik} > 0 \quad \text{THEN} \ \sum_{k=1}^{6} x_{ik} \geq 6$$

$$\Rightarrow \text{Ref.f}_5 \Rightarrow \forall i: \text{IF} \ \sum_{k=1}^{6} x_{ik} \geq 1 \ \text{THEN} \ \sum_{k=1}^{6} x_{ik} \geq 6 \Rightarrow$$

$$\Rightarrow \text{Ref.f}_{18,}\text{f}_{21} \Rightarrow \forall i: \sum_{k=1}^{6} x_{ik} \geq \omega_i \qquad (7.47)$$

$$\Rightarrow \forall i: \sum_{k=1}^{6} x_{ik} \leq k_i\omega_i \qquad (7.48)$$

$$\Rightarrow \forall i: \sum_{k=1}^{6} x_{ik} \geq 6\omega_i \qquad (7.49)$$

ω_i is the logical calculation that responds to whether a table has been used. The group of constraints (7.49) makes the group (7.47) unnecessary.

Objective Criterion

Minimize mixed tables.

To establish the expression of the objective function, we need to find out if each table has been mixed, that is, if it penalizes in the objective function.

Binary Logical Calculation: Mixed table.
Applied to: Tables $i = 1\ldots30$.
Variables:

$$\delta_i = \begin{cases} 1 \text{ if table } i \text{ is mixed} \\ 0 \ \text{ otherwise} \end{cases}$$

Logical Propositions:

$$\forall i: \text{IF}\left(\sum_{k/C_{k2}=1} x_{ik} > 0 \ \text{AND} \sum_{k/C_{k3}=1} x_{ik} > 0\right) \quad [\text{Ref. S}_\text{V}]$$
$$\text{OR}\left(\sum_{k/P_{k1}=1} x_{ik} > 0 \ \text{AND} \sum_{k/P_{k2}=1} x_{ik} > 0\right) \ \text{THEN} \quad \delta_i = 1$$

Proposition modelling:

$$\Rightarrow \text{Ref.f}_5 \Rightarrow\Rightarrow \begin{array}{l} \forall i: \text{IF}\left(\sum_{k/C_{k2}=1} x_{ik} \geq 1 \ \text{AND} \sum_{k/C_{k3}=1} x_{ik} \geq 1\right) \\ \text{OR}\left(\sum_{k/P_{k1}=1} x_{ik} \geq 1 \ \text{AND} \sum_{k/P_{k2}=1} x_{ik} \geq 1\right) \ \text{THEN} \quad \delta_i = 1 \end{array}$$

$$\Rightarrow \text{Ref.f}_{35} \Rightarrow \boxed{\begin{array}{l} \forall i: \lambda_{i1} = 1 \quad \text{IF AND ONLY IF} \ \sum_{k/C_{k2}=1} x_{ik} \geq 1 \\ \forall i: \lambda_{i2} = 1 \quad \text{IF AND ONLY IF} \ \sum_{k/C_{k3}=1} x_{ik} \geq 1 \\ \forall i: \lambda_{i3} = 1 \quad \text{IF AND ONLY IF} \ \sum_{k/P_{k1}=1} x_{ik} \geq 1 \\ \forall i: \lambda_{i4} = 1 \quad \text{IF AND ONLY IF} \ \sum_{k/P_{k2}=1} x_{ik} \geq 1 \end{array}}$$

$$\Rightarrow \text{Ref.f}_{25} \Rightarrow \boxed{\begin{array}{l} \forall i: \sum_{k/C_{k2}=1} x_{ik} \geq \lambda_{i1} \\ \forall i: \sum_{k/C_{k2}=1} x_{ik} \leq K_i\lambda_{i1} \\ \forall i: \sum_{k/C_{k3}=1} x_{ik} \geq \lambda_{i2} \\ \forall i: \sum_{k/C_{k3}=1} x_{ik} \leq K_i\lambda_{i2} \end{array}} \tag{7.50}$$

$$\begin{array}{l} \forall i: \sum_{k/P_{k1}=1} x_{ik} \geq \lambda_{i3} \\ \forall i: \sum_{k/P_{k1}=1} x_{ik} \leq K_i\lambda_{i3} \\ \forall i: \sum_{k/P_{k2}=1} x_{ik} \geq \lambda_{i4} \\ \forall i: \sum_{k/P_{k2}=1} x_{ik} \leq K_i\lambda_{i4} \end{array} \tag{7.51}$$

$$\Rightarrow \text{Ref. f}_{37} \Rightarrow \ \forall\, i: \text{IF}\ (\lambda_{i1} + \lambda_{i2} \geq 2)\ \text{OR}\ (\lambda_{i3} + \lambda_{i4} \geq 2) \quad \text{THEN} \quad \delta_i = 1$$

$$\Rightarrow \text{Ref.f}_{35} \Rightarrow \begin{array}{l} \forall i : (\lambda_{i1} + \lambda_{i2} \geq 2) \quad \text{IF AND ONLY IF} \quad \pi_{i1} = 1 \\ \forall i : (\lambda_{i3} + \lambda_{i4} \geq 2) \quad \text{IF AND ONLY IF} \quad \pi_{i2} = 1 \end{array}$$

$$\Rightarrow \text{Ref.f}_{25} \Rightarrow \boxed{\begin{array}{l} \forall i : \lambda_{i1} + \lambda_{i2} \geq 2\pi_{i1} \\ \forall i : \lambda_{i1} + \lambda_{i2} \leq (1 - \pi_{i1}) + 2\pi_{i1} \\ \forall i : \lambda_{i3} + \lambda_{i4} \geq 2\pi_{i2} \\ \forall i : \lambda_{i3} + \lambda_{i4} \leq (1 - \pi_{i2}) + 2\pi_{i2} \end{array}} \qquad (7.52)$$

$\Rightarrow$Ref. f$_{34}$ $\Rightarrow$ $\forall\ i$: IF π_{i1} + $\pi_{i2} \geq 1$ THEN $\delta_i = 1$
$\Rightarrow$Ref. f$_3$ $\Rightarrow$ $\forall\ i$: IF $\delta_i = 0$ THEN π_{i1} + $\pi_{i2} \leq 0$

$$\text{Ref.f}_7 \Rightarrow \forall i : \text{IF } 1 - \delta_i = 1 \ \text{ THEN } \quad \pi_{i1} + \pi_{i2} \leq 0$$
$$\Rightarrow \boxed{\forall i : \pi_{i1} + \pi_{i2} \leq 2\delta_i} \qquad (7.53)$$

F.O.: Min $\sum_{i=1}^{30} \delta_i$

Let us take a look at a comparison of both models (Table 7.21).

Constraints considered:

Individual Version: 7.30, 7.31, 7.32, 7.36, 7.37, 7.39, 7.40, 7.41, 7.42

Collective Version: 7.43, 7.44, 7.45, 7.46, 7.48, 7.49, 7.50, 7.51, 7.52, 7.53

The collective version is much more efficient with respect to the size than the individual version, especially in the number of variables.

7.7 Individual Elements with Capacity to Be Grouped in Collective Elements: Through Small Changes in the Data Values

These are systems in which it is permissible to make small variations in the data, in order to simplify the problem. With this we can make groups of elements as items of a collective element. It may also be necessary to make small changes in the statement to avoid referring to each item in particular and to do so by means of numerals of the collective element. Let us see an illustration that groups the citizens of a city according to their address, assuming identical distances of citizens who live in the same street or stretch of street. Related problems in Wang et al. (2018)

Illustration 7.7: Allocation of Health Centers to Citizens

There is a city formed by five health centers. Due to changes in locations, it has been decided to restructure the allocation of health centers to its 50,000 citizens. We know

Table 7.21 Comparison of version sizes in Illustration 7.6

	Individual Version	Collective Version
Number of binary variables	6240	420
Number of integer variables	0	180
Number of constraints	680+ (30 × N° of children)	606

the address of each citizen, and therefore, the system allows us to know the distance from his address to each health center. Each health center has a capacity that is expressed as the number of citizens that can be attended to per day. It is estimated that 1% of people go to the doctor daily. The objective is to minimize the sum of the distances from each citizen address to the assigned health center.

In the statement each citizen must be considered as an individual element of the system, since each one will have a distance from each health center, without there being many coincidences. On the other hand, they are unitary because they do not have measurable data or properties. In the second version, we will group citizens who share a common area, such as a street or a section of street. These citizens will therefore have the same values of distance to health centers.

Individual Version

The table of elements would be as follows:

Table of Elements (Table 7.22)

In the definition of data, we logically ignore the address, since it is not a numeric attribute.

Decision Activities

Action: Allocate citizens to health centers.
Decision variables:
α_{ij}: 1 if I allocate citizen i to health center j; 0 otherwise.
$i = 1...50.000; j = 1...5$.

Specifications

1. *Implicit Specifications*

I1. Based on data: Capacity specification of each health center (although it can also be considered as explicit by the description of the capacity).

$$\forall j: \quad \sum_{i=1}^{50000} 0,01\alpha_{ij} \leq K_j \tag{7.54}$$

I2. Quantitative selection rules: Variables α_{ij} define quantitative selection rules (Table 7.23).

Table 7.22 Elements of Illustration 7.7 – Individual version

			Data				
Elements	Set	QN	Name	Param	Type	Belonging	Value
Citizens	$i = 1\ldots50.000$	I_U	Distance	D_{ij}	C	S	–
Health centers	$j = 1\ldots5$	I_U	Capacity	K_j	I	W	–
				D_{ij}			

Table 7.23 Selection rules in Illustration 7.7

Activity	Elements selecting	Selectable elements	Selection	Constraints
Allocate	Citizens $i = 1\ldots50{,}000$	Health centers	$= 1$	$\forall i: \quad \sum_{j=1}^{5} \alpha_{ij} = 1$ (7.55)
	Health centers $j = 1\ldots5$	Citizens	$\leq K_i$	I1

I3. Logical conditions between activities: they do not exist.
I4. Bounds of discrete measurable activities: they do not exist.
I5. Flow balance constraints: they do not exist.

2. Explicit Specifications

They do not exist.

Objective Criterion

$$\text{O.F.: MIN} \sum_{i=1}^{50000} \sum_{j=1}^{5} D_{ij}\alpha_{ij}$$

Collective Version

In the collective version, we assign the same distance value to the citizens who live in the same area, street or street section.

This version requires a slight modification of the statement, to stop referring to each citizen in particular. We can refer in particular to each group of citizens, but we must avoid referring to each citizen individually.

On the other hand, we need to assign an attribute to each group of citizens with the number of citizens it has. Assuming that a total of m groups has been formed, we will use the data N_i, $i = 1\ldots$m to name the number of citizens of each group.

The statement would look like this:

There is a city formed by five health centers. Due to changes in locations, it has been decided to restructure the allocation of health centers to its 50,000 citizens. We know the address of each citizen, and therefore, the system allows us to know the distance from his area to each health center. Each health center has a capacity that is expressed as the number of citizens that can be attended to per day. It is estimated that 1% of people go to the doctor daily. The objective is to minimize the sum of the distances from citizens to the assigned health centers.

Table of Elements (Table 7.24)

Table 7.24 Elements of Illustration 7.7 – Collective Version

			Data				
Elements	Set	QN	Name	Param	Type	Belonging	Value
Groups of citizens	$i = 1 \ldots m$	C_D	Distance	D_{ij}	C	S	–
			Quantity	N_i	I	W	–
Health centers	$j = 1 \ldots 5$	I_U	Capacity	K_j	I	W	–
				D_{ij}			

Table 7.25 Elements of Illustration 7.7 – Collective Version with continent and content elements

			Data				
Elements	Set	QN	Name	Param	Type	Belonging	Value
Group of citizens	$i = 1 \ldots m$	I_U	Distance	D_{ij}	C	S	–
			Quantity	N_i	I	S	
Health centers	$j = 1 \ldots 5$	I_U	Capacity	K_j	I	S	–
				D_{ij}			
Citizens	–	C_D		K_j, N_i			

In the table of elements, we have considered each group of citizens as a collective, and we have introduced them all in a set of m elements. This scenario, as in Illustration 7.2 on elements where we could make the content and the continent explicit, also allows a table of elements in which we consider each group as unitary (continent) and the citizen as collective (content). In that case, the table would have the following form (Table 7.25).

In this table we have grouped the content in a single collective element since the citizens are identical with respect to their own data. The capacity attribute of each health center is shared with the Citizens element. This integer attribute had been own, while there was no need to create the collective element Citizens. Either table is valid to represent the elements of the problem.

Decision Activities

Action: Allocate groups of citizens to Health centers.
Decision variables:
x_{ij}: Number of citizens of group i allocated to Health center j.

Specifications

1. Implicit Specifications

I1. Based on data:

I1.1. Capacity of each health center: We keep it as implicit. Now the activity is x_{ij}.

$$\forall j: \quad \sum_{i=1}^{m} 0,01 x_{ij} \leq K_j \tag{7.56}$$

I1.2. The citizen stock data of each group generates a balance specification:

$$\forall i: \quad \sum_{j=1}^{5} x_{ij} = N_i \tag{7.57}$$

I2. Quantitative selection rules: they do not exist.
I3. Logical conditions between activities: they do not exist.
I4. Bounds of discrete measurable activities: they do not exist.
I5. Flow balance constraints: they do not exist.

2. Explicit Specifications

They do not exist.

Objective Criterion

O.F.: MIN $\sum_{i=1}^{m} \sum_{j=1}^{5} D_{ij} x_{ij}$

Comparison of both models (Table 7.26).

7.8 Items of Indeterminate Collective Elements that Need to Be Defined Individually

According to most of the illustrations seen so far, if a system has an element with a number of indeterminate items, that element is considered indeterminate collective, by default. However, if the system description refers individually to the possible items that the element will have, it is necessary to consider each possible item as an individual element. This means that the collective element must be converted into a set of individual elements. As we do not know how many items we can have, from the description of the system, it is necessary to calculate an upper bound of the number of items, keeping it as small as possible to avoid an unnecessary increase in the number of variables that can be generated in the problem.

By slightly modifying statement 7.6, we can illustrate this case.

Illustration 7.8: Illustration 7.6 with an Indeterminate Number of Tables
There is a celebration hall that should place the guests of a wedding. The living room can use tables of two sizes (tables of 8 seats and tables of 12 seats).

Table 7.26 Comparison of versions of Illustration 7.7

	Individual Version	Collective Version
Number of binary variables	250.000	0
Number of integer variables	0	5 m
Number of constraints	50.005	m + 5

200 guests attend the wedding, including seniors, young people, and children. The guests are also classified as the groom's guest or the bride's guest. Therefore, each guest has the information of age category and origin.

It has been decided that guests who are children must be placed at tables just for them. However, older people and young people can mix at the tables, but what is intended is to have the least number of tables where older and younger people mix (i.e., if at a table they seat young and old people, that table penalizes in the objective function).

On the other hand, if a table is used, it must have at least six people seated at it.

Regarding the groom's guests and bride's guests, we also intend to mix as little as possible at the wedding. In this way, in the objective function we will minimize the tables where both the mix of older and younger people will be produced, as groom's guests and bride's guests, but if in a table there are both circumstances, that table only penalizes once, not twice.

The modification we have made has been to disregard a certain number of tables, both 8 and 12 seats. Therefore, the tables become indeterminate, which could lead us to visualize each type of table as an indeterminate collective element. However, as in the statement there are several specifications that allude to (tables) individual items (if a table is used, it must have at least six people seated at it, if at a table they seat young and old people, that table penalizes in the objective function), it is necessary to keep the tables as individual elements. The problem is that its number is indeterminate; therefore it is necessary to calculate an upper bound of the number of tables of 8 and 12 people that could be used without losing possible solutions to the problem. If we have 200 guests, the maximum number of tables of 8 that could be used is obtained by rounding the quotient between the number of guests and the number of places of that type of tables: $\left\lceil \frac{200}{8} \right\rceil = 25$tables. The same applies to tables of 12 places: $\left\lceil \frac{200}{12} \right\rceil = 17$tables. In this way, the table of elements for the individual guest version would have a total of 42 tables:

Table of Elements (Table 7.27)

Let us take a look at another illustration to finish the chapter.

Illustration 7.9

There is a set of n tasks that must be processed in a machining factory. The processing is done in a machine model for which we must subcontract units. Each task has a process time and is processed completely on the same machine. The

Table 7.27 Elements of Illustration 7.8

Elements	Set	QN	Data				
			Name	Param	Type	Belonging	Value
Tables	$i = 1\ldots42$	I_U	Capacity	K_i	I	S	–
Seats	–	C_D		K_i			
Guests	$k = 1\ldots200$	I_U	Origin	P_{kp}	B	S	–
			Age category	C_{kc}	B	S	–
Origins	$p = 1{,}2$	I_U		P_{kp}			
Age categories	$c = 1{,}2{,}3$	I_U		C_{kc}			

Table 7.28 Elements of Illustration 7.9

Elements	Set	QN	Data				
			Name	Param	Type	Belonging	Value
Factory	–	I_U	Time limit	TL	C	W	–
Tasks	$i = 1\ldots n$	I_U	Process time	t_i	C	W	–
Machines	$j = 1\ldots n$	I_U					

objective of the problem is to minimize the number of machines needed to process all the tasks in a given time limit.

The machines could be considered as an indeterminate collective element, but the system needs its individual consideration, since it is necessary to know in which machine each task is processed. Therefore, an upper bound of its number is necessary. This level can be obtained in various ways. The simplest is to use as many machines as jobs. The tasks are processed completely in the same machine, so they must be considered non-divisible and therefore unitary.

Table of Elements (Table 7.28)

References

Arkin, E. M., & Silverberg, E. B. (1987). Scheduling jobs with fixed start and end times. *Discrete Applied Mathematics, 18*, 1–8.

Kroon, L. G., Salomon, M., & Van Wassenhove, L. N. (1995). Exact and approximation algorithms for the operational fixed interval scheduling problem. *European Journal of Operational Research, 82*, 190–205.

Lewis, R., & Carroll, F. (2016). Creating seating plans: A practical application. *Journal of the Operational Research Society, 67*, 1353–1362.

Wang, L., Huan, S., & Lu, G. (2018). Healthcare facility location-allocation optimization for China's developing cities utilizing a multi-objective decision support approach. *Sustainability, 10*, 4580.

Chapter 8
Practical Examples

8.1 Production with Fixed Costs

A company is dedicated to the manufacture of steel sheets. The manufacturing of the material is composed of three phases: Fusion of Steel (R1), Format (R2), and Cooling (R3). Three types of sheets are manufactured (P1, P2, and P3).

The consumption of time required to manufacture each sheet in each phase is indicated in the table, as well as the total time available for each phase. The marginal benefits obtained by each unit of the products are indicated in the table. But there is also a fixed cost that is incurred if units of a product are manufactured (e.g., if units of P2 are manufactured, a cost of $500 is incurred regardless of the number of units manufactured); if not manufactured, the cost is zero. The objective is to maximize profits.

Table: System data

	P1	P2	P3	
Profit/unit	35	50	40	
Fixed cost	1000	500	1500	Time available
R1	2	3	6	500
R2	7	2	3	400
R3	4	5	2	300

Table of Elements (Table 8.1)

Decision Activities

Action: Produce sheets.
Decision variables: x_j = Number of sheets j produced.

J. M. García Sánchez, *Modelling in Mathematical Programming*, International Series in Operations Research & Management Science 298,
https://doi.org/10.1007/978-3-030-57250-1_8

Table 8.1 Elements of example 8.1

Elements	Set	QN	Data				
			Name	Param	Type	Belonging	Value
Phases	$i = 1\ldots3$	I_U	Time available	D_i	C	W	–
			time phase-sheet	t_{ij}	C	S	–
Sheets	$j = 1\ldots3$	C_D	Profit	B_j	C	W	–
			Fixed cost	C_j	C	W	–
				t_{ij}			

Specifications

1. *Implicit Specifications*

   I1. Based on data:

   I1.1. Time availability of each phase: specification of capacity consumption.

$$\forall i: \quad \sum_{j=1}^{3} t_{ij}x_j \leq D_i$$

   I2. Quantitative selection rules: they do not exist.
   I3. Logical conditions between activities: they do not exist.
   I4. Bounds of discrete measurable activities: they do not exist.
   I5. Flow balance constraints: they do not exist.

2. *Explicit Specifications*
   Not considered

Objective Criterion
Maximize Total Profit – Costs

$$\text{Total Profit} = \sum_{j=1}^{3} I_j x_j$$

Costs: The statement mentions that fixed costs are incurred in the case of producing units of each sheet. It is signalling a conditioned action, incurring fixed costs, which corresponds to a logical calculation:

Binary logical Calculation: Incur in fixed cost.
Applied to: Each sheet j.
Variables:
$\omega_j = 1$ if fixed cost is incurred for the production of sheets j; 0 otherwise.
Logical Proposition:
$\forall j : \omega_j = 1 \leftrightarrow x_j > 0 \Rightarrow \text{Ref } S_V \Rightarrow \forall j : \text{IF } x_j > 0 \text{ THEN } \omega_j = 1$
Proposition modelling:
$\Rightarrow \text{Ref. } f_3 \Rightarrow \forall j : \text{IF } \omega_j = 0 \text{ THEN } x_j \leq 0$
$\Rightarrow \text{Ref. } f_{14} \Rightarrow \forall j : x_j \leq \text{CS}_{x_j}\ \omega_j$

The cost expression would be: $\text{Costs} = \sum_{j=1}^{3} C_j \omega_j$

8.2 Graph Coloring Problem [Jensen and Toft (1995)]

There is a map with a set of n regions. We have four markers or different colors to paint the regions. We must paint each region with a color so that two adjacent regions are not painted with the same color. The objective is to minimize the number of colors used.

Table of Elements (Table 8.2)

The table of elements reflects the colors as individual and unitary although we do not reflect any data that distinguishes them. The statement alludes in particular to each color: "two adjacent regions are not painted with the same color."

In the phrase we do not refer to a number of colors, but to any color individually:

"Two adjacent regions are not painted with the color $j = 1$"
"Two adjacent regions are not painted with the color $j = 2$"
...

On the other hand, the adjacency between pairs of regions is known attribute that is represented in binary form: $a_{ik} = 1$ if the region i and k are adjacent; 0 if not.

Decision Activities

Action: Painting regions with colors.
Decision variables: $\alpha_{ij} = 1$ if I paint region i with color j; 0 otherwise.

Specifications

1. *Implicit Specifications*

I1. Based on data: they do not exist.
I2. Quantitative selection rules: there are logical activities, so it is necessary to analyze the quantitative selection rules. However, the only norm is explicitly described in the statement ("We have to paint each region with a color") (Table 8.3).

Table 8.2 Elements of example 8.2

			Data				
Elements	Set	QN	Name	Param	Type	Belonging	Value
Regions	$i, k = 1...n$	I_U	Adjacency	a_{ik}	B	S	...
Colors	$j = 1...4$	I_U					

Table 8.3 Scheme of selection of the activity Painting regions with colors

Activity	Elements selecting	Selectable elements	Selection	Constraints
Paint	Colors $j = 1...4$	Regions	–	
	Regions $i = 1...n$	Colors	=1	E1

I3. Logical conditions between activities: they do not exist.
I4. Bounds of discrete measurable activities: they do not exist.
I5. Flow balance constraints: they do not exist.

2. *Explicit Specifications*

E1. "We have to paint each region with a color."

$$\forall i: \quad \sum_{j=1}^{4} \alpha_{ij} = 1$$

E2. "Two adjacent regions are not painted with the same color."

It is an imposition of negation that is constructed by propositional logic. The specification refers to each adjacent pair of regions *i*, *k* and to any color. Therefore:

$\forall j, \forall\ i, k/a_{ik} = 1: \quad \text{NOT}(\alpha_{ij} = 1 \text{ AND } \alpha_{kj} = 1)$
$\Rightarrow$ Ref $f_{37} \Rightarrow \forall j, \forall\ i, k/a_{ik} = 1: \quad \text{NOT}(\alpha_{ij} + \alpha_{kj} \geq 2)$
$\Rightarrow$ Ref $f_{11} \Rightarrow \forall j, \forall\ i, k/a_{ik} = 1: \quad \alpha_{ij} + \alpha_{kj} \leq 1$

Objective Criterion

Minimize the number of colors used.

Colors used ⇒ Since the colors are individual, we have to analyze it individually ⇒ color used ⇒ information that is not found in any variable and that therefore is a calculation to be made. It is a conditioned characteristic associated with each color (whether it has been used or not):

Binary logical calculation: Color used.
Applied to: Each color *j*.
Variables: $\beta_j = 1$ if color *j* is used; 0 otherwise.
Logical proposition:

$$\forall j: \beta_j = 1 \quad \text{IF AND ONLY IF} \quad \sum_{i=1}^{n} \alpha_{ij} \geq 1$$

$$\Rightarrow \text{Ref } S_V \Rightarrow \forall j: \text{IF } \sum_{i=1}^{n} \alpha_{ij} \geq 1 \text{ THEN } \beta_j = 1$$

Proposition Modelling:

$$\Rightarrow \text{Ref. } f_3 \Rightarrow \forall j: \text{IF } \beta_j = 0 \text{ THEN } \sum_{i=1}^{n} \alpha_{ij} < 1$$

$$\Rightarrow \text{Ref } f_{7,}\ f_4 \Rightarrow \forall j: \text{IF } 1 - \beta_j = 1 \quad \text{THEN } \sum_{i=1}^{n} \alpha_{ij} \leq 0$$

$$\Rightarrow \text{Ref. } f_{14} \Rightarrow \forall j: \sum_{i=1}^{n} \alpha_{ij} \leq n\ \beta_j$$

The objective function would finally be:

$$\text{Min} \quad \sum_{j=1}^{4} \beta_j$$

8.3 Configuration of Work Centers

It is a company that manufactures a part model for the automotive industry. The company has designed ten work centers in its plant. The manufacture of the parts has two possibilities or modes of production:

Mode 1: the work center needs an M1 machine, an H1 tool and two H2 tools.
Mode 2: the work center needs an M2 machine, an M3 machine, an H2 tool and an H3 tool.

The company must decide on the production mode that is installed in each center. Work centers 1, 2, and 3 can accommodate any mode, while the rest only mode 1.

The purchase costs of each type of machine and tool are the following:

Element	Cost
Machine M1	\$$Cm_1$
Machine M2	\$$Cm_2$
Machine M3	\$$Cm_3$
Tool H1	\$$Ch_1$
Tool H2	\$$Ch_2$
Tool H3	\$$Ch_3$

The start-up of a Center has a cost of \$F/year. The production cost of each mode is \$C1/part and \$C2/part. The production obtained with each mode is K1 parts/day and K2 parts/day, respectively. The year has 260 working days.

The M1 machine emits too many gases, so if its number is greater than 3, it is necessary to buy a gas heatsink valued at \$E.

On the other hand, due to acoustic issues, centers 3 and 4 cannot simultaneously hold Mode 1, if centers 1 and 2 hold Mode 2.

The company must value both productivity and costs for its new structure. For this, it establishes on the one hand a production goal of 10,000 parts per year and an annual objective of minimizing costs, considering:

- *An annual amortization of 10% for purchase costs.*
- *The tool provider applies a discount of 15%/tool if more than 30 total tools are purchased.*
- *The machine supplier applies a bonus of \$b if a total of at least 20 machines are purchased.*

Table of Elements (Table 8.4)

In the table of elements, almost all the elements and numerical data of the statement have been reflected. As we discussed in Chap. 3, it is not necessary for the methodology to represent all the data or elements, only the elements that are going to be alternatives for the decision activities, or the data associated with sets of elements.

Table 8.4 Elements of Example 8.3

			Data				
Elements	Set	QN	Name	Param	Type	Belonging	Value
Centers	$i = 1\ldots10$	I_U	Compatibility	A_{ij}	B	S	-
			Cost	F_i	C	W	€F
Modes	$j = 1,2$	I_U		A_{ij}			
			Cost	C_j	C	S	-
			Production	P_j	I	S	-
			Number of machines	MM_{jm}	I	S	-
			Number of tools	HM_{jh}	I	S	-
Machines	$m = 1,2,3$	C_I	Cost	CM_m	C	W	-
				MM_{jm}			
Tools	$h = 1,2,3$	C_I	Cost	CH_h	C	W	-
				HM_{jh}			
Parts	–	C_D	Goal	G	I	W	10.000
				C_j ; P_i			
Gas heatsink	–	I_U	Cost	CD	C	W	€E
Tool provider	–	I_U	Discount	D	C	W	0.15
			Tools threshold	UH	I	W	31
Machine provider	–	I_U	Bonus	B	C	W	€b
			Machines threshold	UM	I	W	20
Year	–	I_U	Working days	WD	I	W	260

The system will have a gas heatsink or none; therefore it can be considered as an indeterminate collective with a maximum quantity of one, or simply unitary.

We could have used a smaller table in which we did not explicitly specify the parts, the gas heatsink, the providers, and the year, since they do not participate in any decision activity or do so implicitly. In that case, the data and any specification would be attributed to the system.

We can even convert the daily production of parts into annual production by multiplying the daily production by the number of working days and thereby eliminating the Working days attribute (Table 8.5).

Decision Activities

Action: Install Modes in Centers.
Decision variables: $\alpha_{ij} = 1$ if I install mode j in center i; 0 otherwise.

The purchase of machines and tools is not a decision activity in the system, but a calculation motivated by the installation of modes in centers. It is an auxiliary calculation because it is enunciated by an equality function. Machines and tools purchased:

Table 8.5 Reduced table of elements of Example 8.3

Elements	Set	QN	Data				
			Name	Param	Type	Belonging	Value
Centers	$i = 1...10$	I_U	Compatibility	A_{ij}	B	C	-
			Cost	F_i	C	P	\$F
Modes	$j = 1, 2$	I_U		A_{ij}			
			Cost	C_j	C	P	–
			Annual Production	AP_j	E	P	–
			Number of machines	MM_{jm}	E	C	–
			Number of tools	HM_{jh}	E	C	–
Machines	$m = 1,2,3$	C_I	Cost	CM_m	C	P	–
				MM_{jm}			
Tools	$h = 1,2,3$	C_I	Cost	CH_h	C	P	–
				HM_{jh}			
System	–	I_U	Parts	G	E	P	10.000
			Gas heatsink cost	SC	C	P	\$E
			Discount	Dto	C	P	0.15
			Tools threshold	UH	E	P	31
			Bonus	B	C	P	\$b
			Machines threshold	UM	E	P	20

$$\forall m : x_m = \sum_{i=1}^{10} \sum_{j=1}^{2} \mathrm{MM}_{jm}\alpha_{ij}$$

$$\forall h : y_h = \sum_{i=1}^{10} \sum_{j=1}^{2} \mathrm{HM}_{jh}\alpha_{ij}$$

The data MM_{jm} and HM_{jh} that contain the number of m machines and h tools that each mode needs, respectively, are used.

Specifications

1. *Implicit Specifications*

I1. Based on data: There is a demand attribute in the system, the production goal of 10.000 parts. It is necessary to formulate a specification of demand contribution that falls on the system:

$$\sum_{i=1}^{10} \sum_{j=1}^{2} P_j D\alpha_{ij} \geq 10.000$$

The contribution is made by the activity α_{ij}. The unit contribution corresponds to the parts that are manufactured per year with each mode (AP_j).

I2. Quantitative selection rules: Let's analyze the quantitative norms of selection of the activity α_{ij} (Table 8.6).

It seems correct to express a norm of at most one mode in each center. We must allow none to be installed because it is not necessary to obtain more production.

Table 8.6 Diagram of the decision activity of Installing modes in centers

Activity	Elements selecting	Selectable elements	Selection	Constraints
Install	Modes $j = 1,2$	Centers	–	
	Centers i=1...10	Modes	≤ 1	$\forall i : \sum_{j=1}^{2} \alpha_{ij} \leq 1$

Nowhere in the statement is there a mode installed in each center. On the other hand, it is understood that the installed mode remains throughout the year, since the system does not talk about mode changes in the same center. If so, it would be necessary to consider the property installation time as measurable in each center.

I3. Logical conditions between activities: they do not exist.
I4. Bounds of discrete measurable activities: they do not exist.
I5. Flow balance constraints: they do not exist.

2. *Explicit Specifications*

E1. "Work centers 1,2 and 3 can accommodate any mode, while the rest only Mode 1."

The phrase imposes restriction on centers 4 to 10. Centers 4 to 10 can only accommodate Mode 1. We propose the specification in a negative way: Centers 4 to 10 cannot accommodate Mode 2:

$\forall i = 4...10 : \quad \text{NOT}(\alpha_{i2} = 1) \Rightarrow \forall i = 4...10 : \quad \alpha_{i2} = 0$

E2. "The M1 machine emits too many gases, so if its number is greater than 3, it is necessary to buy a gas heatsink valued at \$E."

This sentence contains a logical proposition that defines a conditioned action, which is "buy a gas heatsink." We could have considered it within the definition of the objective function since the gas heatsink only supposes a purchase cost in the system.

Binary logical calculation: Buy gas heatsink.
Applied to: The system.
Variable: $\beta = 1$ if I buy gas heatsink; 0 otherwise.
Logical proposition:
$\forall j : \beta = 1$ IF AND ONLY IF $x_1 \geq 4 \Rightarrow$ Ref $S_V \Rightarrow$ IF $x_1 \geq 4$ THEN $\beta = 1$
Proposition modelling:
$\Rightarrow$ Ref. $f_3 \Rightarrow$ IF $\beta = 0$ THEN $x_1 < 4$
$\Rightarrow$ Ref $f_{7,}$ $f_4 \Rightarrow$ IF $1 - \beta = 1$ THEN $x_1 \leq 3$
$\Rightarrow$ Ref. $f_{14} \Rightarrow x_1 \leq 10 \ \beta$ [We have taken 10 as the upper bound of x_1, since there are 10 centers and one M1 machine is installed at most in a center].

E3. "Centers 3 and 4 cannot simultaneously hold Mode 1, if centers 1 and 2 hold Mode 2."

It is a specification stated as a logical proposition:

"If centers 1 and 2 hold Mode 2 then centers 3 and 4 cannot simultaneously hold Mode 1."

Mathematically:

IF ($\alpha_{12} = 1$ AND $\alpha_{22} = 1$) THEN NOT($\alpha_{31} = 1$ AND $\alpha_{41} = 1$)

Modelling:

$\Rightarrow$ Ref. $f_{37} \Rightarrow$ IF $\alpha_{12} + \alpha_{22} \geq 2$ THEN NOT($\alpha_{31} + \alpha_{41} \geq 2$)
$\Rightarrow$ Ref. $f_{11} \Rightarrow$IF $\alpha_{12} + \alpha_{22} \geq 2$ THEN $\alpha_{31} + \alpha_{41} \leq 1$

$$\alpha_{12} + \alpha_{22} \geq 2\omega$$

$\Rightarrow$ Ref. f_{18}, $f_{20} \Rightarrow$ $\alpha_{12} + \alpha_{22} \leq 1 + \omega$

$$\alpha_{31} + \alpha_{41} \leq 2 - \omega$$

Objective Criterion
Minimize costs $\Rightarrow$ Minimize purchase costs subject to amortization + Start-up cost of centers + Production costs − Discount for tool purchase − Bonus for the purchase of machines.
Purchase costs = Purchase of tools + Purchase of machines + Purchase of the heatsink

$$\textit{Purchase costs} = 0,1\left(\sum_{h=1}^{3} \mathrm{CH}_h y_h + \sum_{m=1}^{3} \mathrm{CM}_m x_m + \mathrm{SC}\beta\right)$$

$$\text{Start-up cost of centers} = \sum_{i=1}^{10} \sum_{j=1}^{2} F_i \alpha_{ij}.$$

A calculation was not necessary to know if a center had been put into operation. We directly use the α_{ij} variables that establish if a mode is installed, which is equivalent to putting the center into operation.

$$\text{Production cost}s = \sum_{i=1}^{10} \sum_{j=1}^{2} C_j \mathrm{AP}_j \alpha_{ij}$$

Discount for tool purchase**:** it is necessary to define a non-binary logical calculation that will directly collect the value of the discount.

Non-binary logical calculation: Apply discount.
Applied to: The system.
Variables: yD = Discount value.
Logic propositions:

$$\text{IF } \sum_{h=1}^{3} y_h > 30 \text{ THEN } yD = \sum_{h=1}^{3} 0,15 Ch_h y_h$$

$$\text{IF } \sum_{h=1}^{3} y_h \leq 30 \text{ THEN } yD = 0$$

Proposition modelling:

IF $\sum_{h=1}^{3} y_h > 30$ THEN $yD = \sum_{h=1}^{3} 0,15Ch_h y_h$

$\Rightarrow$ Ref. $f_5 \Rightarrow$ IF $\sum_{h=1}^{3} y_h \geq 31$ THEN $yD = \sum_{h=1}^{3} 0,15Ch_h y_h$

$\Rightarrow$IF $\sum_{h=1}^{3} y_h \geq 31$ THEN $yD - \sum_{h=1}^{3} 0,15Ch_h y_h = 0$

$\Rightarrow$ Ref. $f_{LB} \Rightarrow$ IF $\sum_{h=1}^{3} y_h \geq 31$ THEN $yD - \sum_{h=1}^{3} 0,15Ch_h y_h \leq 0$

$\Rightarrow$ Ref. $f_{18} \Rightarrow \sum_{h=1}^{3} y_h \geq 31\omega H_1$

$\Rightarrow$ Ref. $f_{18} \Rightarrow \sum_{h=1}^{3} y_h \leq 30(1 - \omega H\ \ ) + \mathrm{UB}_{\sum_{h=1}^{3} y_h} \omega H_1$

$\Rightarrow$ Ref. $f_{20} \Rightarrow yD - \sum_{h=1}^{3} 0,15Ch_h y_h \leq \mathrm{UB}_{y_D - \sum_{h=1}^{3} 0,15Ch_h y_h} (1 - \omega H_1)$

IF $\sum_{h=1}^{3} y_h \leq 30$ THEN $yD = 0 \Rightarrow$

$\Rightarrow$ Ref. $f_{LB} \Rightarrow$IF $\sum_{h=1}^{3} y_h \leq 30$ THEN $yD \leq 0$

$\Rightarrow$ Ref. $f_{17} \Rightarrow \sum_{h=1}^{3} y_h \leq 30 + \left(\mathrm{UB}_{\sum_{h=1}^{3} y_h} - 30 \right) (1 - \omega H_2)$

$\Rightarrow$ Ref. $f_{17} \Rightarrow \sum_{h=1}^{3} y_h \geq 31(1 - \omega H_2)$

$\Rightarrow$ Ref. $f_{20} \Rightarrow yD \leq \mathrm{UB}_{yD}(1 - \omega H_2)$

Bonus for the purchase of machines: it is also necessary to define a logical calculation, in this case binary because the bonus action has a certain value.

Binary logical calculation: Bonus purchase of machines.
Applied to: The system.
Variables: $\pi = 1$ if I apply bonus €b; 0 otherwise.
Logical proposition:

$$\pi = 1 \quad \leftrightarrow \quad \sum_{m=1}^{3} x_m \geq 20$$

Proposition modelling:
$\Rightarrow$ Ref. $f_{25} \Rightarrow$

$$\sum_{m=1}^{3} x_m \geq 20\pi$$

$$\sum_{m=1}^{3} x_m \leq 19(1 - \pi) + \mathrm{UB}_{\sum_{m=1}^{3} x_m} \pi$$

The objective function would finally be:

$$\text{Min} \quad 0,1\left(\sum_{h=1}^{3} \text{CH}_h y_h + \sum_{m=1}^{3} \text{CM}_m x_m + \text{SC}\beta\right) + \sum_{i=1}^{10}\sum_{j=1}^{2} F_i \alpha_{ij} + \sum_{i=1}^{10}\sum_{j=1}^{2} C_j P_j D \alpha_{ij} - yD - b\pi$$

8.4 Production and Delivery of Solar Panels

The EPRS Factory is a producer of solar panels. It manufactures two models of plates daily (P1, P2) that differ in size, among other technical details. It has a portfolio of three client companies (E1, E2, E3) that have requested the following amounts for the month of April:

Client	*Panel*	*Units*
E1	*P1*	*35*
E2	*P1*	*100*
E3	*P1*	*40*
E1	*P2*	*60*
E2	*P2*	*10*

Currently, its warehouse has 4 P1 panels and 3 P2 panels. The capacity of the warehouse is not considered as it is large enough to store any number of panels that the company handles. The factory has a daily production capacity of six P1 panels and three P2 panels. In the month of April, there are 21 working days.

For the daily transport of panels, there is a trailer with a capacity of 120 m^3. The volume of each plate is as follows: P1 Volume = 12 m^3; P2 Volume = 16 m^3

The trailer can do, at most, one of the following routes daily:

Route	*Companies served*	*Cost*
1	*E1*	*C1*
2	*E2*	*C2*
3	*E3*	*C3*
4	*E1+E2*	*C4*
5	*E2+E3*	*C5*

In manufacturing planning, due to the characteristics of the process, only one type of panel can be produced each day. The day on which the type of panel is changed in the production process, the factory incurs an extra cost of $E.

Table 8.7 Elements of Example 8.4

Elements	Set	QN	Data				
			Name	Param	Type	Belong	Value
Clients	$i = 1\ldots3$	$\mathrm{I_U}$	Demand	D_{ij}	I	S	–
			Client-route	CR_{ir}	B	S	–
Panels	$j = 1,2$ (P1, P2)	$\mathrm{C_D}$		D_{ij}; I_j; K_j			
			Volume	VP_j	C	W	{16,10}
			Non-delivery cost	cn_j	C	W	–
Routes	$r = 1\ldots5$	$\mathrm{I_U}$		A_{ir}			
			Cost	c_r	C	W	–
Vehicle	–	$\mathrm{I_U}$	Volume	V	C	W	120
Warehouse	–	$\mathrm{I_U}$	Stock	I_j	I	S	{4,3}
Factory	–	$\mathrm{I_U}$	Daily production capacity	K_j	I	S	{6,3}
			Extra cost	CE	C	W	–
Days of April	$t = 1\ldots21$	$\mathrm{I_U}$		K_{jt}			

Regarding panels that are not delivered due to lack of capacity, the company estimates a unit cost cn_1 and cn_2, for panels P1 and P2, respectively, for each panel not delivered from the April demand.

The factory wants to manage the production and delivery of panels with the main objective of minimizing costs.

Table of Elements (Table 8.7)

The daily production capacity is not necessary to associate it with the days $t = 1–31$, since it is constant for each day and therefore, we can consider it as being only associated to the factory.

Decision Activities

A priori, diverse activities are identified on two collective elements, the panels P1 and P2, over a set of time periods. For this reason, we can design a directed graph to collect the activities and calculations on those measurable elements. We must create a graph for each plate, which will be analogous. Therefore, we simply create the graph for Panels P1 (Fig. 8.1).

From the labelling of arcs in the graph, we can extract the variables of the problem; some of them will be identified as decision activities and others as auxiliary calculations.

x_{jt}: Number of P_j panels produced in the factory on day t
A_{jt}: Number of P_j panels stored in the Warehouse after day t
DV_{jt}: Number of P_j panels distributed in the vehicle on day t
R_{jrt}: Number of P_j panels distributed via the route r on day t

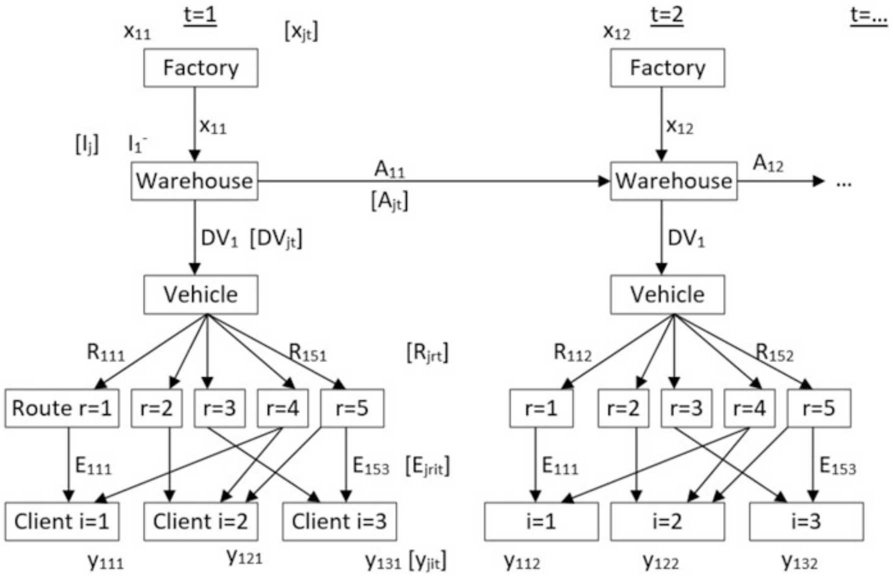


Fig. 8.1 Graph for Panels P1

E_{jrit}: Number of P_j panels delivered to client i via route r on day t
y_{jit}: Number of P_j panels delivered to client i on day t

From all these variables, we can only define as decision variables:

– The panels that we produce: x_{jt}
– The panels that we deliver to each client i via each route: E_{jrit}

The rest corresponds to auxiliary calculations obtained by applying the flow balance equation in the nodes:

$$\forall j, i, t : y_{jit} = \sum_{r/CR_{ir}=1} E_{jirt}$$

$$\forall j, r, t : R_{jrt} = \sum_{i/CR_{ir}=1} E_{jirt}$$

$$\forall j, t : \mathrm{DV}_{jt} = \sum_{r=1}^{5} R_{jrt}$$

$$\forall j, t = 1 : A_{jt} = x_{jt} + I_j - \mathrm{DV}_{jt}$$

$$\forall j, t > 1 : A_{jt} = x_{jt} + A_{jt-1} - \mathrm{DV}_{jt}$$

Specifications

1. *Implicit Specifications*
   I1. Based on data: there is a capacity attribute in the system, the daily production capacity of the factory each day. The warehouse is not considered.

   I1.1. $\forall j,\ t : x_{jt} \leq K_j$

Also, the vehicle has a capacity attribute, its volume, to be controlled daily:

I1.2. $\forall t : \sum_{j=1}^{2} \mathrm{VP}_j \mathrm{DV}_{jt} \leq V$

Regarding the demand, we are not obliged to meet it, but undelivered units may remain, which will entail a cost in the objective function. Therefore, it is not necessary to propose a demand contribution specification. Only, the demand could be treated as an upper bound of the number of units to be delivered; although the manufacture of panels involves a cost and therefore the system should not manufacture more units than those requested by the customers, production costs are not mentioned in the statement, only change of production of panels and routes. This can lead to the fact that if we do not specify the quantities to be delivered, the resolution of the model offers an optimal solution in which more panels than those requested are delivered. Therefore, to maintain the consistency of the solution, we define the demand as an upper bound. The specification falls on each client and each type of panel:

I1.3. $\forall i, j : \sum_{t=1}^{25} y_{jit} \leq D_{ij}$

I2. Quantitative selection rules: they do not exist.
I3. Logical conditions between activities: they do not exist.
I4. Bounds of discrete measurable activities: they do not exist.
I5. Flow balance constraints: those previously associated with the auxiliary calculations.

2. *Explicit Specifications*
   E1. "The trailer can do, at most, one of the following routes daily . . . ."

This specification works with the R_{jrt} variables. It establishes a relationship in which at most one variable can be positive. This can be stated by different propositions:

$$\forall t, r = 1 : \text{IF} \quad \sum_{j=1}^{2} R_{j1t} > 0 \quad \text{THEN} \quad \sum_{r \neq 1} \sum_{j=1}^{2} R_{j1t} = 0$$

$$\forall t, r = 2 : \text{IF} \quad \sum_{j=1}^{2} R_{j2t} > 0 \quad \text{THEN} \quad \sum_{r \neq 2} \sum_{j=1}^{2} R_{j1t} = 0$$

...

But the simplest way is to first define the logical calculation on each route to know if it has been chosen or not and then define a quantitative selection rule with the logical variables:

Binary logical calculation: Route chosen every day.
Applied to: Each route r and each day t.

Variables: $\beta_{rt} = 1$ if I choose route r on day t; 0 otherwise.
Logical proposition:

$$\forall r,t : \beta_{rt} = 1 \quad \leftrightarrow \quad \sum_{j=1}^{2} R_{jrt} \geq 1$$

$$\Rightarrow \text{Ref } S_v \Rightarrow \forall r,t : \text{IF } \sum_{j=1}^{2} R_{jrt} \geq 1 \quad \text{THEN } \beta_{rt} = 1$$

Quantitative selection rule:

$$\forall t : \sum_{r=1}^{5} \beta_{rt} \leq 1$$

E2. "Only one type of panel can be produced each day":

It is a logical specification: the two types of panels cannot be produced each day:

$$\forall t : \text{NOT}(x_{1t} > 0 \quad \text{AND} \quad x_{2t} > 0)$$
$$\Rightarrow \text{Ref. } f_5 \Rightarrow \forall t : \text{NOT}(x_{1t} \geq 1 \quad \text{AND} \quad x_{2t} \geq 1)$$
$$\Rightarrow \text{Ref. } f_{35} \Rightarrow \begin{matrix} \forall t : \omega_{1t} = 1 \leftrightarrow x_{1t} \geq 1 \\ \forall t : \omega_{2t} = 1 \leftrightarrow x_{2t} \geq 1 \end{matrix}$$
$$\Rightarrow \text{Ref. } f_{25} \Rightarrow \begin{matrix} \forall t : x_{1t} \geq \omega_{1t} \\ \forall t : x_{1t} \leq UB_{x_{1t}}\omega_{1t} \\ \forall t : x_{2t} \geq \omega_{2t} \\ \forall t : x_{2t} \leq UB_{x_{2t}}\omega_{2t} \end{matrix}$$
$$\Rightarrow \text{Ref. } f_{37} \Rightarrow \forall t : \text{NOT}(\omega_{1t} + \quad \omega_{2t} \geq 2) \Rightarrow \text{Ref. } f_{11} \Rightarrow \forall t : \omega_{1t} + \quad \omega_{2t} \leq 1$$

Objective Criterion

The objective, although not defined explicitly, is to minimize costs. The costs of the system are:

- Cost for panel production change: CE
- Cost of each route: C_r
- Cost for non-delivery of panels: cn_j

The costs for production change and route can be produced on each day t. The cost for non-delivery of units does not depend on t.

Cost for panel production change: It will be necessary to define a logical calculation on each day t, starting from the second day, to know if there has been a change in the panels that are produced. But the definition is not so simple, since there may be days on which the system does not produce and we have to keep track of which type of panel was the last to produce before t to know if there has been a change on day t. Therefore, we will define a previous logical calculation every day and with each panel, to know if that panel was the last to be produced. For that calculation, we are going to use the logical calculation already defined in E2, ω_{jt}, which told us if panel j had been produced on day t.

Binary logical calculation: Last panel produced.
Applied to: Each day t and each panel j.

Variables: $\delta_{jt} = 1$ if the last panel produced up to day *t* was panel *j*; 0 otherwise.
Logical proposition: It can be considered in several equivalent forms:
Form 1:
$t = 1 : \delta_{j1} = 1$ IF AND ONLY IF $\omega_{j1} = 1$
$\forall t, t > 1, j = 1 : \delta_{1t} = 1$ IF AND ONLY IF $\omega_{1t} = 1$ OR $(\omega_{2t} = 0$ AND $\delta_{1t-1} = 1)$
$\forall t, t > 1, j = 2 : \delta_{2t} = 1$ IF AND ONLY IF $\omega_{2t} = 1$ OR $(\omega_{1t} = 0$ AND $\delta_{2t-1} = 1)$
Form 2:
$\forall t :$ IF $\omega_{1t} = 1$ THEN $\delta_{1t} = 1$
$\forall t :$ IF $\omega_{2t} = 1$ THEN $\delta_{1t} = 0$
$\forall t :$ IF $\omega_{2t} = 1$ THEN $\delta_{2t} = 1$
$\forall t :$ IF $\omega_{1t} = 1$ THEN $\delta_{2t} = 0$
$\forall t / t > 1 :$ IF $\omega_{1t} = 0$ AND $\omega_{2t} = 0$ THEN $\delta_{1t} = \delta_{1t-1}$ AND $\delta_{2t} = \delta_{2t-1}$

Modelling of logical proposition (FORM 1):
$t = 1 : \delta_{j1} = 1$ IF AND ONLY IF $\omega_{j1} = 1$
$\Rightarrow$Ref. $f_{23} \Rightarrow t = 1 : \delta_{j1} = \omega_{j1}$
$\forall t, t > 1, j = 1 : \delta_{1t} = 1$ IF AND ONLY IF $\omega_{1t} = 1$ OR $(\omega_{2t} = 0$ AND $\delta_{1t-1} = 1)$
$\Rightarrow$ Ref.$f_{37} \Rightarrow \forall t, t > 1, j = 1 :$
$\delta_{1t} = 1$ IF AND ONLY IF $\omega_{1t} = 1$ OR $((1 - \omega_{2t}) + \delta_{1t-1} \geq 2)$
$\Rightarrow$ Ref.$f_{35} \Rightarrow \lambda_{1t} = 1 \leftrightarrow ((1 - \omega_{2t}) + \delta_{1t-1} \geq 2)$
$\Rightarrow$ Ref.$f_{25} \Rightarrow (1 - \omega_{2t}) + \delta_{1t-1} \geq 2\lambda_{1t}$
$\Rightarrow$ Ref.$f_{25} \Rightarrow (1 - \omega_{2t}) + \delta_{1t-1} \leq (1 - \lambda_{1t}) + 2\lambda_{1t}$
$\Rightarrow$ Ref.$f_{34} \Rightarrow \forall t, t > 1, j = 1 : \delta_{1t} = 1$ IF AND ONLY IF $\omega_{1t} + \lambda_{1t} \geq 1$
$\Rightarrow$ Ref.$f_{25} \Rightarrow \omega_{1t} + \lambda_{1t} \geq \delta_{1t}$
$\Rightarrow$ Ref.$f_{25} \Rightarrow \omega_{1t} + \lambda_{1t} \leq 2\delta_{1t}$
$\forall t, t > 1, j = 2 : \delta_{2t} = 1$ IF AND ONLY IF $\omega_{2t} = 1$ OR $(\omega_{1t} = 0$ AND $\delta_{2t-1} = 1)$
$\Rightarrow$ Ref.$f_{37} \Rightarrow \forall t, t > 1, j = 2 :$
$\delta_{2t} = 1$ IF AND ONLY IF $\omega_{2t} = 1$ OR $((1 - \omega_{1t}) + \delta_{2t-1} \geq 2)$
$\Rightarrow$ Ref.$f_{35} \Rightarrow \lambda_{2t} = 1 \leftrightarrow ((1 - \omega_{1t}) + \delta_{2t-1} \geq 2)$
$\Rightarrow$ Ref.$f_{25} \Rightarrow (1 - \omega_{1t}) + \delta_{2t-1} \geq 2\lambda_{2t}$
$\Rightarrow$ Ref.$f_{25} \Rightarrow (1 - \omega_{1t}) + \delta_{2t-1} \leq (1 - \lambda_{2t}) + 2\lambda_{2t}$
$\Rightarrow$ Ref.$f_{34} \Rightarrow \forall t, t > 1, j = 1 : \delta_{2t} = 1$ IF AND ONLY IF$\omega_{2t} + \lambda_{2t} \geq 1$
$\Rightarrow$ Ref.$f_{25} \Rightarrow \omega_{2t} + \lambda_{2t} \geq \delta_{2t}$
$\Rightarrow$ Ref.$f_{25} \Rightarrow \omega_{2t} + \lambda_{2t} \leq 2\delta_{2t}$

Binary logical calculation: Production change each day.
Applied to: Each day *t, t*>1.
Variables: $\alpha_t = 1$ if production change is on day *t*; 0 otherwise.
Logical proposition:

$\forall t/t > 1 : \alpha_t = 1$ IF AND ONLY IF $(\delta_{1t} = 1$ AND $\delta_{2t-1} = 1)$ OR $(\delta_{2t} = 1$ AND $\delta_{1t-1} = 1)$

Proposition Modelling:

$\Rightarrow$ Ref. $f_{37} \Rightarrow$

$\forall t/t > 1 : \alpha_t = 1$ IF AND ONLY IF $(\delta_{1t} + \delta_{2t-1} \geq 2)$ OR $(\delta_{2t} + \delta_{1t-1} \geq 2)$

$$\Rightarrow \text{Ref. } f_{35} \Rightarrow \begin{array}{l} \forall t/t > 1 : \pi_{1t} = 1 \quad \leftrightarrow (\delta_{1t} + \delta_{2t-1} \geq 2) \\ \forall t/t > 1 : \pi_{2t} = 1 \quad \leftrightarrow (\delta_{2t} + \delta_{1t-1} \geq 2) \end{array}$$

$$\Rightarrow \text{Ref. } f_{25} \Rightarrow \Rightarrow \begin{array}{l} \forall t/t > 1 : \delta_{1t} + \delta_{2t-1} \geq 2\pi_{1t} \\ \forall t/t > 1 : \delta_{1t} + \delta_{2t-1} \leq (1 - \pi_{1t}) + 2\pi_{1t} \\ \forall t/t > 1 : \delta_{2t} + \delta_{1t-1} \geq 2\pi_{2t} \\ \forall t/t > 1 : \delta_{2t} + \delta_{1t-1} \leq (1 - \pi_{2t}) + 2\pi_{2t} \end{array}$$

$\Rightarrow$ Ref. $f_{34} \Rightarrow \forall t/t > 1 : \alpha_t = 1$ IF AND ONLY IF $\pi_{1t} + \pi_{2t} \geq 1$

$$\Rightarrow \text{Ref. } f_{25} \Rightarrow \begin{array}{l} \forall t/t > 1 : \ \pi_{1t} + \pi_{2t} \geq \alpha_t \\ \forall t/t > 1 : \ \pi_{1t} + \pi_{2t} \leq \alpha_t \end{array}$$

The expression of this cost will be: $\sum_{t=2}^{25} \text{CE}\alpha_t$.

Cost of each route: since we have already defined the logical calculation on the route chosen on each day t, β_{rt}, the expression of this cost would be:

$$\sum_{r=1}^{5} \sum_{t=1}^{25} C_r \beta_{rt}$$

Cost for non-delivery of panels: panels j that are not delivered to each client i correspond to the expression:

$$D_{ij} - \sum_{t=1}^{25} y_{jit}$$

The expression of the cost will then be: $\sum_{j=1}^{2} \sum_{i=1}^{3} \text{cn}_j \left(D_{ij} - \sum_{t=1}^{25} y_{jit} \right)$.

Final Expression of the Objective Function:

$$\text{Min} \quad \sum_{t=2}^{25} \text{CE}\alpha_t + \sum_{r=1}^{5} \sum_{t=1}^{25} C_r \beta_{rt} + \sum_{j=1}^{2} \sum_{i=1}^{3} \text{cn}_j \left(D_{ij} - \sum_{t=1}^{25} y_{jit} \right) \Rightarrow$$

$$\Rightarrow \text{Min} \quad \sum_{t=2}^{25} \text{CE}\alpha_t + \sum_{r=1}^{5} \sum_{t=1}^{25} C_r \beta_{rt} - \sum_{j=1}^{2} \sum_{i=1}^{3} \sum_{t=1}^{25} \text{cn}_j y_{jit}$$

8.5 Selection of a Tree in a Graph

There is an undirected graph G (N, E), n=number of nodes, e= number of edges. The nodes have a weight associated with them and the edges a cost. We must obtain a tree (connected graph without cycles) of G with at least r nodes, r≤n, maximizing the total weight of the selected nodes minus the total cost of the edges.

We are faced with a problem associated with an undirected graph. In Chap. 3 we presented the table of generic elements of directed and undirected graphs. Based on that information, we will define our table of elements.

This type of system requires certain knowledge of operational research in the field of graphs, particularly knowledge of the specifications, which are supposed to be known and are not usually presented explicitly in the statement. As the statement mentions, the objective is to obtain a tree. Obtaining a tree carries with it a series of specifications associated with the selection of edges, which would be our decision activity.

It is also implicit that in an optimization problem associated with an undirected graph, the decision activities are limited to the selection of edges and/or nodes. In our example, it is sufficient to define only the selection of edges. The selection of nodes, although it can be defined as a decision activity, is really a logical calculation. A node is selected if one of the edges that connect it has been selected. Our network has at least *r* nodes, which corresponds to an explicit specification of the problem.

When problems arise in obtaining trees in graphs, the main problem is selecting edges that do not form cycles. This can be achieved by using various strategies of formulating the problem. The most efficient strategies convert the undirected graph into a directed one, assigning two arcs for each edge. On this directed graph, there are strategies that incorporate a flow concept in the graph and others that use the definition of levels.

For our example, we are going to model the problem using two strategies. The first one does not turn the graph into a directed one, but it needs to collect all the cycles that the graph has, to later state in the specifications that there must not be any of them in the tree (Dantzig et al. 1954). To obtain all the cycles of a graph, a recursive and exponential algorithm is needed that explores the paths that can be generated from each node. The second strategy is based on the formulation of Miller et al. (1960) for obtaining a tree in a graph. The strategy needs to use a directed graph, but it is not necessary to incorporate a flow concept. Also it is incorporated an indeterminate continuous property to the nodes, their depth.

Strategy 1: Formulation Based on the Formulation of Dantzig et al. (1954)

Table of Elements

Table 8.8 Elements of Example 8.5 – Strategy 1

			Data				
Elements	Set	QN	Name	Param	Type	Belonging	Value
Nodes	$i = 1...n$	I_U	Edge-Node	EN_{ik}	B	S	...
			Weight	W_i	C	W	...
Edges	k=1...m	I_U	Cost	C_k	C	W	...
				EN_{ik}; EC_{ck}			
Cycles	c=1...v	I_U	Edge-Cycle	EC_{ck}	B	S	...
			Number	N_c	E	W	...

In the table of elements, we include the list of the cycles of the graph, a total of v. From each cycle we store the edges that form it and a data calculation, the number of them (Table 8.8).

Decision Activities

Action: Select edges.
Decision variables: $\alpha_k = 1$ if I select edge k; 0 otherwise.

Specifications

The specifications for forming a tree of an undirected graph are as follows:

1. The number of selected nodes must be equal to the number of selected edges plus one.
2. We cannot select all the edges belonging to a cycle.

We must, before specification 1, define the logical calculation of the selected node:

Binary logical calculation: Selected node.
Applied to: Each node i.
Variables: $\beta_i = 1$ if I select node i; 0 otherwise.
Logical proposition:
$\forall i : \beta_i = 1$ IF AND ONLY IF $\sum\limits_{k/\mathrm{EN}_{ik}=1} \alpha_k \geq 1$

$\Rightarrow$ Ref. $S_V \Rightarrow \forall i$: IF $\sum\limits_{k/\mathrm{AN}_{ik}=1} \alpha_k \geq 1$ THEN $\beta_i = 1$

Specification 1. $\sum\limits_{i=1}^{n} \beta_i = \sum\limits_{k=1}^{m} \alpha_k + 1$

Specification 2. $\forall c$: $\sum\limits_{k/\mathrm{EC}_{ck}=1} \alpha_k \leq N_c - 1$

It is also necessary to make it explicit that there must be at least r selected nodes:

Table 8.9 Elements of Example 8.5 – Strategy 2

			Data				
Elements	Set	QN	Name	Param	Type	Belonging	Value
Nodes	$i = 1...n$	I_M	Source node	O_{ik}	B	S	. . .
			Destination node	D_{ik}	B	S	. . .
			Weight	W_i	C	W	. . .
Arcs	k=1. . .2m	I_U		O_{ik}; D_{ik}			
			Cost	C_k	C	W	. . .

Specification 3. $\sum_{i=1}^{n} \beta_i \geq r$

Objective Criterion

Maximize total of node weights minus total of edge costs:

$$\text{O.F. : Max } \sum_{i=1}^{n} W_i \beta_i - \sum_{k=1}^{m} C_k \alpha_k$$

Strategy 2: Formulation Based on the Formulation of Miller, Tucker, and Zemlin (1960)

Table of Elements

In this case we use the table of directed graphs, converting each edge into two arcs. In addition, the nodes become individual measurable elements due to the indeterminate property introduced on their depth in the solution tree (Table 8.9).

Decision Activities

Action: Select Arcs.
Decision variables: $\alpha_k = 1$ if I select arc k; 0 otherwise.

In addition to the activity of selecting, an activity is required that assigns a depth level value to the nodes.

Action: Assign depth to the nodes.
Decision variables: xi = Depth level assigned to the node i.

Specifications

The logical calculation is maintained to know the selected nodes, but in this version, we only consider that for a node to be selected, some arc must have been selected with a destination in that node:

Binary logical calculation: Selected node.
Applied to: Each node i.
Variables: $\beta_i = 1$ if I select node i; 0 otherwise.
Logical proposition:
$\forall i : \beta_i = 1$ IF AND ONLY IF $\sum_{k/D_{ik}=1 \ \vee O_{ik}=1} \alpha_k \geq 1$

$$\Rightarrow \text{Ref. } S_V \Rightarrow \forall i : \text{IF} \sum_{k/D_{ik}=1 \ \ \vee O_{ik}=1} \alpha_k \geq 1 \ \text{THEN} \ \beta_i = 1$$

The specifications to form a tree are:

1. The number of selected nodes must be equal to the number of selected arcs plus one.
   $$\sum_{i=1}^{n} \beta_i = \sum_{k=1}^{2m} \alpha_k + 1$$
2. To prevent the formation of cycles, it is imposed that if an arc is selected, the depth of the destination node is lower than that of the source node. It is a logical proposition that is stated as follows:
   $$\forall k, i/O_{ik} = 1, j/D_{jk} = 1 : \text{IF} \ \alpha_k = 1 \quad \text{THEN} \quad x_i > x_j$$

MTZ establishes a minimum difference between depths of 1, although any positive value would work:

$$\forall k, i/O_{ik} = 1, j/D_{jk} = 1 : \text{IF} \ \alpha_k = 1 \quad \text{THEN} \quad x_i \geq x_j + 1 \Rightarrow$$
$$\Rightarrow \forall \ k, i/O_{ik} = 1, j/D_{jk} = 1 : \text{IF} \ \alpha_k = 1 \quad \text{THEN} \quad x_i - x_j \geq 1$$
$$\Rightarrow \text{Ref.f}_{15} \Rightarrow$$
$$\forall k, i/O_{ik} = 1, j/D_{jk} = 1 : x_i - x_j \geq \alpha_k + (1-n)(1-\alpha_k)$$
$$[LB_{x_i - x_k} = 0 - (n-1) = 1 - n]$$
$$\Rightarrow \forall k, i/O_{ik} = 1, j/D_{jk} = 1 : x_i - x_j - n\alpha_k \geq 1 - n$$

3. At least r selected nodes:
   $$\sum_{i=1}^{n} \beta_i \geq r$$

Objective Criterion

$$\text{O.F.:Max} \sum_{i=1}^{n} W_i \beta_i - \sum_{k=1}^{2m} C_k \alpha_k$$

8.6 Programming of Pilots for an Airline's Flights

The calculation of the pilots needed to cover the scheduled flights of an airline in a given period, as well as the specific programming of them, is an optimization problem that would fall within the interval of scheduling problems (Spieksma 1999).

It is a system that can be transferred to other work environments, because we can consider it as a system of assigning work shifts. What differentiates shift systems are the specific restrictions regarding the assignments that workers have. For our system, we will work with the real specifications of a well-known airline, although we will simplify certain aspects that are irrelevant to the modelling exercise.

Description of the system:

The company configures as a trip an airplane journey that begins and ends at the same airport. That is, the company considers a trip to be a double flight with a brief 20-minute break at the destination airport on the outward journey. Therefore, the planning of pilots is carried out independently by each city. The pilots are assigned a city as a base, and all their flights begin and end in the same city. This is realistic and possible because the company does not carry out long flights, such as ocean liners, where the round trip would not be viable in the same day. In our example we will plan the flights of a city.

The necessary information of the flight is the following:

- *Departure time*
- *Arrival time*
- *Type of flight (requires pilot with experience or not): this can be defined in a binary way: 1 = Need experience; 0 = Experience not needed.*

Pilots will be associated with a type related to their experience. We will have pilots of less than 1000 flight hours (without experience) and pilots with at least 1000 hours (experienced).

The working specifications for a pilot are the following:

- *Maximum daily activity time (TMAX_1): the activity starts at the departure time of the first flight of the day assigned to the arrival time of the last flight.*
- *Maximum number of flight hours in the month (TMAX_2).*
- *Maximum activity time in the month (TMAX_3).*
- *Minimum rest time between the last flight of a day and the first flight of the next day (DM).*
- *Each pilot must adjust to a model in which he works 5 consecutive days and rests 4. This means that each pilot has to adjust to one of these five patterns:*

W: Work; R: Rest

Days:	1	2	3	4	5	6	7	8	9	10	11	...
Pattern 1:	W	W	W	W	W	R	R	R	R	W	W	...
Pattern 2:	R	W	W	W	W	W	R	R	R	R	W	...
Pattern 3:	R	R	W	W	W	W	W	R	R	R	R	...
Pattern 4:	R	R	R	W	W	W	W	W	R	R	R	...
Pattern 5:	R	R	R	R	W	W	W	W	W	R	R	...

In addition to these specifications, there is the basic specification of any shift or interval scheduling system, which is that a pilot cannot perform two overlapping flights over time. In our case, we are going to impose that there should be at least 20 min between two trips that are assigned to the same pilot.

Table of Elements

With the pilots of the company, which can be of two types, we are in principle looking at two undetermined collective elements, since one of the tasks of the problem is to calculate the number of pilots of each type. However, the statement

Table 8.10 Elements of Example 8.6

Elements	Set	QN	Data				
			Name	Param	Type	Belong	Value
Pilots	$i = 1...m$	I_U	Experience	E_i	B	W	–
Trips (double flights)	$j = 1...n$	I_U	Starting time	I_j	C	W	–
			End time	F_j	C	W	–
			Day	D_j	E	W	–
			Type	T_j	B	W	–
Work patterns	$p = 1...5$	I_U	Daily rule	A_{pt}	B	S	
Days	$t = 1...30$	I_U		A_{pt}			
System	–	I_U	Maximum daily activity time	TMAX_1	C	W	–
			Maximum month hours	TMAX_2	C	W	–
			Maximum month activity time	TMAX_3	C	W	–
			Minimum rest time	DM	C	W	–

always alludes individually to each pilot who will own the system, which means that it is necessary to calculate an upper bound of the number of pilots needed and to work with each pilot as an individual element. In our case we will take m_1 for experienced pilots and m_2 for inexperienced pilots. To reduce the number of subscripts in the table, we will group all the pilots into a single set, a total of $m = m_1 + m_2$. The first m_1 pilots will be the pilots with experience and the rest without experience (Table 8.10).

The daily rule A_{pt} has value 1 if the day t with pattern p is a working day, and 0 if it is a day of rest.

Decision Activities

The main activity of the system is the assignment of pilots to trips.

Action: Assign Pilots to Trips.
Decision variables: $\alpha_{ij} = 1$ if I assign Pilot i to Trip j; 0 otherwise.
But it is not the only activity. In the statement it is mentioned that each pilot must be assigned to a work pattern. The activity will be configured as follows:
Action: Assign Work patterns to Pilots.
Decision variables: $\beta_{ip} = 1$ if I assign pattern p to pilot i; 0 otherwise.
Specifications

1. **Implicit Specifications**
   I1. Based on data: Based on flight typologies, we have the specification that an inexperienced pilot cannot manage a type of flight that needs an experienced pilot:
   $\forall j/T_j = 1, \forall\ i/E_i = 0 : \quad \text{NOT}(\alpha_{ij} = 1)$
   I2. Quantitative selection rules: Regarding the main activity of the system, we know that each flight needs one pilot (Table 8.11).

Table 8.11 Diagram of the decision activity Assign pilots to trips

Activity	Elements selecting	Selectable elements	Selection	Constraints
Assign	Pilots $i = 1...m$	Trips	–	
	Trips $j = 1...n$	Pilots	$= 1$	$\forall j : \sum_{i=1}^{m} \alpha_{ij} = 1$

Regarding the other activity of choice, it is explicitly described that each pilot has a pattern (E5).

I3. Logical conditions between activities:

Implicitly we have a relationship between activities α_{ij} and β_{ip}: A pilot cannot carry out a flight that is on one of the rest days of his work schedule. Since it is not explicitly described, we introduce it in this section:

The condition will be applied to each pilot, each pattern and each trip of each day in which the working rule for that pattern and day that it is one of rest.

$\forall i, p, t/A_{pt} = 0, j/D_j = t$: IF $\beta_{ip} = 1$ THEN $\alpha_{ij} = 0$

I4. Bounds of discrete measurable activities: they do not exist.

I5. Flow balance constraints: they do not exist.

2. **Explicit Specifications**

E1. *Maximum daily activity time: the activity starts at the departure time of the first flight of the day assigned to the arrival time of the last flight.*

It is necessary to declare bound calculations to obtain the starting and end time of each pilot:

Bound calculations must collect a lower and upper bound on the trips that the pilot made each day.

If we define, for example, the lower bound calculation as:

Lower Bound Calculation: Lower bound of the starting times of the trips assigned to the Pilot i on each day t.
Applied to: Each pilot i, each day t.
Variables:
$x^{\text{MIN}}{}_{it}$ = lower bound of the starting time of the activity of pilot i on day t.
Constraints defining the calculation:
$\forall i, \forall t, \forall j/D_j = t : x^{\text{MIN}}{}_{it} \leq I_j \alpha_{ij}$

We would be making an error, since when pilot i does not perform trip j, $\alpha_{ij} = 0$, and therefore, $I_j \alpha_{ij} = 0$, the lower bound would always be 0. We want to calculate the bound between the flights that the pilot has made. Those that you have not made should not influence the lower bound. For this, we must use the definition of bounds explained in Sect. 5.4.1 applied to variables that are not determined a priori. The constraints that define the calculation are based on a logical proposition:

Lower Bound calculation: Lower bound of the starting times of the trips assigned to the Pilot i on each day t
Applied to: Each pilot i, each day t.
Variables:
$x^{\mathrm{MIN}}{}_{it}$ = lower bound of the starting time of the activity of pilot i on day t.
Constraints defining the calculation:
$\forall i, \forall\, t, \forall\, j/D_j = t$: IF $\alpha_{ij} = 1$ THEN $x^{\mathrm{MIN}}{}_{it} \le I_j\alpha_{ij}$

For the arrival value of the trips made by each pilot, it is not necessary to use a logical proposition, since the upper bound can be imposed on all the pairs (i, j). In the case of not performing the trip, $\alpha_{ij} = 0$, and therefore, $F_j\alpha_{ij} = 0$, it will not influence the calculation of a upper bound.

Upper Bound calculation: Upper bound of the end times of the trips assigned to the Pilot i on each day t.
Applied to: Each pilot i, each day t.
Variables:
$x^{\mathrm{MAX}}{}_{it}$ = upper bound of the activity of pilot i on day t.
Constraints defining the calculation:
$\forall i, \forall\, t, \forall\, j/D_j = t : x^{\mathrm{MAX}}{}_{it} \ge F_j\alpha_{ij}$

The specification is an imposition of upper bound to the activity range:

$\forall i, \forall\, t : x^{\mathrm{MAX}}{}_{it} - x^{\mathrm{MIN}}{}_{it} \le TMAX\_1$

E2. *Maximum number of flight hours in the month.*

This specification is a capacity constraint on each pilot. The consumption is performed by the variables α_{ij} with a unitary consumption equal to the time of trip j.

$$\forall i : \sum_{j=1}^{n} \left(F_j - I_j\right)\alpha_{ij} \le TMAX\_2$$

E3. *Maximum activity time in the month.*

It would suffice to add the activity of each day and impose the upper bound:

$$\forall i : \sum_{t=1}^{30} \left(x^{MAX}{}_{it} - x^{MIN}{}_{it}\right) \le TMAX\_3$$

E4. *Minimum rest time between the last trip of a day and the first trip of the next day.*

We apply it to each pilot every couple of consecutive days of the month:

$\forall i, \forall\, t/t < 30 : x^{\mathrm{MAX}}{}_{it} - x^{\mathrm{MIN}}{}_{it+1} \le DM$

E5. *Each pilot must adjust to one of these five patterns.*

$$\forall i : \sum_{p=1}^{5} \beta_{ip} = 1$$

E6. *There should be at least 20 minutes between two flights that are assigned to the same pilot.*

The specification falls on each pilot and every two trips that do not have an interval of 20 min between them:

$$\forall i, \forall j, j'/I_{j'} < F_j + 20\,\text{min}\,\&I_{j'} \geq I_j : \text{NOT}\big(\alpha_{ij} = 1 \quad \text{AND} \quad \alpha_{ij'} = 1\big)$$

Objective Criterion

The objective function will minimize the number of pilots needed. Therefore, we are going to minimize the number of pilots that make a trip. We will have to perform a logical calculation on each pilot that has or has not carried out flights and then minimize the sum of the logical variables.

Binary logical calculation: Pilot who has performed trips.
Applied to: Each pilot *i*.
Variables: $\forall i : \delta_i = 1$ if pilot *i* performs trips; 0 otherwise.
Logical proposition:

$$\forall i : \delta_i = 1 \text{ IF AND ONLY IF } \sum_{j=1}^{n} \alpha_{ij} > 0$$

$$\Rightarrow \text{Ref. } S_V \Rightarrow \forall i : \text{IF } \sum_{j=1}^{n} \alpha_{ij} > 0 \text{ THEN } \delta_i = 1$$

Objective Function

$$\text{Min} \quad \sum_{i=1}^{m} \delta_i$$

References

Dantzig, G. B., Fulkerson, D. R., & Johnson, S. M. (1954). Solution of a large scale traveling salesman problem. *Operations Research, 2*, 393–410.

Jensen, T. R., & Toft, B. (1995). *Graph coloring problems*. New York: Wiley-Interscience. ISBN 0-471-02865-7.

Miller, C. E., Tucker, A. W., & Zemlin, R. A. (1960). Integer programming formulations and traveling salesman problems. *Journal of the ACM, 7*, 326–329.

Spieksma, F. C. R. (1999). On the approximability of an interval scheduling problem. *Journal of Scheduling, 2*(5), 215–227.

Correction to: Modelling in Mathematical Programming

Correction to:
J. M. García Sánchez, ***Modelling in Mathematical Programming,*** **International Series in Operations Research & Management Science,**
https://doi.org/10.1007/978-3-030-57250-1

This book was inadvertently published without updating the following corrections:

The city information in the affiliation of the author has been corrected as:

José Manuel García Sánchez
IO and Business Management
University of Seville
Sevilla, Spain

In Chapter 4,
On page 99; section 4.5: the line "If $y = 0$ then $x = 0$ and $\alpha = 0$" should be "If $y = 0$ then $x = 0$ or $\alpha = 0$"

On page 104: the caption for Table 4.15 was corrected to read as "Table 4.15 Version 4.2 of the Elements in Example 4.6.1"

In Chapter 6, page 181, the following note has been included under section 6.8.4.2.

NOTE: Only for some compound propositions, the following expression for the OR operator may also be valid:

$\forall_i / \psi_i \in Z \vee \psi_i \in \mathfrak{R}: \ \psi_i \leftarrow \omega_i = 1$

The updated online versions of these chapters can be found at
https://doi.org/10.1007/978-3-030-57250-1_4
https://doi.org/10.1007/978-3-030-57250-1_6
https://doi.org/10.1007/978-3-030-57250-1

J. M. García Sánchez, *Modelling in Mathematical Programming*, International Series in Operations Research & Management Science 298,
https://doi.org/10.1007/978-3-030-57250-1_9