1

Basic Concepts in Optimization and Graph Theory

1.1. Introduction

An optimization problem is a formal specification of a set of proposals related to a specific framework that includes one or numerous decision makers, one or several objectives to be achieved and a set of structural constraints. Optimization has been practiced in numerous fields of study as it provides a primary tool for modeling and solving complex and hard constrained problems. After the 1960s, integer programming formulations and approximate approaches have received considerable attention as useful tools for solving optimization problems. Depending on the problem structure and its complexity, appropriate solution approaches were proposed to generate appropriate solutions in a reasonable computation time. Several optimization studies are formulated as a problem whose goal is to find the best solution, which corresponds to the minimum or maximum value of a single-objective function. The challenge of solving combinatorial problems lies in their computational complexity since most of them are non-deterministic polynomial-time (NP)-hard [GAR 79]. This complexity can mainly be expressed in terms of the relationship between the

search space and the difficulty of finding a solution. The search space in combinatorial optimization problems is discrete and multidimensional. The higher the dimensionality, the larger the search space, and the harder the problem. The remainder of this chapter is organized as follows. In section 1.2, we highlight the terminology adopted throughout this book. Section 1.3 deals with the mathematical structure of an optimization problem and enumerates its main variants. Section 1.4 illustrates the previously announced principles by a didactic example. Section 1.5 outlines the main features of an optimization problem.

1.2. Notation

We present in the following the major symbols used for defining an optimization problem:

Symbols	Description
n	the number of decision variables
k	the number of objectives
$x = (x_1, \ldots, x_n)^T$	the vector of decision variables
$c_{(p,n)}$	the cost matrix
A	the matrix of constraints
B	Resources limitations
E_O	The set of efficient solutions in the objective space
E_D	The set of efficient solutions in the decision space

1.3. Problem structure and variants

Assuming the linearity of an optimization problem, its mathematical modeling is outlined as follows:

$$Max \quad p.x \tag{1.1}$$

$$S.t.\ A.x \leq B \tag{1.2}$$

$$x \in \mathcal{X} \tag{1.3}$$

where $x = (x_1, \ldots, x_n)^T$ denotes the vector of decision variables, p, b and A are constant vectors and matrix of coefficients, respectively.

Many variants of this formulation can be pointed out:

– *Continuous linear programming (CLP)*: The optimization model [1.1]-[1.3] is a CLP if the decision variables are continuous. For continuous linear optimization problems, efficient exact algorithms such as the simplex-type method [BUR 12] or interior point methods exist [ANS 12].

– *Integer linear programming (ILP)*: The optimization model [1.1]-[1.3] is an ILP if $\mathcal{X}$ is the set of feasible integer solutions (i.e. decision variables are discrete). This class of models is very important as many real-life applications are modeled with discrete variables since their handled resources are indivisible (as cars, machines and containers). A large number of combinatorial optimization problems can be formulated as ILPs (e.g. packing problems, scheduling problems and traveling salesman) in which the decision variables are discrete and the search space is finite. However, the objective function and constraints may take any form [PAP 82].

– *Mixed integer programming (MIP)*: The optimization model [1.1]-[1.4] is called MIP, when the decision variables are both discrete and continuous. Consequently, MIP models generalize the CLP and ILP models. MIP problems have improved dramatically of late with the use of advanced optimization techniques such as relaxations and decomposition approaches, branch-and-bound, and cutting plane algorithms when the problem sizes are small [GAR 12, WAN 13, COO 11]. Metaheuristics are also a good candidate for larger instances. They can also be used to generate good lower or upper bounds for exact algorithms and improve their efficiency.

1.4. Features of an optimization problem

Optimization problems can be classified in terms of the nature of the objective function and the nature of the constraints. Special forms of the objective function and the constraints give rise to specialized models that can efficiently model the problem under study. From this point of view, various types of optimization models can be highlighted: linear and nonlinear, single and multiobjective optimization problems, and continuous and combinatorial programming models. Based on such features, we have to define the following points:

– *The number of decision makers*: if one decision maker (DM) is involved, the problem dealt with is an *optimization problem*; otherwise we are concerned with a *game* that can be cooperative or non-cooperative, depending on the DMs' standpoints.

– *The number of objectives*: it determines the nature of the solution to be generated. If only one objective is addressed in the decision problem, the best solution corresponds to the optimal solution. However, if more than one objective is considered, we went to generate a set of efficient solutions that correspond to some trade-offs between the objectives under study.

– *The linearity*: when both the objective(s) and the constraints are linear, the optimization problem is said to be linear. In that case, specific solution approaches can be adapted as the simplex method. Otherwise, the problem is nonlinear in which case the resolution becomes more complex and the decision space is not convex.

– *The nature of the decision variables*: if the decision variables are integer, we deal with a combinatorial optimization problem.

1.5. A didactic example

Let us consider the following optimization problem involving two decision variables x_1 and x_2. We show in this illustrative example how the solution changes in terms of the nature of the decision variables that can be either continuous or binary and the number of objectives $k = 1,\ 2$. Hence, four optimization problems follow:

$k = 1$		$k = 2$
$\boldsymbol{x_1, x_2 \geq 0}$	$\begin{array}{ll} Max & 2x_1 + x_2 \\ S.t. & 5x_1 + 7x_2 \leq 100 \\ & x_1 - 3x_2 \leq 80 \\ & x \geq 0 \end{array}$ $\Rightarrow$ $\boxed{\begin{array}{r}(x_1, x_2) = (20, 0) \\ z(x) = 40\end{array}}$	$\begin{array}{ll} Max & 2x_1 + x_2 \\ & x_1 + 5x_2 \\ S.t. & 5x_1 + 7x_2 \leq 100 \\ & x_1 - 3x_2 \leq 80 \\ & x \geq 0 \end{array}$ $\Downarrow$ $\boxed{\begin{array}{l} E_D = \{(20, 0), (0, 14.285)\} \\ E_O = \{\begin{pmatrix} 40 \\ 20 \end{pmatrix} \begin{pmatrix} 14.285 \\ 71.428 \end{pmatrix}\} \end{array}}$
$\boldsymbol{x_1, x_2 \in \{0, 1\}}$	$\begin{array}{ll} Max & 2x_1 + x_2 \\ S.t. & 5x_1 + 7x_2 \leq 100 \\ & x_1 - 3x_2 \leq 80 \\ & x \in \{0, 1\} \end{array}$ $\Rightarrow$ $\boxed{\begin{array}{l}(x_1, x_2) = (1, 1) \\ z(x) = 3\end{array}}$	$\begin{array}{ll} Max & 2x_1 + x_2 \\ & x_1 + 5x_2 \\ S.t. & 5x_1 + 7x_2 \leq 100 \\ & x_1 - 3x_2 \leq 80 \\ & x \in \{0, 1\} \end{array}$ $\Downarrow$ $\boxed{\begin{array}{l} E_D = \{(1, 1)\} \\ E_O = \{\begin{pmatrix} 3 \\ 6 \end{pmatrix}\} \end{array}}$

As previously mentioned, the resolution of the single-objective optimization problem yields to the finding of the optimal solution that varies depending on the nature of the decision variables. However, if a second objective is added, the resolution generates a set of Pareto-optimal solutions, as it is the case for $k = 2$.

1.6. Basic concepts in graph theory

A graph G is defined as a couple of sets $G = (V, E)$: a vertex set V and an edge set E.

– *The vertex set* states all involved entities that model the original problem.

– *The edge set* is an exhaustive enumeration of all possible connections between two vertices. If $e = \{x, y\}$ is an edge, we say "x is adjacent to y". A graph can also contain a loop whose endpoints are equal. Based on such features, we can point out numerous types connections in a graph:

- *Simple edge*: a connection between two vertices x and y such that $x \neq y$. It is modeled as a set of two nodes $\{x, y\}$.

- *Oriented edge*: an edge represented by a couple of vertices (x, y).

- *Multiple edges*: numerous edges having the same pair of vertices.

- *Loop*: an edge whose endpoints are equal.

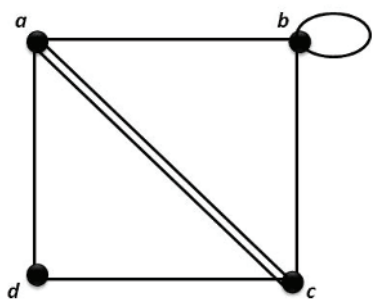

Figure 1.1. *An example of a graph with* $n = 4$ *and* $e = 5$

A simple graph is a graph that contains neither loops nor multiple edges. Simple graphs can be directed as shown in Figure 1.2(a), undirected as is the case of Figure 1.2(b) or/and weighted as shown in Figure 1.2(c). A weighted graph can designate a road network where each edge is labeled by the

distance between the corresponding vertices. Weights can also express the traveling cost between two adjacent vertices.

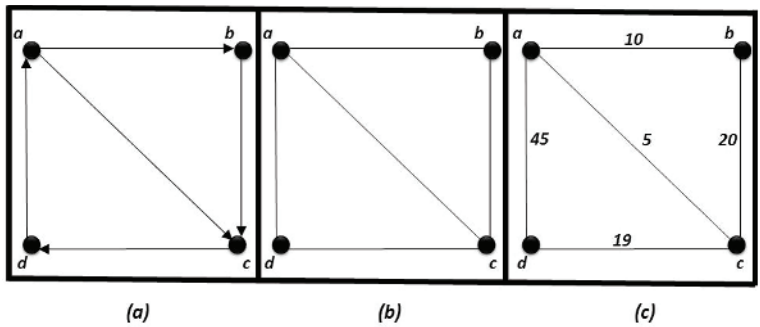

Figure 1.2. *Types of simple graphs*

1.6.1. *Degree of a graph*

The degree of a vertex x in a graph G, denoted by $d_{G(x)}$, is the number of edges where x is one of their endpoints. Note that $e = \frac{\sum_{x \in V} deg(x)}{2}$.

1.6.2. *Matrix representation of a graph*

An alternative representation of a graph is the matrix representation A that designates the adjacency matrix. It is a symmetric square matrix of order n if the graph is undirected. If the graph is weighted, the matrix reports the distances' values if the corresponding vertices are adjacent and 0 elsewhere. Table 1.1 corresponds to the matrix representations of graphs (a), (b) and (c) of Figure 1.2. The matrix representation is adopted mainly for handling a routing problem for which a shortest path has to be generated using the extracted adjacency matrix A.

Graph	(a)	(b)	(c)
Matrix	$\begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 10 & 5 & 45 \\ 10 & 0 & 20 & 0 \\ 5 & 20 & 0 & 19 \\ 45 & 0 & 19 & 0 \end{pmatrix}$

Table 1.1. *Matrix representation of simple graphs*

1.6.3. *Connected graphs*

A graph G is said to be connected if there exists at least a path between each pair of vertices x and y:

$$\forall x, y \in V \exists \text{a path: } x \rightarrow \ldots \rightarrow y \qquad [1.4]$$

We can note that a basic result related to connected graphs is the following:

$$\text{If } e < n-1, \text{ then } G \text{ is not connected} \qquad [1.5]$$

Based on equation [1.5], we can clearly understand that if $e \geq n-1$, G can be either connected or not. In fact, if we observe the graph of Figure 1.3, condition $\underbrace{e}_{8} \geq \underbrace{n-1}_{6}$ holds, but the graph is not connected. Whenever a graph is connected, we can speak about finding shortest paths between pairs of vertices. Routing problems address such topics that are mainly about minimizing the number of vehicles used and determining the shortest itinerary for each vehicle. An itinerary can be either a circuit as is the case of the basic vehicle routing problem or a path when we speak about the open vehicle routing problem.

1.6.4. *Itineraries in a graph*

When trying to choose the most cost-efficient solution, we should handle a weighted graph.

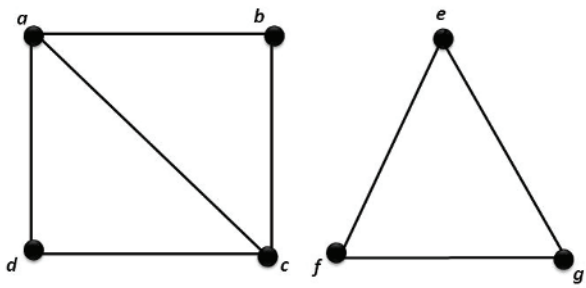

Figure 1.3. *A non-connected graph with* $n = 7$ *and* $e = 8$

A *path* is a succession of adjacent edges that starts from a predefined source x and ends at a destination y such that $x \neq y$. For an optimization problem, we generally speak about the least cost path.

A *circuit* is a path that starts and ends at the same vertex ($x = y$). The traveling salesman problem is the most famous optimization problem that tries to generate the least cost Hamiltonian circuit (a circuit that runs through all vertices of the graph) in the graph under study. Figure 2.1 reports a weighted connected graph with 7 vertices and 12 edges where the vertices correspond to the cities and the edges are direct routes weighted by their corresponding distances. The following are examples of itineraries:

Itinerary	Detail	Distance
Path	• $a \xrightarrow{10} b \xrightarrow{45} d \xrightarrow{12} e$	67
	• $a \xrightarrow{90} c \xrightarrow{24} e$	114
Circuit	• $a \xrightarrow{10} b \xrightarrow{45} e \xrightarrow{12} f \xrightarrow{24} c \xrightarrow{90} a$	181
	• $a \xrightarrow{90} c \xrightarrow{57} b \xrightarrow{10} a$	157

1.6.5. *Tree graphs*

A tree T is a connected graph that does not contain any circuit. The vertex set V contains:

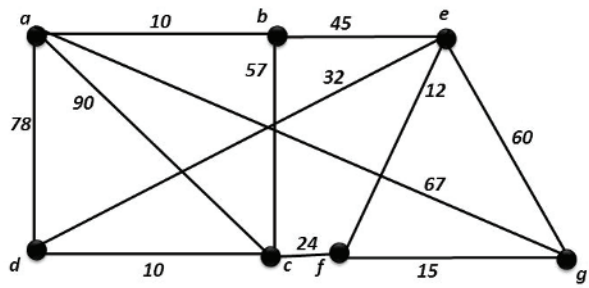

Figure 1.4. *A connex plutot connected graph with* $n = 7$ *and* $e = 12$

– intermediates nodes or branches: x is an intermediate node if $d_T(x) \geq 2$;

– leaves: x is a leave if $d_T(x) = 1$.

A graph $T = (V, E)$ is a tree if:

– there exists only one path between each pair of vertices. This can be explained by the fact that T should not contain any circuit; ‘

– $e = n - 1$. This result can be proved by induction:

Basis of induction:

The unit tree with $n = 1$ does not contain any edge $\begin{cases} n = 1 \\ e = 0 \end{cases}$
Thus, the equality $e = n - 1$ holds. Induction step:

Given a tree T with $e = n - 1$, if we increment the number of edges, three cases follow:

- n remains the same: the new edge links two existing vertices. The graph obtained is not a tree as it will give rise to a circuit.

- $n' \longleftarrow n + 1$: the new vertex becomes a new leaf. In the sequel, the new graph T' with n' vertices and e' edges remains a tree as $\begin{cases} T' \text{is connex} \\ e' = n' + 1 \end{cases}$

– $n \longleftarrow n+2$: the new edge is a disconnected component, hence, the graph obtained is not connected $\Rightarrow$ It is not a tree

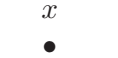

Figure 1.5. *A tree T with $n = 1$*

To sum up the above proof, when adding an edge to a tree, we should add exactly one vertex.

1.6.6. *The bipartite graphs*

– A graph G is said to be bipartite if the the vertex set V can be split into two separate subsets A and B such that:

- $A \cap B = \emptyset$;
- $A \cup B = V$;
- $\nexists \{x, y\} \in E | x \in A$ and $y \in B$.

A and B are disjoint sets of V such that vertices within a subset are not adjacent.

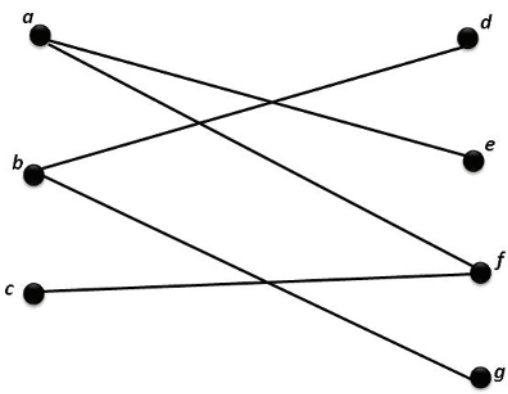

Figure 1.6. *A bipartite graph with $n = 7$ and $e = 5$*

– A complete bipartite graph connects all nodes of the first vertex set A to all nodes of the vertex set B. This yields to a total number of edges accounting to:

$$e = |A| \times |B| \quad [1.6]$$

Bipartite graphs are used to model packing problems such as the knapsack problem, the bin packing and their variants.

1.7. Conclusion

In this chapter, we discussed various aspects of an optimization problem and the main concepts of graph theory. We addressed the key issues to consider when solving a combinatorial optimization problem. The main optimization models for the decision-making were introduced as they provide tools for making optimal and promising decisions. Indeed, optimization models are closely linked to their mathematical formulation as an objective function and a set of system constraints, and hence the definition of the feasible space. The complexity theory and the optimization methods are linked in such a way that the choice of the solution method often depends on the problem complexity. In the case of problems in class $\mathcal{P}$, a polynomial algorithm can be used. However, for $\mathcal{NP}$-hard problems, two possibilities can be adopted. While exact methods are used for finding the optimal solution for small-sized problems, approximate methods are efficient for large-size instances.

The different definitions and concepts provided in this chapter are the foundation of the theoretical and applicative contributions to be evoked in the subsequent chapters of this book.