Appendix

A.1 Linear Programming

Consider program $P_{A.1}$ below which consists in minimizing the linear function $f(x_1, x_2, \ldots, x_n)$; the variables of this function, x_1, x_2,..., x_n, are either Boolean variables, or integer variables or non-negative real variables. These variables are subject to a set of linear constraints and the coefficients c_1, c_2,..., c_n, a_{i1}, a_{i2},..., a_{in} ($i = 1, \ldots, m$), b_1, b_2,..., b_m are arbitrary.

$$
P_{A.1} : \begin{cases} \min \quad f(x_1, x_2, \ldots, x_n) = c_1 x_1 + c_2 x_2 + \cdots + c_n x_n & & \\ \text{s.t.} \begin{cases} a_{i1} x_1 + a_{i2} x_2 + \cdots + a_{in} x_n \leq b_i & i = 1, \ldots, p & \text{(A.1.1)} \\ a_{i1} x_1 + a_{i2} x_2 + \cdots + a_{in} x_n = b_i & i = p+1, \ldots, m & \text{(A.1.2)} \\ x_j \in \{0, 1\} & j = 1, \ldots, r & \text{(A.1.3)} \\ x_j \in \mathbb{N} & j = r+1, \ldots, s & \text{(A.1.4)} \\ x_j \geq 0 & j = s+1, \ldots, n & \text{(A.1.5)} \end{cases} \end{cases}
$$

$P_{A.1}$ is a mathematical program, called a mixed-integer linear program; it can be written in the condensed form $P_{A.2}$.

$$
P_{A.2} : \begin{cases} \min \quad f(x_1, x_2, \ldots, x_n) = \sum_{j=1}^{n} c_j x_j & & \\ \text{s.t.} \begin{cases} \sum_{j=1}^{n} a_{ij} x_j \leq b_i & i = 1, \ldots, p & \text{(A.2.1)} \\ \sum_{j=1}^{n} a_{ij} x_j = b_i & i = p+1, \ldots, m & \text{(A.2.2)} \\ x_j \in \{0, 1\} & j = 1, \ldots, r & \text{(A.2.3)} \\ x_j \in \mathbb{N} & j = r+1, \ldots, s & \text{(A.2.4)} \\ x_j \geq 0 & j = s+1, \ldots, n & \text{(A.2.5)} \end{cases} \end{cases}
$$

Problem $P_{A.1}$ – or $P_{A.2}$ – consists in determining the values of variables x_1, x_2,..., x_n which respect their specificity, defined by constraints A.2.3–A.2.5, satisfy linear constraints A.2.1 and A.2.2, and minimize the linear economic function $\sum_{j=1}^{n} c_j x_j$. The linear constraints are either inequalities, $\sum_{j=1}^{n} a_{ij} x_j \leq b_i,\ i = 1, 2, \ldots, p,$ or equalities, $\sum_{j=1}^{n} a_{ij} x_j = b_i,\ i = p+1, \ldots, m$. As regards the specificity of the variables in program $P_{A.1}$, some of them can only take integer values; this is the case of variables $x_j, j = 1, \ldots, s$. Others can take any non-negative real values; this is the case of variables $x_j,\ j = s+1, \ldots, n$. Among the variables that can only take integer values, some can only take the values 0 or 1; this is the case of variables $x_j\,, j = 1, \ldots, r$. If all the variables of $P_{A.1}$ must only take integer values, we have an integer linear program. If all its variables must only take the values 0 or 1, it is a 0–1 linear program or a linear program in Boolean variables. It can always be assumed that all the variables in a mixed-integer linear program are positive or zero. This is because any variable that is not constrained in sign can be expressed as the difference between two non-negative variables. Thus, a real variable can be expressed as the difference between two positive or zero real variables and a variable belonging to the set of integers, as the difference between two variables belonging to the set of natural numbers.

Example A.1. Consider program $P_{A.3}$, which consists of minimizing a linear function of the five variables x_1, x_2, x_3, x_4, and x_5, subject to two linear constraints – one equality and one inequality. Variables x_1 and x_2 can only take the values 0 or 1, variable x_3 must take a positive or zero integer value, and both variables x_4 and x_5 must take real, positive or zero values.

$$P_{A.3}: \begin{cases} \min \quad f(x_1, x_2, x_3, x_4, x_5) = -x_1 - 3x_2 + 3x_3 - 4x_4 + 7x_5 & \\ \text{s.t.} \begin{cases} 2x_1 - 3x_2 + 3x_3 - 6x_4 - 2x_5 \leq 10 & \text{(A.3.1)} \\ x_1 - 4x_2 + 2x_3 - 5x_4 + 3x_5 = 14 & \text{(A.3.2)} \\ x_1, x_2 \in \{0, 1\} & \text{(A.3.3)} \\ x_3 \in \mathbb{N} & \text{(A.3.4)} \\ x_4, x_5 \geq 0 & \text{(A.3.5)} \end{cases} \end{cases}$$

Many software packages for solving linear programs are available. Using one of these software packages, the following optimal solution of $P_{A.3}$ is obtained: ($x_1 = 1$, $x_2 = 0$, $x_3 = 4$, $x_4 = 0.0714$, $x_5 = 1.7857$). This solution gives the economic function the value 23.2143. A solution that satisfies all the constraints is called a feasible solution. For example, the solution ($x_1 = 0$, $x_2 = 1$, $x_3 = 5$, $x_4 = 0.5$, $x_5 = 3.5$) is feasible but not optimal since it gives the economic function the value 34.5.

Some integer linear programs are easy to solve because any feasible basic solution of their continuous relaxation is an integer solution. A matrix is said to be totally unimodular if the determinants of all its square sub-matrices are 0, 1 or −1. The coefficients of such a matrix can, therefore, only take the values 0, 1 or −1. There are some simple characterizations of totally unimodular matrices. Consider

the set of solutions, assumed to be not empty, $\{x \in \mathbb{R}^n : Ax \leq b, x \geq 0\}$. In a general way, if A is totally unimodular and if all the entries of the vector b are integer, then $\{x \in \mathbb{R}^n : Ax \leq b, x \geq 0\}$ is an integral polyhedron, *i.e.*, a polyhedron whose vertices all have integer coordinates. Consider the mathematical program $\mathrm{P_{A.4}}$ and its continuous relaxation, $\mathrm{P_{A.5}}$. It is assumed that $\mathrm{P_{A.4}}$ admits an optimal solution.

$$\mathrm{P_{A.4}} : \begin{cases} \min \ \sum_{j=1}^{n} c_j x_j & \\ \text{s.t.} \begin{cases} \sum_{j=1}^{n} a_{ij} x_j \leq b_i & i = 1,\ldots,p \quad \text{(A.4.1)} \\ x_j \in \{0,1\} & j = 1,\ldots,r \quad \text{(A.4.2)} \\ x_j \in \mathbb{N} & j = r+1,\ldots,s \quad \text{(A.4.3)} \end{cases} \end{cases}$$

$$\mathrm{P_{A.5}} : \begin{cases} \min \ \sum_{j=1}^{n} c_j x_j & \\ \text{s.t.} \begin{cases} \sum_{j=1}^{n} a_{ij} x_j \leq b_i & i = 1,\ldots,p \quad \text{(A.5.1)} \\ x_j \leq 1 & j = 1,\ldots,r \quad \text{(A.5.2)} \\ x_j \geq 0 & j = 1,\ldots,s \quad \text{(A.5.3)} \end{cases} \end{cases}$$

If the matrix associated with constraints A.5.1 and A.5.2 is totally unimodular and if the coefficients b_i, $i = 1,\ldots, p$, are integers, then any basic solution of $\mathrm{P_{A.5}}$ is integer-valued, and this is the case, in particular, of its optimal solution(s). To solve the integer linear program $\mathrm{P_{A.4}}$, it is therefore sufficient to solve the continuous linear program $\mathrm{P_{A.5}}$.

There is an extensive literature on linear programming, a central problem in operational research. A few works, either entirely devoted to linear programming or more general but with parts devoted to linear programming, are mentioned below.

References and Further Reading

Baillargeon G. (1999) *Introduction à la programmation linéaire*. Edition SMG.

Billionnet A. (2017) *Optimisation discrète*. Dunod.

Brezinski C. (2000) *Initiation à la programmation linéaire et à l'algorithme du simplexe*. Ellipses.

Chen D.S., Batson R.G., Dang Y. (2010) *Applied integer programming: modeling and solution*. John Wiley & Sons.

Charon I., Germa A., Hudry O. (1996) *Méthodes d'optimisation combinatoire*. Masson.

Chrétienne P., Grandjean J.C., Pesqueux Y. (1980) *Algorithmes et pratique de la programmation linéaire*. Technip.

Chvatal V. (1983) *Linear Programming*. Freeman.

Dantzig G.B. (1963) *Linear programming and extensions*. Princeton University Press.

De Werra D. (1990) *Éléments de programmation linéaire avec application aux graphes*. Presses Polytechniques Romandes.

Eppen G.D., Gould F.J., Schmidt C.P., Moore J.H., Weatherford L.R. (1998) *Introductory management science*. Prentice Hall.

Faure R., Lemaire B., Picouleau C. (2014) *Précis de recherche opérationnelle*. Dunod.

Guéret C., Prins C., Sevaux M. (2000) *Programmation linéaire avec 65 problèmes d'optimisation modélisés et résolus avec Visual Xpress*. Eyrolles.
Hillier F.S., Lieberman G.J. (2015) *Introduction to operations research*. McGraw-Hill.
Jacquet-Lagrèze E. (1998) *Programmation linéaire, modélisation et mise en œuvre informatique*. P.I.Q poche.
Matousek J., Gärtner B. (2007) *Understanding and using linear programming*. Springer.
Minoux M. (2008) *Programmation mathématique, théorie et algorithmes*. Lavoisier.
Murty K.G. (1983) *Linear programming*. Wiley.
Nering E.D., Tucker A.W. (1993) *Linear programs and related problems*. Academic Press.
Padberg M. (1995) *Linear optimization and extensions*. Springer.
Roos C., Terlaky T., Vial J.-P. (1996) *Linear optimization with interior point methods*. Wiley.
Roseaux (collective of authors) (2003) Exercices et problèmes résolus de recherche opérationnelle (Tome 3), *Programmation linéaire et extensions, problèmes classiques*. Dunod.
Sakarovitch M. (1997) *Optimisation combinatoire*. Hermann.
Salkin H.M., Haas R., Mathur K. (1989) *Foundations of integer programming*. North-Holland.
Schrijver A. (1986) *Theory of linear and integer programming*. Wiley.
Sierksma G. (1996) *Linear and integer programming, theory and practice*. Marcel Dekker.
Simonnard M. (1962) *Programmation linéaire*. Dunod.
Teghem J. (2003) *Programmation linéaire*. Ellipses.
Vajda S. (1981) *Linear programming, algorithms and applications*. Chapman and Hall.
Vanderbei R.J. (1996) *Linear programming: foundations and extensions*. Kluwer Academic Publishers.
Vaserstein L.N. (2003) *Introduction to linear programming*, Prentice Hall.
Williams H.P. (1993) *Model solving in mathematical programming*. John Wiley.

A.2 Quadratic Programming

Linear programming is a powerful tool for formulating and solving a wide variety of optimization problems. However, in some cases, the economic function or constraints associated with the problems of interest do not possess the linearity property. These are referred to as non-linear optimization problems and non-linear mathematical programs. Solving a general non-linear mathematical program is a difficult task. Here we are interested in quadratic programs. Such a program is generally written as $\mathrm{P}_{\mathrm{A.6}}$.

$$
\mathrm{P}_{\mathrm{A.6}}: \begin{cases} \min \quad q(x_1, x_2, \ldots, x_n) = \displaystyle\sum_{j=1}^{n} a_j x_j + \sum_{i=1}^{n}\sum_{j=1}^{n} q_{ij} x_i x_j & \\ \text{s.t.} \begin{cases} \displaystyle\sum_{j=1}^{n} e_{kj} x_j + \sum_{i=1}^{n}\sum_{j=1}^{n} c_{kij} x_i x_j \leq b_k & k = 1, \ldots, p \quad \text{(A.6.1)} \\ \displaystyle\sum_{j=1}^{n} e_{kj} x_j + \sum_{i=1}^{n}\sum_{j=1}^{n} c_{kij} x_i x_j = b_k & k = p+1, \ldots, m \quad \text{(A.6.2)} \\ x_j \in \{0, 1\} & j = 1, \ldots, r \quad \text{(A.6.3)} \\ x_j \in \mathbb{N} & j = r+1, \ldots, s \quad \text{(A.6.4)} \\ x_j \geq 0 & j = s+1, \ldots, n \quad \text{(A.6.5)} \end{cases} \end{cases}
$$

Variables $x_1, x_2, \ldots, x_n$ are the variables of the problem; $a_j\,(j = 1, \ldots, n)$, $q_{ij}\,(i = 1, \ldots, n; j = 1, \ldots, n)$, $e_{kj}\,(k = 1, \ldots, m; j = 1, \ldots, n)$, $c_{kij}\,(k = 1, \ldots, m; i = 1, \ldots, n; j = 1, \ldots, n)$, and $b_k\,(k = 1, \ldots, m)$ are any given coefficients. Constraints A.6.1 are quadratic inequality constraints and constraints A.6.2 are quadratic equality constraints. Solving $\mathrm{P_{A.6}}$ is generally difficult, but there are many interesting and much easier special cases. Some of them are discussed below.

A.3 Convex Quadratic Programming

Consider program $\mathrm{P_{A.7}}$ that satisfies the following properties: the objective, $q(x) = q(x_1, x_2, \ldots, x_n)$, and the left-hand side of constraint A.7.1, $\sum_{j=1}^{n} e_{kj}x_j + \sum_{i=1}^{n}\sum_{j=1}^{n} c_{kij}x_i x_j$, are convex quadratic functions; the right-hand side of this constraint, b_k, is a positive or zero constant. Program $\mathrm{P_{A.7}}$ consists of minimizing a convex function over a convex domain; it is called a convex quadratic program. There are very efficient algorithms to solve it.

$$\mathrm{P_{A.7}}: \begin{cases} \min \quad q(x_1, x_2, \ldots, x_n) = \displaystyle\sum_{j=1}^{n} a_j x_j + \sum_{i=1}^{n}\sum_{j=1}^{n} q_{ij}x_i x_j & \\ \text{s.t.} \begin{vmatrix} \displaystyle\sum_{j=1}^{n} e_{kj}x_j + \sum_{i=1}^{n}\sum_{j=1}^{n} c_{kij}x_i x_j \le b_k & k = 1, \ldots, m \quad \text{(A.7.1)} \\ x_j \ge 0 & j = 1, \ldots, n \quad \text{(A.7.2)} \end{vmatrix} \end{cases}$$

Given n real variables, $x_1, x_2, \ldots, x_n$, an expression of the form $q(x) = \sum_{i=1}^{n} \sum_{j=1}^{n} x_i q_{ij} x_j$ is called a quadratic form. It is written, in matrix form, $q(x) = x^t Q\, x$ where x denotes the vector $(x_1, x_2, \ldots, x_n)$, and Q denotes a symmetric square matrix of dimension $n \times n$. The matrix Q is said to represent the quadratic form $q(x)$. Note that any quadratic form can be represented by one and only one symmetric matrix and that any symmetric matrix represents one and only one quadratic form. The symmetric matrix Q is said to be positive semidefinite if, for all $x \in \mathbb{R}^n$, $x^t Qx \ge 0$. By definition, the quadratic form $q(x) = x^t Qx$ is convex if the symmetric matrix Q is positive semidefinite. There are many characterizations of positive semidefinite matrices.

A special case of $\mathrm{P_{A.7}}$, which can be solved very efficiently, consists in minimizing a convex quadratic function subject to linear constraints. It corresponds to program $\mathrm{P_{A.8}}$ in which, now, b_k is a coefficient of any sign.

$$\mathrm{P_{A.8}}: \begin{cases} \min \quad q(x_1, x_2, \ldots, x_n) = \displaystyle\sum_{j=1}^{n} a_j x_j + \sum_{i=1}^{n}\sum_{j=1}^{n} q_{ij}x_i x_j & \\ \text{s.t.} \begin{vmatrix} \displaystyle\sum_{j=1}^{n} e_{kj}x_j \le b_k & k = 1, \ldots, m \quad \text{(A.8.1)} \\ x_j \ge 0 & j = 1, \ldots, n \quad \text{(A.8.2)} \end{vmatrix} \end{cases}$$

Example A.2. Consider the convex quadratic program $\mathrm{P}_{\mathrm{A.9}}$.

$$\mathrm{P}_{\mathrm{A.9}}: \begin{cases} \min \quad q(x) = -3x_1 - 2x_2 - 3x_3 - 10x_4 + 4x_5 + 2(x_2 - 1)^2 + 5(x_3 - 1)^2 \\ \quad + 2(x_4 - 1)^2 + (x_5 - 2)^2 \\ \text{s.t.} \begin{vmatrix} 2x_1 + 3x_2 + 3x_3 + 5x_4 - 2x_5 \leq 20 & \text{(A.9.1)} \\ x_1, x_2, x_3, x_4, x_5 \geq 0 & \text{(A.9.2)} \end{vmatrix} \end{cases}$$

Many software packages are available to solve $\mathrm{P}_{\mathrm{A.9}}$. The optimal solution obtained with one of these software packages is: ($x_1 = 5.6$, $x_2 = 0.375$, $x_3 = 0.85$, $x_4 = 1.625$, $x_5 = 1.5$). This solution gives the economic function the value -28.425.

There are many books and articles dealing with convex mathematical programming and, in particular, convex quadratic programming. Some of these publications are mentioned below.

References and Further Reading

Best M.J. (2017) *Quadratic programming with computer programs*. Taylor & Francis.
Fletcher R. (1971) A General quadratic programming algorithm, *IMA J. Appl. Math.* **7**, 76.
Gill P.E., Murray W., Wright M.H. (1986) *Practical optimization*. Elsevier.
Hertog D. den (1994) *Interior point approach to linear, quadratic and convex programming, algorithms and complexity*. Kluwer.
Mitra G. (1976) *Theory and application of mathematical programming*. Academic Press.
Nesterov Y. (2004) *Introductory lectures on convex optimization: basic course*. Springer-Verlag.
Quadri D., Soutil E. (2015) Reformulation and solution approach for non-separable integer quadratic programs, *J. Oper. Res. Soc.* **66**, 1270.
Wolf P. (1959) The simplex method for quadratic programming, *Econometrica* **27**, 382.

A.4 Mixed-Integer Quadratic Programs With Convex Continuous Relaxations

In such programs, the variables are either integer or real. They are generally written as $\mathrm{P}_{\mathrm{A.10}}$. In this program, the economic function and the left-hand side of constraint A.10.1 are convex quadratic functions, and b_k, $k = 1, 2, \ldots, m$, is a positive or zero constant.

$$\mathrm{P}_{\mathrm{A.10}}: \begin{cases} \min \quad q(x_1 x_2, \ldots, x_n) = \sum_{j=1}^{n} a_j x_j + \sum_{i=1}^{n} \sum_{j=1}^{n} q_{ij} x_i x_j \\ \text{s.t.} \begin{vmatrix} \sum_{j=1}^{n} e_{kj} x_j + \sum_{i=1}^{n} \sum_{j=1}^{n} c_{kij} x_i x_j \leq b_k & k = 1, \ldots, m & \text{(A.10.1)} \\ x_j \in \{0, 1\} & j = 1, \ldots, r & \text{(A.10.2)} \\ x_j \in \mathbb{N} & j = r+1, \ldots, s & \text{(A.10.3)} \\ x_j \geq 0 & j = s+1, \ldots, n & \text{(A.10.4)} \end{vmatrix} \end{cases}$$

The continuous relaxation of $\mathrm{P}_{\mathrm{A.10}}$ is obtained by relaxing the integrality constraints. Specifically, the constraint $x_j \in \{0, 1\}$, $j = 1, \ldots, r$, is replaced by the

constraint $0 \leq x_j \leq 1$, $j = 1,\ldots,r$, and the constraint $x_j \in \mathbb{N}$, $j = r+1,\ldots,s$, is replaced by the constraint $x_j \geq 0$, $j = r+1,\ldots,s$. The resulting program is called the continuous relaxation of $\mathrm{P_{A.10}}$, and it is easy to see that it is a convex quadratic program. There are efficient algorithms for solving mixed-integer quadratic programs with convex continuous relaxation. They are in fact based on implicit enumeration methods that require, at each node of the search tree, the resolution of a continuous relaxation, which in this case can be done efficiently because of convexity properties.

There are also several methods for converting a mixed-integer quadratic program whose continuous relaxation is not convex into a mixed-integer quadratic program whose continuous relaxation is convex. Some examples of these transformations are presented in the following section.

A.5 Quadratic Programming in 0–1 Variables

Quadratic programs in 0–1 variables allow the formulation of a large number of combinatorial optimization problems in various fields. The general form of these programs is given by $\mathrm{P_{A.11}}$.

$$\mathrm{P_{A.11}}: \begin{cases} \min \quad q(x_1, x_2,\ldots, x_n) = \sum_{j=1}^{n} a_j x_j + \sum_{i=1}^{n}\sum_{j=1}^{n} q_{ij} x_i x_j & \\ \text{s.t.} \begin{cases} \sum_{j=1}^{n} e_{kj} x_j + \sum_{i=1}^{n}\sum_{j=1}^{n} c_{kij} x_i x_j \leq b_k & k = 1,\ldots,p \quad \text{(A.11.1)} \\ \sum_{j=1}^{n} e_{kj} x_j + \sum_{i=1}^{n}\sum_{j=1}^{n} c_{kij} x_i x_j = b_k & k = p+1,\ldots,m \quad \text{(A.11.2)} \\ x_j \in \{0,1\} & j = 1,\ldots,n \quad \text{(A.11.3)} \end{cases} \end{cases}$$

There are many methods to solve this type of program: linearization methods and convexification methods. The linearization methods consist of transforming $\mathrm{P_{A.11}}$ into a mixed-integer linear program, using additional variables. The convexification methods consist in transforming $\mathrm{P_{A.11}}$ into a quadratic problem whose continuous relaxation is convex, possibly using additional variables. Some solvers accept programs $\mathrm{P_{A.11}}$ directly, and automatically perform a pre-processing – linearization or convexification – that transforms the program into an equivalent one whose continuous relaxation is a linear or a convex quadratic program.

A.5.1 Linearizations

One way to solve $\mathrm{P_{A.11}}$ is to linearize it and then solve the mixed-integer linear program thus constructed using a mixed-integer linear programming solver. A first linearization method consists in replacing, in the economic function and in the constraints, each product of variables $x_i x_j$ by variable y_{ij}, and in adding to the obtained program constraints which force variable y_{ij} to be equal to product $x_i x_j$. Thus, we obtain program $\mathrm{P_{A.12}}$ in which IJ designates the set of index pairs $(i,j) \in$

$\{1,\ldots,n\}^2$ such that the product $x_i x_j$ appears either in the economic function or in the constraints of $\mathrm{P}_{\mathrm{A.11}}$.

$$\mathrm{P}_{\mathrm{A.12}}: \begin{cases} \min \quad q(x_1, x_2, \ldots, x_n) = \sum_{j=1}^{n} a_j x_j + \sum_{i=1}^{n}\sum_{j=1}^{n} q_{ij} y_{ij} & \\ \text{s.t.} \begin{cases} \sum_{j=1}^{n} e_{kj} x_j + \sum_{i=1}^{n}\sum_{j=1}^{n} c_{kij} y_{ij} \le b_k & k = 1,\ldots,p \quad (\text{A.12.1}) \\ \sum_{j=1}^{n} e_{kj} x_j + \sum_{i=1}^{n}\sum_{j=1}^{n} c_{kij} y_{ij} = b_k & k = p+1,\ldots,m \quad (\text{A.12.2}) \\ y_{ij} \le x_i;\ y_{ij} \le x_j;\ 1 - x_i - x_j + y_{ij} \ge 0 & (i,j) \in \mathrm{IJ} \quad (\text{A.12.3}) \\ y_{ij} \ge 0 & (i,j) \in \mathrm{IJ} \quad (\text{A.12.4}) \\ x_j \in \{0,1\} & j = 1,\ldots,n \quad (\text{A.12.5}) \end{cases} \end{cases}$$

It is easy to verify, by examining the two possible values of variables x_j, $j = 1,\ldots,n$, that any feasible solution of $\mathrm{P}_{\mathrm{A.12}}$ satisfies $y_{ij} = x_i x_j$.

Example A.3. Consider program $\mathrm{P}_{\mathrm{A.13}}$ which consists of minimizing a quadratic function of 5 Boolean variables subject to one linear constraint.

$$\mathrm{P}_{\mathrm{A.13}}: \begin{cases} \min \quad -3x_1 - 2x_2 + 3x_3 - 10x_4 + 4x_5 + 2\,x_1 x_2 & \\ \quad -4x_1 x_3 + 5x_2 x_3 + 6x_2 x_5 - 4x_3 x_4 + 6x_3 x_5 - 2x_4 x_5 & \\ \text{s.t.} \begin{cases} 2x_1 + 3x_2 + 3x_3 + 5x_4 - 2x_5 \le 20 & (\text{A.13.1}) \\ x_1, x_2, x_3, x_4, x_5 \in \{0,1\} & (\text{A.13.2}) \end{cases} \end{cases}$$

By applying the linearization shown above, we obtain program $\mathrm{P}_{\mathrm{A.14}}$.

$$\mathrm{P}_{\mathrm{A.14}}: \begin{cases} \min \quad -3x_1 - 2x_2 + 3x_3 - 10x_4 + 4x_5 & \\ \quad + 2y_{12} - 4y_{13} + 5y_{23} + 6y_{25} - 4y_{34} + 6y_{35} - 2y_{45} & \\ \text{s.t.} \begin{cases} 2x_1 + 3x_2 + 3x_3 + 5x_4 - 2x_5 \le 20 \quad (\text{A.14.1}) & | \quad y_{34} \le x_3;\ y_{34} \le x_4 \quad (\text{A.14.6}) \\ 1 - x_1 - x_2 + y_{12} \ge 0 \quad (\text{A.14.2}) & | \quad 1 - x_3 - x_5 + y_{35} \ge 0 \quad (\text{A.14.7}) \\ y_{13} \le x_1;\ y_{13} \le x_3 \quad (\text{A.14.3}) & | \quad y_{45} \le x_4;\ y_{45} \le x_5 \quad (\text{A.14.8}) \\ 1 - x_2 - x_3 + y_{23} \ge 0 \quad (\text{A.14.4}) & | \quad x_1, x_2, x_3, x_4, x_5 \in \{0,1\} \quad (\text{A.14.9}) \\ 1 - x_2 - x_5 + y_{25} \ge 0 \quad (\text{A.14.5}) & | \quad y_{12}, y_{13}, y_{23}, y_{25}, y_{34}, y_{35}, y_{45} \ge 0 \quad (\text{A.14.10}) \end{cases} \end{cases}$$

Note that, given the signs of the coefficients of variables y_{ij} in the economic function, some linearization constraints are unnecessary. An optimal solution for $\mathrm{P}_{\mathrm{A.14}}$ is: ($x_1 = 1$, $x_2 = 0$, $x_3 = 1$, $x_4 = 1$, $x_5 = 0$). This solution gives the economic function the value -18.

We now present a second linearization method by applying it to program $\mathrm{P}_{\mathrm{A.15}}$, which consists of minimizing a quadratic economic function whose variables are subject to linear constraints. Here, we assume that the quadratic part of the economic function is written as $\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} q_{ij} x_i x_j$. Note that, since $x_i^2 = x_i$, we can assume that the economic function does not have any terms x_i^2.

$$\mathrm{P_{A.15}}: \begin{cases} \min \quad q(x_1, x_2, \ldots, x_n) = \sum_{j=1}^{n} a_j x_j + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} q_{ij} x_i x_j \\ \text{s.t.} \begin{vmatrix} \sum_{j=1}^{n} e_{kj} x_j \leq b_k & k = 1, \ldots, m & \text{(A.15.1)} \\ x_j \in \{0,1\} & j = 1, \ldots, n & \text{(A.15.2)} \end{vmatrix} \end{cases}$$

This second linearization consists in rewriting the economic function of $\mathrm{P_{A.15}}$ by factoring variables x_i in the quadratic part of this function and then replacing, for all $i = 1, \ldots, n-1$, the expression $x_i \sum_{j=i+1}^{n} q_{ij} x_j$ by variable z_i. By proceeding in this way, the economic function is written $\sum_{j=1}^{n} a_j x_j + \sum_{i=1}^{n-1} z_i$. We must then add constraints A.16.2 and A.16.3 to force variable z_i to take, at the optimum of the program obtained, the value of the expression $x_i \sum_{j=i+1}^{n} q_{ij} x_j$. We finally obtain program $\mathrm{P_{A.16}}$.

$$\mathrm{P_{A.16}}: \begin{cases} \min \quad q(x_1, x_2, \ldots, x_n) = \sum_{j=1}^{n} a_j x_j + \sum_{i=1}^{n-1} z_i \\ \text{s.t.} \begin{vmatrix} \sum_{j=1}^{n} e_{kj} x_j \leq b_k & k = 1, \ldots, m & \text{(A.16.1)} \\ z_i \geq x_i \sum_{\substack{j = i+1, \ldots, n : \\ q_{ij} < 0}} q_{ij} & i = 1, \ldots, n-1 & \text{(A.16.2)} \\ z_i \geq \sum_{j=i+1, \ldots, n} q_{ij} x_j - (1 - x_i) \sum_{\substack{j = i+1, \ldots, n : \\ q_{ij} > 0}} q_{ij} & i = 1, \ldots, n-1 & \text{(A.16.3)} \\ x_j \in \{0,1\} & j = 1, \ldots, n & \text{(A.16.4)} \\ z_i \in \mathbb{R} & i = 1, \ldots, n-1 & \text{(A.16.5)} \end{vmatrix} \end{cases}$$

Let us look at constraints A.16.2 and A.16.3. If $x_i = 0$, constraint A.16.2 becomes $z_i \geq 0$ and constraint A.16.3, $z_i \geq \sum_{j=i+1, \ldots, n} q_{ij} x_j - \sum_{j=i+1, \ldots, n : q_{ij} > 0} q_{ij}$. The right-hand side of the latter constraint is negative or zero whatever the values taken by variables x_j. Finally, in this case and because we are seeking to minimize variable z_i, this variable takes the value 0 at the optimum of $\mathrm{P_{A.16}}$. If $x_i = 1$, constraint A.16.2 becomes $z_i \geq \sum_{j=i+1, \ldots, n : q_{ij} < 0} q_{ij}$ and constraint A.16.3 becomes $z_i \geq \sum_{j=i+1, \ldots, n} q_{ij} x_j$. Note that $\sum_{j=i+1, \ldots, n} q_{ij} x_j \geq \sum_{j=i+1, \ldots, n : q_{ij} < 0} q_{ij}$ whatever the values taken by the Boolean variables x_j. Because we are seeking to minimize z_i, this variable takes, at the optimum of $\mathrm{P_{A.16}}$, the greater of the 2 values $\sum_{j=i+1, \ldots, n : q_{ij} < 0} q_{ij}$, $\sum_{j=i+1, \ldots, n} q_{ij} x_j$, that is to say the value $\sum_{j=i+1, \ldots, n} q_{ij} x_j$. Finally, at the optimum of $\mathrm{P_{A.16}}$, $z_i = x_i \sum_{j=i+1, \ldots, n} q_{ij} x_j$. Note that the technique just presented for linearizing the economic function could be applied in the same way to linearize quadratic constraints.

Example A.4. Let us go back to $\mathrm{P_{A.13}}$ and apply this second linearization method to it. The quadratic part of the economic function can be rewritten $x_1(2x_2 - 4x_3) + x_2(5x_3 + 6x_5) + x_3(-4x_4 + 6x_5) + x_4(-2x_5)$. We thus obtain the mixed-integer linear program $\mathrm{P_{A.17}}$ which is equivalent to program $\mathrm{P_{A.13}}$.

$$\mathrm{P_{A.17}}: \begin{cases} \min \quad q(x_1, x_2, \ldots, x_5) = -3x_1 - 2x_2 + 3x_3 - 10x_4 + 4x_5 + z_1 + z_2 + z_3 + z_4 \\ \text{s.t.} \begin{vmatrix} 2x_1 + 3x_2 + 3x_3 + 5x_4 - 2x_5 \leq 20 & | & z_4 \geq -2x_4;\ z_4 \geq -2x_5 \\ z_1 \geq -4x_1;\ z_1 \geq 2x_2 - 4x_3 - 2(1 - x_1) & | & x_1, x_2, x_3, x_4, x_5 \in \{0,1\} \\ z_2 \geq 0;\ z_2 \geq 5x_3 + 6x_5 - 11(1 - x_2) & | & z_1, z_2, z_3, z_4, z_5 \in \mathbb{R} \\ z_3 \geq -4x_3;\ z_3 \geq -4x_4 + 6x_5 - 6(1 - x_3) & | \end{vmatrix} \end{cases}$$

The optimal solution for $\mathrm{P_{A.17}}$ is: $(x_1 = 1, x_2 = 0, x_3 = 1, x_4 = 1, x_5 = 0, z_1 = -4, z_2 = 0, z_3 = -4,\ z_4 = 0)$, and this solution gives the economic function the value -18.

A.5.2 Convexifications

There are several methods to transform $\mathrm{P_{A.11}}$ into an equivalent quadratic program with a convex continuous relaxation. Examples of these methods are given below, in the particular case of minimizing a quadratic function whose variables are subject to linear constraints. Let us therefore consider program $\mathrm{P_{A.18}}$.

$$\mathrm{P_{A.18}}: \begin{cases} \min \quad q(x_1, x_2, \ldots, x_n) = \sum_{j=1}^{n} a_j x_j + \sum_{i=1}^{n} \sum_{j=1}^{n} q_{ij} x_i x_j \\ \text{s.t.} \begin{vmatrix} \sum_{j=1}^{n} e_{kj} x_j \leq b_k & k = 1, \ldots, m & \text{(A.18.1)} \\ x_j \in \{0,1\} & j = 1, \ldots, n & \text{(A.18.2)} \end{vmatrix} \end{cases}$$

Let $Q^+ = \{(i,j) : q_{ij} > 0\}$ and $Q^- = \{(i,j) : q_{ij} < 0\}$. Since variables x_i, $i = 1,\ldots, n$, are Boolean variables, the function $\tilde{q}$ below is equal to the economic function of $\mathrm{P_{A.18}}$, $q(x_1, x_2, \ldots, x_n)$, for all $x \in \{0,1\}^n$.

$$\tilde{q}(x_1, x_2, \ldots, x_n) = \sum_{j=1}^{n} a_j x_j + \frac{1}{2} \sum_{(i,j) \in Q^+} q_{ij}((x_i + x_j)^2 - (x_i + x_j)) - \frac{1}{2} \sum_{(i,j) \in Q^-} q_{ij}((x_i - x_j)^2 - (x_i + x_j))$$

In addition, $\tilde{q}(x_1, x_2, \ldots, x_n)$ is a convex function. Program $\mathrm{P_{A.18}}$ is equivalent to program $\mathrm{P_{A.19}}$ and the continuous relaxation of $\mathrm{P_{A.19}}$ is a convex quadratic program.

$$\mathrm{P_{A.19}}: \begin{cases} \min \quad \tilde{q}(x_1, x_2, \ldots, x_n) \\ \text{s.t.} \mid \text{(A.18.1)}, \text{(A.18.2)} \end{cases}$$

Let us now consider another convexification method. Let us rewrite program $\mathrm{P_{A.18}}$ using a matrix writing of the quadratic part of the economic function. We obtain program $\mathrm{P_{A.20}}$ in which the matrix M, of general term m_{ij}, is the symmetric matrix associated with the quadratic form $\sum_{i=1}^{n} \sum_{j=1}^{n} q_{ij} x_i x_j$. We have, for all $i \leq j$, $m_{ji} = m_{ij} = (q_{ij} + q_{ji})/2$.

$$\mathrm{P}_{\mathrm{A.20}}: \begin{cases} \min \;\; q(x_1, x_2, \ldots, x_n) = \sum_{j=1}^{n} a_j x_j \; + x^t \, M x \\ \text{s.t.} \mid (\mathrm{A.18.1}), (\mathrm{A.18.2}) \end{cases}$$

One way of reformulating program $\mathrm{P}_{\mathrm{A.20}}$ – and thus program $\mathrm{P}_{\mathrm{A.18}}$ – as an equivalent quadratic program whose continuous relaxation is a convex quadratic program, is to add to the function $q(x_1, x_2, \ldots, x_n)$ – assumed to be non-convex – the quantity $\sum_{j=1,\ldots,n} \lambda_{\min}(x_j^2 - x_j)$ where $\lambda_{\min}$ denotes the absolute value of the smallest eigenvalue of the square and symmetric matrix, M. This gives the convex quadratic function $\hat{q}(x_1, x_2, \ldots, x_n) = q(x_1, x_2, \ldots, x_n) + \sum_{j=1,\ldots,n} \lambda_{\min}\left(x_j^2 - x_j\right)$ which is equal to $q(x_1, x_2, \ldots, x_n)$ for all $x \in \{0,1\}^n$. This method requires a relatively simple pre-processing of program $\mathrm{P}_{\mathrm{A.20}}$, the calculation of the smallest eigenvalue of the matrix M. Finally, one can thus solve $\mathrm{P}_{\mathrm{A.20}}$ by solving the quadratic program $\mathrm{P}_{\mathrm{A.21}}$ whose continuous relaxation is convex.

$$\mathrm{P}_{\mathrm{A.21}}: \begin{cases} \min \;\; q(x_1, x_2, \ldots, x_n) + \sum_{j=1}^{n} \lambda_{\min}\left(x_j^2 - x_j\right) \\ \text{s.t.} \mid (\mathrm{A.18.1}), (\mathrm{A.18.2}) \end{cases}$$

Example A.5. Let us consider again program $\mathrm{P}_{\mathrm{A.13}}$ and transform it into a quadratic program whose continuous relaxation is a convex program. To do this, let us add to the economic function the quantity – zero for any feasible solution – $\lambda_{\min} \sum_{j=1}^{n} \left(x_j^2 - x_j\right)$ where $\lambda_{\min}$ is equal to the absolute value of the smallest eigenvalue of the matrix associated with the quadratic form $2x_1x_2 - 4x_1x_3 + 5x_2x_3 + 6x_2x_5 - 4x_3x_4 + 6x_3x_5 - 2x_4x_5$, *i.e.*, of the matrix

$$M = \begin{pmatrix} 0 & 1 & -2 & 0 & 0 \\ 1 & 0 & 2.5 & 0 & 3 \\ -2 & 2.5 & 0 & -2 & 3 \\ 0 & 0 & -2 & 0 & -1 \\ 0 & 3 & 3 & -1 & 0 \end{pmatrix}.$$

The smallest eigenvalue of M is equal to -4.19. Program $\mathrm{P}_{\mathrm{A.13}}$ is therefore equivalent to program $\mathrm{P}_{\mathrm{A.22}}$ whose continuous relaxation is convex.

$$\mathrm{P}_{\mathrm{A.22}}: \begin{cases} \min \;\; -3x_1 - 2x_2 + 3x_3 - 10x_4 + 4x_5 + 2x_1x_2 \\ \quad -4x_1x_3 + 5x_2x_3 + 6x_2x_5 - 4x_3x_4 + 6x_3x_5 - 2x_4x_5 \\ \quad + 4.2 \sum_{i=1}^{n} \left(x_i^2 - x_i\right) \\ \text{s.t.} \left| \begin{array}{ll} 2x_1 + 3x_2 + 3x_3 + 5x_4 - 2x_5 \leq 20 & (\mathrm{A.22.1}) \\ x_1, x_2, x_3, x_4, x_5 \in \{0,1\} & (\mathrm{A.22.2}) \end{array} \right. \end{cases}$$

There are other, more elaborate pre-treatments of program $P_{A.18}$ – whose continuous relaxation is non-convex – allowing it to be rewritten as an equivalent quadratic program whose continuous relaxation is a convex quadratic program. These methods, which are based on positive semidefinite programming, also allow the processing of quadratic programs containing simultaneously Boolean variables, integer variables and real variables. There are many publications dealing with these linearization and convexification methods. Some of them are mentioned below.

References and Further Reading

Adams W.P., Sherali H.D. (1990) Linearization strategies for a class of zero–one mixed integer programming problems, *Oper. Res.* **38**, 217.

Adams W.P., Forrester R.J., Glover F.W. (2004) Comparisons and enhancement strategies for linearizing mixed 0–1 quadratic programs, *Discr. Opti.* **1**, 99.

Balas E., Mazzola J.B. (1984a) Nonlinear 0–1 programming: I. Linearization techniques, *Math. Prog.* **30**, 1.

Balas E., Mazzola J.B. (1984b) Nonlinear 0–1 programming: II, Dominance relations and algorithms, *Math. Prog.* **30**, 22.

Billionnet A., Elloumi S. (2007) Using a mixed integer quadratic programming solver for the unconstrained quadratic 0–1 problem, *Math. Prog.* **109**, 55.

Billionnet A., Elloumi S., Plateau M. (2009) Improving the performance of standard solvers for quadratic 0–1 programs by a tight convex reformulation: the QCR method, *Discr. Appl. Math.* **157**, 1185.

Billionnet A., Elloumi S., Lambert A. (2012) Extending the QCR method to the case of general mixed integer programs, *Math. Prog.* **131**, 381.

Billionnet A., Elloumi S., Lambert A. (2016) Exact quadratic convex reformulations of mixed-integer quadratically constrained problems, *Math. Prog.* **158**, 235.

Billionnet A., Elloumi S., Lambert A., Wiegele A. (2017) Using a conic bundle method to accelerate both phases of a quadratic convex reformulation,. *Info. J. Comp.* **29**, 318.

Chardaire P., Sutter A. (1995) A decomposition method for quadratic zero–one programming, *Manage. Sci.* **41**, 704.

Chen D.S., Batson R.G., Dang Y. (2010) *Applied integer programming: modeling and solution.* John Wiley & Sons.

Elloumi S., Faye A., Soutif E. (2000) Decomposition and linearization for 0–1 quadratic programming, *Ann. Oper. Res.* **99**, 79.

Faye A., Roupin F. (2007) Partial Lagrangian relaxation for general quadratic programming, *4OR* **5**, 75.

Fortet R. (1959) L'algèbre de Boole et ses applications en recherche opérationnelle, *Cahiers du Centre d'Etudes de Recherche Opérationnelle* **1**, 5.

Fortet R. (1960) Applications de l'algèbre de Boole en recherche opérationnelle, *Revue Française de Recherche Opérationnelle* **4**, 17.

Glover F., Woolsey E. (1973) Further reduction of zero–one polynomial programming to zero–one linear programming problems, *Oper. Res.* **21**, 156.

Glover F., Woolsey E. (1974) Converting the 0–1 polynomial programming problem to a linear 0–1 program, *Oper. Res.* **22**, 180.

Hammer P.L., Rudeanu S. (1968) *Boolean methods in operations research and related areas.* Springer.

Hammer P.L., Rubin A.A. (1970) Some remarks on quadratic programming with 0–1 variables, *RIRO* **3**, 67.

Rodriguez-Heck E. (2018) Linear and quadratic reformulations of nonlinear optimization problems in binary variables, Thèse de l'Univesité de Liège.

Sherali H.D., Adams W.P. (1986) A tight linearization and an algorithm for 0–1 quadratic programming problems, *Manage. Sci.* **32**, 1274.

A.6 Fractional Programming

The general fractional optimization problem can be written in the form of the mathematical program $\mathrm{P}_{\mathrm{A.23}}$.

$$\mathrm{P}_{\mathrm{A.23}} : \begin{cases} \max \ \ f(x)/g(x) \\ \text{s.t.} \ \ x \in X \end{cases}$$

The set X is a compact, non-empty subset of $\mathbb{R}^n$. The functions $f(x)$ and $g(x)$ are continuous functions with real values defined on the set X. It is assumed here that $g(x) > 0$ for any x belonging to X. There are many methods to solve this problem. We present below one of these methods, the Dinkelbach algorithm.

Let λ be a parameter belonging to the set of real numbers. Let us consider the parametric problem $\mathrm{P}_{\mathrm{A.24}}(\lambda)$ associated with $\mathrm{P}_{\mathrm{A.23}}$.

$$\mathrm{P}_{\mathrm{A.24}}(\lambda) : \begin{cases} \max \ \ f(x) - \lambda \ g(x) \\ \text{s.t.} \ \ x \in X \end{cases}$$

Let us denote by $v\,(\lambda)$ the optimal value of $\mathrm{P}_{\mathrm{A.24}}(\lambda)$ and by x^*_λ, an optimal solution to this program. We can prove that $v\,(\lambda) = 0$ if and only if λ is the optimal value of $\mathrm{P}_{\mathrm{A.23}}$ and x^*_λ, an optimal solution to this problem. Thus, we obtain another formulation of program $\mathrm{P}_{\mathrm{A.23}}$:

Find $\lambda \in R$ such that $v\,(\lambda) = 0$, where $v\,(\lambda) = \max\{f(x) - \lambda g(x) : x \in X\}$.

From this formulation, we will be able to build algorithms to solve $\mathrm{P}_{\mathrm{A.23}}$, based on classical methods to determine the root of a function – the Newton method. This is the case of the Dinkelbach algorithm presented below.

The Dinkelbach Algorithm

Step 1. $\lambda \leftarrow f(x_0)/g(x_0)$ where x_0 is a point of X.

Step 2. calculate $v(\lambda) = \max\{\, f(x) - \lambda g(x) : x \in X\}$ and let x_λ be such that $v(\lambda) = f(x_\lambda) - \lambda g(x_\lambda)$.

Step 3. **if** $v(\lambda) \neq 0$ **then** $\lambda \leftarrow f(x_\lambda)/g(x_\lambda)$ and go to 2 **else** x_λ is an optimal solution **endif.**

The difficulty of this algorithm depends on the difficulty of the optimization problem of Step 2.

In the case where the functions $f(x)$ and $g(x)$ are linear or affine and X is a convex polyhedron, $\mathrm{P}_{\mathrm{A.23}}$ is a linear or hyperbolic fractional optimization problem. It is written as $\mathrm{P}_{\mathrm{A.25}}$.

$$\mathrm{P}_{\mathrm{A.25}} : \begin{cases} \max \ \ \left(b_0 + \sum_{j=1}^n b_j x_j\right) \Big/ \left(c_0 + \sum_{j=1}^n c_j x_j\right) \\ \text{s.t.} \left| \begin{array}{ll} \sum\limits_{j=1}^{n} a_{ij} x_j \leq d_i & i = 1,\ldots, m \quad \text{(A.25.1)} \\ x_j \geq 0 & j = 1,\ldots, n \quad \text{(A.25.2)} \end{array} \right. \end{cases}$$

where b_0, c_0, b_j $(j = 1,\ldots, n)$, c_j $(j = 1,\ldots, n)$, and $a_{ij} (i = 1,\ldots, m, j = 1,\ldots, n)$ are real coefficients such that, for any feasible solution of $\mathrm{P}_{\mathrm{A.25}}$, $c_0 + \sum_{j=1}^n c_j x_j > 0$. Program

$P_{A.25}$ can be solved by the Dinkelbach algorithm. In this case, the optimization problem of Step 2 consists in solving a continuous linear program. However, by performing the variable changes $y_j = x_j/(c_0 + \sum_{j=1}^{n} c_j x_j)$, $j = 1,\ldots,n$, and $t = 1/(c_0 + \sum_{j=1}^{n} c_j x_j)$, $P_{A.25}$ can be rewritten as the equivalent linear program $P_{A.26}$.

$$
P_{A.26}: \begin{cases} \max \quad b_0 t + \sum_{j=1}^{n} b_j y_j & \\ \text{s.t.} \begin{cases} c_0 t + \sum_{j=1}^{n} c_j y_j = 1 & & \text{(A.26.1)} \\ \sum_{j=1}^{n} a_{ij} y_j \le d_i t & i = 1,\ldots,m & \text{(A.26.2)} \\ y_j \ge 0 & j = 1,\ldots,n & \text{(A.26.3)} \\ t \ge 0 & & \text{(A.26.4)} \end{cases} \end{cases}
$$

Let us now consider the mixed-integer linear fractional optimization problem $P_{A.27}$.

$$
P_{A.27}: \begin{cases} \max \quad \left(b_0 + \sum_{j=1}^{n} b_j x_j\right) \Big/ \left(c_0 + \sum_{j=1}^{n} c_j x_j\right) & \\ \text{s.t.} \begin{cases} \sum_{j=1}^{n} a_{ij} x_j \le d_i & i = 1,\ldots,m & \text{(A.27.1)} \\ x_j \ge 0 & j = 1,\ldots,p & \text{(A.27.2)} \\ x_j \in \{0,1\} & j = p+1,\ldots,n & \text{(A.27.3)} \end{cases} \end{cases}
$$

As before, it is assumed that $c_0 + \sum_{j=1}^{n} c_j x_j > 0$ for any feasible solution. The Dinkelbach algorithm, presented above, can be used to solve $P_{A.27}$. In this case, X is defined by Constraints A.27.1–A.27.3, and Step 2 of the algorithm consists in solving a mixed-integer linear program.

Fractional optimization problems are very diverse. For example, one can look at the ratio of two quadratic functions or at the sum of several ratios. For more information on fractional optimization, the reader can consult the references cited below.

References and Further Reading

Bajalinov E.B. (2003) *Linear-fractional programming: theory, methods, applications and software.* Springer Science + Business Media.

Billionnet A. (2002) Approximate and exact solution methods for the hyperbolic 0–1 knapsack problem, *INFOR: Info. Syst. Oper. Res.* **40**, 97.

Billionnet A., Djebali K. (2006) Résolution d'un problème combinatoire fractionnaire par la programmation linéaire mixte, *RAIRO-Oper. Res.* **40**, 97.

Borrero J.S., Gillen C., Prokopyev O.A. (2016) A simple technique to improve linearized reformulations of fractional (hyperbolic) 0–1 programming problems, *Oper. Res. Lett.* **44**, 479.

Charnes A., Cooper W.W. (1962) Programming with linear fractional functionals, *Naval Res. Logis. Quart.* **9**, 181.

Craven B.D. (1988) *Fractional programming.* Heldermann Verlag.

Dinkelbach W. (1967) On nonlinear fractional programming, *Manage. Sci.* **13**, 492.

Frenk J.B.G., Schaible S. (2005) Fractional programming. *Handbook of generalized convexity and generalized monotonicity* (N. Hadjisavvas, S.Komlósi, S. Schaible, Eds). Nonconvex optimization and its applications, Springer, p. 76.

Freund R.W., Jarre F. (2001) Solving the sum-of-ratios problem by an interior point method, *J. Global Opti.* **19**, 83.

Jeflea A. (2003) A parametric study for solving nonlinear fractional problems, *An. St. Univ. Ovidius Constanta* **11**, 87.

Konno H. (2001) Minimization of the sum of several linear fractional functions, *Generalized convexity and generalized monotonicity* (N. Hadjisavvas, J.E. Martínez-Legaz, J.P. Penot, Eds.) Lecture Notes in Economics and Mathematical Systems. Springer Verlag, Vol. 502, pp. 3–20.

Kuno T., (2002) A branch-and-bound algorithm for maximizing the sum of several linear ratios, *J. Global Opti.* **22**, 155.

Li H. (1994) A Global approach for general fractional programming, *Eur. J. Oper. Res.* **73**, 590.

Nagih A., Plateau G. (1999) Problèmes fractionnaires: tour d'horizon sur les applications et méthodes de resolution, *RAIRO – Oper. Res.* **33**, 383.

Radzik T. (1998) Fractional combinatorial optimization, *Handbook of combinatorial optimization* (D.-Z. Du, P.M. Pardalos). Kluwer Academic Publishers, pp. 429–478.

Rodenas R.G., Lopez M.L., Verastegui D. (1999) Extensions of Dinkelbach's algorithm for solving non-linear fractional programming problems, *Sociedad de Estadistica e Investigacion Operativa* **7**, 33.

Saipe A.L. (1975) Solving a 0–1 hyperbolic program by branch-and-bound, *Naval Res. Logis. Quart.* **22**, 397.

Schaible S., Ibaraki T. (1983) Fractional programming, *Eur. J. Oper. Res.* **12**, 325.

Schaible S. (1995) Fractional programming, *Handbook of global optimization* (R. Horst, P.M. Pardalos). Kluwer Academic Publishers, pp. 495–608.

Schaible S., Shi J. (2003) Fractional programming: the sum-of-ratios case, *Opti. Meth. Soft.* **18**, pp. 219–229.

Stancu-Minasian I.M. (1997) *Fractional programming*. Kluwer Academic Publishers.

Wu T. (1997) A note on a global approach for general 0–1 fractional programming, *Eur. J. Oper. Res.*, 229.

You F., Castro P.M., Grossmann I.E. (2009) Dinkelbach's algorithm as an efficient method to solve a class of MINLP models for large-scale cyclic scheduling problems, *Comp. Chem. Eng.* **33**, 1879.

A.7 Piecewise Linear Functions

In this section, we are interested in mathematical programs involving piecewise linear functions and linear functions in the economic function and/or in the constraints. In fact, in the general case, such programs can be rewritten as mixed-integer linear programs. Note that this notion of piecewise linearity is interesting since any continuous function of one variable can be approximated by a piecewise linear function, the quality of the approximation depending on the size of the segments. Let $f(x)$ be a piecewise linear function defined on the interval $[b_0,\ b_p]$ in the following way:

$$\begin{array}{ll} f(x) = a_1 x + d_1 & b_0 \le x \le b_1 \\ f(x) = a_2 x + d_2 & b_1 \le x \le b_2 \\ f(x) = a_3 x + d_3 & b_2 \le x \le b_3 \\ \ldots\ldots\ldots\ldots\ldots\ldots & \ldots\ldots\ldots\ldots \\ f(x) = a_p x + d_p & b_{p-1} \le x \le b_p \end{array}$$

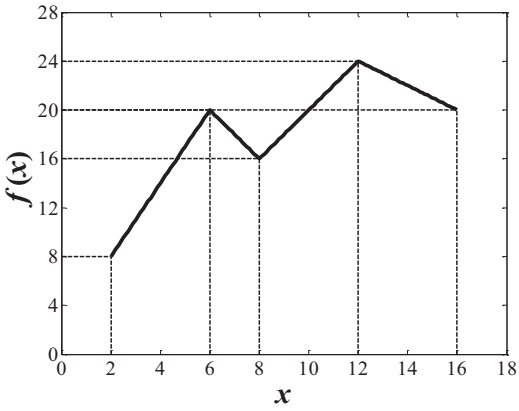

FIG. A.1 – A piecewise linear function, $f(x)$, defined by the 5 points of coordinates (2,8), (6,20), (8,16), (12,24), and (16,20).

The coefficients $b_0, b_1, \ldots, b_p$ are real numbers such that $0 \leq b_0 < b_1 < \cdots < b_p$. The coefficients $a_1, \ldots, a_p$ represent the slope of the different segments. Figure A.1 shows a piecewise linear function, $f(x)$, defined on the interval [2, 16] by the 5 points of coordinates (2, 8), (6, 20), (8, 16), (12, 24), and (16, 20). The 4 corresponding linear – or affine – functions are: $f_1(x) = 3x + 2$, $f_2(x) = -2x + 32$, $f_3(x) = 2x$, and $f_4(x) = -x + 36$.

A first formulation. This formulation allows a piecewise linear function to be expressed as a linear function subject to linear constraints. This formulation uses additional Boolean variables and also additional real variables. Note, first of all, that, for any x between b_i and b_{i+1}, two non-negative reals, λ_i and λ_{i+1}, can be defined, whose sum is 1 and such that $x = \lambda_i b_i + \lambda_{i+1} b_{i+1}$. It is thus deduced that, for any x between b_i and b_{i+1}, the piecewise linear function $f(x)$ can be written $f(x) = \lambda_i f(b_i) + \lambda_{i+1} f(b_{i+1})$ where λ_i and λ_{i+1} satisfy the above properties. Finally, we can therefore write the function $f(x)$ in the form $f(x) = \sum_{i=0}^{p} \lambda_i f(b_i)$ where all λ_i are non-negative real numbers such that $\sum_{i=0}^{p} \lambda_i = 1$, and satisfying the following conditions: if $b_i \leq x \leq b_{i+1}$ then $\lambda_i + \lambda_{i+1} = 1$ and $x = \lambda_i b_i + \lambda_{i+1} b_{i+1}$. We can therefore write $f(x)$ in the form $\sum_{i=0}^{p} \lambda_i f(b_i)$, where variables z_i and λ_i satisfy constraints $\mathrm{C}_{\mathrm{A.1}}$ below.

$$
\mathrm{C}_{\mathrm{A.1}} : \begin{cases}
x = \sum_{i=0}^{p} \lambda_i b_i & \text{(CA.1.1)} \quad | \quad \sum_{i=0}^{p} \lambda_i = 1 & \text{(CA.1.5)} \\
\lambda_0 \leq z_0 & \text{(CA.1.2)} \quad | \quad \sum_{i=0}^{p-1} z_i = 1 & \text{(CA.1.6)} \\
\lambda_i \leq z_{i-1} + z_i \quad 1 \leq i < p & \text{(CA.1.3)} \quad | \quad \lambda_i \geq 0 \quad i = 0, 1, \ldots, p & \text{(CA.1.7)} \\
\lambda_p \leq z_{p-1} & \text{(CA.1.4)} \quad | \quad z_i \in \{0, 1\} \quad i = 0, 1, \ldots, p-1 & \text{(CA.1.8)}
\end{cases}
$$

Indeed, constraints CA.1.6 and CA.1.8 require that one and only one variable z_i be equal to 1. Moreover, if $z_i = 1$, then constraints CA.1.2, CA.1.3, CA.1.4, CA.1.5, and CA.1.7 imply $\lambda_i + \lambda_{i+1} = 1$ and $\lambda_k = 0$, for all k different from i or $i+1$.

Example A.6. Let us apply the approach presented above to the mathematical program $\mathrm{P}_{\mathrm{A.28}}$. In this program, the economic function and one of the constraints are expressed as the sum of a piecewise linear function of variable x, and a linear function of variable x and other variables, t_1, t_2, and t_3. The piecewise linear function $f(x)$ is defined on the interval [1, 7] by the points of coordinates (1, 3), (3, 5), (5, 3), and (7, 5).

$$\mathrm{P}_{\mathrm{A.28}}: \begin{cases} \min\ f(x) + t_1 - 2t_2 + t_3 - 2x & & & & \\ \text{s.t.} \begin{vmatrix} 4.5x + t_1 - t_3 \le 27.5 & (\text{A.28.1}) & | & t_1 \le 6 & (\text{A.28.4}) \\ x + 2f(x) + 2t_1 + 2t_2 \le 24 & (\text{A.28.2}) & | & t_2 \le 4 & (\text{A.28.5}) \\ 1 \le x \le 7 & (\text{A.28.3}) & | & t_1, t_2, t_3 \ge 0 & (\text{A.28.6}) \end{vmatrix} \end{cases}$$

The solution of $\mathrm{P}_{\mathrm{A.28}}$ can be obtained by solving the mixed-integer linear program $\mathrm{P}_{\mathrm{A.29}}$ in which variable e represents the value of the piecewise linear function $f(x)$.

$$\mathrm{P}_{\mathrm{A.29}}: \begin{cases} \min \quad e + t_1 - 2t_2 + t_3 - 2x & & & & & & & \\ \text{s.t.} \begin{vmatrix} 4.5x + t_1 - t_3 \le 27.5 & | & \lambda_0 \le z_0 & | & z_0 + z_1 + z_2 = 1 & | & t_2 \le 4 \\ x + 2e + 2t_1 + 2t_2 \le 24 & | & \lambda_1 \le z_0 + z_1 & | & \lambda_0 + \lambda_1 + \lambda_2 + \lambda_3 = 1 & | & \lambda_0, \lambda_1, \lambda_2, \lambda_3 \ge 0 \\ e = 3\lambda_0 + 5\lambda_1 + 3\lambda_2 + 5\lambda_3 & | & \lambda_2 \le z_1 + z_2 & | & 1 \le x \le 7 & | & t_1, t_2, t_3 \ge 0 \\ x = \lambda_0 + 3\lambda_1 + 5\lambda_2 + 7\lambda_3 & | & \lambda_3 \le z_2 & | & t_1 \le 6 & | & z_0, z_1, z_2 \in \{0, 1\} \end{vmatrix} \end{cases}$$

Note that the values of both variables e and x are entirely defined by the values of variables λ_i, $i = 0, 1, 2, 3$. The optimal solution of $\mathrm{P}_{\mathrm{A.29}}$ is: ($x = 6.1111$, $t_1 = 0$, $t_2 = 4$, $t_3 = 0$); the corresponding values of variables e, λ_i and z_i are: $e = 4.1111$, $\lambda_0 = 0$, $\lambda_1 = 0$, $\lambda_2 = 0.4444$, $\lambda_3 = 0.5556$, $z_0 = 0$, $z_1 = 0$, $z_2 = 1$. The value of the optimal solution is equal to -16.1111.

A second formulation. We present below another way of expressing the piecewise linear function $f(x)$ defined at the beginning of this section. In this formulation, $f(x)$ is expressed as a linear function of real variables, u_i, and Boolean variables, z_i, $i = 1, \ldots, p$, these variables being subject to linear constraints. It is indeed easy to verify that, for any x belonging to the interval $[b_0, b_p]$, $f(x) = \sum_{i=1}^{p} (a_i u_i + d_i z_i)$, provided that variables u_i and z_i satisfy constraints $\mathrm{C}_{\mathrm{A.2}}$ below.

$$\mathrm{C}_{\mathrm{A.2}}: \begin{cases} b_{i-1} z_i \le u_i \le b_i z_i \quad i = 1, \ldots, p & (\text{CA.2.1}) \\ x = \sum_{i=1}^{p} u_i & (\text{CA.2.2}) \\ \sum_{i=1}^{p} z_i = 1 & (\text{CA.2.3}) \\ z_i \in \{0, 1\} \quad i = 1, \ldots, p & (\text{CA.2.4}) \end{cases}$$

Example A.7. Let us apply this second method to the previous program $P_{A.28}$. Since the piecewise linear function $f(x)$ is defined on the interval [1, 7] by the points of coordinates (1, 3), (3, 5), (5, 3), and (7, 5), we have: $a_1 = 1, a_2 = -1, a_3 = 1$ and $d_1 = 2, d_2 = 8, d_3 = -2$. The equivalent mixed-integer linear program $P_{A.30}$ is obtained in which variable e represents the value of the piecewise linear function $f(x)$.

$$P_{A.30}: \begin{cases} \min \quad e + t_1 - 2t_2 + t_3 - 2x \\ \text{s.t.} \begin{cases} x + 2e + 2t_1 + 2t_2 \le 24 & | \quad x = u_1 + u_2 + u_3 \\ 4.5x + t_1 - t_3 \le 27.5 & | \quad z_1 + z_2 + z_3 = 1 \\ e = u_1 + 2z_1 - u_2 + 8z_2 + u_3 - 2z_3 & | \quad 0 \le t_1 \le 6;\ 0 \le t_2 \le 4;\ t_3 \ge 0 \\ z_1 \le u_1 \le 3z_1;\ 3z_2 \le u_2 \le 5z_2;\ 5z_3 \le u_3 \le 7z_3 & | \quad z_1, z_2, z_3 \in \{0, 1\} \end{cases} \end{cases}$$

Note that the possible values of variables u_i, $i = 1, 2, 3$, are completely defined by the values of variables z_i, $i = 1, 2, 3$, that the value of variable x is completely defined by the values of variables u_i, $i = 1, 2, 3$, and that the value of variable e is completely defined by the values of variables u_i and z_i, $i = 1, 2, 3$. The optimal solution of $P_{A.30}$ is: $(x = 6.1111,\ t_1 = 0,\ t_2 = 4,\ t_3 = 0)$; the corresponding values of variables e, and z_i are: $e = 4.1111,\ z_1 = 0,\ z_2 = 0,\ z_3 = 1$. The value of the optimal solution is −16.1111.

Maximization of a concave piecewise linear function. In the concave case, a piecewise linear function can be expressed as the maximum of a linear function of additional real variables, these variables being subject to linear constraints. In this case, the use of Boolean variables becomes unnecessary. Recall that a function $f(x)$ defined on a domain D is concave if and only if.

$$\forall x_1, x_2 \in D, \forall \lambda \in [0, 1] : f(x) = f(\lambda x_1 + (1 - \lambda)\, x_2) \ge \lambda f(x_1) + (1 - \lambda)\, f(x_2).$$

Figure A.2 shows an example of a concave piecewise linear function.

Let us consider a concave piecewise linear function, $f(x)$, defined on the interval $[b_0, b_p]$ by the points of coordinates (b_0, y_0), (b_1, y_1),..., (b_p, y_p) with $0 \le b_0 < b_1 < \cdots < b_p$. It can be shown that, for all x belonging to the interval $[b_0, b_p]$,

$$f(x) = \max_{u_1, \ldots, u_p} \left\{ y_0 + \sum_{i=1}^{p} \frac{y_i - y_{i-1}}{b_i - b_{i-1}}\, u_i : x = \sum_{i=1}^{p} u_i, 0 \le u_i \le b_i - b_{i-1}\ (i = 1, \ldots, p) \right\}.$$

Example A.8. Let us apply the previous method to program $P_{A.31}$ which consists in maximizing the sum of two concave piecewise linear functions subject to linear constraints.

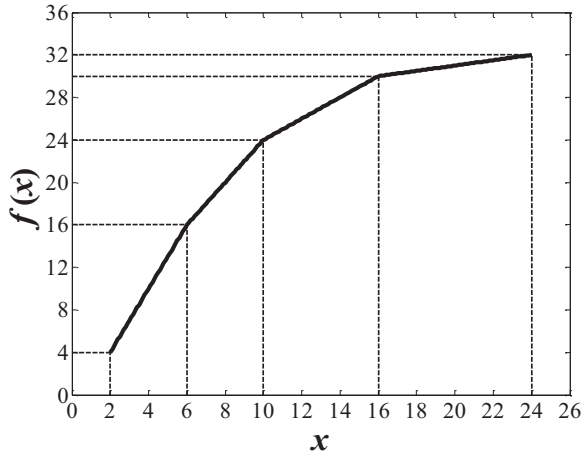

FIG. A.2 – A concave piecewise linear function, $f(x)$, defined by the 5 points of coordinates (2, 4), (6, 16), (10, 24), (16, 30), and (24, 32). The successive slopes of the 4 segments are decreasing and equal to 3, 2, 1, and 0.25, respectively.

$$\mathrm{P}_{\mathrm{A.31}} : \begin{cases} \max f_1(x_1) + f_2(x_2) \\ \text{s.t.} \left| \begin{array}{lcl} x_1 + 2x_2 \leq 6 & | & 0 \leq x_1 \leq 6 \\ 3x_1 + x_2 \geq 10 & | & 0 \leq x_2 \leq 8 \end{array} \right. \end{cases}$$

The function $f_1(x_1)$ is defined by the points of coordinates (0, 0), (1, 2), (3, 4), and (6, 5), and the function $f_2(x_2)$ is defined by the points of coordinates (0, 0), (2, 4), (4, 6), and (8, 7). The linear program $\mathrm{P}_{\mathrm{A.31}}$ is equivalent to $\mathrm{P}_{\mathrm{A.32}}$.

$$\mathrm{P}_{\mathrm{A.32}} : \begin{cases} \max \quad 2u_{11} + u_{12} + \frac{1}{3}u_{13} + 2u_{21} + u_{22} + \frac{1}{4}u_{23} \\ \text{s.t.} \left| \begin{array}{lcl} x_1 + 2x_2 \leq 6 & | & 0 \leq u_{11} \leq 1;\ 0 \leq u_{12} \leq 2;\ 0 \leq u_{13} \leq 3 \\ 3x_1 + x_2 \geq 10 & | & 0 \leq u_{21} \leq 2;\ 0 \leq u_{22} \leq 2;\ 0 \leq u_{23} \leq 4 \\ x_1 = u_{11} + u_{12} + u_{13} & | & 0 \leq x_1 \leq 6 \\ x_2 = u_{21} + u_{22} + \alpha_{23} & | & 0 \leq x_2 \leq 8 \end{array} \right. \end{cases}$$

Note that constraints $0 \leq x_1 \leq 6$ and $0 \leq x_2 \leq 8$ are useless since the value of variable x_1 (resp. x_2) is completely defined by the values of variables u_{1k} (resp. u_{2k}), $k = 1, 2, 3$. The optimal solution of $\mathrm{P}_{\mathrm{A.32}}$ is $(x_1 = 3, x_2 = 1.5)$. Its value is 7. The corresponding values of variables u_{ik} are: $u_{11} = 1, u_{12} = 2, u_{13} = 0$ and $u_{21} = 1.5, u_{22} = 0, u_{23} = 0$.

The concave piecewise linear function $f(x)$, defined on the interval $[b_0,\ b_p]$ by the points of coordinates (b_0, y_0), (b_1, y_1),..., (b_p, y_p) with $0 \leq b_0 < b_1 < \cdots < b_p$, can also be expressed as follows:

$$\forall x \in [b_0, b_p], f(x) = \min_{i=1,\ldots,p} \{a_i x + d_i\}$$

where, for $i = 1,\ldots, p$, $a_i = (y_i - y_{i-1})/(b_i - b_{i-1})$ and $d_i = y_i - a_i b_i$. Applying this property to program $\mathrm{P}_{\mathrm{A.31}}$ results in the equivalent linear program $\mathrm{P}_{\mathrm{A.33}}$ in which variable e_1 represents the value of the piecewise linear function $f_1(x)$ and variable e_2 the value of the piecewise linear function $f_2(x)$.

$$\mathrm{P}_{\mathrm{A.33}} : \left\{ \begin{array}{lll} \max & e_1 + e_2 & \\ \text{s.t.} & e_1 \leq 2x_1 & e_2 \leq 0.25x_2 + 5 \\ & e_1 \leq x_1 + 1 & x_1 + 2x_2 \leq 6 \\ & e_1 \leq (1/3)x_1 + 3 & 3x_1 + x_2 \geq 10 \\ & e_2 \leq 2x_2 & 0 \leq x_1 \leq 6 \\ & e_2 \leq x_2 + 2 & 0 \leq x_2 \leq 8 \end{array} \right.$$

Note that there is no need to further constrain the – non negative – variables e_1 and e_2. The optimal solution of $\mathrm{P}_{\mathrm{A.33}}$ is: $(x_1 = 3, x_2 = 1.5, e_1 = 4, e_2 = 3)$; the value of this solution is equal to 7.

For a more complete presentation of the possible processing of piecewise linear functions, the reader can consult the references cited below.

References and Further Reading

Billionnet A. (2017) *Programmation discrète*. Dunod, Paris.

Chen D.-S., Batson R.G., Dang Y. (2010) *Applied integer programming, modeling and solution*. John Wiley & Sons.

Li H.-L., Lu H.-C., Huang C.-H., Hu N.-Z. (2009) A superior representation method for piecewise linear functions, *INFORMS J. Comp.* **21**, 314.

Lin M.-H., Carlsson J.G., Ge D., Shi J., Tsai J.-F. (2013) A review of piecewise linearization methods, *Mathematical problems in engineering*. Hindawi Publishing Corporation.

Vielma J.P., Ahmed S., Nemhauser G. (2010) A note on "A superior representation method for piecewise linear functions", *INFORMS J. Comp.* **22**, 493.

A.8 Robustness in Mathematical Programming

It is often necessary to take into account, in a mathematical program, some uncertainty in the data since this uncertainty can strongly influence the quality and feasibility of the selected solution. The robust approach allows this uncertainty to be taken into account to some extent.

Consider the linear program $\mathrm{P}_{\mathrm{A.34}}$.

$$\mathrm{P}_{\mathrm{A.34}} : \left\{ \begin{array}{lll} \max & \sum_{j=1}^{n} c_j x_j & \\ \text{s.t.} & \sum_{j=1}^{n} a_{ij} x_j \leq b_i \quad i = 1,\ldots, m & \text{(A.34.1)} \\ & l_j \leq x_j \leq u_j \quad j = 1,\ldots, n & \text{(A.34.2)} \end{array} \right.$$

The coefficients c_j ($j = 1,\ldots, n$), a_{ij} ($i = 1,\ldots, m$, $j = 1,\ldots, n$), b_i ($i = 1,\ldots, m$), l_j ($j = 1,\ldots, n$), and u_j ($j = 1,\ldots, n$) are data, and all the coefficients l_j are positive or zero. For all $i \in \{1,\ldots, m\}$, let J_i be the set of indices j such that the coefficient a_{ij} is uncertain. It is assumed here that, for all $i \in \{1,\ldots, m\}$, each entry a_{ij}, $j \in J_i$, corresponds to a bounded symmetric random variable, $\tilde{a}_{ij}$, $j \in J_i$, which takes its values in the interval $[a_{ij} - \hat{a}_{ij}, a_{ij} + \hat{a}_{ij}]$ where $\hat{a}_{ij}$ is a positive or zero constant. Below we present a robust approach proposed by Bertsimas and Sim (2004).

Maximal protection against uncertainty. In this case, the optimal solution of $\mathrm{P}_{\mathrm{A.34}}$ is the one that maximizes the value of the economic function and is feasible regardless of the values taken by the coefficients a_{ij}, in the set of possible values. The linear program $\mathrm{P}_{\mathrm{A.35}}$ allows this solution to be determined.

$$\mathrm{P}_{\mathrm{A.35}} : \begin{cases} \max \displaystyle\sum_{j=1}^{n} c_j x_j & \\ \text{s.t.} \left| \begin{array}{ll} \displaystyle\sum_{j=1}^{n} a_{ij} x_j + \sum_{j \in J_i} \hat{a}_{ij} x_j \leq b_i & i = 1,\ldots, m \quad \text{(A.35.1)} \\ l_j \leq x_j \leq u_j & j = 1,\ldots, n \quad \text{(A.35.2)} \end{array} \right. \end{cases}$$

Indeed, any feasible solution of $\mathrm{P}_{\mathrm{A.35}}$ remains feasible for all possible values of the random variables $\tilde{a}_{ij}$, *i.e.*, for all values belonging to the interval $[a_{ij} - \hat{a}_{ij}, a_{ij} + \hat{a}_{ij}]$. The optimal solution of $\mathrm{P}_{\mathrm{A.35}}$ is said to be the optimal robust solution of the problem under consideration.

A less conservative approach. Here, it is considered unlikely that all uncertain coefficients will differ simultaneously from their nominal value. It is thus assumed that Γ_i coefficients a_{ij} can differ from their nominal value – at most from the quantity $\hat{a}_{ij}$. Γ_i is therefore an integer belonging to $[0, |J_i|]$. As before, the optimal solution of $\mathrm{P}_{\mathrm{A.34}}$ is then the one that maximizes the value of the economic function, and which is feasible regardless of the values taken by the coefficients a_{ij}, given the uncertainty assumptions. The search for this solution can be formulated as the mathematical program $\mathrm{P}_{\mathrm{A.36}}$.

$$\mathrm{P}_{\mathrm{A.36}} : \begin{cases} \max \displaystyle\sum_{j=1}^{n} {}_j x_j & \\ \text{s.t.} \left| \begin{array}{ll} \displaystyle\sum_{j=1}^{n} a_{ij} x_j & \\ \quad + \displaystyle\max_{S_i \subseteq J_i,\, |S_i| \leq \Gamma_i} \left[\sum_{j \in S_i} \hat{a}_{ij} x_j \right] \leq b_i & i = 1,\ldots, m \quad \text{(A.36.1)} \\ l_j \leq x_j \leq u_j & j = 1,\ldots, n \quad \text{(A.36.2)} \end{array} \right. \end{cases}$$

Given a feasible solution of $\mathrm{P}_{\mathrm{A.36}}$, $\tilde{x}$, let us denote by $\beta_i(\tilde{x}, \Gamma_i)$ the quantity $\max_{S_i \subseteq J_i,\, |S_i| \leq \Gamma_i} [\sum_{j \in S_i} \hat{a}_{ij} \tilde{x}_j]$. This quantity can be determined by solving the linear program $\mathrm{P}_{\mathrm{A.37}}$.

$$\mathrm{P}_{\mathrm{A.37}}: \begin{cases} \max \sum\limits_{j\in J_i} \hat{a}_{ij}\tilde{x}_j\alpha_{ij} & \\ \text{s.t.} \begin{vmatrix} \sum\limits_{j\in J_i} \alpha_{ij} \leq \Gamma_i & \text{(A.37.1)} \\ 0 \leq \alpha_{ij} \leq 1 \quad j \in J_i & \text{(A.37.2)} \end{vmatrix} \end{cases}$$

$\mathrm{P}_{\mathrm{A.37}}$ admits an optimal finite solution, which implies that its dual admits one too (duality theory). Moreover, the values of these two optimal solutions are equal. By associating the dual variable z_i to constraint A.37.1 and the dual variables p_{ij}, $j \in J_i$, to constraints A.37.2, this dual problem is written.

$$\beta_i(\tilde{x}, \Gamma_i) = \begin{cases} \min \sum\limits_{j\in J_i} p_{ij} + \Gamma_i z_i & \\ \text{s.t.} \begin{vmatrix} z_i + p_{ij} \geq \hat{a}_{ij}\tilde{x}_j & j \in J_i \\ p_{ij} \geq 0 & j \in J_i \\ z_i \geq 0 & \end{vmatrix} \end{cases}.$$

From this, it can be deduced that the optimal robust solution to the problem under consideration can be determined by solving program $\mathrm{P}_{\mathrm{A.38}}$.

$$\mathrm{P}_{\mathrm{A.38}}: \begin{cases} \max \sum\limits_{j=1}^{n} c_j x_j & \\ \text{s.t.} \begin{vmatrix} \sum\limits_{j=1}^{n} a_{ij}x_j + \sum\limits_{j\in J_i} p_{ij} + \Gamma_i z_i \leq b_i & i = 1,\ldots,m & \text{(A.38.1)} \\ z_i + p_{ij} \geq \hat{a}_{ij}x_j & i = 1,\ldots,m; j \in J_i & \text{(A.38.2)} \\ l_j \leq x_j \leq u_j & j = 1,\ldots,n & \text{(A.38.3)} \\ z_i \geq 0 & i = 1,\ldots,m & \text{(A.38.4)} \\ p_{ij} \geq 0 & i = 1,\ldots,m; j \in J_i & \text{(A.38.5)} \end{vmatrix} \end{cases}$$

Example A.9. Consider the linear program $\mathrm{P}_{\mathrm{A.39}}$.

$$\mathrm{P}_{\mathrm{A.39}}: \begin{cases} \max \quad 6x_1 + 2x_2 + 9x_3 + 10x_4 + x_5 & \\ \text{s.t.} \begin{vmatrix} 4x_1 + x_2 - 5x_3 - 3x_4 - 2x_5 \leq 10 & | & 5x_1 - x_2 - 7x_3 + 6x_4 + 7x_5 \leq 20 & \\ 3x_1 + 2x_2 + 9x_3 - 3x_4 - 10x_5 \leq 2 & | & 0 \leq x_i \leq 10 & i = 1,\ldots,5 \end{vmatrix} \end{cases}$$

The optimal solution of $P_{A.39}$ is $x = (0, 10, 10, 6.25641, 8.92308)$ and its value is 181.4872. Now, suppose that all the coefficients of the constraints are uncertain. The values of the coefficients $\hat{a}_{ij}$ are given by the matrix below.

$$(\hat{a}_{ij}) = \begin{pmatrix} 0.8 & 0.2 & 1.0 & 0.6 & 0.4 \\ 0.6 & 0.4 & 1.8 & 0.6 & 2.0 \\ 1.0 & 0.2 & 1.4 & 1.2 & 1.4 \end{pmatrix}$$

Table A.1 gives the optimal robust solutions and their values for different values of the parameter Γ_i. In this example, it is assumed that this parameter is not dependent on i and we set $\Gamma = \Gamma_i$ for all i.

Table A.1 shows that when the uncertainty is substantial ($\Gamma = 5$), the protection cost against this uncertainty is very high since the value of the economic function decreases, for example, from 91.0816 when $\Gamma = 1$, to 44.0547 when $\Gamma = 5$ (about -52%). Note that in the case where $\Gamma = 5$ the optimal robust solution can be calculated by using the formulation $P_{A.35}$.

TAB. A.1 – Optimal robust solutions associated with program $P_{A.39}$, for different uncertainty domains defined by the parameter Γ.

Γ	Optimal robust solution	Value
1	(0, 0, 9.0204, 0, 9.8980)	91.0816
2	(0, 1.7778, 2.2222, 2.5926, 2.2222)	51.7037
3	(0, 3.3182, 0.7374, 3.0379, 0.6636)	44.3156
4	(0, 0, 0.9701, 3.5323, 0)	44.0547
5	(0, 0, 0.9701, 3.5323, 0)	44.0547

The approach can be extended to mixed-integer linear programs. Consider program $P_{A.40}$ in which some variables are integer while others are real.

$$P_{A.40}: \begin{cases} \max \sum_{j=1}^{n} c_j x_j & \\ \text{s.t.} \begin{cases} \sum_{j=1}^{n} a_{ij} x_j \leq b_i & i = 1,\ldots,m \quad \text{(A.40.1)} \\ x_j \in \mathbb{N} & j = 1,\ldots,p \quad \text{(A.40.2)} \\ x_j \geq 0 & j = p+1,\ldots,n \quad \text{(A.40.3)} \end{cases} \end{cases}$$

In this case, the optimal robust solution can be determined by solving the mixed-integer linear program $P_{A.41}$.

$$
\mathrm{P}_{\mathrm{A.41}}: \left\{ \begin{array}{l} \max \sum_{j=1}^{n} c_j x_j \\ \text{s.t.} \left| \begin{array}{lll} \sum_{j=1}^{n} a_{ij} x_j + \sum_{j \in J_i} p_{ij} + \Gamma_i z_i \leq b_i & i = 1,\ldots,m & \text{(A.41.1)} \\ z_i + p_{ij} \geq \hat{a}_{ij} x_j & i = 1,\ldots,m, j \in J_i & \text{(A.41.2)} \\ x_j \in \mathbb{N} & j = 1,\ldots,p & \text{(A.41.3)} \\ x_j \geq 0 & j = p+1,\ldots,n & \text{(A.41.4)} \\ z_i \geq 0 & i = 1,\ldots,m & \text{(A.41.5)} \\ p_{ij} \geq 0 & i = 1,\ldots,m, j \in J_i & \text{(A.41.6)} \end{array} \right. \end{array} \right.
$$

In everything we have just seen, the uncertainty affecting some coefficients is defined by intervals. We now consider the case where the uncertainty is represented by a set of (discrete) scenarios. A scenario is a set of assumptions about the evolution of the factors that may influence the value of the coefficients, and several scenarios are possible. In such an approach, the value of the different coefficients of the mathematical program considered depends on the scenario.

Consider the linear program $\mathrm{P}_{\mathrm{A.42}}$ where the coefficients a_{ij} are uncertain.

$$
\mathrm{P}_{\mathrm{A.42}}: \left\{ \begin{array}{l} \max \sum_{j=1}^{n} c_j x_j \\ \text{s.t.} \left| \begin{array}{lll} \sum_{j=1}^{n} a_{ij} x_j \leq b_i & i = 1,\ldots,m & \text{(A.42.1)} \\ x_j \geq 0 & j = 1,\ldots,n & \text{(A.42.2)} \end{array} \right. \end{array} \right.
$$

A set of scenarios, $\mathrm{Sc} = \{\mathrm{sc}_1, \mathrm{sc}_2, \ldots, \mathrm{sc}_p\}$, is envisaged and the values of the coefficients a_{ij} depend on the scenario. It is assumed that for each of these p scenarios the values of all the coefficients a_{ij} are known. For $i = 1,\ldots,m$, $j = 1,\ldots,n$, and $\omega = 1,\ldots,p$, a_{ij}^{ω} denotes the value of the coefficient a_{ij} in the case of the scenario sc_{ω}.

The problem of finding a solution to $\mathrm{P}_{\mathrm{A.42}}$ that is feasible for all the scenarios and that is the least costly – an optimal robust solution – can then be formulated as $\mathrm{P}_{\mathrm{A.43}}$.

$$
\mathrm{P}_{\mathrm{A.43}}: \left\{ \begin{array}{l} \max \sum_{j=1}^{n} c_j x_j \\ \text{s.t.} \left| \begin{array}{lll} \sum_{j=1}^{n} a_{ij}^{\omega} x_j \leq b_i & i = 1,\ldots,m;\ \omega = 1,\ldots,p & \text{(A.43.1)} \\ x_j \geq 0 & j = 1,\ldots,n & \text{(A.43.2)} \end{array} \right. \end{array} \right.
$$

We have just shown how to take into account some uncertainty about the coefficients a_{ij} of program $P_{A.42}$. Let us now consider how to take into account uncertainty about the coefficients of the economic function, c_j. Consider the linear program $P_{A.42}$ where the coefficients c_j are uncertain. We consider a set of possible scenarios, $Sc = \{sc_1, sc_2, \ldots, sc_p\}$, and denote the value of the coefficient c_j in the case of the scenario sc_ω as c_j^ω. Several robustness criteria can be considered (see, for example, Kouvelis and Yu, 1997). We consider here a "max–min" criterion to measure the quality of a solution. In this approach, a solution is better than all others if its worst performance – over all scenarios – is better than the worst performance of all other solutions. We first illustrate this robustness criterion on an optimization problem in graphs (example A.10) and then formulate the search for an optimal robust solution for program $P_{A.42}$ with this robustness criterion.

Example A.10. Let us consider the problem of the path of minimum value in a graph with uncertainty in the arc values, this uncertainty being modelled by a set of possible scenarios. Note that this is a minimisation problem unlike program $P_{A.43}$. Let $G = (X, U)$ be a graph where $X = \{x_1, \ldots, x_n\}$ is the set of vertices and $U = \{a_1, \ldots, a_m\}$, the set of arcs. Let $Sc = \{sc_1, sc_2, \ldots, sc_p\}$ be the set of possible scenarios. For each scenario $sc_\omega \in Sc$, the value of the arc $a_i \in U$ is denoted by c_i^ω. The objective is to determine a path of minimum value, from vertex x_1 to vertex x_n, the value of a path being equal to the sum of the value of its arcs. Here, we use a min–max criterion, *i.e.*, we consider the problem of determining, among all the paths in the graph from x_1 to x_n, the one whose maximal length, over all the scenarios, is minimal. It is obviously possible to consider other criteria to take into account this uncertainty on the arc values. Let us consider an example of the problem with two scenarios.

The considered graph is represented by figure A.3 where each double arrow connecting two vertices x_i and x_j actually corresponds to the 2 symmetric arcs (x_i, x_j) and (x_j, x_i) with identical associated values. For each arc in the graph, the values for

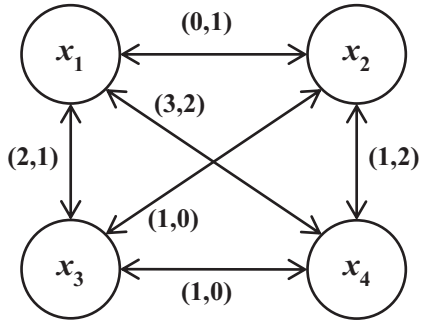

FIG. A.3 – A symmetric graph with 4 vertices. The value of each arc depends on the scenario and is indicated in brackets next to the arc: (value in the scenario sc_1, value in the scenario sc_2). For example, the value of the arc (x_3, x_4) and the arc (x_4, x_3) is equal to 1 for the scenario sc_1 and 0 for the scenario sc_2.

TAB. A.2 – List of all elementary paths, from x_1 to x_4, in the graph of figure A.3, with their values in both scenarios.

Name of the path	Description of the path	Value of the path in the scenario sc_1	Value of the path in the scenario sc_2
π_1	$x_1 \rightarrow x_2 \rightarrow x_3 \rightarrow x_4$	2	1
π_2	$x_1 \rightarrow x_2 \rightarrow x_4$	1	3
π_3	$x_1 \rightarrow x_3 \rightarrow x_2 \rightarrow x_4$	4	3
π_4	$x_1 \rightarrow x_3 \rightarrow x_4$	3	1
π_5	$x_1 \rightarrow x_4$	3	2

the scenarios sc_1 and sc_2 are given in brackets (arc value for the scenario sc_1, arc value for the scenario sc_2). Let us determine the optimal robust solution, by enumeration. Table A.2 gives, for all elementary paths from x_1 to x_4, their respective values in both scenarios.

We can deduce from table A.2 that, in this example, the optimal path is $\pi_1 = x_1 \rightarrow x_2 \rightarrow x_3 \rightarrow x_4$ and the optimal value is equal to 2.

The program for determining an optimal robust solution of program $P_{A.42}$, when the coefficients of the economic function, c_j, are uncertain, is $P_{A.44}$.

$$P_{A.44}: \begin{cases} \max \ \alpha \\ \text{s.t.} \begin{cases} \alpha \leq \sum_{j=1}^{n} c_j^{\omega} x_j & \omega = 1,\ldots,p \quad (A.44.1) \\ \sum_{j=1}^{n} a_{ij} x_j \leq b_i & i = 1,\ldots,m \quad (A.44.2) \\ x_j \geq 0 & j = 1,\ldots,n \quad (A.44.3) \end{cases} \end{cases}$$

Choosing the best possible solution in the worst-case scenario can have a significant drawback. Indeed, if one of the scenarios is very pessimistic – regarding the value of the economic function coefficients –, then the solution chosen will essentially take into account this single scenario. To overcome this disadvantage, other criteria can be chosen to evaluate a solution of $P_{A.42}$ when the coefficients of the economic function are uncertain. For example, we may be interested in the solution that minimizes the largest relative gap or "regret" – over all scenarios – between the value of the selected solution and the value of the optimal solution in the scenario under consideration. To solve this problem, one must first determine the optimal solutions in each scenario, *i.e.*, solve program $P_{A.45}(\omega)$ for each scenario, *i.e.*, for $\omega = 1,\ldots, p$.

$$P_{A.45}(\omega): \begin{cases} \max \sum_{j=1}^{n} c_j^{\omega} x_j \\ \text{s.t.} \begin{cases} \sum_{j=1}^{n} a_{ij} x_j \leq b_i & i = 1,\ldots,m \quad (A.45_{\omega}.1) \\ x_j \geq 0 & j = 1,\ldots,n \quad (A.45_{\omega}.2) \end{cases} \end{cases}$$

Let $V^{*\omega}$ be the optimal value of program $P_{A.45}(\omega)$ – for the scenario sc_{ω}. The optimal robust solution can be determined by solving program $P_{A.46}$.

$$
\mathrm{P}_{\mathrm{A.46}}:\left\{\begin{array}{l}\min\ \alpha\\ \text{s.t.}\left|\begin{array}{lll}\alpha\geq\left(V^{*\omega}-\sum\limits_{j=1}^{n}c_j^{\omega}x_i\right)\Big/\,V^{*\omega} & \omega=1,\ldots,p & \text{(A.46.1)}\\ \sum\limits_{j=1}^{n}a_{ij}x_j\leq b_i & i=1,\ldots,m & \text{(A.46.2)}\\ x_j\geq 0 & j=1,\ldots,n & \text{(A.46.3)}\end{array}\right.\end{array}\right.
$$

Robust optimization is a rapidly growing branch of mathematical optimization that attempts to solve an optimization problem by taking into account as best as possible the various uncertainties that affect it. For a more detailed presentation of the basics of this optimization field, the reader can consult the references cited below.

References and Further Reading

Ben-Tal A., El Ghaoui L., Nemirovski A. (2009) *Robust optimization*. Princeton University Press.

Bertsimas D., Sim M. (2004) Price of robustness, *Oper. Res.* **52**, 35.

Bertsimas D., Brown B., Caramanis C. (2011) Theory and applications of robust optimization, *SIAM Rev.* **53**, 464.

Billionnet A., Costa M-C., Poirion P.-L. (2014) 2-Stage Robust MILP with continuous recourse variables, *Discr. Appl. Math.* **170**, 21.

Gabrel V., Murat C., Thiele A. (2014) Recent advances in robust optimization: an overview, *Eur. J. Oper. Res.* **235**, 471.

Kouvelis P., Yu G. (1997) *Robust discrete optimization and its applications*. Kluwer Academic Publishers.

Roy B. (2007) La robustesse en recherche opérationnelle et aide à la décision: une préoccupation multi facettes, *Annales du LAMSADE* **7**.

A.9 Set-Covering and Set-Partitioning Problems

We consider a set of elements, $E=\{e_1, e_2,\ldots, e_m\}$, and a set of parts of E, $F=\{F_1, F_2,\ldots, F_n\}$. With each element, F_j, of F is associated a cost, c_j. The set covering problem consists in determining a subset of F, of minimal cost, which covers all the elements of E. In other words, the set covering problem consists in determining $X\subseteq F$ such that $\cup_{j\,:\,F_j\in X}F_j=E$, and which minimizes the cost of X, that is the quantity $\sum_{j\,:\,F_j\in X}c_j$. The set X is said to be a cover for E.

One can also look at minimal covers in the inclusion sense. A cover, X, of E is minimal in the inclusion sense if there are no other covers of E strictly included in X.

Mathematical program associated with the set-covering problem. With each element F_j of F, is associated a Boolean variable, x_j, which, by definition, is equal to 1 if and only if F_j is selected to form the minimal cost cover, X, of E, *i.e.*, if $F_j\in X$. Let J_i be the set of indices j belonging to $\{1,\ldots,n\}$ and such that $e_i\in F_j$. The set-covering problem – finding a minimal cost cover – can be formulated as the linear program in Boolean variables $\mathrm{P}_{\mathrm{A.47}}$.

$$\mathrm{P}_{\mathrm{A.47}}: \begin{cases} \min \sum_{j=1}^{n} c_j x_j & \\ \text{s.t.} \left| \begin{array}{ll} \sum_{j \in J_i} x_j \geq 1 & i = 1,\ldots,m \quad \text{(A.47.1)} \\ x_j \in \{0,1\} & j = 1,\ldots,n \quad \text{(A.47.2)} \end{array} \right. \end{cases}$$

If the inequality constraints in $\mathrm{P}_{\mathrm{A.47}}$ are replaced by equality constraints, the resulting program is associated with what is called a set-partitioning problem. This problem indeed consists in determining a set of elements of F of minimal cost, and which form a partition of E.

Example A.11. Consider the set-covering problem in which $E = \{e_1, e_2, e_3, e_4, e_5\}$, $F = \{F_1, F_2, F_3, F_4\}$ with $F_1 = \{e_1, e_2\}$, $F_2 = \{e_2, e_3, e_5\}$, $F_3 = \{e_2, e_4, e_5\}$, $F_4 = \{e_3, e_4\}$ and $c = \{3, 4, 5, 2\}$. The associated linear program in Boolean variables is $\mathrm{P}_{\mathrm{A.48}}$.

$$\mathrm{P}_{\mathrm{A.48}}: \begin{cases} \min \quad 3x_1 + 4x_2 + 5x_3 + 2x_4 & \\ \text{s.t.} \left| \begin{array}{ll|ll} x_1 \geq 1 & & x_3 + x_4 \geq 1 \\ x_1 + x_2 + x_3 \geq 1 & & x_2 + x_3 \geq 1 \\ x_2 + x_4 \geq 1 & & x_1, x_2, x_3, x_4 \in \{0,1\} \end{array} \right. \end{cases}$$

The very particular structure of the programs associated with set-covering and set-partitioning problems make that many simple and effective pre-processing operations are possible. Suppose, for example, that the set of indices, J_r, appearing in a constraint r is contained in the set of indices, J_s, appearing in a constraint s. In the case of the set-covering problem, constraint s can be removed. In the case of the set-partitioning problem, we can set to 0 all variables whose indices belong to J_s without belonging to J_r, and remove the constraint s. Thus, in program $\mathrm{P}_{\mathrm{A.48}}$, the second constraint can be removed. Furthermore, the resolution of the continuous relaxation of $\mathrm{P}_{\mathrm{A.47}}$ often results in an optimal solution in which all the variables take integer values – 0 or 1. The continuous relaxation of program $\mathrm{P}_{\mathrm{A.47}}$ is obtained by replacing in this program the constraints $x_j \in \{0,1\}$, $j = 1,\ldots,n$, by the constraints $0 \leq x_j \leq 1$, $j = 1,\ldots,n$. If this happens, the resolution of $\mathrm{P}_{\mathrm{A.47}}$ is particularly easy since it can be deduced that the optimal solution of the continuous relaxation of $\mathrm{P}_{\mathrm{A.47}}$ – a continuous linear program – is the optimal solution of $\mathrm{P}_{\mathrm{A.47}}$.

Set-covering and partitioning problems are two important issues in operations research with many applications. For more information on the different properties of these problems and how to approach their resolution, the reader can consult the references mentioned below.

References and Further reading

Al-Sultan K.S., Hussain M.F., Nizami J.S. (1996) A genetic algorithm for the set covering problem, *J. Oper. Res. Soc.* **47**, 702.

Balas E., Padberg M.W. (1976) Set partitioning: a survey, *SIAM Rev.* **18**, 710.

Beasley J.E. (1987) An algorithm for set covering problems, *Eur. J. Oper. Res.* **31**, 85.

Beasley J.E., Chu P.C. (1996) A genetic algorithm for the set covering problem, *Eur. J. Oper. Res.* **94**, 392.

Billionnet A. (1978) Transformation du problème de partitionnement en un problème d'ensemble stable de poids maximal, *Revue d'Automatique, d'Informatique et de Recherche Opérationnelle (RAIRO), série verte* **12**, 319.

Billionnet A. (1981) Une nouvelle méthode pour le problème de partitionnement fondée sur une évaluation par excès de la solution, *RAIRO Recherche Opérationnelle* **15**, 139.

Caprara A., Toth P., Fischetti M. (2000) Algorithms for the set covering problem, *Ann. Oper. Res.* **98**, 353.

Chen D.S., Batson R.G., Dang Y. (2010) *Applied integer programming: modeling and solution.* John Wiley & Sons.

Chvatal V. (1979) A greedy heuristic for the set covering problem, *Math. Oper. Res.* **4**, 233.

Galinier P., Hertz A. (2007) Solution techniques for the large set covering problem, *Discr. Appl. Math.* **155**, 312.

Garfinkel R.S., Nemhauser G.L. (1969) The set-partitioning problem: set covering with equality constraints, *Oper. Res.* **17**, 848.

Garfinkel R.S., Nemhauser G.L. (1972) *Integer programming.* John Wiley & Sons.

Vermuganti R.R. (1998) Application of set covering, set packing, and set partitioning models: a survey, *Handbook of combinatorial optimization* (D.Z. Du, P.M. Pardalos, Eds). Springer, pp. 573–746.

A.10 Elements of Graph Theory

An undirected graph, G, is a pair, (X, E), composed of a set of vertices, $X = \{x_1, x_2, \ldots, x_n\}$, and a set of edges, E, each edge connecting two vertices of X called the ends of the edge. In general, two vertices can be connected by more than one edge. If this is the case, we are dealing with a multi-graph. In the rest of this section we consider simple graphs. In such graphs, two vertices are connected by no more than one edge and there is no loop, *i.e.*, an edge whose two ends are identical. Each edge is therefore defined by a pair of distinct vertices, $\{x_i, x_j\}$. The two vertices x_i and x_j are said to be adjacent. The degree of a vertex is equal to the number of edges of which this vertex is one end. An adjacency matrix can be associated with G. It is a $n \times n$-matrix, M, whose general term, m_{ij}, is equal to 1 if and only if vertices x_i and x_j are adjacent. If these two vertices are not adjacent $m_{ij} = 0$.

A directed graph – or oriented graph – G is a pair, (X, A), composed of a set of vertices, $X = \{x_1, x_2, \ldots, x_n\}$, and a set of arcs, A, each arc being defined by a – oriented – pair of vertices, (x_i, x_j). One says that x_i is the initial end of the arc, that x_j is its terminal end, that x_i is a predecessor of x_j, and that x_j is a successor of x_i. An arc whose two ends are identical is called an oriented loop. We are interested here in oriented graphs without loops – oriented – and for which, for any pair of vertices (x_i, x_j), there is at most one arc going from x_i to x_j. The indegree of a vertex is the number of arcs of which this vertex is the terminal end, and the outdegree of a vertex is the number of arcs of which this vertex is the initial end.

Graphs are so named because they can be represented graphically. Each vertex is represented by a point, each edge by a line connecting its ends, *i.e.*, two points, each arc by an arrow from its initial end to its terminal end. A graph can be drawn in several ways: the positions of the points representing the vertices and the shape of the lines or arrows connecting these vertices can vary.

Let $G = (X, U)$ be a directed or undirected graph. An induced sub-graph of G is a graph having for vertices a subset, $\hat{X}$, of the vertices of G, and for arcs/edges only those of G joining the vertices of $\hat{X}$; a partial sub-graph of G is a graph having for vertices a subset, $\hat{X}$, of the vertices of G and for arcs/edges a subset of those of G joining the vertices of $\hat{X}$. In an oriented graph, a path originating at x_i and ending at x_j is defined by a sequence of consecutive arcs, connecting x_i to x_j. If x_i and x_j are identical, then we have a circuit. In the oriented graph in figure A.4, the 3 arcs $(x_2, x_3), (x_3, x_4)$, and (x_4, x_2) define a circuit.

In a graph, oriented or not, a chain connecting x_i to x_j is a sequence of arcs or edges placed end to end, and connecting these two vertices. A chain connecting x_i to x_j is a cycle if x_i and x_j are identical and if the edges of the chain are all distinct. The length of a chain is the number of edges or arcs that compose it, and the length of a path is the number of arcs that compose it. In the undirected graph of figure A.4, the 4 edges $\{x_1, x_2\}, \{x_2, x_5\}, \{x_5, x_4\}$, and $\{x_4, x_6\}$ form a chain connecting vertices x_1 and x_6 and in the directed graph of the same figure, the 4 arcs $(x_2, x_1), (x_2, x_3), (x_3, x_4)$, and (x_6, x_4) form a chain connecting these same two vertices.

A graph – oriented or not – is connected if and only if any pair of vertices is linked by a chain. The distance between two vertices of a connected graph is the length of the chain that links them, with the smallest number of edges – or arcs. The diameter of a connected graph is the greatest distance between two vertices of that graph, among all pairs of vertices. A real value – sometimes called a weight – can be associated with each arc of G. The value of a path/chain is then equal to the sum of the values of the arcs/edges that compose it. A connected component of a graph G is a sub-graph, G_0, of G, which is connected and maximal in the inclusion sense – no other connected sub-graph of G contains G_0.

An oriented graph is strongly connected if and only if for any oriented pair of vertices, (x_i, x_j), there is a path from x_i to x_j. A strongly connected component of an oriented graph, G, is a sub-graph, G_0, of G, which is strongly connected and maximal in the inclusion sense – no other strongly connected sub-graph of G contains G_0. The two graphs in figure A.4 are connected. The sub-graph of the oriented graph in

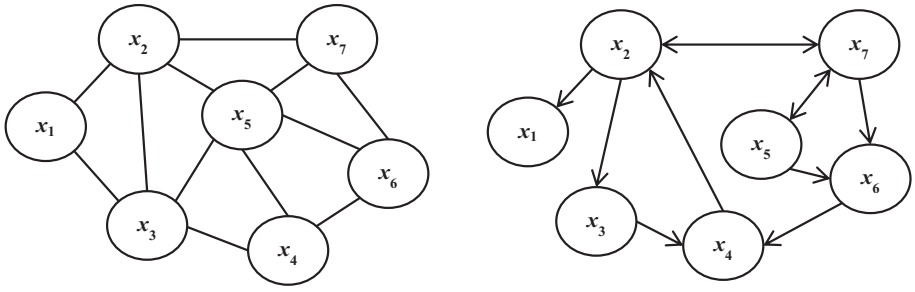

FIG. A.4 – An example of an undirected graph and a directed graph with 7 vertices. Here, a double arrow connecting two vertices x_i and x_j – for example x_2 and x_7 – means that the graph includes an arc from x_i to x_j and an arc from x_j to x_i.

this figure, induced by the vertex set $\{x_2, x_3, x_4, x_5, x_6, x_7\}$, is a strongly connected component of this graph.

A tree is an undirected graph that has no cycle and is connected (figure A.5). A tree can be defined in many ways. For example, G is a tree if G is without cycles and has $n-1$ edges, or G is a tree if G is connected and has $n-1$ edges – n denotes the number of vertices of the graph.

If in a tree a particular vertex, r, is chosen and the edges of this tree are oriented so that there is a – unique – path from r to all other vertices, one obtains an arborescence of root r (figure A.6).

We can also define an arborescence as an oriented graph without circuits admitting a particular vertex, r, called root, and such that there is a single path from r to all the other vertices of the graph.

Given a connected undirected graph, G, a spanning tree of G is a partial sub-graph of G whose vertices are those of G, and which is a tree. A classical problem, when values are assigned to the edges of G, is to determine a spanning tree of minimal value, the value of a tree being equal to the sum of the values of its edges. There are efficient algorithms to solve this problem. One can also look at the

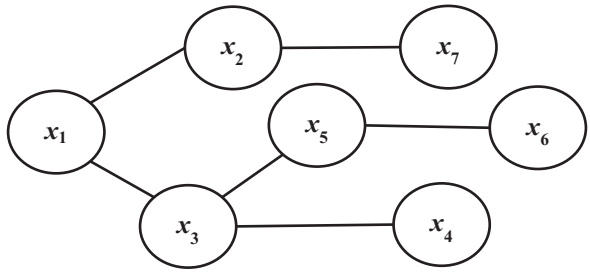

FIG. A.5 – An example of a tree with 7 vertices.

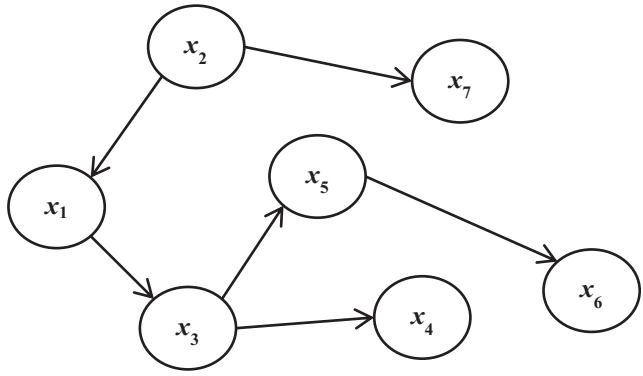

FIG. A.6 – An example of an arborescence constructed from the tree in figure A.5 and whose root is vertex x_2.

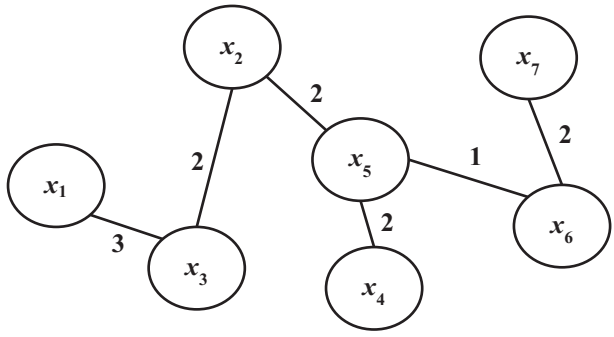

FIG. A.7 – A spanning tree of minimal value (12) associated with the graph in figure A.8.

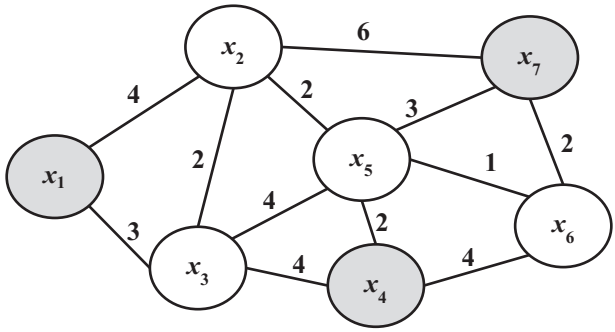

FIG. A.8 – A connected undirected graph with values associated to each edge and a set of mandatory vertices $\hat{X} = \{x_1, x_4, x_7\}$.

spanning tree of minimal value in an oriented graph. The tree in figure A.7 is a spanning tree of minimal value for the graph in figure A.8.

Consider an undirected graph, $G = (X, E)$, and a subset of vertices, $\hat{X}$, included in X. With each edge $\{x_i, x_j\}$ of E, is associated a positive or zero value. The Steiner tree problem consists in determining a partial sub-graph of G that includes all the vertices of $\hat{X}$, which is a tree, and whose value is as small as possible. Recall that the value of a tree is equal to the sum of the values of its edges. This problem is usually difficult. Consider the graph in figure A.8. The values of the edges are shown next to the edges. Suppose that the set $\hat{X}$ consists of vertices x_1, x_4, and x_7 – the required vertices. The Steiner tree of minimal value is given in figure A.9.

A transportation network is defined by an oriented graph, $G = (X, A)$, with two particular vertices, x_1, which is the source of the network and x_n, which is its sink. To simplify the presentation, it is assumed that no arc ends at x_1 and no arc starts at x_n. Each arc in the graph has an associated capacity. It is assumed that there is no useless vertex, *i.e.*, for any vertex x_i of X, there is a path from x_1 to x_n passing through x_i. In a transportation network, G, a flow is the assignment of a

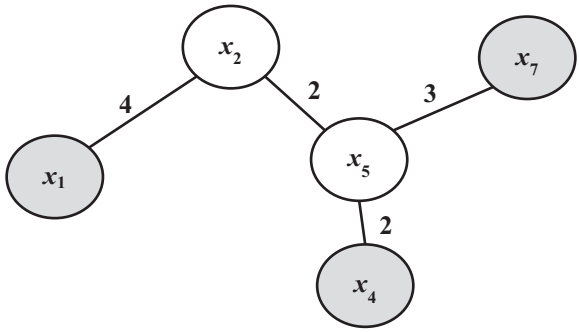

FIG. A.9 – A Steiner tree of minimal value (11) associated with the graph in figure A.8 when the set of mandatory vertices is $\hat{X} = \{x_1, x_4, x_7\}$.

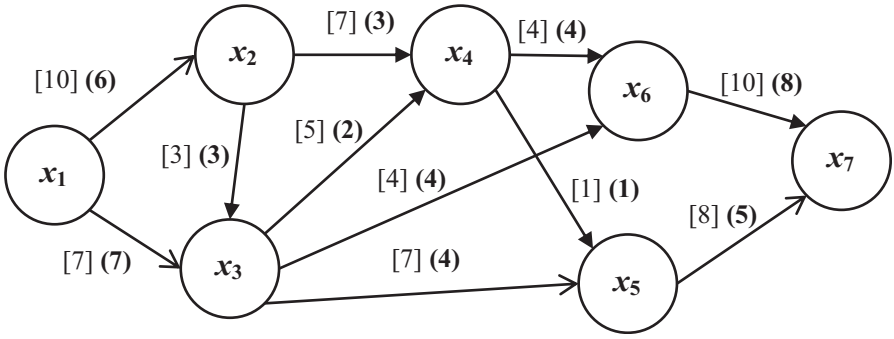

FIG. A.10 – A transportation network whose source is vertex x_1 and sink is vertex x_7, and a flow of value 13 on this network.

non-negative real value to each arc of G – the flow on that arc – which can be interpreted, for example, as a quantity of matter transported on that arc, such that, in each vertex, the sum of the incoming flows – on the arcs of which this vertex is the terminal end – is equal to the sum of the outgoing flows – on the arcs of which this vertex is the initial end. This flow must take into account the capacity assigned to each arc, this capacity reflecting an upper limit of the flow allowed on that arc. The value of the flow is equal to the sum of the flows emanating from x_1 or entering x_n. It is easy to show that these two quantities are equal. A classical problem, for which efficient algorithms exist and which has many applications, consists in determining a flow of maximal value on the considered network. Let us consider the transportation network in figure A.10. The capacities of the arcs are given in square brackets next to the arcs. A flow from x_1 to x_7, of value 13, is shown in figure A.10. The corresponding flow of each arc is noted between brackets next to it.

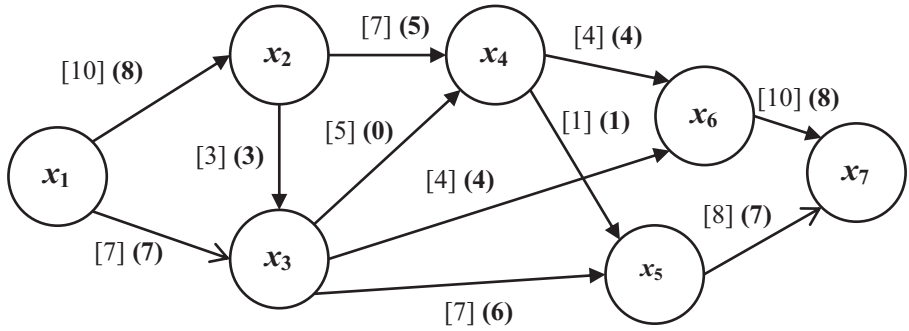

FIG. A.11 – A flow of maximal value (15) on the transportation network in figure A.10.

The flow indicated in figure A.10 is not a maximal flow because there is a flow with a value of 15 from x_1 to x_7 (figure A.11). It can be shown that the value of this new flow is maximal.

Publications concerning graph theory, a branch of mathematics in its own right, are extremely numerous. For more information on the basic notions of this very dynamic discipline, the reader can consult the references cited below.

References and Further Reading

Ahuja R., Magnanti T., Orlin J. (1993) *Network flows: Theory, algorithms, and applications*. Prentice Hall.

Berge C. (1958) *Théorie des graphes et ses applications*. Dunod.

Berge C., Ghouila-Houri A. (1962) *Programmes jeux et réseaux de transport*. Dunod.

Berge C. (1970) *Graphes et Hypergraphes*. Dunod.

Bondy A., Murty U. (2008) Graph theory. Graduate texts in mathematics 244, Springer, Traduction française disponible: https://www-sop.inria.fr/members/Frederic.Havet/Traduction-Bondy-Murty.pdf.

Bretto A., Faisant A., Hennecart F. (2018) *Éléments de théorie des graphes*. Hermes Science Publications.

Promel H.G., Steger A. (2002) *The Steiner tree problem: a tour through graphs, algorithms, and complexity*. Vieweg.

Roseaux (collective of authors) (2005) Exercices et problèmes résolus de recherche opérationnelle (Tome 1), *Graphes, leurs usages, leurs algorithmes*. Dunod.

Roy B. (1969–1970) Algèbre moderne et théorie des graphes orientées vers les sciences économiques et sociales, *Tome 1: notions et résultats fondamentaux, Tome 2: applications et problèmes spécifiques*. Dunod.

Sessions J. (1992) Solving for network connections as a Steiner network problem, *For. Sci.* **38**, 203.

West D. (2000) *Introduction to graph theory*. Prentice Hall.

A.11 Markov Chains

A Markov chain allows to model the dynamic evolution of a random system with N states, S_1, S_2,..., S_N. The system – or the chain – is initially in one of the states, and passes successively from one state to another. Each movement constitutes a step or a transition. If the chain is, at a given instant, in state S_i, then it passes into state

S_j, at the following instant with a probability denoted by p_{ij}, and this probability does not depend on the state in which the chain was previously – nor on the considered instant for a homogeneous chain. The probabilities p_{ij} are called transition probabilities. The process can also remain in state S_i in which it was, and this happens with a probability p_{ii}. If this probability is equal to 1, state S_i is said to be absorbing. The set of these transition probabilities constitutes the transition probability matrix. A graph can be easily associated with this matrix. Figure A.12 presents such a graph for a chain defined on 5 states. Note that, for all $i \in \{1,\ldots, N\}$, $\sum_{j=1}^{N} p_{ij} = 1$.

The starting state of the chain is defined by the initial probability distribution of states S_1, S_2,…, S_N. This is often done by specifying a particular state as the starting state. Markov chains allow a large number of situations to be modelled in a variety of fields. The study of a Markov chain aims at studying the evolution of the system described by this chain. One can be interested, for example, in the probability of being in state S_j at the end of p transitions when the initial state is S_i. One can also seek to determine the probability distribution of the states after a very large (infinite) number of transitions – if this limiting distribution exists. One can also be interested in the probability of entering state S_j for the first time after p transitions, starting from state S_i. Some chains are said to be absorbing. In this case, there is at least one absorbing state and, from any non-absorbing state, an absorbing state can be reached in one or several transitions. For any absorbing Markov chain and for any starting state, the probability of being in an absorbing state after p transitions tends to 1 when p tends to infinity. In such chains, one can look at the probability of ending up in a given absorbing state – if there are several absorbing states – or at the expected number of transitions through the non-absorbing state S_j, starting from the non-absorbing state S_i, before ending up in an absorbing state. One can also look at the number of transitions it will take on average to reach an absorbing state, taking into account the initial state of the chain.

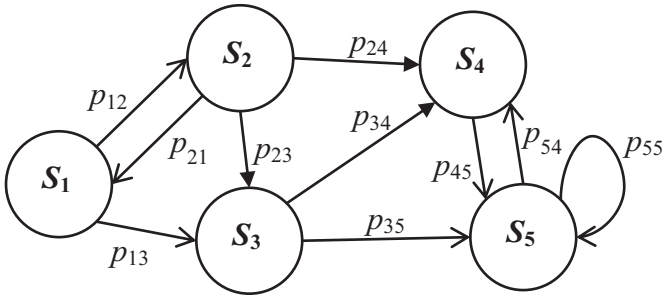

FIG. A.12 – Graph associated with a Markov chain with 5 states S_1, S_2,…, S_5. If, at the time n, the chain is in state S_5, then at the time $n + 1$ it will be in one of the 2 following states: again in state S_5 with the probability p_{55} or in state S_4 with the probability p_{54} ($p_{54} + p_{55} = 1$).

Example A.12. Consider a Markov chain with the 5 states, S_1, S_2, S_3, S_4, and S_5, and whose transition probability matrix is the matrix M below. The value at the intersection of row i and column j is the transition probability from state S_i to state S_j, *i.e.*, the probability p_{ij}. Thus, when the chain is at the instant n in state S_3, it can be at the instant $n+1$ either in state S_1, with the probability 0.4, or in state S_3, with the probability 0.4, or in state S_4, with the probability 0.1, or in state S_5, with the probability 0.1. This chain has 3 transient states, S_1, S_2, and S_3, and 2 absorbing states, S_4 and S_5. A state is transient if the system, being in this state, may not return to this state.

$$M = \begin{pmatrix} 0.4 & 0.5 & 0 & 0 & 0.1 \\ 0 & 0.5 & 0.3 & 0.1 & 0.1 \\ 0.4 & 0 & 0.4 & 0.1 & 0.1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Let us denote by Z the sub-matrix of transition probabilities between transient states and by D the sub-matrix of transition probabilities from a transient state to an absorbing state. In this example,

$$Z = \begin{pmatrix} 0.4 & 0.5 & 0 \\ 0 & 0.5 & 0.3 \\ 0.4 & 0 & 0.4 \end{pmatrix} \quad \text{and} \quad D = \begin{pmatrix} 0 & 0.1 \\ 0.1 & 0.1 \\ 0.1 & 0.1 \end{pmatrix}.$$

Let us denote by π_{ij} the expected number of passages through state S_j – transient – starting from state S_i – transient – before absorption, and by Π the matrix whose general term is π_{ij}. We can show that $\Pi = (I - Z)^{-1}$ where I designates the identity matrix of the same dimension as Z. In our example, Π is equal to the inverse of the matrix

$$I - Z = \begin{pmatrix} 1 - 0.4 & -0.5 & 0 \\ 0 & 1 - 0.5 & -0.3 \\ -0.4 & 0 & 1 - 0.4 \end{pmatrix}, \quad \textit{i.e.}, \quad \Pi = \begin{pmatrix} 2.5 & 2.5 & 1.25 \\ 1 & 3 & 1.5 \\ 5/3 & 5/3 & 2.5 \end{pmatrix}.$$

Thus, starting from state S_2, the system will go on average 3 times through this same state before absorption. Let us now look at the probability, being in state S_i, $i = 1, 2, 3$, of ending up in the absorbing state S_j, $j = 4, 5$. Let a_{ij} be this probability and A be the matrix of general term a_{ij}. We can show that $A = \Pi . D$. In our example,

$$A = \begin{pmatrix} 0.375 & 0.625 \\ 0.45 & 0.55 \\ 1.25/3 & 1.75/3 \end{pmatrix}.$$

Thus, starting from state S_1, the system will end up in the absorbing state S_5 with a probability equal to 0.625.

Markov chain theory has proven to be very effective in modelling and studying many concrete or theoretical random phenomena. For a more detailed presentation of this theory the reader can consult the references cited below.

References and Further Reading

Brémaud P. (2009) *Initiation aux probabilités et aux chaînes de markov*. Springer.
Ching W.-K., Ng M.K. (2006) *Markov chains: models, algorithms and applications*. Springer.
Douc R., Moulines E., Priouret P., Soulier P. (2019) *Markov chains*. Springer.
Gagniuc P.A. (2017) *Markov chains: from theory to implementation and experimentation*. Wiley.
Gordon P. (1964) *Théorie des chaînes de markov finies et ses applications*. Dunod, Paris.
Graham C. (2008) *Chaînes de markov - cours et exercices corrigés*. Dunod.
Grinstead C.M., Snell J.L. (1997) *Introduction to probability*, 2nd edition 2006. American Mathematical Society, Providence.
Kemeny J.G., Snell J.L. (1983) *Finite markov chains*, Springer.
Méléard S. (2016) *Modèles aléatoires en écologie et évolution*. Springer.
Privault N. (2018) *Understanding Markov chains, examples and applications*. Springer.
Roseaux (collective of authors) (2004) Exercices et problèmes résolus de recherche opérationnelle (Tome 2), *Phénomènes aléatoires en recherche opérationnelle*. Dunod, Paris.
Sericola B. (2013) *Chaînes de markov, theorie, algorithmes et applications*. Hermes-Lavoisier.