Algorithmes

Domaines d'applications
Industrie
Transport
Secteur public
Télécoms
Finance

Christian Prins - Marc Sevaux

Programmation linéaire avec Excel

55 problèmes d'optimisation modélisés pas à pas et résolus avec Excel

En complément sur www.editions-eyrolles.com :

- des applications supplémentaires
- les fichiers Excel des 55 problèmes

EYROLLES

Programmation linéaire avec Excel

*Professeur à l'Université de Technologie de Troyes, **Christian Prins** enseigne l'optimisation et la logistique. Il dirige l'équipe de recherche OSI de l'Institut Charles Delaunay et est responsable de l'axe « Gestion des crises » de l'UMR STMR. Il effectue des recherches en optimisation du transport.*

*Professeur à l'Université de Bretagne-Sud à Lorient, **Marc Sevaux** enseigne la recherche opérationnelle. Il est directeur-adjoint du Lab-STICC et responsable du groupe Recherche Opérationnelle. La conception de méthodes d'optimisation pour des applications réelles est sa principale activité de recherche.*

Premier du genre en français, cet ouvrage montre comment utiliser le logiciel Excel et son solveur pour résoudre des problèmes d'optimisation complexes dans les domaines les plus divers : gestion de production, ordonnancement, transport, économie et finances, gestion des organisations, aide à la décision, etc.

Le livre met l'accent sur la démarche de modélisation, appliquée à la résolution de 55 problèmes concrets regroupés par discipline. À partir du descriptif de chaque cas, vous apprendrez, pas à pas, à construire un modèle de programmation linéaire, à le traduire dans une feuille de calcul Excel et à le résoudre par l'intermédiaire du solveur, avec l'aide de macros VBA dans certains cas.

À qui s'adresse ce livre ?

- **Aux utilisateurs d'Excel confrontés à des problèmes d'optimisation au quotidien.**
- **Aux décideurs, industriels, ingénieurs et responsables de services ayant à résoudre en entreprise des problèmes complexes d'optimisation et d'aide à la décision.**
- **Aux développeurs d'applications complètes sous Excel.**
- **Aux étudiants et élèves-ingénieurs des disciplines scientifiques et économiques.**
- **Aux enseignants de ces disciplines à la recherche d'un ouvrage complet avec études de cas sur la programmation linéaire.**

Sommaire

En complément sur www.editions-eyrolles.com

- Les fichiers Excel des 55 problèmes du livre, en versions 2003 (compatible 2007) et 2010.
- Un chapitre supplémentaire proposant trois puzzles résolus par programmation linéaire.

Plate-forme requise :
PC avec Microsoft Excel 2003 ou ultérieur.

www.editions-eyrolles.com
Groupe Eyrolles | Diffusion Geodif

Programmation linéaire avec Excel

Dans la même collection

A. Cornuéjols, L. Miclet. – **Apprentissage artificiel.**
Concepts et algorithmes.
N°12471, 2e édition, 2010, 804 pages.

G. Dreyfus *et al*. – **Apprentissage statistique.**
Réseaux de neurones – Cartes topologiques – Machines à vecteurs supports.
N°12229, 2008, 450 pages.

P. Naïm, P.-H. Wuillemin, P. Leray, O. Pourret, A. Becker. – **Réseaux bayésiens.**
N°11972, 3e édition, 2007, 424 pages.

G. Fleury, P. Lacomme et A. Tanguy. – **Simulation à événements discrets.**
Modèles déterministes et stochastiques – Exemples d'applications implémentés en Delphi et en C++.
N°11924, 2006, 444 pages avec CD-Rom.

J. Richalet *et al*. – **La commande prédictive.**
Mise en œuvre et applications industrielles.
N°11553, 2004, 256 pages.

P. Lacomme, C. Prins, M. Sevaux. – **Algorithmes de graphes.**
N°11385, 2003, 368 pages, avec CD-Rom.

J. Dréo, A. Pétrowski, P. Siarry, E. Taillard. – **Métaheuristiques pour l'optimisation difficile.**
Recuit simulé, recherche tabou, algorithmes évolutionnaires et algorithmes génétiques, colonies de fourmis…
N°11368, 2003, 368 pages.

Y. Collette, P. Siarry. – **Optimisation multiobjectif.**
N°11168, 2002, 316 pages (disponible en version numérique uniquement).

C. Guéret, C. Prins, M. Sevaux. – **Programmation linéaire.**
65 problèmes d'optimisation modélisés et résolus avec Visual XPress.
N°9202, 2000, 365 pages, avec CD-ROM.

Programmation linéaire avec Excel

Christian Prins - Marc Sevaux

EYROLLES

ÉDITIONS EYROLLES
61, bd Saint-Germain
75240 Paris Cedex 05
www.editions-eyrolles.com

© Groupe Eyrolles, 2011, ISBN : 978-2-212-12659-4

Avant-propos

En 2000, les deux auteurs ont publié avec Christelle Guéret aux éditions Eyrolles un livre intitulé *Programmation linéaire : 65 problèmes d'optimisation modélisés et résolus avec Visual Xpress*. Cet ouvrage partait d'un constat : de nombreux étudiants en sciences, économie et écoles d'ingénieurs sont initiés à l'optimisation, mais d'une manière très académique ne permettant pas de résoudre des problèmes réels. D'après notre expérience de l'enseignement de cette discipline, cette incapacité provient du manque de pratique à la modélisation et de logiciels *ad hoc*, permettant d'éviter de fastidieux calculs à la main.

Le livre de 2000 visait à résoudre ces problèmes en offrant un entraînement systématique à la modélisation, à travers des problèmes résolus, regroupés en chapitres d'applications comme les industries minières et de process, les télécommunications et le transport. Les problèmes étaient volontairement dimensionnés pour ne pas être résolubles à la main et ils étaient présentés en trois temps. Chaque problème faisait d'abord l'objet d'un énoncé très réaliste. Il était ensuite modélisé sous forme mathématique pour donner un programme linéaire indépendant de la syntaxe des logiciels disponibles. Enfin, le modèle était traduit et résolu dans un logiciel, à l'époque *Visual Xpress*. Aux chapitres d'applications s'ajoutaient trois chapitres préliminaires : deux apportant les bases théoriques suffisantes pour la programmation linéaire et la programmation linéaire en nombres entiers, et un pour présenter le logiciel utilisé et offrir un panorama des autres produits existants.

L'utilisation de ce livre en enseignement nous a permis d'observer une nette augmentation du rendement pédagogique et de la motivation des étudiants. Cependant, à l'époque, les PC commençaient à se répandre mais étaient plus chers qu'aujourd'hui. Les établissements d'enseignement supérieur disposaient tous de salles de travaux pratiques équipées de PC, mais la plupart des étudiants n'avaient pas d'ordinateur personnel. Beaucoup de logiciels étaient certes puissants, mais complexes pour des débutants ou réservés aux cursus d'informatique.

Dix ans après, la situation a évolué. Presque tous les étudiants de l'enseignement supérieur possèdent un PC et utilisent des outils bureautiques comme ceux de Microsoft ou la suite Open Office du domaine public. Or, le tableur Excel de Microsoft et celui d'OpenOffice incluent un solveur peu connu de programmation linéaire et non linéaire. Il permet de traiter un large éventail d'applications, en évitant le recours à la programmation et en bénéficiant des services du tableur en matière de présentation des données et de réalisation de graphiques.

Nous avons testé l'enseignement de ce solveur et avons obtenu de très bons résultats, même en premier cycle universitaire et en IUT. De plus, le produit est une étape commode avant de passer à des logiciels spécialisés plus complexes.

L'idée nous est donc venue de démocratiser encore plus la programmation linéaire, en reprenant le principe du livre de 2000 mais en utilisant Excel pour la résolution. Le

chapitre 1 sur la programmation linéaire et le chapitre 2 sur la programmation linéaire en nombres entiers ont été conservés, mais en complétant leur bibliographie par des références récentes.

Le chapitre 3 consacré en 2000 au logiciel Visual Xpress décrit maintenant le solveur d'Excel. Le contenu s'applique intégralement au tableur d'OpenOffice, qui possède une interface identique. Le chapitre propose aussi une méthode de modélisation, qui permet au débutant de modéliser étape par étape un problème d'optimisation puis de le traduire sous Excel. Des fonctions peu connues d'Excel sont aussi présentées, ainsi qu'un répertoire de ressources du Web sur les principaux logiciels d'optimisation linéaire et non linéaire.

Le nouveau chapitre 4 décrit l'utilisation de *Visual Basic for Applications* (VBA) avec le solveur. En effet, il est commode dans certaines applications de cacher le modèle à l'utilisateur ou de déclencher l'optimisation en cliquant sur un bouton, ce qui nécessite d'écrire quelques lignes de VBA. Le chapitre commence par une introduction expresse à VBA, pour les lecteurs ayant des rudiments de programmation dans un autre langage. Il montre ensuite comment contrôler le solveur depuis VBA. Les lecteurs réfractaires à la programmation pourront ignorer sans inconvénient ce chapitre 4 et la minorité de problèmes qui utilisent ces techniques dans les chapitres d'application.

Les dix autres chapitres reprennent la plupart des problèmes d'application du livre de 2000. Cependant, certains d'entre eux ont été modifiés pour être davantage compatibles avec la logique bidimensionnelle d'Excel. Il a même fallu supprimer quelques problèmes nécessitant des variables à trois indices ou plus. En règle générale, l'énoncé de chaque problème et sa modélisation mathématique ont été repris. Les parties sur la traduction dans le logiciel ont été complètement réécrites pour Excel, en expliquant avec des copies d'écran la disposition de la feuille de calcul, les formules, le modèle à saisir dans la boîte de dialogue du solveur et les résultats. De plus, la bibliographie à la fin de chaque chapitre a été revue et complétée.

Le chapitre 5 rassemble des problèmes de mélange ou de séparation de composés, rencontrés dans les industries minières et de process. Ces problèmes, qui font rarement appel à des variables entières, sont les plus simples : ils conviennent donc bien aux débutants. Les deux chapitres suivants sont consacrés à deux autres groupes importants d'applications industrielles : les problèmes d'ordonnancement (chapitre 6) et les problèmes de planification de production (chapitre 7).

Dans beaucoup d'applications, il faut remplir un espace limité (caisses, cales de navires, disques d'ordinateurs) avec des objets ou, au contraire, découper des matériaux pour en extraire des motifs tout en minimisant les chutes. Ce genre de problème fait l'objet du chapitre 8 sur la découpe et le conditionnement (*cutting and packing* en anglais).

Les questions de flux et de placement d'installations industrielles soulèvent de très nombreux problèmes intéressants en optimisation. Nous avons choisi de les aborder au travers de trois domaines représentatifs : le transport terrestre (chapitre 9), le transport aérien (chapitre 10), et enfin le monde foisonnant des télécommunications (chapitre 11). Plusieurs modèles théoriques sont transposables entre ces trois chapitres, mais on y trouve aussi des applications plus spécifiques, comme les tournées de véhicules en transport routier, les problèmes de correspondance d'avions en transport aérien, et tout ce qui concerne le dimensionnement et la fiabilité des réseaux en télécommunications.

Pour éviter de nous adresser seulement aux scientifiques, industriels et logisticiens, nous avons tenu à présenter des domaines d'application plus récents ou moins connus de la programmation linéaire.

Le chapitre 12 est ainsi consacré aux applications en économie et en optimisation financière. Les problèmes d'emploi du temps et de gestion du personnel font l'objet du chapitre 13. La programmation linéaire peut également rendre de grands services aux collectivités locales, aux administrations, et au secteur public en général : le chapitre 14 présente différents exemples. Les copies d'écran ayant entraîné une augmentation du nombre de pages, nous avons décidé pour rester en deçà des 400 pages de placer le dernier chapitre du livre de 2000 (jeux et casse-tête) sur le site Internet des éditions Eyrolles (*www.editions-eyrolles.com*).

En tout, 55 problèmes sont modélisés pas à pas et résolus. En fin d'ouvrage, une première annexe présente une liste récapitulative des problèmes du livre et indique les correspondances entre les applications traitées et les modèles théoriques classiques. Une deuxième annexe liste les fichiers Excel des problèmes. Enfin, la troisième annexe rassemble les références bibliographiques. Un grand soin a été apporté à la sélection des références, qui mixent des ouvrages didactiques, des articles de synthèse et des articles plus pointus.

Pour la résolution des modèles, nous avons utilisé Excel 2007, mais sauvegardé les fichiers au format d'Excel 2003 pour des raisons de compatibilité ascendante. La compatibilité avec Excel 2010 a été vérifiée pour tous les modèles. L'annexe 2 donne la liste des fichiers obtenus, qui peuvent être téléchargés gratuitement sur le site Internet des éditions Eyrolles. Elle explique aussi les quelques modifications mineures à effectuer pour Excel 2010.

Pour l'enseignement, nous donnons en général un des énoncés aux étudiants qui doivent d'abord établir le modèle mathématique. Cette étape est la plus importante car, en entreprise, on peut leur imposer un autre logiciel qu'Excel. Or, le modèle mathématique reste valable quel que soit le logiciel de programmation linéaire. Seulement ensuite, les étudiants peuvent passer à la traduction en Excel, en réfléchissant soigneusement à la disposition des données et des informations à calculer pour les contraintes. La méthode expliquée au chapitre 4 a d'ailleurs été élaborée en observant les erreurs les plus communes.

Nous espérons que cet ouvrage rendra service aux enseignants, étudiants, ingénieurs, informaticiens, industriels et décideurs intéressés par la programmation linéaire, et qu'il permettra à cette discipline de se diffuser encore plus largement.

Christian Prins et Marc Sevaux

Table des matières

CHAPITRE 1

Programmation linéaire

1.1 Introduction

Bien qu'on puisse modéliser des problèmes d'optimisation et utiliser des logiciels sans connaître la théorie qui se cache derrière, quelques notions sont utiles pour démystifier le sujet. Ce chapitre présente donc des bases de programmation linéaire suffisantes pour les chapitres 5 à 14 consacrés à la modélisation. Le § 1.2 donne un exemple de programme linéaire puis définit les programmes linéaires en général, sous leurs deux principales formes. Les programmes linéaires à deux variables peuvent être résolus par une méthode géométrique, présentée au § 1.3, qui permet de saisir physiquement les principes.

Les problèmes à plus de deux variables sont résolus par une méthode algébrique célèbre, *l'algorithme du simplexe*, introduite au § 1.4. La forme tableau de l'algorithme, qui facilite les calculs, est présentée au § 1.5. Quelques situations particulières font l'objet du § 1.6. La dualité et l'analyse de sensibilité sont riches en propriétés qui dépassent le cadre de ce livre. Les paragraphes 1.7 et 1.8 consacrés à ces sujets ne présentent que des rudiments. Quelques évolutions récentes comme les méthodes de points intérieurs sont évoquées au § 1.9. La dernière partie est consacrée aux compléments et aux références pour approfondir le sujet.

1.2 Notion de programme linéaire

1.2.1 Composantes d'un programme linéaire et exemple

Soit une usine qui produit deux ciments rapportant 50 $ et 70 $ la tonne. Pour fabriquer une tonne de ciment 1, il faut 40 min de calcination dans un four et 20 min de broyage. Pour fabriquer une tonne de ciment 2, il faut 30 min de four et 30 min de broyage. Le four et l'atelier de broyage sont disponibles 6 h et 8 h par jour. Combien de ciment de chaque type peut-on produire par jour pour maximiser le bénéfice ? Ce problème se modélise par le *programme linéaire* (PL) suivant, en notant x_1 et x_2 les quantités de ciment à fabriquer.

(1) Max $z = 50x_1 + 70x_2$

(2) $40x_1 + 30x_2 \leq 360$

(3) $20x_1 + 30x_2 \leq 480$

(4) $x_1, x_2 \geq 0$

La ligne (1) représente le profit total z qui est le critère à optimiser, appelé aussi *fonction-objectif* (nom composé), *fonction de coût* ou *fonction économique*. *Max* signifie que ce critère doit être maximisé ; on écrirait *Min* pour minimiser. Les autres lignes désignent des *contraintes*. La contrainte (2) concerne la disponibilité du four : elle stipule que le temps total de calcination requis par les ciments ne doit pas dépasser 360 min ou 6 h. La contrainte (3) décrit de même la disponibilité du broyeur. Les contraintes (4) précisent les domaines des variables.

1.2.2 Forme générale d'un programme linéaire et extensions

Plus généralement, on appelle *programme mathématique* un problème d'optimisation d'une fonction-objectif de plusieurs variables en présence de contraintes. Le programme est dit *linéaire* si la fonction et les contraintes sont toutes des combinaisons linéaires de variables. Il a la forme générique suivante. Il comporte n variables non négatives (3), m contraintes d'égalité ou d'inégalité (2), et la fonction-objectif à optimiser (1). Le coefficient de coût ou de profit de la variable x_j est noté c_j, celui de la variable x_j dans la contrainte i est noté a_{ij}. La contrainte i a un second membre constant b_i. Les contraintes simples de positivité ne sont pas incluses dans les m contraintes, car elles sont gérées à part par les algorithmes.

(1) Max ou Min $z = \sum_{j=1}^{n} c_j x_j$

(2) $\forall i = 1 \ldots m : \sum_{j=1}^{n} a_{ij} x_j \leq, =$ ou $\geq b_i$

(3) $\forall j = 1 \ldots n : x_j \geq 0$

Des valeurs de variables qui vérifient toutes les contraintes, comme $x_1 = 4$ et $x_2 = 2$ dans l'exemple des ciments, forment une *solution réalisable* (SR) du PL. Une solution réalisable est *optimale* si aucune autre solution n'a un profit supérieur.

Si les variables sont astreintes à être entières, on obtient un *programme linéaire en nombres entiers* (PLNE). Un *programme linéaire en 0-1* est un cas particulier de PLNE dont les variables ne peuvent prendre que deux valeurs 0-1 ; ces variables sont dites *booléennes*, *binaires* ou *de décision*. Un *PL mixte* comprend à la fois des variables continues et des variables entières. Enfin, à partir du moment où au moins une contrainte ou la fonction-objectif n'est plus une combinaison linéaire de variables, on a affaire à un *programme non linéaire* (PNL). Les PLNE et PL en 0-1, qui font l'objet du chapitre 2, sont plus difficiles à résoudre que les PL ordinaires. Les PNL sont encore plus difficiles, sauf dans quelques cas particuliers comme des fonctions-objectifs convexes. Les algorithmes actuels ne trouvent en général qu'un optimum local et ils sortent du cadre de cet ouvrage.

1.2.3 Formes matricielles classiques et conversions

Notons $x = (x_1, x_2, \ldots, x_n)^T$ le vecteur des variables, $b = (b_1, b_2, \ldots, b_m)^T$ celui des seconds membres des contraintes, $c = (c_1, c_2, \ldots, c_n)$ les coûts ou profits associés aux variables, et A la matrice $m \times n$ des a_{ij}. Dans la suite, nous n'écrirons plus les signes de transposition pour alléger l'écriture. On peut alors écrire un PL sous forme matricielle. Deux formes sont courantes : la *forme canonique* avec des contraintes ≤, utilisée pour la résolution graphique, et la *forme standard* avec égalités, pour la résolution algébrique par des algorithmes. Par convention, la forme standard est souvent exprimée avec des seconds membres positifs.

Forme canonique	*Forme standard*
Max $c.x$	Max $c.x$
$A.x \leq b$	$A.x = b$
$x \geq 0$	$x \geq 0$

Ces formes ne servent qu'à simplifier les présentations théoriques. Dans la réalité, un PL peut présenter des égalités et des inégalités, et les logiciels du marché acceptent heureusement ces mélanges. On peut facilement convertir les formes mixtes en formes classiques. Ainsi, toute contrainte d'égalité peut être remplacée par deux inégalités.

$$\sum_{j=1}^{n} a_{ij}x_j = b_i \Leftrightarrow \begin{cases} \sum_{j=1}^{n} a_{ij}x_j \leq b_i \\ -\sum_{j=1}^{n} a_{ij}x_j \leq -b_i \end{cases}$$

On peut convertir une inégalité en égalité en ajoutant ou soustrayant une *variable d'écart* $e_i \geq 0$, propre à chaque contrainte i. À l'optimum, pour une inégalité ≤ concernant la disponibilité d'une ressource i, cette variable indique la quantité inutilisée de la ressource.

$$\sum_{j=1}^{n} a_{ij}x_j \leq b_i \text{ et } e_i \geq 0 \Leftrightarrow \sum_{j=1}^{n} a_{ij}x_j + e_i = b_i$$

D'autres conversions sont possibles. Ainsi, on peut passer d'une maximisation à une minimisation, car maximiser z revient à minimiser $-z$. Il ne faut pas oublier alors de multiplier par -1 la valeur de la fonction-objectif trouvée par la minimisation ! L'exigence de variables positives n'est pas restrictive, car une variable x_j non contrainte en signe peut toujours s'écrire comme une différence $x'_j - x''_j$ de deux variables non négatives.

1.2.4 Interprétation économique

Un PL a une *interprétation économique* très large. Soit un acteur économique qui exerce n activités avec des intensités x_j à déterminer. Ces activités utilisent m ressources. On connaît la quantité a_{ij} de ressource i nécessaire pour exercer l'activité j avec une intensité 1. On connaît aussi le profit ou le coût c_j pour une intensité 1 de l'activité j. On veut trouver les intensités des activités, compatibles avec les ressources, pour maximiser le profit ou minimiser le coût. Ce problème est modélisable par un PL sous forme canonique.

La programmation linéaire définit donc une classe très large de modèles, mais dans laquelle on prend deux hypothèses restrictives fondamentales : la *proportionnalité* des coûts et des consommations de ressources aux intensités d'activités, et l'*additivité* des consommations de ressources (pas d'interactions entre activités). Nous verrons en fait que de nombreux problèmes, en apparence non linéaires, peuvent être rendus linéaires.

Par exemple, l'hypothèse de proportionnalité n'est pas respectée quand on produit un bien en grande série. Le prix de vente par unité bénéficie souvent de tarifs dégressifs, grâce aux économies d'échelle. Le prix de vente en fonction de la quantité est alors une fonction linéaire par morceaux qui croît de moins en moins vite. Il n'empêche que ce genre de fonction se linéarise facilement, moyennant des variables supplémentaires. Ainsi, le champ d'application de la programmation linéaire est bien plus vaste qu'il n'y paraît.

1.3 Résolution graphique

Appelée aussi *résolution géométrique*, elle est possible pour un PL sous forme canonique avec deux ou trois variables ($n = 2$ ou $n = 3$). Soit par exemple le PL suivant, qu'on interprétera comme le calcul des quantités à produire x_1 et x_2 de deux produits pour maximiser un profit z.

$$\begin{array}{lrcrcl} \text{Max } z = & x_1 & + & 2 \cdot x_2 & & \\ & x_1 & + & x_2 & \leq & 6 \\ & & & x_2 & \leq & 3 \\ & x_1, & & x_2 & \geq & 0 \end{array}$$

Ce PL est bien sous forme canonique. En utilisant les notations matricielles, on peut écrire $m = 2$, $n = 2$, $c = (1, 2)$, $x = (x_1, x_2)^T$, $b = (6, 3)^T$ et :

$$A = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$$

Pour ce PL à deux variables, on trace dans le plan les axes des coordonnées pour les valeurs positives de x_1 et x_2. On finit de délimiter le *domaine des solutions réalisables*, qui forme un *polygone convexe*, en traçant les droites d'équations $x_1 + x_2 = 6$ et $x_2 = 3$ (figure 1.1 page suivante). Pour chaque droite, on indique par des flèches le demi-plan à conserver. On peut aussi hachurer les demi-plans interdits, mais la figure est souvent moins lisible.

Traçons la droite de profit $z = 0$. Son intersection avec le domaine des solutions réalisables est réduite au point O et correspond à la solution triviale où on ne produit rien, ce qui ne rapporte rien. Une droite d'équation $z = k$, où k est une constante entre 0 et 9 (exclus) donne par intersection avec le domaine tout un segment de plans de production possibles, ayant le même profit k. On peut augmenter k jusqu'à 9, ce qui donne la solution correspondant au point J : $x_1 = 3$, $x_2 = 3$. Cette solution est optimale, car si on augmente encore k, on sort du domaine. En remplaçant dans le PL les variables par leurs valeurs, on peut vérifier le coût et le respect des contraintes.

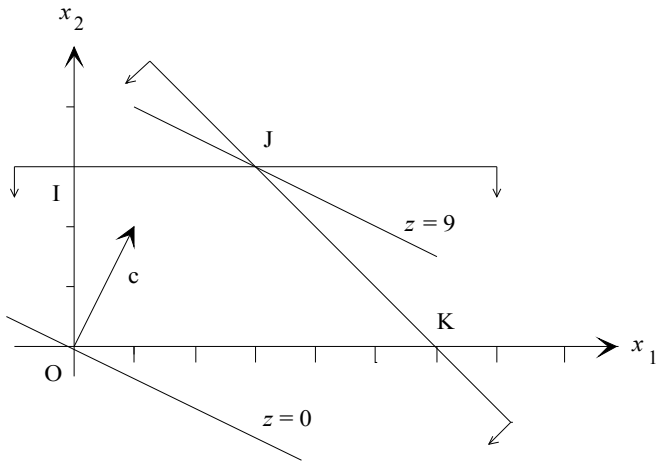

Figure 1.1 – Exemple de résolution graphique

On peut formuler sur la figure des remarques générales. Les contraintes de positivité confinent les solutions dans le quart de plan positif et chaque autre contrainte définit un demi-espace (ici un demi-plan). L'intersection de tous ces espaces forme un *polygone convexe* non vide, ici OIJK. La *convexité* signifie que pour deux points quelconques A et B du polygone, le segment [A, B] est contenu dans le polygone. Pour un nombre quelconque n de variables, on parle de *polyèdre convexe*. Pour k croissant, les droites d'équation $z = k$ forment une famille de droites parallèles. z augmente dans la direction du vecteur c de la fonction-objectif. L'optimum x^*, ici unique, est atteint au sommet J = (3, 3) du polyèdre. Le coût optimal correspondant est $z^* = 9$.

Les cas spéciaux suivants pourraient se produire. En supprimant $x_1 + x_2 \leq 6$, le domaine des solutions serait *non borné*, ainsi que l'optimum. L'optimum peut cependant être fini même si le domaine est non borné : ce serait le cas pour $c = (-1, 2)$, qui donnerait le point I. Si on ajoutait la contrainte $x_1 - x_2 \leq -4$, le polyèdre serait *vide* et il n'y aurait aucune solution réalisable. Enfin, si on maximisait $z = x_1 + x_2$, il y aurait *plusieurs optima* (tous les points de l'arête JK). Remarquons cependant que l'ensemble des points optimaux, s'il est non vide, comprend toujours un sommet du polyèdre.

Pour un problème réel comme un plan de production, le domaine n'est normalement pas vide car il y a au moins une solution : le plan de production *actuel* de l'entreprise. De plus, l'optimum est borné en pratique à cause de limites sur les ressources. Dans un problème réel, l'absence de solution indique en général un problème surcontraint, tandis que l'oubli d'une contrainte peut donner un optimum non borné. Une erreur de saisie du modèle dans un logiciel, comme une inversion de signe, peut également produire ces phénomènes. Dans tous ces cas anormaux, il faut revoir soigneusement la formulation.

Si on est bon dessinateur, la résolution graphique est encore possible pour trois variables : les contraintes définissent des demi-espaces dont l'intersection forme un polyèdre à trois dimensions (une sorte de cristal), et les solutions de profit z constant forment une famille de plans parallèles. Il est clair qu'il faut utiliser une autre méthode en présence de plus de trois variables. C'est le but de la *résolution algébrique* des sections suivantes.

1.4 Principes de la résolution algébrique

1.4.1 Bases et solutions de base

Cette résolution utilise la forme standard. Rappelons que le système d'égalités linéaires d'un PL en forme standard s'écrit matriciellement $Ax = b$, avec A une matrice $m \times n$ telle que $m \leq n$, x un vecteur $n \times 1$, et b un vecteur $m \times 1$. On suppose aussi qu'il n'y a pas de contrainte redondante, c'est-à-dire combinaison linéaire d'autres contraintes : le *rang* de A vaut m, en termes mathématiques. Heureusement, les logiciels du commerce fonctionnent quand même correctement si ces hypothèses ne sont pas vérifiées !

On appelle *base* de A, ou *matrice de base*, toute sous-matrice carrée inversible B, $m \times m$, de A. Pour une base B, on peut (en réarrangeant les colonnes si nécessaire) partitionner A en (B, N) et x en (x_B, x_N). N est la matrice $m \times (n\text{-}m)$ des colonnes hors base, ou *matrice hors base*. Le vecteur x_B a pour composantes les *m variables de base* (associées aux colonnes de B). x_N a pour composantes les *n-m variables hors base*. Le système de contraintes et la fonction-objectif peuvent alors s'écrire de manière équivalente :

$$A \cdot x = b \Leftrightarrow B \cdot x_B + N \cdot x_N = b \Leftrightarrow x_B = B^{-1} \cdot b - B^{-1} \cdot N \cdot x_N$$
$$z = c \cdot x = c_B \cdot x_B + c_N \cdot x_N = c_B \cdot B^{-1} \cdot b + \left(c_N - c_B \cdot B^{-1} \cdot N\right) \cdot x_N$$

Pour une base choisie B, le système équivalent est simplement une expression des variables de base en fonction des variables hors base. On a une solution évidente en forçant x_N à 0 : on a alors $x_B = B^{-1}.b$. Cette solution du PL est la *solution de base* (SB) associée à la base B. Elle peut violer des contraintes de positivité : on appelle *solution de base réalisable* (SBR) une solution réalisable dont toutes les variables sont positives ou nulles.

1.4.2 Exemple d'énumération des bases et changements de base

Reprenons le PL traité par la résolution géométrique, mais mis sous forme standard en ajoutant une *variable d'écart* x_3 pour la première contrainte, et une autre x_4 pour la seconde.

$$\begin{array}{lrcrcrcrcl} \text{Max } z = & x_1 & + & 2 \cdot x_2 & & & & & & \\ & x_1 & + & x_2 & + & x_3 & & & = & 6 \\ & & & x_2 & & & + & x_4 & = & 3 \\ & x_1, & & x_2, & & x_3, & & x_4 & \geq & 0 \end{array}$$

En notant a_i la colonne i de A, la matrice A (2×4) de ce programme linéaire est :

$$(a_1, a_2, a_3, a_4) = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}$$

Les bases de A sont ses sous-matrices carrées inversibles 2×2, en fait toutes sauf (a_1, a_3). On trouve ainsi les cinq matrices de base de la page suivante, pour lesquelles on a calculé les solutions de base. Toutes ces solutions de base sont réalisables, sauf celle correspondant à B_4.

$$B_1 = (a_1, a_2) = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} \Rightarrow x_B = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = B^{-1}b = \begin{pmatrix} 1 & -1 \\ 0 & 1 \end{pmatrix} \times \begin{pmatrix} 6 \\ 3 \end{pmatrix} = \begin{pmatrix} 3 \\ 3 \end{pmatrix}$$

$$B_2 = (a_1, a_4) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \Rightarrow x_B = \begin{pmatrix} x_1 \\ x_4 \end{pmatrix} = B^{-1}b = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \times \begin{pmatrix} 6 \\ 3 \end{pmatrix} = \begin{pmatrix} 6 \\ 3 \end{pmatrix}$$

$$B_3 = (a_2, a_3) = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \Rightarrow x_B = \begin{pmatrix} x_2 \\ x_3 \end{pmatrix} = B^{-1}b = \begin{pmatrix} 0 & 1 \\ 1 & -1 \end{pmatrix} \times \begin{pmatrix} 6 \\ 3 \end{pmatrix} = \begin{pmatrix} 3 \\ 3 \end{pmatrix}$$

$$B_4 = (a_2, a_4) = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \Rightarrow x_B = \begin{pmatrix} x_2 \\ x_4 \end{pmatrix} = B^{-1}b = \begin{pmatrix} 1 & 0 \\ -1 & 1 \end{pmatrix} \times \begin{pmatrix} 6 \\ 3 \end{pmatrix} = \begin{pmatrix} 6 \\ -3 \end{pmatrix}$$

$$B_5 = (a_3, a_4) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \Rightarrow x_B = \begin{pmatrix} x_3 \\ x_4 \end{pmatrix} = B^{-1}b = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \times \begin{pmatrix} 6 \\ 3 \end{pmatrix} = \begin{pmatrix} 6 \\ 3 \end{pmatrix}$$

Dans cet exemple, on ne peut pas visualiser le polyèdre en quatre dimensions. Mais le polyèdre utilisé pour la résolution graphique en est une projection à deux dimensions, sans les variables d'écart (figure 1.1). *On constate alors que les SBR correspondent aux sommets J, K, I et O du polyèdre*. La base B_5 est la matrice identité associée aux variables d'écart. La SBR associée correspond au point O de la résolution graphique. Dans un contexte de production, elle consiste à ne rien produire : on met à zéro les variables du PL canonique d'origine, et les variables d'écart sont égales aux quantités de ressources disponibles (non consommées), définies par le vecteur b.

Les opérations précédentes ont l'air compliquées sous forme matricielle. Elles sont en fait simples si on détaille les équations en exprimant les variables de base en fonction des variables hors base. Par exemple, pour retrouver la solution de base associée à B_1 (point J de la résolution graphique), on part du système $Ax = b$ de la forme standard.

$$x_1 + x_2 + x_3 = 6$$
$$x_2 + x_4 = 3$$

Il faut transformer ce système pour exprimer les variables de base x_1 et x_2 en fonction des autres variables (hors base) x_3 et x_4. Il se trouve que x_2 est déjà exprimée en fonction de x_4 dans la ligne 2. En substituant son expression ($x_2 = 3 - x_4$) dans la première équation, on obtient le système suivant, traduction non matricielle du système $x_B = B^{-1}.b - B^{-1}.N.x_N$. En fixant les variables hors base x_3 et x_4 à 0, on retrouve effectivement la solution réalisable de base $x_1 = 3$ et $x_2 = 3$ correspondant au point J.

$$x_1 = 3 - x_3 + x_4$$
$$x_2 = 3 - x_4$$

Les bases pour deux sommets adjacents du polyèdre ne diffèrent que par une variable. Si on dispose du système exprimant les variables de base en fonction de celles hors base, on peut déduire le système pour l'autre base, sans inversion de matrice. Par exemple, supposons que l'on parte de la base B_5 des variables d'écart x_3 et x_4, correspondant au point O.

$$x_3 = 6 - x_1 - x_2$$
$$x_4 = 3 - x_2$$

Pour passer à la base B_3 formée des variables x_2 et x_3 et correspondant au point I, x_2 remplace x_4 dans la base. Ceci s'effectue dans la ligne définissant x_4 en déplaçant x_2 dans le membre de gauche et x_4 dans le membre de droite. Il faut ensuite éliminer x_2 dans les membres de droite des autres équations, réservées aux variables hors base. Ce qui s'effectue simplement en remplaçant x_2 par son expression, 3 - x_4. Cette transformation du système est appelée *pivotage* de x_4 vers x_2. On obtient ainsi le système suivant :

$$x_3 = 3 - x_1 + x_4$$
$$x_2 = 3 - x_4$$

1.4.3 Propriétés fondamentales de la programmation linéaire

Les méthodes de résolution algébriques reposent sur deux propriétés fondamentales à admettre, mais vues intuitivement dans les paragraphes précédents. Les précautions de style sont nécessaires à cause des polyèdres non bornés ou vides :

- Si une fonction linéaire atteint son maximum (ou son minimum) sur le polyèdre X défini par les contraintes, alors cet optimum a lieu en un sommet de X.
- Si X est non vide, alors il existe au moins un sommet, et l'ensemble des sommets correspond aux solutions de base réalisables.

Ainsi, bien qu'un polyèdre non vide contienne une infinité de solutions réalisables, il suffit de consulter ses sommets, en nombre fini, pour trouver l'optimum. Le nombre de sommets peut être énorme : pour 100 contraintes et 400 variables, on peut avoir autant de matrices de base que de façons de choisir 100 colonnes parmi 400, soit de l'ordre de 10^{100} sommets ! L'algorithme du simplexe va seulement générer une suite de sommets de profits croissants.

1.4.4 Algorithme du simplexe à la main

Reprenons notre PL favori en forme standard, avec ses deux contraintes d'égalité :

$$\begin{array}{lrcrcrcrcl} \text{Max } z = & x_1 & + & 2 \cdot x_2 & & & & & & \\ & x_1 & + & x_2 & + & x_3 & & & = & 6 \\ & & & x_2 & & & + & x_4 & = & 3 \\ & x_1, & & x_2, & & x_3, & & x_4 & \geq & 0 \end{array}$$

L'algorithme du simplexe a été publié par Dantzig en 1949 [Dantzig 1949]. Il construit une suite de solutions de base réalisables de profit croissant, jusqu'à ce qu'il n'y ait plus de gain possible. Géométriquement, il visite une suite de sommets adjacents du polyèdre. Le passage d'une base à l'autre s'effectue par des opérations de pivotage (*cf.* § 1.4.2). On part de la base évidente (matrice identité) formée par les variables d'écart x_3 et x_4. La réécriture des contraintes pour obtenir les expressions de x_3 et x_4 en fonction des variables hors base x_1 et x_2 est immédiate. La fonction-objectif est ajustée de la même manière, mais au début il n'y a rien à faire puisque les variables d'écart y ont un coefficient nul. On obtient :

$$x_3 = 6 - x_1 - x_2$$
$$x_4 = 3 - x_2$$
$$z = 0 + x_1 + 2x_2$$

La SBR initiale associée s'obtient en mettant les variables hors base des seconds membres à 0. On trouve $x_1 = 0$, $x_2 = 0$, $x_3 = 6$, $x_4 = 3$ et $z = 0$. Elle correspond au point O de la résolution géométrique. Pour changer de SBR, on imagine que les variables hors base sont nulles dans le système précédent. Une variable hors base, actuellement à 0, va être choisie pour entrer en base et augmenter jusqu'à annuler une variable de base. À la suite de cette opération de pivotage, la variable hors base qu'on augmente, dite *entrante*, remplace celle qui s'annule dans la solution de base, dite *sortante*. Géométriquement, on passe sur un sommet adjacent du polyèdre.

Pour déterminer la variable entrante, on examine les coefficients des variables hors base dans z, appelés *profits marginaux* ou *réduits* (*coûts marginaux* en minimisation). Ils donnent le gain obtenu en augmentant de 1 la variable associée. En fait, l'algorithme converge si on augmente une des variables de profit marginal strictement positif, mais plus rapidement en moyenne si on choisit parmi elles celle *de profit marginal maximal*, ici x_2.

Voyons maintenant comment trouver la variable sortante. D'après la première contrainte, x_2 ne peut pas augmenter au-delà de 6, sinon x_3 deviendrait négative. La seconde contrainte est encore plus limitante, car x_2 peut augmenter jusqu'à 3 seulement. Ainsi, x_4 s'annule. *La variable sortante est donc la première qui s'annule quand on augmente la variable entrante*. Les nouvelles variables de base sont donc x_3 et x_2, et on doit les écrire, ainsi que z, uniquement en fonction des nouvelles variables hors base x_1 et x_4. On obtient :

$$x_2 = 3 - x_4$$
$$x_3 = 6 - x_1 - x_2 = 6 - x_1 - (3 - x_4) = 3 - x_1 + x_4$$
$$z = 0 + x_1 + 2(3 - x_4) = 6 + x_1 - 2x_4$$

La SBR actuelle se lit en plaçant à 0 les variables hors base : $x_2 = 3$, $x_3 = 3$, $x_1 = x_2 = 0$, avec un profit $z = 6$. Elle correspond au point I du polyèdre. Pour augmenter encore le profit, on peut seulement augmenter x_1, car c'est la seule variable hors base de profit unitaire strictement positif. x_1 peut augmenter jusqu'à 3 et remplace dans la base x_3, qui s'annule. On écrit donc x_1, x_2 et z en fonction de x_3 et x_4 :

$$x_1 = 3 - x_3 + x_4$$
$$x_2 = 3 - x_4$$
$$z = 6 + (3 - x_3 + x_4) - 2x_4 = 9 - x_3 - x_4$$

À ce stade la SBR associée est $x = (3,3,0,0)$, de profit 9. Elle correspond au sommet J du polyèdre. On est à l'optimum, car toutes les variables hors base ont un profit unitaire négatif : le profit diminuerait si on augmentait l'une d'entre elles. Finalement, sur les cinq bases possibles pour ce PL, l'algorithme du simplexe n'en a consulté que trois. Pour un PL volumineux, l'économie par rapport à une énumération complète des solutions de base est énorme : des tests numériques montrent qu'en moyenne le nombre de pivotages reste proportionnel au nombre m de contraintes.

Le processus est facilement adaptable au cas d'une *minimisation*. La seule différence est la règle pour la variable entrante : il faut prendre celle ayant le plus petit coût marginal strictement négatif (le plus grand en valeur absolue), le but étant d'obtenir un coût minimal.

1.5 Algorithme du simplexe forme tableau

Dans le § 1.4.4, nous avons manipulé directement le PL en opérant par calcul matriciel sans nous en apercevoir. En effet, à toute itération et pour une base B, le PL était écrit sous la forme équivalente définie à la fin du § 1.4.1 :

$$x_B = B^{-1} \cdot b - B^{-1} \cdot N \cdot x_N = b' - B^{-1} \cdot N \cdot x_N$$
$$z = c_B \cdot B^{-1} \cdot b + \left(c_N - c_B \cdot B^{-1} \cdot N\right) \cdot x_N = z_B + \Delta_N \cdot x_N$$

Pour les contraintes, on reconnaît chaque variable de base $x_B(i)$, égale à un terme constant b'_i (sa valeur actuelle) plus une combinaison linéaire des variables hors base. L'expression de z comporte un terme constant noté z_B (valeur actuelle de la fonction-objectif) plus une combinaison linéaire dans laquelle Δ_N désigne le vecteur des profits marginaux.

La *forme tableau* de l'algorithme du simplexe facilite les calculs précédents et se prête bien à la programmation. Reprenons le programme linéaire vu pour la résolution géométrique, une fois mis sous forme standard.

$$\begin{array}{lrcrcrcrcl} \text{Max } z = & x_1 & + & 2 \cdot x_2 & & & & & & \\ & x_1 & + & x_2 & + & x_3 & & & = & 6 \\ & & & x_2 & & & + & x_4 & = & 3 \\ & x_1, & & x_2, & & x_3, & & x_4 & \geq & 0 \end{array}$$

Le tableau initial de ce PL est composé de quatre sous-tableaux. Le tableau central T est chargé avec la matrice A du PL. Le tableau-colonne *Base* donne les indices des variables de base actuelles, les variables d'écart x_3 et x_4. Le tableau-colonne b' contient les seconds membres. La ligne Δ est la fonction-objectif sous la forme $0.x_B + \Delta_N.x_N = -z_B$ (les variables de base ont un profit marginal nul). Notez que la case $-z_B$ contient bien la valeur de la fonction-objectif, mais multipliée par -1. Actuellement cette valeur est nulle.

Base	T	1	2	3	4	b'	$b'_i / T_{ie} > 0$
3	1	1	1	1	0	6	6
4	2	0	**1**	0	1	3	$3 \rightarrow s$
	Δ	1	2	0	0	0	
			↑			$-z_B$	
			e				

Effectuons la première itération, tout à fait équivalente à ce que nous avons réalisé en manipulant directement les équations. Elle construit le tableau de la prochaine base.

- La variable hors base x_e entrant en base est x_2, car elle a le coût réduit strictement positif le plus grand sur la ligne Δ. On repère sa colonne (*colonne pivot*) avec un *e*.
- La variable sortant de la base est celle s'annulant en premier quand x_e augmente. C'est celle qui minimise les b'_i / T_{ie} avec $T_{ie} > 0$ ou, comme on suppose des seconds membres positifs dans la forme standard, les $b'_i / T_{ie} > 0$. Ceci se produit à la ligne $s = 2$, dite *ligne pivot*. *Base*[*s*] donne l'indice de la variable de base correspondante, x_4.
- On entoure le *pivot* T_{se}. Pour exprimer x_2 en fonction des variables hors base, il faut écrire dans le tableau suivant la ligne du pivot, divisée par le pivot, pour obtenir un 1 à la place du pivot. Le pivot valant déjà 1, on se contente de recopier la ligne du pivot.
- On fait apparaître ensuite des 0 dans les T_{ie} des lignes $i \neq s$: on multiplie la ligne du pivot (déjà calculée dans le tableau suivant) par T_{ie} puis on la soustrait à la ligne *i*. Les lignes obtenues sont copiées dans le tableau suivant. Ceci élimine x_e des équations autres que la ligne *s*. Ici, le traitement équivaut à soustraire la ligne 2 à la ligne 1.
- On applique la même opération à la ligne Δ, y compris pour $-z_B$: la ligne du pivot est multipliée par Δ_e et soustraite à la ligne Δ pour obtenir la nouvelle ligne Δ du tableau suivant. Ici, ceci revient à multiplier la ligne 2 par 2 et à la soustraire à la ligne Δ.
- On n'oublie pas de mettre à jour l'indice de la variable de base correspondant désormais à la ligne *s* dans *Base* : 2 au lieu de 3. On obtient le tableau suivant :

Base	*T*	1	2	3	4	*b*'	$b'_i / T_{ie} > 0$
3	1	**1**	0	1	-1	3	$3 \rightarrow s$
2	2	0	1	0	1	3	–
	Δ	1	0	0	-2	-6	
		↑				$-z_B$	
		e					

Ce tableau se lit comme suit :

$$x_1 + x_3 - x_4 = 3$$
$$x_2 + x_4 = 3$$
$$x_1 - 2x_4 = -6$$

On retrouve le système obtenu par l'algorithme du § 1.4.4, sauf que les variables de base ne sont plus dans les membres de gauche : c'est le tableau *Base* qui donne leurs colonnes. La solution de base actuelle est donc $x_B = (x_2, x_3) = (3, 3)$, $x_N = (x_1, x_4) = (0, 0)$. La matrice identité des variables de base s'est déplacée dans *T* aux colonnes 3 et 2. Le profit actuel est $z_B = 6$. Le vecteur des profits unitaires des variables hors base est $\Delta_N = (1, -2)$.

L'algorithme n'est pas terminé car on peut encore augmenter le profit en augmentant x_1. On n'a pas le choix car c'est la seule variable hors base de profit unitaire > 0. Le minimum des $b'i / T_{ie} > 0$ est obtenu sur la ligne 1. Le tiret sur la ligne 2 signifie que x_1 peut augmenter autant qu'on veut sans affecter la variable de base x_2. On trouve $e = 1$, $s = 1$, *Base*[*s*] = 1 (x_1 entre en base et x_3 en sort). Le pivotage sur $T_{se} = T_{11}$ revient à soustraire la ligne du pivot à la ligne Δ pour faire apparaître un 0 dans Δ(1). On obtient le tableau suivant.

Base	T	1	2	3	4	b'
1	1	1	0	1	-1	3
2	2	0	1	0	1	3
	Δ	0	0	-1	-1	-9

$-z_B$

C'est la fin de l'algorithme car les profits marginaux sont négatifs ou nuls. On retrouve évidemment la solution optimale de la méthode géométrique et de l'algorithme non matriciel : $x_1^* = 3$, $x_2^* = 3$, les autres variables à 0, et un coût total $z^* = 9$.

1.6 Cas spéciaux pour l'algorithme du simplexe

1.6.1 Optimum non borné

Considérons le programme linéaire suivant sous forme canonique :

$$\begin{aligned} \text{Max } z = \quad & x_1 + 2 \cdot x_2 \\ -2 \cdot x_1 + \quad & x_2 \le 2 \\ -x_1 + \quad & 2 \cdot x_2 \le 5 \\ x_1 - \quad & 4 \cdot x_2 \le 4 \\ x_1, \quad & x_2 \ge 0 \end{aligned}$$

La résolution géométrique montre qu'il n'a pas d'optimum borné (figure 1.2).

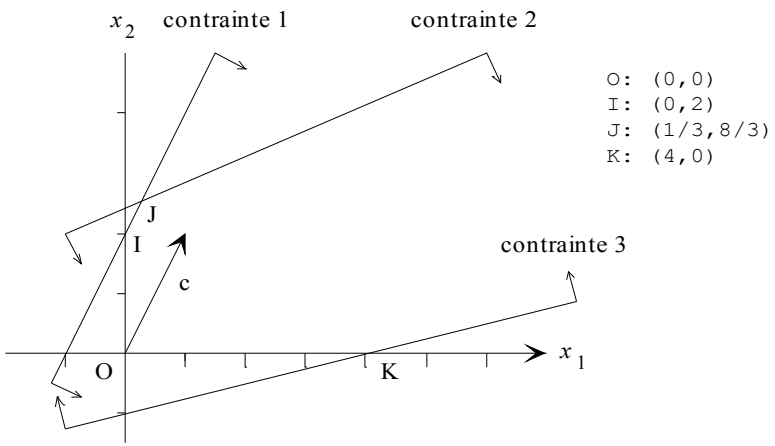

Figure 1.2 – Résolution géométrique d'un cas sans optimum borné

Résolvons-le maintenant par l'algorithme du simplexe. On passe à la forme standard en introduisant trois variables d'écart x_3, x_4, et x_5. Le premier tableau correspond au sommet O du polyèdre de la forme canonique.

Base	T	1	2	3	4	5	b'	$b'_i / T_{ie} > 0$
3	1	-2	**1**	1	0	0	2	$2 \rightarrow s$
4	2	-1	2	0	1	0	5	2.5
5	3	1	-4	0	0	1	4	–
	Δ	1	2	0	0	0	0	
			↑				$-z_B$	
			e					

Le deuxième tableau correspond au point I :

Base	T	1	2	3	4	5	b'	$b'_i / T_{ie} > 0$
2	1	-2	1	1	0	0	2	–
4	2	**3**	0	-2	1	0	1	$1/3 \rightarrow s$
5	3	-7	0	4	0	1	12	–
	Δ	5	0	-2	0	0	-4	
		↑					$-z_B$	
		e						

Le troisième et dernier tableau correspond au point J. On peut ensuite augmenter le coût en faisant entrer x_3 en base. *L'optimum n'est pas borné*, car on peut augmenter x_3 sans annuler une variable hors base : il n'y a pas de $b'_i / T_{ie} > 0$. Cette situation est parfaitement détectée par les logiciels commerciaux, avec un message du genre *Unbounded optimum*.

Base	T	1	2	3	4	5	b'	$b'_i / T_{ie} > 0$
2	1	0	1	-1/3	2/3	0	8/3	–
1	2	1	0	-2/3	1/3	0	1/3	–
5	3	0	0	-2/3	7/3	1	43/3	–
	Δ	0	0	4/3	-5/3	0	-17/3	$-z_B$

1.6.2 Absence de base initiale évidente

Le problème

Soit le programme linéaire suivant sous forme canonique :

$$
\begin{aligned}
\text{Max } z = \quad & x_1 + 2 \cdot x_2 \\
& x_1 + x_2 \leq 6 \\
& x_2 \leq 3 \\
& x_1 + x_2 \geq 1 \\
& x_1 \;, \; x_2 \geq 0
\end{aligned}
$$

Pour la forme standard, il faut *soustraire* une variable d'écart à la troisième contrainte. En effet, une variable d'écart doit être non négative, comme toute autre variable :

$$
\begin{array}{lrcrcrcrcrcl}
\text{Max } z = & x_1 & + & 2 \cdot x_2 & & & & & & & & \\
 & x_1 & + & x_2 & + & x_3 & & & & & = & 6 \\
 & & & x_2 & & & + & x_4 & & & = & 3 \\
 & x_1 & + & x_2 & & & & & - & x_5 & = & 1 \\
 & x_1 & , & x_2 & , & x_3 & , & x_4 & , & x_5 & \geq & 0
\end{array}
$$

On n'a plus la matrice identité habituelle, qui nous fournissait une base initiale évidente. Ceci est confirmé par l'interprétation géométrique (forme canonique) : l'optimum est atteint en J = (3,3), mais l'origine O n'est plus réalisable (figure 1.3).

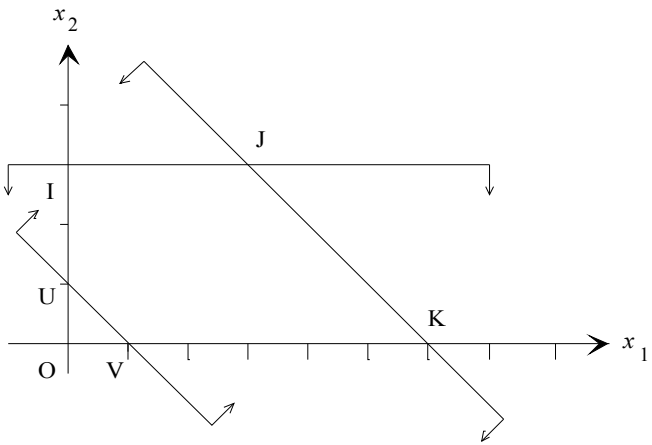

Figure 1.3 – Un cas où le point O n'est pas solution de base

Pour démarrer l'algorithme du simplexe, on pourrait trouver par tâtonnement une sous-matrice *B* inversible, mais l'effort de calcul peut être énorme pour un grand PL. Les logiciels utilisent en pratique deux méthodes basées sur l'emploi de variables artificielles : la méthode des deux phases et la méthode du grand *M*. L'idée est la suivante : on fait apparaître une matrice identité en ajoutant aux contraintes qui en ont besoin une *variable artificielle* (VA), avec un coefficient 1.

Les variables x_1 à x_5 sont dites *légitimes* : elles sont nécessaires à l'obtention de la forme standard. Les variables d'écart x_3 et x_4 nous donnent deux colonnes de matrice identité, les contraintes correspondantes (première et deuxième) n'ont donc pas besoin de VA. On ajoute x_6 à la troisième contrainte pour compléter la base. Une telle variable est réellement artificielle et n'a aucun sens économique : à l'optimum, elle doit être nulle (hors base) sinon la contrainte n'est pas vérifiée avec égalité ! Les deux méthodes des 2 phases et du grand M ont précisément pour but de s'en débarrasser.

$$
\begin{array}{lrcrcrcrcrcrcl}
\text{Max } z = & x_1 & + & 2 \cdot x_2 & & & & & & & & & & \\
& x_1 & + & x_2 & + & x_3 & & & & & & & = & 6 \\
& & & x_2 & & & + & x_4 & & & & & = & 3 \\
& x_1 & + & x_2 & & & & & - & x_5 & + & x_6 & = & 1 \\
& x_1 & , & x_2 & , & x_3 & , & x_4 & , & x_5 & , & x_6 & \geq & 0
\end{array}
$$

Méthode des deux phases

Dans la *phase 1*, on ignore la vraie fonction-objectif et on cherche à minimiser la somme des p variables artificielles qu'on a dû introduire. On résout donc le PL auxiliaire suivant, où x_A désigne le vecteur des p VA, et e un vecteur avec p composantes à 1. Notez qu'on est en minimisation, le choix de la variable entrante dans l'algorithme du simplexe est modifié : c'est celle de plus petit coût marginal négatif.

$$\text{Min } z = e \cdot x_A = \sum_{j=1}^{p} x_A(j)$$

$$A \cdot x + x_A = b$$

$$x, x_A \geq 0$$

Si tout se passe bien, on parvient à annuler la somme des VA et à trouver une base sans VA. On obtient donc une solution réalisable pour le problème de départ. La *phase 2* consiste à éliminer du tableau les colonnes des VA, devenues inutiles, à remplacer la ligne Δ par la vraie fonction-objectif, et à continuer le simplexe sur le tableau obtenu. C'est en phase 2 qu'on peut détecter si le PL d'origine n'a pas d'optimum borné. En réalité, d'autres cas spéciaux peuvent se produire en fin de phase 1 :

- Si $x_A \neq 0$ (fonction-objectif non nulle), le PL d'origine n'a pas de solution réalisable.
- Si $x_A = 0$ avec des VA dans la base, on peut montrer que les contraintes associées à ces variables sont redondantes. On peut les éliminer et attaquer la phase 2.

À notre connaissance, cette technique de variables artificielles est la seule qui permette de détecter assez simplement les PL sans solution et les contraintes redondantes. Elle peut aussi servir à savoir si un système d'inégalités linéaires a ou non des solutions, en lui ajoutant des variables d'écart et des variables artificielles comme pour un programme linéaire.

Appliquons la méthode des deux phases à l'exemple. La base est formée des colonnes 3, 4 et 6. Mais dans l'algorithme du simplexe, les variables de base doivent avoir des coûts réduits nuls, ce qui n'est pas le cas pour x_6. Transformons Δ en une ligne correcte Δ', en lui soustrayant la ligne 3 du tableau. Ceci fait apparaître le véritable coût de la solution de base actuelle : 1, c'est-à-dire qu'elle contient une VA. Étant en minimisation, il faut faire entrer en base la variable de coût réduit négatif minimal. On a le choix entre $e = 1$ et $e = 2$. Par convention, nous prenons le plus petit indice.

Base	T	1	2	3	4	5	6	b'	$b'_i / T_{ie} > 0$
3	1	1	1	1	0	0	0	6	6
4	2	0	1	0	1	0	0	3	–
6	3	**1**	1	0	0	-1	1	1	$1 \rightarrow s$
	Δ	0	0	0	0	0	1	0	
	Δ'	-1	-1	0	0	1	0	-1	
		↑						$-z_B$	
		e							

Après pivotage sur T_{31}, on obtient le tableau suivant :

Base	T	1	2	3	4	5	6	b'	
3	1	0	0	1	0	1	-1	5	
4	2	0	1	0	1	0	0	3	
1	3	1	1	0	0	-1	1	1	
	Δ'	0	0	0	0	0	1	0	$-z_B$

La phase 1 se termine ici avec succès, puisque tous les coûts marginaux sont positifs ou nuls (on est en minimisation). La solution réalisable trouvée correspond au point V du polyèdre de la forme canonique.

Base	T	1	2	3	4	5	b'	$b'_i / T_{ie} > 0$
3	1	0	0	1	0	**1**	5	$5 \rightarrow s$
4	2	0	1	0	1	0	3	–
1	3	1	1	0	0	-1	1	–
	Δ'	1	2	0	0	0	0	
	Δ''	0	1	0	0	1	-1	
						↑	$-z_B$	
						e		

On peut mener la phase 2 car la VA x_6 a été éjectée de la base. Pour débuter, les colonnes des VA (ici une seule) sont supprimées du tableau de fin de phase 1, ce qui donne le tableau ci-dessus. La fonction-objectif d'origine $x_1 + 2.x_2$ est réintroduite. Elle doit être exprimée en fonction des variables hors base, comme les autres lignes. Pour cela, on soustrait la ligne 3 à la ligne Δ', ce qui donne une ligne correcte Δ''.

Après pivotage sur T_{15} on parvient au point K du polyèdre de la forme canonique (tableau suivant). Notez qu'on a un peu "triché" en faisant entrer en base x_5 au lieu de x_2, pour gagner une itération : on a ainsi emprunté la suite de sommets V, K, J au lieu de V, U, I, J !

Base	T	1	2	3	4	5	b'	$b'_i / T_{ie} > 0$
5	1	0	0	1	0	1	5	–
4	2	0	**1**	0	1	0	3	$3 \rightarrow s$
1	3	1	1	1	0	0	6	6
	Δ"	0	1	-1	0	0	-6	
			↑				$-z_B$	
			e					

Un dernier pivotage nous amène à l'optimum, au point J = (x_1, x_2) = (3,3), de coût 9.

Base	T	1	2	3	4	5	b'	
5	1	0	0	1	0	1	5	
2	2	0	1	0	1	0	3	
1	3	1	0	1	-1	0	3	
	Δ"	0	0	-1	-1	0	-9	$-z_B$

La méthode du grand M

Cette méthode procède en une phase. Elle ajoute les VA à la fonction-objectif du PL d'origine, mais pénalisées par un coût -*M*, *M* étant un grand nombre positif (10^6 par exemple). Avec les mêmes notations que pour les deux phases, on résout en fait le PL :

$$\text{Max } z = c \cdot x - M \cdot e \cdot x_A = \sum_{j=1}^{n} c_j x_j - M \sum_{j=1}^{p} x_A(j)$$

$$A \cdot x + x_A = b$$

$$x, x_A \geq 0$$

Si tout se passe bien, le simplexe se débarrasse des VA dès les premières itérations et ne les fait plus entrer en base à cause de leur coût énorme. On peut ignorer les colonnes des VA dès qu'elles ne sont plus en base. Les cas spéciaux suivants peuvent aussi se produire :

- Le PL modifié n'a pas d'optimum borné et $x_A = 0$: le PL d'origine est aussi non borné.
- Le PL modifié n'a pas d'optimum borné et $x_A \neq 0$: le PL d'origine n'a pas de solution.
- Le PL modifié a un optimum borné, mais $x_A \neq 0$: le PL d'origine n'a pas de solution.

En général, la méthode du grand *M* donne moins d'itérations que celle des deux phases. En revanche, sur ordinateur, la coexistence de *M* avec des petits nombres engendre souvent des problèmes de précision, ce qui fait que les logiciels commerciaux utilisent plutôt la méthode des deux phases.

Voyons la méthode sur l'exemple. Les tableaux *T*, *Base* et *b*' sont identiques à ceux du début de la phase 1 dans la méthode des deux phases. La ligne Δ est chargée avec les coûts des variables légitimes et un coût -*M* pour chaque VA. On rend nuls les coûts marginaux des variables de base en ajoutant la ligne 3, multipliée par *M*, à la ligne Δ, ce qui donne la ligne Δ'.

Base	T	1	2	3	4	5	6	b'	$b'_i / T_{ie} > 0$
3	1	1	1	1	0	0	0	6	6
4	2	0	1	0	1	0	0	3	3
6	3	1	[1]	0	0	-1	1	1	$1 \rightarrow s$
	Δ	1	2	0	0	0	$-M$	0	
	Δ'	M+1	M+2	0	0	$-M$	0	M	
			↑					$-z_B$	
			e						

Une itération mène à une solution réalisable (point U du polyèdre de la forme canonique).

Base	T	1	2	3	4	5	6	b'	$b'_i / T_{ie} > 0$
3	1	0	0	1	0	1	-1	5	5
4	2	-1	0	0	1	[1]	-1	2	$2 \rightarrow s$
2	3	1	1	0	0	-1	1	1	–
	Δ'	-1	0	0	0	2	$-M$-2	-2	
						↑		$-z_B$	
						e			

L'algorithme du simplexe se poursuit en ignorant la colonne 6 :

Base	T	1	2	3	4	5	b'	$b'_i / T_{ie} > 0$
3	1	[1]	0	1	-1	0	3	$3 \rightarrow s$
5	2	-1	0	0	1	1	2	–
2	3	0	1	0	1	0	3	–
	Δ'	1	0	0	-2	0	-6	
		↑					$-z_B$	
		e						

Base	T	1	2	3	4	5	b'	$b'_i / T_{ie} > 0$
1	1	[1]	0	1	-1	0	3	$3 \rightarrow s$
5	2	0	0	1	0	1	5	–
2	3	0	1	0	1	0	3	–
	Δ'	0	0	-1	-1	0	-9	$-z_B$

On retrouve l'optimum calculé par la méthode des deux phases !

1.7 Dualité

1.7.1 Définition et exemple

À tout programme linéaire, appelé par convention *PL primal*, on peut associer un autre PL appelé son *dual*. Voici un PL primal sous forme générale, c'est-à-dire non mis sous forme canonique ou standard, et son dual. On a séparé dans ce primal les variables positives de celles non contraintes en signe, d'une part, et les contraintes d'égalité des contraintes d'inégalité, d'autre part. Ceci partitionne la matrice A en quatre sous-matrices.

Primal	Dual
Max $z = c_1^T x_1 + c_2^T x_2$	Min $w = b_1 y_1 + b_2 y_2$
$A_{1,1} x_1 + A_{1,2} x_2 \le b_1$	$y_1 \ge 0$
$A_{2,1} x_1 + A_{2,2} x_2 = b_2$	y_2 de signe quelconque
$x_1 \ge 0$	$A_{1,1}^T y_1 + A_{2,1}^T y_2 \ge c_1$
x_2 de signe quelconque	$A_{1,2}^T y_1 + A_{2,2}^T y_2 = c_2$

Voici un exemple plus simple, avec un PL primal sous forme canonique :

Max $z = 4x_1 + 7x_2$	Min $w = 6y_1 + 8y_2$
$3x_1 + 5x_2 \le 6$	$3y_1 + y_2 \ge 4$
$x_1 + 2x_2 \le 8$	$5y_1 + 2y_2 \ge 7$
$x_1, x_2 \ge 0$	$y_1, y_2 \ge 0$

Notez comment on est passé mécaniquement au dual pour ce dernier PL canonique :

- Une variable duale $y_i \ge 0$ est associée à chaque contrainte i du primal.
- Dans l'objectif, c est remplacé par b et le sens de l'optimisation est inversé.
- Dans les contraintes, le second membre b est remplacé par c, et $\le$ est remplacé par $\ge$.
- La matrice A est transposée.

La notion de primal et de dual est relative, car la transformation peut avoir lieu dans les deux sens : le dual du dual est le primal.

1.7.2 Interprétation économique

Reprenons l'interprétation économique d'un PL primal en forme canonique, vue au § 1.2.4. Supposons qu'une autre firme B veuille racheter les ressources de l'entreprise A à moindre coût, avec un prix unitaire de y_i pour la ressource i. A n'acceptera de vendre que si, pour chaque activité j, le prix de vente des ressources nécessaires à l'exercice de cette activité avec une intensité 1 est au moins égal au profit unitaire c_j auquel elle renoncera. Les prix y_i optimaux sont précisément solutions du dual suivant, dans lequel la somme dans chaque contrainte j est la somme des prix de vente des ressources pour l'activité j :

$$\text{Min} \sum_{i=1}^{m} y_i b_i$$

$$\forall j = 1 \ldots n : \sum_{i=1}^{m} a_{ij} y_i \geq c_j$$

$$\forall i = 1 \ldots m : y_i \geq 0$$

La dualité est également utilisable par la firme qui possède les biens. Dans le primal, la valeur optimale d'une variable duale y_i correspond à l'amélioration du profit quand on relâche la contrainte i d'une unité (c'est-à-dire quand on augmente de 1 la capacité b_i de la ressource i pour une contrainte ≤, ou quand on la diminue pour ≥). Pour ces raisons, y_i est souvent appelé *coût réduit*, ou *marginal*, de la contrainte i.

Les économistes sont friands de cette interprétation. Les variables duales sont des sortes de *coûts d'opportunité* : augmenter la capacité d'une ressource est une opportunité d'augmenter le profit, et la variable duale est le coût de cette opportunité non réalisée. Les coûts réduits des contraintes sont très utilisés par les décideurs. Heureusement, il n'est pas nécessaire de construire le dual et de le résoudre : les logiciels commerciaux fournissent pour un PL primal un listing de résultats avec le coût réduit de chaque contrainte.

1.7.3 Propriétés du dual

Une présentation complète de la dualité dépasse le cadre de ce livre, aussi citerons-nous seulement quelques propriétés intéressantes. Les deux premières ont déjà été mentionnées dans les deux paragraphes précédents.

- Le dual du dual est le primal.
- La solution du dual est égale aux coûts réduits du primal, et *vice versa*.
- Étant donné un primal et son dual, soit les deux problèmes ont un optimum fini, soit ils sont tous deux sans solution, soit l'un est sans solution et l'autre sans optimum fini.
- Pour un primal en maximisation, le coût d'une solution réalisable est inférieur ou égal au coût d'une solution réalisable du dual. Ces coûts sont égaux si et seulement si les deux solutions sont optimales.
- Pour toute contrainte ≤ du primal, soit la contrainte est *saturée* (vérifiée avec égalité), soit la variable correspondante du dual est nulle. Pour toute contrainte ≥ du dual, soit la contrainte est saturée, soit la variable associée du primal est nulle.

La dernière propriété, connue sous le nom de *théorème des écarts complémentaires* ou de *relations d'exclusion*, est très utilisée. Intuitivement, dans l'interprétation économique, elle signifie qu'une contrainte non saturée (non vérifiée avec égalité à l'optimum) a un coût réduit nul : en effet, la ressource n'étant pas utilisée complètement, augmenter sa capacité ne rapporte rien. Le théorème peut s'écrire matriciellement pour les deux formes classiques d'un PL. Pour un primal canonique et deux solutions optimales x^* et y^* du primal et du dual, on a $y^*.(b - A.x^*) = 0$ et $(y^*.A - c).x^* = 0$. Pour un primal standard, on a toujours $y^*.(b - A.x^*) = 0$ grâce aux égalités, la propriété se réduit donc à $(y^*.A - c).x^* = 0$.