Recherche Opérationnelle

Pierre L. Douillet

30 décembre 2016

Le module OPREA a été enseigné durant les années 2001-2010
tant en promotion A3 qu'en promotion E3.

La Recherche Opérationnelle (Operations Research) est apparue comme service spécialisé au sein des structures de Commandement lors de la seconde guerre mondiale. Avec pour objectif l'aide à la décision. Un proverbe dit: "décider dans le doute, agir dans la foi". Une structure RO permet de développer les questionnements, y compris à posteriori, tout en évitant de paralyser la prise de décision.

Enseigner cela en formation initiale n'est pas évident. Les techniques d'optimisation viennent souvent masquer les difficultés de la modélisation. Et il est difficile d'imaginer les difficultés de la collecte d'information lorsque l'on ne les a pas expérimentées dans la réalité d'un système de grande taille.

En outre, la réalité d'un enseignement est pilotée par les évaluations. Et dans une évaluation "sur table" de 2 heures, on a tout juste le temps d'optimiser le monde en inversant une matrice 2x2. Ou peut-être 4x4 si l'on utilise un ordinateur. Pour la promotion A3, on pouvait considérer qu'une présence à mi-temps en entreprise constituait une antidote suffisante.

Pour la promotion E3, l'évaluation s'est faite sous forme de mini-projets dont nous rendons compte par ailleurs. Cette forme est très exigeante pour les étudiants... et plus encore pour l'enseignant qui doit "laisser faire, de façon directive".

Je ne me serais probablement pas lancé dans une telle aventure si j'en avais connu par avance les difficultés... sauf à en connaître par avance les bénéfices ! Un autre proverbe est "en théorie, théorie et pratique coïncident ; tandis qu'en pratique...".

En raison des contraintes horaires, le projecteur a été mis sur les techniques d'optimisation linéaire et la méthode du simplexe. C'est un petit sous-ensemble d'un grand corpus, mais comment entrer sans une porte d'entrée ?

Table des matières

Introduction

L'objectif de la Recherche Opérationnelle est l'aide à la décision. Les deux termes sont essentiels. En premier lieu, ce que l'on cherche à garantir, ce n'est pas la valeur du résultat, mais l'optimalité de la décision. En effet, nous ne sommes pas maîtres des aléas qui constituent le contexte de notre action. Ce que maîtrisons, ce sont nos choix... et il est souhaitable de ne pas avoir à les regretter par la suite.

En second lieu, la Recherche Opérationnelle n'a pas pour but de prendre des décisions, mais de clarifier le contexte dans lequel ces décisions sont prises. Un choix réel ne peut se faire qu'en connaissance de cause... mais on ne peut attendre de tout connaître avant d'agir.

Pour citer Chinneck (2007) :

> Students need to have a solid intuitive understanding of how and why optimization methods work. This enables them to recognize when things have gone wrong, and to diagnose the source of the difficulty and take appropriate action. It also permits students to see how methods can be combined or modified to solve non-standard problems.
>
> Explanation and diagrams are more effective in transmitting a real understanding than a total reliance on mathematics.
>
> There should be some exposure to how things are done in practice, which can be significantly different than how they are usually done in textbooks.
>
> The goal is for students to be able to recognize when problems that they encounter in their jobs or in thesis work can be successfully tackled by optimization methods. Students should be able to abstract a useful formulation of a problem from a messy real world description.

Le présent document est une introduction à l'une des branches de la recherche opérationnelle : la "programmation linéaire".

Chapitre 1

Optimisation non linéaire

1.1 Théorèmes généraux

Dans tout ce qui suit, on s'intéresse à des ensembles métriques, c'est à dire des ensembles sur lesquels est définie une distance, en analogie avec la distance usuelle de l'espace géométrique ordinaire.

1.1.1 Propriétés de l'ensemble des réels

Definition 1.1.1. On dit que deux suites (a_n) et (b_n) sont adjacentes lorsque $\forall n : a_n \leq a_{n+1} \leq b_{n+1} \leq b_n$ et que, de plus, $b_n - a_n \to 0$.

Axiom 1.1.2. *Lorsque deux $\mathbb{R}$ suites sont adjacentes, chacune d'elles admet une limite (et alors ces limites sont égales).*

Definition 1.1.3. On appelle borne supérieure d'un sous ensemble A de $\mathbb{R}$ le plus petit majorant de A (s'il en existe).

Theorem 1.1.4. *Toute partie non vide majorée de $\mathbb{R}$ admet une borne supérieure.*

Preuve. Soit A cette partie, $a_0 \in A$ et $b_0 \geq A$. Par récurrence sur n, on pose $m_n = (a_n + b_n)/2$. Si $m_n \geq A$, on pose $a_{n+1} = a_n$ et $b_{n+1} = m_n$. Sinon, il existe $a_{n+1} \in A$ avec $m_n \leq a_{n+1}$ et on pose $b_{n+1} = b_n$. Il est immédiat que les deux suites (a_n) et (b_n) sont adjacentes. Leur limite commune est à la fois un majorant de A en tant que limite des (b_n) et est plut petit que tout majorant de A en tant que limite des (a_n). □

1.1.2 Un peu de topologie

Definition 1.1.5. On dit que $x \in E$ est intérieur au sous-ensemble $A \subset E$ lorsqu'il existe une distance de sécurité $\delta > 0$ telle que $\forall y \in E : d(x, y) \leq \delta \implies y \in A$. On note cela par $x \in \dot{A}$.

Definition 1.1.6. On dit que $x \in E$ est extérieur au sous-ensemble $A \subset E$ lorsqu'il est intérieur au complémentaire de A. Un point "non-extérieur" est dit adhérent.

Definition 1.1.7. On dit que $x \in E$ est un point frontière du sous-ensemble $A \subset E$ lorsqu'il n'est ni intérieur ni extérieur.

Definition 1.1.8. On dit qu'un ensemble est ouvert lorsqu'il ne contient aucun de ses points frontière. On dit qu'un ensemble est fermé lorsqu'il contient tous ses points frontière.

Definition 1.1.9. On dit qu'un sous ensemble K d'un espace métrique E est compact lorsque de toute K suite on peut extraire une sous-suite convergente (dans K).

1.1.3 Recherche d'extrema

Theorem 1.1.10. *L'image d'un compact par une fonction continue est un compact. En particulier, pour K compact non vide, une fonction continue $K \hookrightarrow \mathbb{R}$ atteint ses bornes.*

Preuve. Soit (y_n) une $f(K)$-suite. Par définition de l'image, il existe une K-suite (x_n) telle que $y_n = f(x_n)$. Par définition d'un compact, il existe une fonction d'extraction φ et un $\lambda \in K$ tels que $\left(x_{\varphi(n)}\right) \to \lambda$. Par définition de la continuité $\left(f\left(x_{\varphi(n)}\right)\right) = \left(y_{\varphi(n)}\right) \to f(\lambda) \in K$.

On conclut en remarquant qu'un compact réel non vide contient sa borne supérieure (élément maximal atteint). □

Theorem 1.1.11. *Lorsqu'une fonction dérivable $f : A \hookrightarrow R$ atteint un extremum en un point intérieur de A, le gradient est nul en ce point.*

Preuve. Soit ω ce point, δ une distance de sécirité et x un vecteur unitaire quelconque. Pour $|\lambda| < \delta$, on a :

$$f(\omega + \lambda x) = f(\omega) + \lambda \overrightarrow{grad} f\, x + \mathrm{o}(\lambda)$$

Si on suppose $\overrightarrow{grad} f\, x$ non nul, il est possible de choisir $\lambda > 0$ assez petit pour que le signe de $f(\omega + \lambda x) - f(\omega)$ et de $f(\omega - \lambda x) - f(\omega)$ soit celui du terme en λ. Malheureusement, ces deux signes seraient différents. On voit donc que $\overrightarrow{grad} f$ est orthogonal à tout vecteur x donné d'avance. □

Remark 1.1.12. Une recherche d'extrema se décompose donc, le plus souvent, en deux étapes : une recherche des points intérieurs par annulation du gradient et une recherche des points frontières par une technique ou une autre.

Chapitre 2

Programmation linéaire

2.1 Application linéaire tangente

Commençons par rappeler que la notion de fonction dérivée est issue de la notion de linéarisation, c'est à dire d'approximation par une fonction linéaire (Fig. 2.1).

1. La dérivée d'une fonction (dérivable) est la pente de la tangente à la courbe. En désignant par $\mathrm{d}x$ et $\mathrm{d}y$ les variations de coordonnées le long de la tangente, la relation

$$\frac{\mathrm{d}y}{\mathrm{d}x} = p$$

ne dépend pas du fait que $\mathrm{d}y$ soit petit ou non : une droite "va en ligne droite" sur toute sa longueur !

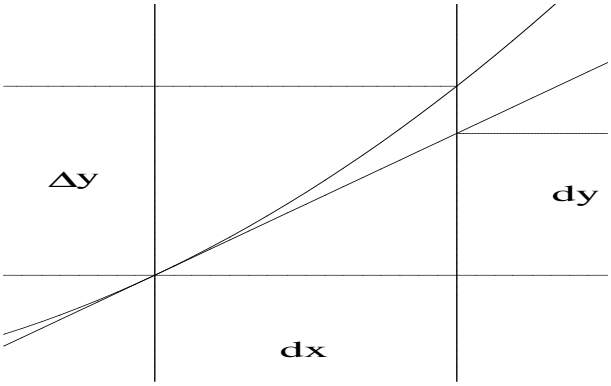

Fig. 2.1: Distinguer Delta y et d y !

2. Par contre la relation $\frac{\Delta y}{\Delta x} \approx p$ dépend, elle, de ce que Δx est supposé petit. Plus précisément, on a

$$\Delta y = p\,\Delta x + \mathrm{o}\,(\Delta x)$$

et cette formule n'est significative que dans un certain domaine de validité. En dehors de ce domaine, il faut changer de tangente.

3. Tout cela se généralise à des fonctions vectorielles d'une variable vectorielle. Dans ce cas la "pente" n'est plus un simple nombre, mais la matrice d'une application linéaire.

2.2 Un exemple caricatural

Voici un exemple d'optimisation sous contraintes. Cet exemple est deux fois caricatural. En premier lieu, il constitue un modèle bien trop simple pour pouvoir capturer la réalité d'un processus industriel réel. En second lieu, il est tellement simple que l'on peut en trouver la

solution d'un seul coup, uniquement en traçant la figure correspondante. L'objectif poursuivi est simplement de décrire une problématique.

1. Soit à optimiser la production d'un atelier fabriquant deux sortes de produits. On se demande comment choisir les quantités x_1 et x_2 fabriquées de façon à optimiser le bénéfice $z = 4\,x_1 + 12\,x_2$. Bien entendu, la réponse n'est pas $+\infty$, $+\infty$, car il y a inévitablement des contraintes sur les valeurs des x_j.
2. En premier lieu, il est clair que $x_1 \geq 0$ et que $x_2 \geq 0$.
3. Ensuite, on peut imaginer des contraintes liées à la saturation du marché, conduisant par exemple à $x_1 \leq 1000$ et $x_2 \leq 500$, et une contrainte de fabrication $3\,x_1 + 6\,x_2 \leq 4200$.
4. On voit tout de suite que (0, 0) est un point admissible (vérifiant toutes les contraintes). La question est de faire un meilleur bénéfice que 0 !
5. Chacune des contraintes a pour effet de séparer le plan en deux régions : un demi-plan ouvert où la contrainte n'est pas satisfaite et un demi-plan fermé (i.e. frontière comprise) où la contrainte est satisfaite. L'ensemble des points admissibles (i.e. vérifiant toutes les contraintes) est donc une surface polygonale convexe.
6. La Fig. 2.2 représente en grisé les régions du plan où l'une au moins des contraintes n'est pas satisfaite. L'ensemble des points admissibles apparaît en blanc sur la figure.

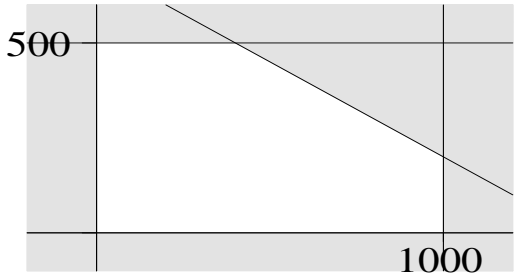

Fig. 2.2: Polygone admissible.

7. Les deux droites de la Fig. 2.3 correspondent à $z = 5000$ et $z = 7600$. Il est clair que toutes les droites $z = Cte$ sont parallèles entre elles, et que z augmentant, ces droites finissent par ne plus couper le polygone des points admissibles : le point cherché est donc le sommet (400, 500).

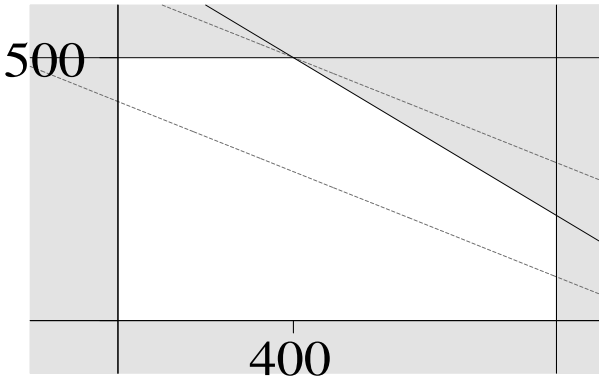

Fig. 2.3: Point optimal.

2.3 Quelques remarques

Definition 2.3.1. Un système facile est un système tel que l'origine des coordonnées soit un point de fonctionnement du système (c.à.d. vérifie l'ensemble des contraintes). En pareil cas, le problème consiste à trouver le meilleur point de fonctionnement.

Remark 2.3.2. Lorsqu'un problème "n'est pas facile", il faut commencer par trouver un point de bon fonctionnement, ou bien démontrer qu'il ne peut exister de points de bon fonctionnement.

Remark 2.3.3. Contrainte égalité. Une égalité $f(M) = Cte$ ne peut pas être utilisée directement pour réduire le nombre de variables, car il faut continuer à exprimer les contraintes sur chaque variable.

Remark 2.3.4. Contrainte inégalité. Une telle contrainte peut être transformée en une égalité par introduction d'une nouvelle variable, en remplaçant $f(M) \leq Cte$ par $\{f(M) + y = Cte\,;\, 0 \leq y\}$.

2.4 Ensembles convexes et "programmes linéaires"

Definition 2.4.1. Un "programme linéaire" est un modèle dans lequel à la fois les contraintes et la fonction objectif ont été modélisés par des fonctions du premier degré.

Definition 2.4.2. On dit qu'une partie A de $\mathbb{R}^n$ est convexe lorsque $x,\, y \in A$ implique $[a,\, b] \subset A$.

Proposition 2.4.3. *L'intersection d'un nombre quelconque de convexes est encore convexe (et éventuellement vide...).*

Definition 2.4.4. L'enveloppe convexe d'un ensemble A est le plus petit convexe contenant A. Cette enveloppe est l'intersection de tous les convexes contenant A.

Theorem 2.4.5. *Le domaine de validation des contraintes d'un programme linéaire est un ensemble convexe à faces hyperplanes (polytope), et l'extremum de la fonction linéaire à optimiser est atteint en un ou plusieurs sommets du polytope.*

Proposition 2.4.6. *Le maximum est atteint par exactement tous les points appartenant à l'enveloppe convexe des sommets optimaux (hyper-arête du polytope).*

Proposition 2.4.7. *L'hyperplan $z = max$ est un hyperplan d'appui du polytope, c'est à dire qu'ils ont au moins un point en commun, et que tous leurs points communs sont des points frontière du polytope.*

Chapitre 3

Algorithme du simplexe

L'objectif de ce chapitre n'est pas l'efficacité du calcul (il y a d'excellents logiciels pour cela) mais la compréhension des calculs entrepris.

3.1 Un système facile en dimension 2

Reprenons l'exemple de la Section 2.2.

1. Nous récrivons les données du problème en introduisant trois nouvelles variables, déterminées par le système d'équations :

$$(S) \quad \left\{ \begin{array}{rcl} x_1 + x_3 & = & 1000 \\ x_2 + x_4 & = & 500 \\ 3\,x_1 + 6\,x_2 + x_5 & = & 4200 \end{array} \right.$$

2. Les contraintes s'écrivent maintenant $0 \leq x_i$ pour $1 \leq i \leq 5$: chaque x_i sert à contrôler que nous sommes dans le bon demi-plan relatif à la contrainte correspondante. Le problème consiste à partir du point $x_1 = x_2 = 0$, qui est évidemment un point de bon fonctionnement, et à maximiser la fonction de bénéfice $z = 4\,x_1 + 12\,x_2$.
3. Du point de vue du sommet $x_1 = x_2 = 0$, les variables principales sont x_1 et x_2, et le système s'écrit préférentiellement :

$$\left. \begin{array}{rcl} x_3 & = & 1000 - x_1 \\ x_4 & = & 500 - x_2 \\ x_5 & = & 4200 - 3\,x_1 - 6\,x_2 \end{array} \right\} \quad z = 4\,x_1 + 12\,x_2$$

   Nous avons le choix de modifier x_1 ou x_2 puisque les deux coefficients $\frac{dz}{dx_i}$ sont positifs. Nous choisissons de modifier x_1, et donc de garder $x_2 = 0$. Les contraintes de positivité imposent $x_1 \leq 1000$ et $x_1 \leq 1400$. Le meilleur mouvement est donc $x_1 = 1000$, qui conduit au sommet :

$$x_2 = x_3 = 0 \quad ; \quad x_1 = 1000,\ x_4 = 500,\ x_5 = 1200$$

4. Du point de vue de ce nouveau sommet, le système se récrit

$$\left. \begin{array}{rcl} x_1 & = & 1000 - x_3 \\ x_4 & = & 500 - x_2 \\ x_5 & = & 1200 - 6\,x_2 + 3\,x_3 \end{array} \right\} \quad z = 4000 + 12\,x_2 - 4\,x_3$$

   Vu les coefficients de z, le seul mouvement possible est celui consistant à augmenter x_2 (et donc à garder $x_3 = 0$). Les contraintes de positivité imposent $x_2 \leq 500$ et $x_2 \leq \frac{1}{6}1200$. Le meilleur mouvement est donc $x_2 = 200$, qui conduit au sommet :

$$x_3 = x_5 = 0 \quad ; \quad x_1 = 1000,\ x_2 = 200,\ x_4 = 300$$

5. Du point de vue de ce sommet, le système se récrit

$$\left.\begin{array}{rcl} x_1 &=& 1000 - x_3 \\ x_2 &=& 200 + \frac{1}{2}x_3 - \frac{1}{6}x_5 \\ x_4 &=& 300 - \frac{1}{2}x_3 + \frac{1}{6}x_5 \end{array}\right\} \quad z = 6400 + 2\,x_3 - 2\,x_5$$

La seule variable modifiable est x_3. Vu les contraintes de positivité sur les trois autres variables, cette modification doit vérifier $x_3 \leq 1000$, $-400 \leq x_3$ et $x_3 \leq 600$. Le meilleur mouvement possible est donc $x_3 = 600$, qui conduit au sommet :

$$x_4 = x_5 = 0 \quad ; \quad x_1 = 400,\, x_2 = 500,\, x_3 = 600$$

6. Une dernière réécriture conduit à :

$$\left.\begin{array}{rcl} x_1 &=& 400 + 2x_4 - \frac{1}{3}x_5 \\ x_2 &=& 500 - x_4 \\ x_3 &=& 600 - 2x_4 + \frac{1}{3}x_5 \end{array}\right\} \quad z = 7600 - 4\,x_4 - \frac{4}{3}\,x_5$$

Comme tous les coefficients de la fonction de bénéfice sont négatifs, on est arrivé à un sommet optimal (et comme tous les coefficients sont non nuls, ce sommet optimal est le seul). Le bénéfice maximal escompté est donc $z = 7600$.

7. Le résultat de ce calcul n'est pas la valeur du bénéfice escompté (qui dépend de la validité du modèle et des diverses influences extérieures) mais la ***décision conseillée***, qui est

$$x_1 = 400,\, x_2 = 500$$

Exercise 3.1.1. *Maximiser* $z = 3x_1 + 4x_2$ *sous les contraintes* $x_1 + 7x_2 \leq 1000$, $7x_1 + x_2 \leq 1200$ *et* $3x_1 + 2x_1 \leq 2000$.

3.2 Un système facile en dimension 3

On se pose le problème suivant :

$$\textit{maximiser} \quad z = 4\,x_1 + 12\,x_2 + 3\,x_3 \quad \textit{sachant que} \left\{\begin{array}{rcl} x_1 &\leq& 1000 \\ x_2 &\leq& 500 \\ x_3 &\leq& 1500 \\ 3\,x_1 + 6\,x_2 + 2\,x_3 &\leq& 6750 \end{array}\right.$$

1. On introduit une nouvelle variable par contrainte, et le système devient :

$$\left.\begin{array}{rcl} x_4 &=& 1000 - x_1 \\ x_5 &=& 500 - x_2 \\ x_6 &=& 1500 - x_3 \\ x_7 &=& 6750 - 3\,x_1 - 6\,x_2 - 2\,x_3 \end{array}\right\} \quad z = 4\,x_1 + 12\,x_2 + 3\,x_3$$

Les trois variables sont modifiables, et nous choisissons de modifier x_1 (et donc de garder $x_2 = x_3 = 0$). Les contraintes de positivité imposent $x_1 \leq 1000$ et $x_1 \leq 2250$. Le meilleur mouvement est donc $x_1 = 1000$, qui conduit au sommet :

$$x_2 = x_3 = x_4 = 0 \quad ; \quad x_1 = 1000,\, x_5 = 500,\, x_6 = 1500,\, x_7 = 3750$$

2. Du point de vue de ce sommet, le système se récrit :

$$\left.\begin{array}{rcl} x_1 &=& 1000 - x_4 \\ x_5 &=& 500 - x_2 \\ x_6 &=& 1500 - x_3 \\ x_7 &=& 3750 + 3\,x_4 - 6\,x_2 - 2\,x_3 \end{array}\right\} \quad z = 4000 + 12\,x_2 + 3\,x_3 - 4\,x_4$$

Deux variables sont modifiables. Nous choisissons de modifier x_2 (et de ne pas modifier $x_3 = x_4 = 0$). Vu les contraintes de positivité, cette modification doit vérifier $x_2 \leq 500$ et $x_2 \leq 3750/6 = 625$. Le meilleur mouvement possible est donc $x_2 = 500$, qui conduit au sommet :

$$x_3 = x_4 = x_5 = 0 \quad ; \quad x_1 = 1000,\ x_2 = 500,\ x_6 = 1500,\ x_7 = 750$$

3. Du point de vue de ce sommet, le système se récrit :

$$\left.\begin{array}{rcl} x_1 &=& 1000 - x_4 \\ x_2 &=& 500 - x_5 \\ x_6 &=& 1500 - x_3 \\ x_7 &=& 750 + 3\,x_4 + 6\,x_5 - 2\,x_3 \end{array}\right\} \quad z = 10000 + 3\,x_3 - 4\,x_4 - 12\,x_5$$

La seule variable modifiable est x_3. Les contraintes de positivité imposent $x_3 \leq 1500$ et $x_3 \leq 375$. Le meilleur mouvement possible est donc $x_3 = 375$, qui conduit au sommet :

$$x_4 = x_5 = x_7 = 0 \quad ; \quad x_1 = 1000,\ x_2 = 500,\ x_3 = 375,\ x_6 = 1125$$

4. Du point de vue de ce sommet, le système se récrit :

$$\left.\begin{array}{rcl} x_1 &=& 1000 - x_4 \\ x_2 &=& 500 - x_5 \\ x_3 &=& 375 + \frac{3}{2}x_4 + 3\,x_5 - \frac{1}{2}x_7 \\ x_6 &=& 1125 - \frac{3}{2}\,x_4 - 3\,x_5 + \frac{1}{2}x_7 \end{array}\right\} \quad z = 11125 + \frac{1}{2}x_4 - 3\,x_5 - \frac{3}{2}x_7$$

La seule variable modifiable est x_4, avec pour contraintes $x_4 \leq 1000$ et $x_4 \leq 1125 \times \frac{2}{3} = 750$. Le meilleur mouvement est donc $x_4 = 750$, conduisant au sommet :

$$x_5 = x_6 = x_7 = 0 \quad ; \quad x_1 = 250,\ x_2 = 500,\ x_3 = 1500,\ x_4 = 750$$

5. Du point de vue de ce sommet, le système se récrit :

$$\left.\begin{array}{rcl} x_1 &=& 250 + 2\,x_5 + \frac{2}{3}x_6 - \frac{1}{3}x_7 \\ x_2 &=& 500 - x_5 \\ x_3 &=& 1500 - x_6 \\ x_4 &=& 750 - 2x_5 - \frac{2}{3}x_6 + \frac{1}{3}x_7 \end{array}\right\} \quad z = 11500 - 4\,x_5 - \frac{1}{3}x_6 - \frac{4}{3}x_7$$

Il ne reste plus de variables modifiables, provoquant l'arrêt de l'algorithme. La décision proposée est donc :

$$x_1 = 250,\ x_2 = 500,\ x_3 = 1500$$

qui vérifie toutes les contraintes et conduit à un bénéfice prévisionnel de 11500.

Exercise 3.2.1. *Reprendre les calculs en commençant par modifier x_2 au point 1 ci-dessus*

3.3 Système général

1. Si l'origine n'est pas un point de bon fonctionnement, un travail supplémentaire doit être effectué. Il faut ou bien trouver un sommet acceptable, ou bien démontrer qu'il n'en existe pas. Dans le premier cas, on pourra partir de ce sommet et appliquer l'algorithme décrit ci-dessus. Dans le deuxième cas, il faut évidemment obtenir une démonstration de non existence, et pas seulement un constat d'échec.
2. Méthode des variables auxiliaires. En plus des variables naturelles x_k du problème, on introduit autant de variables y_n que d'inégalités (comme ci-dessus), puis autant de variables z_m que de relations non vérifiées par l'origine. Ainsi
   $x_1 + 2\,x_2 \leq 100$ devient $x_1 + 2\,x_2 + y_1 = 100$ (contrainte majoration)
   $x_1 + 2\,x_2 = 100$ devient $x_1 + 2\,x_2 + z_1 = 100$ (contrainte égalité)
   $x_1 + 2\,x_2 \geq 100$ devient $x_1 + 2\,x_2 + z_2 = 100 + y_2$ (contrainte minoration)

3. Un sommet acceptable du problème initial est caractérisé par $\forall m\ :\ z_m = 0$, et nous sommes dans une situation où l'origine ne vérifie pas cette condition.
4. Nous nous posons alors un ***nouveau*** problème, dont
   — les variables principales sont tous les x_k et ceux des y_n qui sont issus d'une contrainte minoration.
   — les équations linéarisées sont celles du problème initial
   — la fonction de bénéfice est $f = -\sum z_m$
5. La nouvelle origine (la dimension ayant augmenté, ce n'est plus le même espace) est un point acceptable pour le nouveau problème.
6. Si l'on arrive à $\max f < 0$, il est alors prouvé que le problème initial est insoluble (contraintes contradictoires).
7. Si l'on arrive à $\max f = 0$, on a nécessairement $\forall m\ :\ z_m = 0$ puisque ces variables sont positives. Et alors les variables x_k et y_n sont caractéristiques d'un sommet acceptable du problème initial.

3.4 Un exemple en deux phases

1. Considérons le problème

$$\max\left(4\,x_1 + 12\,x_2\right) \quad \textit{sachant que} \begin{cases} x_1 \leq 1000 \\ x_2 \leq 500 \\ 3\,x_1 + 6\,x_2 \leq 4200 \\ 1000 \leq x_1 + x_2 \\ 1200 \leq x_1 + 2\,x_2 \end{cases}$$

2. La procédure `mk_simplex` permet de le récrire en :

$$\left.\begin{array}{rcl} x_3 &=& 1000 - x_1 \\ x_4 &=& 500 - x_2 \\ x_5 &=& 4200 - 3\,x_1 - 6\,x_2 \\ x_8 &=& 1000 - x_1 - x_2 + x_6 \\ x_9 &=& 1200 - x_1 - 2\,x_2 + x_7 \end{array}\right\},\ f \doteq -x_8 - x_9 = -2200 + 2\,x_1 + 3\,x_2 - x_6 - x_7$$

   Les variables $x_3 \cdots x_7$ sont les variables de contrôle des faces et les variables x_8, x_9 ont été introduites pour tenir compte du non respect des contraintes par le point $(x_1 = x_2 = 0)$. Le problème comportant le même nombre d'équations qu'avant, le nombre de variables principales augmente de 2, et l'on part du point $(x_1 = x_2 = x_6 = x_7 = 0)$.
3. On choisit la variable x_2 comme pivot, et on obtient :

$$\left.\begin{array}{rcl} x_2 &=& 500 - x_4 \\ x_3 &=& 1000 - x_1 \\ x_5 &=& 1200 - 3\,x_1 + 6\,x_4 \\ x_8 &=& 500 - x_1 + x_4 + x_6 \\ x_9 &=& 200 - x_1 + 2\,x_4 + x_7 \end{array}\right\},\ f = -700 + 2\,x_1 - 3\,x_4 - x_6 - x_7$$

4. La variable x_1 s'impose comme pivot, et on obtient :

$$\left.\begin{array}{rcl} x_1 &=& 200 + 2\,x_4 + x_7 - x_9 \\ x_2 &=& 500 - x_4 \\ x_3 &=& 800 - 2\,x_4 - x_7 + x_9 \\ x_5 &=& 600 - 3\,x_7 + 3\,x_9 \\ x_8 &=& 300 - x_4 + x_6 - x_7 + x_9 \end{array}\right\},\ f = -300 + x_4 - x_6 + x_7 - 2\,x_9$$

5. Puis, x_4 étant pris comme pivot :

$$\left.\begin{array}{rcl} x_1 &=& 800 + 2\,x_6 - x_7 - 2\,x_8 + x_9 \\ x_2 &=& 200 - x_6 + x_7 + x_8 - x_9 \\ x_3 &=& 200 - 2\,x_6 + x_7 + 2\,x_8 - x_9 \\ x_4 &=& 300 + x_6 - x_7 - x_8 + x_9 \\ x_5 &=& 600 - 3\,x_7 + 3\,x_9 \end{array}\right\},\ f = -x_8 - x_9$$

Le fait remarquable n'est pas que $f = -x_8 - x_9$ (cette égalité à lieu depuis le début, puisque c'est la définition de f !) mais que ces deux variables (qui expriment la violation des contraintes) soient enfin nulles.

6. Ainsi, nous venons d'atteindre un sommet admissible pour le problème initial, à savoir $x_6 = x_7 = 0$. Nous pouvons donc nous débarrasser des variables "z" (i.e. x_8 et x_9) et revenir au problème initial, qui s'écrit désormais :

$$\left.\begin{array}{rcl} x_1 &=& 800 + 2\,x_6 - x_7 \\ x_2 &=& 200 - x_6 + x_7 \\ x_3 &=& 200 - 2\,x_6 + x_7 \\ x_4 &=& 300 + x_6 - x_7 \\ x_5 &=& 600 - 3\,x_7 \end{array}\right\},\ z \doteq 4\,x_1 + 12\,x_2 = 5600 - 4\,x_6 + 8\,x_7$$

et se résout par la méthode usuelle.

7. On trouve, en une étape,

$$\left.\begin{array}{rcl} x_1 &=& 600 + 2\,x_6 + \frac{1}{3}x_5 \\ x_2 &=& 400 - x_6 - \frac{1}{3}x_5 \\ x_3 &=& 400 - 2\,x_6 - \frac{1}{3}x_5 \\ x_4 &=& 100 + x_6 + \frac{1}{3}x_5 \\ x_7 &=& 200 - \frac{1}{3}x_5 \end{array}\right\},\ z = 7200 - \frac{8}{3}x_5 - 4\,x_6$$

La décision à recommander est donc $x_1 = 600$ et $x_2 = 400$, pour un bénéfice escompté de $z = 7200$.

3.5 Un exemple de contraintes redondantes

1. Reprenons le système du paragraphe précédent, à savoir :

$$\left.\begin{array}{rcl} x_3 &=& 1000 - x_1 \\ x_4 &=& 500 - x_2 \\ x_5 &=& 4200 - 3\,x_1 - 6\,x_2 \\ x_8 &=& 1000 - x_1 - x_2 + x_6 \\ x_9 &=& 1200 - x_1 - 2\,x_2 + x_7 \end{array}\right\},\ z = -2200 + 2\,x_1 + 3\,x_2 - x_6 - x_7$$

2. Si nous choisissons maintenant x_1 comme pivot, au lieu de x_2, nous obtenons :

$$\left.\begin{array}{rcl} x_1 &=& 1000 - x_3 \\ x_4 &=& 500 - x_2 \\ x_5 &=& 1200 + 3\,x_3 - 6\,x_2 \\ x_8 &=& -x_2 + x_3 + x_6 \\ x_9 &=& 200 - 2\,x_2 + x_3 + x_7 \end{array}\right\},\ z = -1200 + 3\,x_2 - 2\,x_3 - x_6 - x_7$$

Une fois arrivé là, il n'est plus possible de bouger. En effet, la variable x_2 est la seule à influer positivement sur la fonction de bénéfice, et d'autre part $x_8 = -x_2 + x_3 + x_6$ avec $x_3 = x_6 = 0$ et $0 \leq x_2$, $0 \leq x_8$ impose $x_2 = x_8 = 0$.

3. L'équation $x_2 = x_3 + (x_6 - x_8)$ exprime que l'information véhiculée par la contrainte $x_2 \geq 0$ est une simple conséquence des contraintes $x_1 \leq 1000$ et $1000 \leq x_1 + x_2$. On peut donc se ramener à un système avec une inconnue de moins et une contrainte de moins par la substitution $\xi_1 = x_1$, $\xi_2 = 500 - x_2$. Il vient :

$$\max\left(6000 + 4\,\xi_1 - 12\,\xi_2\right) \quad \textit{sachant que} \begin{cases} \xi_1 \leq 1000 \\ 3\,\xi_1 - 6\,\xi_2 \leq 1200 \\ 500 \leq \xi_1 - \xi_2 \\ 200 \leq \xi_1 - 2\,\xi_2 \end{cases}$$

Exercise 3.5.1. *Résoudre le système ci-dessus. Vérifier qu'il conduit au résultat déjà obtenu.*

Exercise 3.5.2. *Représentation et résolution graphiques de ce système.*

Exercise 3.5.3. *Reprendre ce système en modifiant légèrement une équation, de façon à rompre la liaison. Comparer les résultats obtenus.*

Annexe A

Programmation effective

A.1 Listings commentés (Maple)

Une optimisation évidente de l'algorithme du simplexe consiste à confier le soin des additions et multiplications à un ordinateur. On sera néanmoins attentif à ne pas sous-traiter la réflexion, qui constitue le coeur du métier d'ingénieur.

Le programme `mk_simplex` (Incl. A.1) a pour objectif de créer toutes les variables auxiliaires nécessaires à la transformation des inéquations en équations et à la transformation des contraintes non satisfaites par l'origine du problème initial en contraintes satisfaites par l'origine d'un problème de plus grande taille. Pour des raisons de commodité, les trois sortes de variables s'appellent toutes x, l'indice variant de 1 à N pour les variables naturelles, de $N+1$ à $N+M$ pour les variables "y" et de $N+M+1$ à la fin pour les variables "z".

En sortie, *eqs* contient les équations récrites, *tous* est l'ensemble de toutes les variables, *spec* est l'ensemble des variables "z" et *lieu* est la liste des coordonnées dont l'égalité à 0 caractérise le point où l'on se trouve.

Le programme `pt_vue` (Incl. A.2) est un programme "cosmétique" destiné à récrire l'ensemble des équations du point de vue du sommet où l'on se trouve. L'écriture effective de ce programme est tout à fait simple (on utilise *solve* pour résoudre), mais est loin d'être optimale. En effet, une étape est caractérisée par l'échange de rôles entre une variable "actuellement principale" et une variable "actuellement auxiliaire" : une résolution de proche en proche serait plus rapide.

Le programme `avance` (Incl. A.5) exécute le déplacement consistant en l'échange des rôles entre la variable principale *piv*, choisie par l'utilisateur, et la variable auxiliaire exerçant la contrainte la plus forte. Le cas exceptionnel où l'on trouve un mouvement $x_{piv} = 0$ ne se produit qu'en présence de contraintes redondantes.

Enfin, le programme `randavance` (Incl. A.6) détermine quels sont les pivots possibles (variables ayant une influence positive sur la fonction de bénéfice) et en choisit un aléatoirement.

A.2 Scilab 4.1 utilise un algorithme quadratique

Scilab A.2.1. La commande Scilab 4.1 de résolution des "programmes linéaires" est linpro. Consulter l'aide en ligne `help(linpro)` pour une description de la syntaxe. Etre attentif au fait que f est considérée comme une fonction de coût...

Exercise A.2.2. Examiner la feuille de calcul linpro.sci et en déduire la méthode utilisée par le logiciel pour calculer l'optimum voulu.

Scilab A.2.3. Pour résoudre l'exemple de la Section 2.2, nous écrivons :

```
. ci=[0;0]        // contrainte inférieure sur x1, x2
. cs=[1000;500]   // contraintes supérieures, matérialisées par x3, x4
. p=-[4;12]       // les coefficients de la fonction de côut
```

Listing A.1: Traitement des inégalités, création des variables supplémentaires.

```
mk_simplex := proc (benef)
 1: global x, z, eqb, eqb2, eqs, tous, spec, lieu, lessubs
 2: local lesargs, lesvarsi, lesvarsx, ni, nx, k, n, m, li, N, M, eq, tmp
 3: indets([args]) ; lesvarsi, lesvarsx := selectremove(type, %, indexed) ;
 4: x :=' x' ; z :=' z' ; nx, ni := nops(lesvarsx), nops(lesvarsi) ;
 5: if ni ≠ 0 and nx ≠ 0 then
 6:   error "mélange entre variables indexées et non indexées" ;
 7: end if
 8: if nx ≠ 0 then
 9:   if member(x, lesvarsx) then
10:     tmp := sort(convert(lesvarsx minus{x}, list)) ;
11:     lessubs := x = x[1], seq(tmp[js] = x[js + 1], js = 1 .. nx − 1) ;
12:   else
13:     lessubs := seq(lesvarsx[js] = x[js], js = 1 .. nx) ;
14:   end if
15: else
16:   tmp := map2(op, 0, lesvarsi) ;
17:   if nops(tmp) ≠ 1 then
18:     error "plusieurs noms de variables indexées";
19:   end if
20:   tmp := op(tmp) ; map2(op, 1, lesvarsi) ;
21:   if % ≠ {$1 .. ni} then error "numérotation lacunaire" end if
22:   lessubs := seq(tmp[js] = x[js], js = 1 .. ni) ;
23: end if
24: eqb := z = subs(lessubs, benef) ; eqb2 := NULL ;
25: N := nops({lessubs}) ; lesargs := subs(lessubs, [args][2 .. nargs]) ;
26: li := NULL ; n := 0 ; m := 0 ; lieu := $1 .. N ;
27: for k in lesargs do
28:   if type(k, ' <= ') then
29:     n := n + 1 ; eq := rhs(k) − lhs(k) − x[N + n] ; tmp := N + n ;
30:   else if type(k, ' = ') then
31:     eq := k ; tmp := NULL ;
32:   else
33:     error "bad type of relation"
34:   end if
35:   if remove(has, eq, x) < 0 then
36:     m := m + 1 ; eq := −eq − x[M + m] ; lieu := lieu, tmp ;
37:   end if
38:   li := li, eq ;
39: end for
40: M := N + n ; eqs := eval(li) ; lieu := [lieu] ;
41: tous := {z, seq(x[ks], ks = 1..M + m)} ;
42: spec := {seq(x[ks], ks = M + 1..M + m)} ;
43: if m > 0 then eqb2 := eqb ; eqb := z = −add(js, js = spec) end if
```

Ensure: initialisation des variables globales x, z, eqb, $eqb2$, eqs, $tous$, $spec$, $lieu$, $lessubs$.

Listing A.2: Affichage du système (du point de vue du sommet en cours).

```
PT_VUE := proc (lieu) ; global sol, eqs, eqb, tous, x, z ; local autres, sor
1: sor := (u, v) → evalb((op@lhs)(u) < (op@lhs)(v))
2: autres := tous minus {z, seq(x[k], k = lieu)} ;
3: solve({eqs}, autres) ; convert(%, list) ; sol := (op@sort)(%, sor) ;
4: subs(sol, rhs(eqb)) ; selectremove(has, %, x) ;
5: return < sol >, lhs(eqb) = "(%[2]) + %[1] ;
```

Listing A.3: Une étape de l'algorithme (avec choix du pivot par l'utilisateur).

```
AVANCE := proc (piv) ; global lieu, sol, x
1: local fixe, constraints, mini, tmp, xxx
2: fixe := seq(x[k] = 0, k = subs(piv = NULL, lieu)) ;
3: constraints := select(has, subs(fixe, [sol]), x[piv]) ;
4: tmp := map(solve@rhs, constraints) ; mini := (min@op@select)(type, tmp, nonnegative)
   ;
5: if mini = 0 then ERROR "le pivot est bloqué" end if
6: member(mini, tmp, 'xxx') ; op(lhs(constraints[xxx])) ;
7: lieu := sort(subs(piv = %, lieu)) ;
```

Listing A.4: randavan

```
RANDAVAN := proc ()
1: global lieu, sol, piv, eqb, x, z
2: local tmp, benef
3: benef := subs(eqb, sol, z) ;
4: [seq(`if`(coeff(benef, k) > 0, k, NULL), k = indets(benef))] ;
5: if nops(%) = 0 then return "maximum atteint" end if
6: %[rand(1..nops(%))()] ; avan(op(%)) ;
```

Listing A.5: Une étape de l'algorithme (avec choix du pivot par l'utilisateur).

```
AVANCE := proc (piv)
1: global lieu, sol ; local fixe, tmp, xxx, qui
2: fixe := seq(x[k] = 0, k = subs(piv = NULL, lieu))
3: tmp := select(has, subs(fixe, [sol]), x[piv])
4: map(solve@rhs, tmp) ; (min@op@select)(type, %, nonnegative)
5: if % = 0 then return `le pivot est bloqué`
6: member(%, %%, xxx) ; qui := (op@lhs)(tmp[xxx])
7: lieu := (sort@subs)(piv = qui, lieu)
```

Listing A.6: Une étape de l'algorithme (avec choix aléatoire du pivot).

```
RANDAVANCE := proc ()
1: global lieu, sol ; local fixe, tmp, xxx, benef, choix, piv, vars
2: benef := subs(eqb, sol, z) ; vars := indets(benef)
3: choix := [seq(if(coeff(benef, k) > 0, op(k), NULL), k = vars)] ;
4: if choix = [] then return `le maximum est atteint`
5: piv := choix[rand(1..nops(choix))()]
6: fixe := seq(x[k] = 0, k = subs(piv = NULL, lieu)) ;
7: tmp := select(has, subs(fixe, [sol]), x[piv]) ;
8: map(solve@rhs, tmp) ; (min@op@select)(type, %, nonnegative)
9: if % = 0 then return `le pivot est bloqué`
10: member(%, %%, xxx) ; qui := (op@lhs)(tmp[xxx])
11: lieu := (sort@subs)(piv = qui, lieu)
Ensure: Ou bien un diagnostic ou bien une progression du bénéfice
```

. `C=[3,6]` // les coefficients des contraintes (ici : x_5)
. `b=[4200]` // les majorants contraignants
. `[xx, lagr,ff]=linpro(p,C,b,ci,cs)`

Et nous recevons :

. `xx`=[400. ; 500.], `lagr`=[0. ; 4. ; 1.333], `ff`=−7600

Exercise A.2.4. *Reprendre les calculs en commençant par modifier x_2 au point 4 ci-dessus*

A.3 Programme simplexe sous Scilab

A.3.1 Incantations initiales

Listing A.7: Incantations initiales

```
1: //Algorithme du simplexe
2: //www.douillet.info
3: cd "~/docs/Ensait/oprea/scilab";
4: funcprot(0); hd=mopen("msg.txt",'w');
5: function mymprintf(aa,bb) mprintf(aa,bb) mfprintf(hd,aa,bb); endfunction
6: function myprintf(aa) mprintf(aa) mfprintf(hd,aa); endfunction
7: memieee=ieee(); ieee(2);
8: myseed=12345; grand('setsd', myseed);
```

Les sorties se font à l'écran et dans un fichier. La fonction mymprint gère cela. Avec funcprot(0), on évite le message "fonction redéfinie". Avec ieee(2), une division par 0 engendre le nombre %inf et non une erreur. Avec myseed, on initialise le générateur aléatoire (utile pour debug).

A.3.2 Entrée des données et affichage

Listing A.8: Entrée des données

```
1: ma=[1 0 0; 0 1 0; 0 0 1; 3 6 2]; mb=[1000 5000 1500 6750]';
2: mza=[4 12 3]; mzb=[0];
3: [neq,nxx]=size(ma); vue=1:nxx;
4: id=eye(ms); ms=[eye(nxx,nxx),zeros(nxx,neq+1)];
5: ms=[ms;[-ma;mza],zeros(neq+1,neq), [mb;mzb]];
6: fvu="vue= "; for j=1:size(vue,'c') do; fvu=fvu+"%d "; end; fvu=fvu+"\n";
7: fff=""; for j=1:size(ms,'c')-1 do; fff=fff+"%+10f "; end; fff=fff+"%+13f\n";
8: mymprintf("seed= %d\n\n", myseed); mymprintf(fvu, vue); mymprintf(fff, ms)
```

La matrice *ms* regroupe toutes les données du système. Les contraintes supérieures sont entrées comme des contraintes ordinaires. Les formats d'affichage (fff pour la matrice et fvu pour la vue) sont construits d'après les données.

Annexe B

Compléments

B.1 Quelques rappels sur les symboles de Landau

Cette annexe a pour objectif de rappeler quelques propriétés des quatre symboles de Landau : $\mathbf{O}(f)$, $\mathrm{o}(f)$, $\sim$ et Θ.

Remark B.1.1. Les notations de Landau (qui se sont imposées comme standard de fait) ne sont pourtant pas des plus heureuses. En effet les deux relations $g_1 = f + \mathrm{o}(f)$ et $g_2 = f + \mathrm{o}(f)$ n'impliquent pas que $g_1 = g_2$. Il vaut mieux les lire et les comprendre comme $g_1 - f \in \mathrm{o}(f)$ et $g_2 - f \in \mathrm{o}(f)$, avec pour conséquence $g_2 - g_1 \in \mathrm{o}(f)$.

Definition B.1.2 (equivalence)**.** La relation "$f \sim g$ quand $x \to 0$" est définie par :

$$\forall x : (f(x) = 0 \Leftrightarrow g(x) = 0) \text{ et } \lim \left[\frac{f(x)}{g(x)} \middle/ \mid x \to 0,\ f(x) \neq 0 \right] = 1$$

Exercise B.1.3. *Quelles sont les fonctions telles que $f \sim 0$?*

Exercise B.1.4. *Montrer que la définition de $f \sim g$ équivaut à l'existence d'une fonction q telle que $f(x) = g(x) \times q(x)$ avec $q(x) \to 1$.*

Definition B.1.5. Une relation d'équivalence est une relation réflexive, symétrique et transitive, i.e. :

$$\begin{array}{lll} R: & \forall x & x \sim x \\ S: & \forall x, y & x \sim y \implies y \sim x \\ T: & \forall x, y, z & (x \sim y \text{ et } y \sim z) \implies x \sim z \end{array}$$

Definition B.1.6. Dire qu'une relation d'équivalence est compatible avec une opération $\perp$ est :

$$(a \sim c \text{ et } b \sim d) \implies (a \perp b \sim c \perp d)$$

Autrement dit, on peut remplacer des équivalents par des équivalents.

Theorem B.1.7. *La relation $\sim$ est une équivalence, et elle est compatible avec le produit.*

Definition B.1.8 (grand O)**.** Définition $\mathbf{O}(f)$. La relation "f contrôle g", notée $g \in \mathbf{O}(f)$, est définie par l'existence d'un facteur de contrôle, i.e.

$$\forall x : |g(x)| \leq k\ |f(x)|$$

Exercise B.1.9. *Montrer que cette relation est RT, mais pas S.*

Definition B.1.10 (même ordre)**.** On dit que deux fonctions sont du même ordre lorsque $f \in \mathbf{O}(g)$ et $g \in \mathbf{O}(f)$. Ceci ce note par $f \,\Theta\, g$.

Exercise B.1.11. *Montrer que cette relation est RST, i.e. est une équivalence.*

Definition B.1.12. La relation "g est négligeable devant f", notée $g \in \mathrm{o}(f)$, est définie par l'existence d'une fonction de "négligeabilité" ε, i.e.

$$\forall x : g(x) = f(x) \times \varepsilon(x) \text{ et } \varepsilon(x) \to 0$$

Exercise B.1.13. *Cette relation est elle R, S ou T ?*

Theorem B.1.14. *Les ensembles* $\mathbf{O}(f)$ *et* $\mathrm{o}(f)$ *sont clos par combinaisons linéaires.*

Bibliographie

Chinneck J.W. *Practical Optimization: A Gentle Introduction* (Carleton University, Ottawa, Ontario) (2007), 15 chapters, 151 pp. http://www.sce.carleton.ca/faculty/chinneck/po.html.