Optimisation linéaire et discrète
Théorie mathématique et Exercices,

Bernhard Beckermann
Laboratoire Paul Painlevé UMR 8524
Université de Lille
59655 Villeneuve d'Ascq CEDEX
e-mail : Bernhard.Beckermann@univ-lille.fr

Ana Cristina Matos
Laboratoire Paul Painlevé UMR 8524
Université de Lille
59655 Villeneuve d'Ascq CEDEX
e-mail : Ana.Matos@univ-lille.fr

10/1/2025

Table des matières

Chapitre 1

Introduction

1.1 A propos de ce texte

Ce texte a pour objectif d'introduire au sujet de la recherche opérationnelle, en particulier à l'optimisation linéaire, l'optimisation combinatoire.Il s'adresse aux étudiants ayant acquis des bases en Algèbre linéaire, et désirant savoir plus sur les outils mathématiques et pratiques (algorithmique, mise en œuvre sur ordinateur) en optimisation, dès la troisième année d'études universitaires ou d'école d'ingénieur. Le texte est basé sur un cours et des travaux dirigés dispensés par les auteurs en Maîtrise d'Ingénierie Mathématique (MIM), en Master professionnel ISN et en Master MA ISN (Master 1 et 2) à l'Université de Lille.

Le lecteur interessé consultera avec profit en parallèle d'autres livres d'optimisation, en particulier

- Michel Minoux, Programmation mathématique, Tome 1, Dunod (1983) ;
- Michel Sakarovitch, Optimisation combinatoire, Tome 1 : Graphes et programmation linéaire (Tome 2 : Programmation discrète), Hermann (1984) ;
- G.B. Dantzig, Applications et prolongements de la programmation linéaire, Dunod (1966).

1.2 Quelques exemples

L'objectif de la Recherche opérationnelle consiste à développer des méthodes efficaces pour la résolution des problèmes du type

$$\max\{f(x) : x \in \mathcal{M}\} \tag{1.1}$$

où $f : \mathcal{M} \mapsto \mathbb{R}$, et $\mathcal{M}$ peut être un ensemble fini (programmation discrète), ou une partie continue du $\mathbb{R}^n$ (par exemple en programmation linéaire un polyèdre, c'est-à-dire, une intersection fini de demi-espaces). Pour fixer des idées, les valeurs $f(x)$ peuvent être les bénéfices pour un plan de production x, et $\mathcal{M}$ décrit les plans de production admissibles, définis par les restrictions de ressources (temps, machines, matières premières) et les demandes de production.

Comme $\min\{f(x) : x \in \mathcal{M}\} = -\max\{-f(x) : x \in \mathcal{M}\}$, ce formalisme permet aussi de

décrire des problèmes de minimisation (par exemple le coût de production). Notons également que dans certains domaines comme l'optimisation des formes, il est utile aussi de prendre pour $\mathcal{M}$ également des sous-espaces de fonctions (décrivant par exemple la forme d'une boule de savon), mais nous allons nous restreindre dans ce texte à $\mathcal{M} \subset \mathbb{R}^n$ (par exemple obtenu après discrétisation d'une surface), ce qui facilite l'analyse, et ce qui permet souvent de donner des interprétations géométriques intéressantes.

Pour fixer les idées, discutons quatre exemples.

1.2.1. Distribution de ressources *(programmation linéaire)*
Un boulanger veut produire des rouleaux de pâte brisée et de pâte feuilletée. La production d'une unité de rouleaux (en milliers) nécessite une certaine quantité de beurre, sel et farine (en tonnes, cf. tableau ci–dessous). On cherche à maximiser le revenu qu'il est possible de tirer des stocks.

	beurre	*sel*	*farine*	*prix de vente*
pâte brisée	*1*	*0*	*2*	*4*
pâte feuilletée	*2*	*1*	*1*	*5*
quantité en stock	*7*	*3*	*8*	

En notant par x_1 (et par x_2) la quantité à produire (en milliers de rouleaux) de pâte brisée (et de pâte feuilletée, respectivement), nous sommes amenés à résoudre

$$\max\{f(x) : x \in \mathcal{M}\}, \quad f(x) = 4x_1 + 5x_2, \tag{1.2}$$
$$\mathcal{M} = \{x = (x_1, x_2)^t \in \mathbb{R}^2 : x_1 \geq 0, x_2 \geq 0, x_1 + 2x_2 \leq 7, x_2 \leq 3, 2x_1 + x_2 \leq 8\},$$

c'est-à-dire, nous devons maximiser une fonction linéaire sur un polyèdre du $\mathbb{R}^2$. Le problème (1.2) est un problème de programmation linéaire *(voir Chapitre 2), et plus précisément du type* distribution de ressources*, qui dans un contexte plus général se décrit comme suit : Trouver un plan de production maximisant les bénéfices tout en respectant les contraintes en termes de matériel, temps, etc., par exemple*

- *la production d'une tonne du produit k ($k = 1, ..., n$) nécessite A_j^k tonnes du matériel j ($j = 1, ..., m$)*
- *on peut vendre le produit k à f^k E/tonne ($k = 1, ..., n$)*
- *on dispose d'un stock de b_j tonnes du matériel j ($j = 1, ..., m$).*

En notant par x_k le chiffre de production (en tonnes) du produit k, on obtient le modèle mathématique (1.1) avec $f(x) = f^1 x_1 + ... + f^n x_n$ et

$$\mathcal{M} = \{x = (x_1, ..., x_n)^t \in \mathbb{R}^n : \forall k = 1, ..., n : x_k \geq 0, \forall j = 1, ..., m : A_j^1 x_1 + ... + A_j^n x_n \leq b_j\}.$$

Parfois il convient d'ajouter la contrainte $x_k \in \mathbb{Z}$ pour modéliser le fait que les objets en considération sont indivisibles : on obtient alors un programme linéaire en nombres entiers*, bien plus difficile à résoudre (voir le chapitre 3.7) où on fait appel aux techniques d'optimisation combinatoire.*

Pour expliquer son résultat au patron d'entreprise, le mathématicien ne devrait pas seulement fournir la solution exacte de ce problème. En pratique, les données sont assujetties à des fluctuations, et il convient d'inclure ces fluctuations dans l'étude. Il faudra alors pouvoir répondre aux questions (post optimisation) comme

— *comment changer le plan de production si f^k baisse de 2 E/tonne ?*
— *est-il intéressant d'acheter de la matière première j à un prix de 5 E/tonne ?*

Un élément essentiel dans l'interprétation économique des résultats peut être fourni par l'étude du problème dual *qui pour notre boulanger prend la forme suivante : le boulanger observe que les matières premières beurre, sel et farine coûtent tellement cher (λ^j Euros par tonne) que, pour les deux types de pâtes, le coût du matériel est supérieur aux bénéfices. On se pose alors la question de savoir combien peut-il gagner au moins s'il vend son stock au lieu de produire (par exemple à un concurrent). Mathématiquement, on se retrouve avec le problème*

$$\min\{7\lambda^1 + 3\lambda^2 + 8\lambda^3 : \lambda^1, \lambda^2, \lambda^3 \geq 0, \lambda^1 + 2\lambda^3 \geq 4, 2\lambda^1 + \lambda^2 + \lambda^3 \geq 5\}, \tag{1.3}$$

les dernières deux inégalités traduisant le fait que, pour les deux types de pâtes, le coût du matériel est supérieur aux bénéfices.

1.2.2. Problèmes de cheminement *(programmation discrète)*
Soit V un ensemble de villes comportant m éléments, et $R \subset V \times V$ un ensemble de routes (x, y) de longueur $d_{(x,y)}$ reliant la ville x à la ville y. Trouver l'itinéraire le plus court pour aller de $x \in V$ à $y \in V$.

Ce problème abstrait peut correspondre à un meilleur parcours dans un réseau de métro/ SNCF/autoroutes, avec l'objectif de minimiser la longueur, le temps de parcours, le péage, trouver l'itinéraire permettant des camions les plus lourds,...

L'ensemble $\mathcal{M}$ dans (1.1) contient l'ensemble des itinéraires imaginables, et il suffit de considérer des itinéraires ne passant pas deux fois par la même ville. En décrivant un itinéraire par la liste $[x_0, x_1, ..., x_k]$ de villes rencontrés, on obtient

$$\begin{aligned} \mathcal{M} \;=\; & \{[x_0, x_1, ..., x_k] : k \geq 1, x_1, ..., x_{k-1} \in V \setminus \{a, b\} \text{ distincts}, \\ & x_0 = a, x_k = b, \forall j = 0, ..., k-1 : (x_j, x_{j+1}) \in R\} \end{aligned}$$

comme l'ensemble des itinéraires intéressants allant de a à b, et

$$f([x_0, x_1, ..., x_k]) = d_{(x_0,x_1)} + ... + d_{(x_{k-1},x_k)}.$$

Notons que $1 \leq k \leq m-1$, ce qui fait que, même pour $R = V \times V$, l'ensemble $\mathcal{M}$ est fini

$$\#\mathcal{M} \leq \sum_{k=1}^{m-1} (m-2)(m-3)....(m-k) < \infty,$$

*souvent caractéristique pour un problème d'*optimisation discrète*. Pourtant, ici $\#S \geq (m-1)!$, ce qui croît trop vite avec m pour envisager de parcourir tout l'ensemble $\mathcal{M}$ pour trouver le meilleur itinéraire. Il faudra alors des stratégies pour construire successivement de meilleurs chemins. Des tels algorithmes (Bellman, Dijkstra, Roy-Warshall) sont classiques et seront étudiés au Chapitre 3, ils ont une complexité (nombre d'opérations arithmétiques et espace mémoire) de $\mathcal{O}(m^3)$, voir $\mathcal{O}(m^2 \log(m))$.*

1.2.3. Voyageur de commerce *(programmation discrète)*
Etant donné le réseau routier de l'exemple 1.2.2, une personne cherche à établir une tournée qui lui permet de passer une et une seule fois par chaque ville, pour revenir finalement dans la

ville de départ, ceci en minimisant la longueur de l'itinéraire parcouru. Ici $\mathcal{M}$ est l'ensemble des permutations non décomposables de l'ensemble V, avec la même fonction f.

Pour ce problème on rencontre un phénomène bien classique en théorie de graphes : même si l'énoncé est facile, la résolution est fortement non trivial. Déjà l'existence et la construction d'un tel parcours semblent être des questions non triviales. On ne connaît aucune méthode de résolution de complexité $\mathcal{O}(m^q)$ quelque soit $q > 0$. La résolution efficace du problème du voyageur de commerce dépasse le cadre du présent texte.

1.2.4. Problème de transport *(programmation linéaire/discrète)*
Comment transporter au moindre coût un certain bien depuis p entrepôts (a_j unités disponibles à l'entrepôt j, $j = 1, ..., p$) vers q magasins (b_k unités demandées au magasin k, $k = 1, ..., q$), sachant que les coûts de transport du j-ème entrepôt vers le k-ème magasin sont de $d_{j,k}$ E/unité.

Pour la mise en équation, notons par $x_{j,k}$ la quantité (en unités) transportée du j-ième entrepôt vers le k-ième magasin. On est amené à résoudre

$$\begin{cases} \min \sum_{j=1}^{p} \sum_{k=1}^{q} d_{j,k} x_{j,k} \\ \forall j = 1, ..., p, \forall k = 1, ..., q : x_{j,k} \geq 0 \\ \forall j = 1, ..., p : \sum_{k=1}^{q} x_{j,k} \leq a_j \\ \forall k = 1, ..., q : \sum_{j=1}^{p} x_{j,k} \geq b_k \end{cases}$$

(plus éventuellement la contrainte $x_k \in \mathbb{Z}$). Il s'agit bien d'un programme linéaire, mais les contraintes comportent beaucoup de coefficients 0, et la matrice correspondante admet une structure bien particulière. Ceci nous permettra de trouver au Chapitre 3 un algorithme plus efficace que celui approprié pour l'exemple 1.2.1.

1.3 Résolution graphique d'un programme linéaire

Nous allons reprendre l'exemple (1.2) du boulanger pour expliquer comment dans le cas de deux variables on peut résoudre graphiquement un programme linéaire.

Sur le graphe de la Figure 1.1, nous avons tracé en bleu les demi-espaces induits par les contraintes $x_1 \geq 0$, $x_2 \geq 0$, et en vert les trois autres demi-espaces. La partie n'appartenant pas à notre ensemble $\mathcal{M}$ a été hachée. Par conséquent, l'ensemble $\mathcal{M}$ est bien un polyèdre (convexe) du $\mathbb{R}^2$ avec 5 sommets.

Nous avons ensuite tracé en rouge les courbes de niveau $\{x : f(x) = c\}$ pour $c = 0$ et pour un autre paramètre c, on obtient des droites parallèles. Comme notre objectif est de maximiser f, il nous faudra rechercher le plus grand paramètre c de sorte que la courbe de niveau aie une intersection non vide avec $\mathcal{M}$. Dans le cas du boulanger on trouve la courbe de niveau passant par l'intersection entre les droites $g3$ et $g5$, ce qui donne comme solution optimale unique le point $x = (3, 2)^t$ ($c = 22$). Le boulanger devrait alors produire 3 milliers de rouleaux de pâte brisée, et 2 milliers de rouleaux de pâte feuilletée, pour obtenir un bénéfice de 22000 Euros.

Notons que, pour l'objectif $f(x) = 2x_1 + x_2$, on trouve comme solutions optimales le segment engendré par les points $(4, 0)$ et $(3, 2)^t$, on peut alors avoir plusieurs solutions optimales.

Conclusions au $\mathbb{R}^2$:

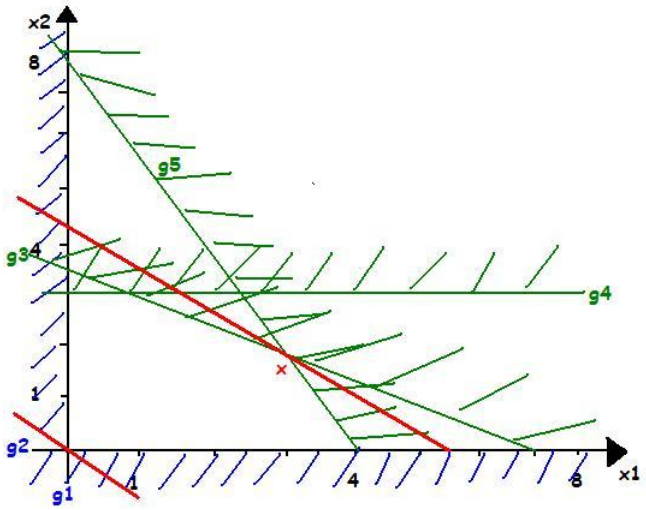


FIGURE 1.1 – RÉSOLUTION GRAPHIQUE POUR L'EXEMPLE DU BOULANGER

— si notre problème admet un optimum fini (le contraire est possible), alors le maximum est atteint au bord de $\mathcal{M}$;
— le maximum semble être toujours atteint aussi en un sommet du polyèdre ?

Pour ceux qui ont une bonne imagination géométrique dans l'espace, une résolution graphique en 3 variables peut aussi être imaginée (les courbes de niveau étant des plans parallèles), ici on semble avoir les mêmes phénomènes que décrit ci-dessus : le max semble être atteint en un sommet. Ceci est une propriété essentielle, car il existe un nombre fini de sommets, et donc un problème d'optimisation semble être un problème d'optimisation combinatoire...

Le premier objectif du chapitre 2 va être d'affirmer cette propriété (voir Corollaire 2.2.5), et de caractériser ces sommets comme solutions d'un certain système d'équations linéaires (voir Théorème 2.2.8), ce qui facilitera la tâche de calculer ces points sur ordinateur. Effectivement, pour trouver les sommets dans la Figure 1.1 il suffit de calculer l'intersection entre deux droites quelconques (mais pas toutes ces intersections donnent des sommets, voir les droites $g4$ et $g5$ ou encore $g2$ et $g4$, qui sont parallèles).

Sur le dessin de Figure 1.1 on a aussi l'impression que l'on n'a pas besoin de connaître tous les sommets : à partir d'un sommet donné, on peut suivre l'arête permettant d'augmenter l'objectif pour rejoindre un des sommets voisins, jusqu'au moment où aucune augmentation n'est possible. Le sommet voisin est obtenu en remplaçant une des contraintes définissant notre sommet de départ par une autre.

Dans le paragraphe précédent on vient d'expliquer géométriquement le déroulement de l'algorithme SIMPLEX, reste à voir les détails dans le chapitre suivant

1.3.1. Exercice : *Résoudre d'une manière graphique les programmes linéaires suivants*

$$(PL_1)\begin{cases}\min(x_2-x_1)\\ -2x_1+x_2\le 2\\ x_1-2x_2\le 2\\ x_1+x_2\le 5\\ x_1\ge 0, x_2\ge 0\end{cases}\qquad (PL_2)\begin{cases}\min(-x_1)\\ 3x_1+4x_2\le 12\\ -2x_1-x_2\ge -6\\ x_1\ge 0, x_2\ge 0\end{cases}\qquad (PL_3)\begin{cases}\min(-2x_1-x_2)\\ 3x_1+4x_2\le 12\\ -2x_1-x_2\ge -6\\ x_1\ge 0, x_2\ge 0\end{cases}$$

$$(PL_4)\begin{cases}\max(3x_1+4x_2)\\ 3x_1+4x_2\le 12\\ 2x_1+x_2\le 3\\ x_1\ge 0, x_2\ge 0\end{cases}\qquad (PL_5)\begin{cases}\max(-x_1-x_2)\\ 3x_1+4x_2\ge 12\\ 2x_1+x_2\ge 4\\ x_1\ge 0, (x_2)\end{cases}\qquad (PL_6)\begin{cases}\min(-x_1-x_2)\\ 3x_1+4x_2\ge 12\\ 2x_1+x_2\ge 4\\ x_1\ge 0, (x_2)\end{cases}$$

1.3.2. Exercice : *Pour bien comprendre la différence entre* dégénérescence *et* redondance*, discuter les deux polyèdres suivants*

$$(P_1)\begin{cases}x_1+x_2+x_3\le 1\\ x_1+x_3\le 1\\ x_1,x_2,x_3\ge 0\end{cases}\qquad (P_2)\begin{cases}x_1+x_2\le 1\\ x_1+x_3\le 1\\ x_1,x_2,x_3\ge 0\end{cases}$$

I	x_1	x_2	x_3	x_4	x_5	*réal. ?*	P
$\{4,5\}$	*0*	*0*	*0*				
$\{3,5\}$	*0*	*0*		*0*			
$\{3,4\}$	*0*	*0*			*0*		
$\{2,5\}$	*0*		*0*	*0*			
$\{2,4\}$	*0*		*0*		*0*		
$\{2,3\}$	*0*			*0*	*0*		
$\{1,5\}$		*0*	*0*	*0*			
$\{1,4\}$		*0*	*0*		*0*		
$\{1,3\}$		*0*		*0*	*0*		
$\{1,2\}$			*0*	*0*	*0*		

I	x_1	x_2	x_3	x_4	x_5	*réal. ?*	P
$\{4,5\}$	*0*	*0*	*0*				
$\{3,5\}$	*0*	*0*		*0*			
$\{3,4\}$	*0*	*0*			*0*		
$\{2,5\}$	*0*		*0*	*0*			
$\{2,4\}$	*0*		*0*		*0*		
$\{2,3\}$	*0*			*0*	*0*		
$\{1,5\}$		*0*	*0*	*0*			
$\{1,4\}$		*0*	*0*		*0*		
$\{1,3\}$		*0*		*0*	*0*		
$\{1,2\}$			*0*	*0*	*0*		

1.3.3. Exercice : *Le directeur d'une station de radio veut minimiser les frais de production d'une émission qui est composée de la publicité (coût 1 E/min), de l'information (5 E/min), et de la musique (6E/min). Proposer une composition (formulation mathématique et résolution graphique) pour une heure sachant que*
(1) le temps consacré à la publicité doit être au plus égal à 15 minutes,
(2) le temps consacré aux informations doit être au plus égal à 10 minutes,
(3) le temps consacré à la musique doit être au moins égal au quadruple du temps consacré à la publicité.
Ecrire le problème mathématique et résoudre.

1.3.4. Exercice : *Une usine utilise matériaux du type A, B pour fabriquer les produits P_1, P_2 sur les deux machines M_1, M_2. Trouver un plan de production pour maximiser les bénéfices.*

	produit P_1	*produit P_2*	*disponible*	*coût*
A	*20 %*	*40 %*	*100 t*	*6E/t*
B	*80 %*	*60 %*	*500 t*	*10E/t*
sur M_1	*1h/t*	*2h/t*	*200 h*	*4E/h*
sur M_2	*3h/t*	*4h/t*	*300 h*	*3E/h*
rendement	*20 E/t*	*22 E/t*		

Ecrire le problème mathématique et résoudre.

1.3.5. Exercice : *Un fabricant d'emballages envisage l'achat de machines à plier le carton de deux types différents : le modèle A et le modèle B. Le modèle A peut plier 30 boîtes par minute et doit être alimenté et contrôlé par une personne, tandis que le modèle B peut plier 50 boîtes par minute et requiert 2 personnes. Le fabricant doit mettre en forme au moins 120 boîtes par minute et ne peut pas consacrer plus de 12 employés pour l'opération de pliage. Si une machine du modèle A coûte 15000 E et une machine du modèle B coûte 20000 E, combien de machines de chaque type doit-il acheter pour minimiser son investissement ? Ecrire le problème mathématique et résoudre.*

1.3.6. Exercice : *India Coffee Mart (ICM) commercialise du café en poudre qu'elle prépare en mélangant du café en provenance du Sud de l'Inde, de l'Assam et importé d'Afrique. Un kg de café du Sud coûte 28 roupies, un kg de café d'Assam coûte 30Rs et le café d'importation revient à 32Rs. Les ventes mensuelles de café s'élèvent à 5000 kg, mais on ne pourra disposer de plus de 1500kg de café du Sud de l'Inde, et il faut utiliser au moins 1000kg de café d'Assam et 1000kg de café d'Afrique pour le mélange. Déterminer la composition du mélange optimale du point de vue du coût. Ecrire le problème mathématique et résoudre.*

1.3.7. Exercice : *Une entreprise dispose du bien aux entrepots A (150 unités) et B (200 unités) et doit livrer ces biens aux consommateurs 1, 2, et 3 qui demandent 100, 160 et 90 unités, respectivement. Proposer un plan de transport le moins cher sachant que les coûts unitaires de transport sont donnés par le tableau suivant (en kE/unité)*

	1	2	3
A	4	1	3
B	1	9	2

1.3.8. Exercice : *Une usine utilise les matières premières du type A, B, C pour fabriquer les produits P_1, P_2, P_3, P_4 (voir tableau). Qu'est–ce qu'il faut produire pour maximiser les bénéfices ? Donner une formulation mathématique (sous forme standard) pour le tableau suivant*

disponible	*produit*	P_1	P_2	P_3	P_4
1500	*A/unité*	*5*	*1*	*9*	*12*
1000	*B/unité*	*4*	*3*	*4*	*1*
800	*C/unité*	*3*	*2*	*5*	*10*
	rendement/unité	*12*	*5*	*15*	*10*

1.4 Compléments d'Algèbre linéaire. Notations

Une liste d'indices $J = (j_1, ..., j_n)$ est un ensemble ordonné d'entiers (généralement distincts). Souvent, avec n le cardinal de J, on prend $J = (1, ..., n)$, une énumération croissante. Une sous-liste I vérifie $I \subset J$, bien souvent (mais pas toujours) écrit dans l'ordre induit par J. On note par $\overline{I}$ le complémentaire $\overline{I} = J \setminus I$ de I dans J (on enlève les élḿents $j \in I$, en gardant l'ordre induit par J). Aussi, on utilise l'intersecton et l'union de listes avec un ordre à préciser. Une union particulière est la concaténation (J_1, J_2), on ajoute les éléments de J_2 dans l'ordre à la fin de la liste J_1. Finalement, $I + s = I \cup (s)$ pour $s \in J \setminus I$ et $I - r = I \setminus (r)$ pour $r \in I$.

1.4.1. Ecriture de matrices et sous-matrices *Soit A une matrice réelle et notons L l'ensemble des indices des lignes de A (en bas) et J l'ensemble des indices des colonnes de A (en haut).*

$$\begin{aligned} \textit{écriture matrice/élément} \quad & A = A_L^J = (A_\ell^j)_{\ell\in L}^{j\in J}, \\ \textit{ℓème ligne} \quad & A_\ell = (A_\ell^j)^{j\in J} \textit{ pour } \ell \in L, \\ \textit{$j$ème colonne} \quad & A^j = (A_\ell^j)_{\ell\in L} \textit{ pour } j \in J, \\ \textit{sous-matrice} \quad & A_K^L \textit{ pour } K \subset L \textit{ et } I \subset J. \end{aligned}$$

$A = A_L^J$. On notera A^j ($j \in J$) la colonne j de la matrice et A_ℓ ($l\,\ell \in L$) la ligne ℓ de la matrice A. Soient $I = \{j_1, \cdots, j_k\} \subset J$ et $K = \{\ell_1, \ell_2, \cdots \ell_i\} \subset L$ Alors A_k^I est la sousmatrice de A formée de l'intersection des lignes de K et des colonnes de I. Par exemple, avec $L = K = J = (1, 2, 3)$, $I = (3, 1)$

$$A_L^J = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix}, \quad A_L^I = \begin{bmatrix} 0 & 1 \\ 0 & 0 \\ 3 & 0 \end{bmatrix}, \quad A_L^{(I,J\setminus I)} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 2 \\ 3 & 0 & 0 \end{bmatrix}.$$

On note $U = U_J^J$ la matrice identité (1 sur la diagonale et sinon que des zéros).

1.4.2. Sous-matrices et permutations *Avec J' une permutation de J et la permutation $\sigma : J \mapsto J'$, nous avons que $x_{J'} = Qx_J$ pour tout vecteur $x = x_J \in \mathbb{R}^J$, avec la matrice $Q = U_{J'}^J$ de permutation (et alors $Q^T = U_J^{J'} = Q^{-1}$). Par conséquent, avc $K \subset L$ et $I \subset J$, la sous-matrice A_K^I n'est pas forcément une sous-matrice principale de A_L^J mais une sous-matrice principale de*

$$A_{(K,L\setminus K)}^{(I,J\setminus I)} = \begin{bmatrix} A_K^I & A_K^{J\setminus I} \\ A_{L\setminus K}^I & A_{L\setminus K}^{J\setminus I} \end{bmatrix} = U_{(K,L\setminus K)}^L A_L^J U_J^{(I,J\setminus I)}$$

(la multiplication à gauche/droite par une matrice de permutation permute l'ordre des lignes/colonnes). Moralité : en travaillant avec les listes, pour former AB avec $A = A_L^J$ et $B = B_{L'}^{J'}$ il faudra que $J = L'$, et alors $AB = (AB)_L^{J'}$.

1.4.3. Exercice : *Soient $A = A_L^J$, $B = B_J$, et $I = (i_1, ..., i_q) \subset J$ une liste.*

(a) Montrer la formule de produit par blocs

$$A \cdot B = A^I \cdot B_I + A^{\overline{I}} B_{\overline{I}}.$$

Peut-on obtenir un résultat similaire pour les sous-matrices de $A \cdot B$?

(b) On suppose dans la suite que $K \subset L$ de sorte que A_K^I soit inversible. Donner des matrices de permutation Q, Q' de sorte que le bloc A_K^I devient une sous-matrice principale de QAQ'.

(c) Trouver une matrice $X = X_L^L$ de sorte que

$$X_K^K = U_K^K, \quad X_{\overline{K}}^{\overline{K}} = U_{\overline{K}}^{\overline{K}}, \quad X_K^{\overline{K}} = 0, \quad (X \cdot A)_{\overline{K}}^K = 0$$

(discuter d'abord le cas particulier $I = K = (1, 2, ..., \ell)$).

(d) Le complément de Schur *est défini par*

$$A^J_L/A^I_K =: S = A^{\overline{I}}_{\overline{K}} - A^I_{\overline{K}} \cdot (A^I_K)^{-1} \cdot A^{\overline{I}}_K.$$

Montrer que A est inversible ssi le complément de Schur $S = A/A^I_K$ est inversible, et que alors $(A/A^I_K)^{-1} = (A^{-1})^{\overline{K}}_{\overline{I}}$.

1.4.4. Exercice : *Soit $B = B^J_L$ une matrice carrée, avec $B^{\overline{I}}_K = 0$, ou $I \subset J$, $K \subset L$, $|I| = |K|$, et $\overline{I} := J \setminus I$, $\overline{K} := L \setminus K$. Donner une CNS en fonction de B^I_K et de $B^{\overline{I}}_{\overline{K}}$ pour que B soit inversible, et déterminer B^{-1}.*

1.4.5. Exercice : *Soient $A = A^J_L$, et A^I une sous-matrice inversible. On cherche $r \in I$ et $s \in J \setminus I$ de sorte que $A^{I'}$ soit inversible, $I' := I + s - r$. Notons $T(I) := (A^I)^{-1} \cdot A$. Vérifier que $T(I)^{I'}$ est une matrice triangulaire par blocs. En déduire l'équivalence*

$$A^{I'} \text{ inversible} \iff T(I)^{I'} \text{ inversible} \iff T(I)^s_r \neq 0$$

et dans ce cas, calculer l'inverse de $T(I)^{I'}$. Donner une expression pour l'inverse de $A^{I'}$.

Soient

$$A = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix}, \quad I = (1, 2, 3).$$

Trouver un indice r pour $s = 5$ (un indice s si on fixe $r = 2$) de sorte que $A^{I'}$ soit inversible.

1.4.6. Exercice : *Soit B une matrice inversible. En utilisant le complément de Schur d'une matrice plus grande, montrer que $B + Y \cdot X$ est inversible ssi $U + X \cdot B^{-1} \cdot Y$ est inversible, et que dans ce cas (formule de* Sherman–Morrison*)*

$$(B + Y \cdot X)^{-1} = B^{-1} - B^{-1} \cdot Y \cdot (U + X \cdot B^{-1} \cdot Y)^{-1} \cdot X \cdot B^{-1}.$$

Chapitre 2

Optimisation linéaire : La méthode Simplex

2.1 Le problème de la programmation linéaire

2.1.1. Définition : Position du problème
On considère dans $\mathbb{R}^n$ *le problème d'optimisation linéaire*

$$(PL)\begin{cases} \textit{maximiser } fx \textit{ sous les contraintes} \\ k \in K^0 : A_k x = a_k \\ k \in K^+ : A_k x \leq a_k \\ k \in K^- : A_k x \geq a_k \quad \textit{(par exemple (*) } \forall k : x_k \geq 0\textit{)} \end{cases}$$

où $a_k \in \mathbb{R}$, *et* f, A_k *pour* $k \in K = K^0 \cup K^- \cup K^+$ *sont les vecteurs lignes (éléments de* $(\mathbb{R}^n)^*$*).*
Terminologie :

- $x \in \mathbb{R}^n$ *est dit solution réalisable si il vérifie les contraintes ;*
- $\widehat{x}$ *est dit solution optimale (avec* $f\widehat{x}$ *valeur optimale) si* $f\widehat{x} \geq \max\{f(x) : x \text{ sol. réal.}\}$ *et* $\widehat{x}$ *est solution réalisable ;*
- *le vecteur* f *est dit fonction économique ou objectif ;*
- (PL) *est dit sous forme* canonique *si* $K^0 = \emptyset$, *et les contraintes de* K^- *sont réduites à* $(*)$ *;*
- (PL) *est dit sous forme* standard *si* $K^+ = \emptyset$, *et les contraintes de* K^- *sont réduites à* $(*)$ *;*
- (PL) *est dit sous forme* standard généralisé *ou* aux variables doublement bornées *si les contraintes de* $K^- = K^+$ *sont réduites à* $\forall k \in K^+ : \mathbb{R} \cup \{-\infty\} \ni b_k \leq x_k \leq c_k \in \mathbb{R} \cup \{+\infty\}$.

2.1.2. Remarque
Notons l'ensemble des solutions réalisables par P. *La notation symbolique* $\max\{f(x) : x \in P\}$ *inclut les cas*

$$(i) \quad P = \emptyset \quad \textit{(le problème est impossible) ;}$$
$$(ii) \quad P \neq \emptyset, \sup\{f(x) : x \in P\} = +\infty \quad \textit{(l'optimum vaut } +\infty\textit{).}$$

On montrera ultérieurement qu'autrement on trouve[1] *une solution optimale* $\widehat{x}$*, voir le corollaire 2.2.5.*

Il sera utile d'adopter une écriture matricielle, ici $x \leq y$ pour $x, y \in \mathbb{R}^n$ signifie que $\forall k$: $x_k \leq y_k$ ($x < y$ n'est pas défini).

2.1.3. Définition : équivalence
Deux problèmes d'optimisation

$$(PL) : \max\{f(x) : x \in P\}, \qquad (\widetilde{PL}) : \max\{\widetilde{f}(\widetilde{x}) : \widetilde{x} \in \widetilde{P}\}$$

sont dits équivalents si $\exists \phi : P \mapsto \widetilde{P}$ *et* $\exists \widetilde{\phi} : \widetilde{P} \mapsto P$ *telles que*

$$\forall x \in P : \widetilde{f}(\phi(x)) \geq f(x), \qquad \forall \widetilde{x} \in \widetilde{P} : f(\widetilde{\phi}(\widetilde{x})) \geq \widetilde{f}(\widetilde{x}).$$

On laissera comme exercice la tâche de démontrer que (PL) et $(\widetilde{PL})$ ont la même valeur optimale, et que, pour toute solution optimale x de (PL), $\phi(x)$ est solution optimale $(\widetilde{PL})$.

2.1.4. Lemme
Tout programme linéaire (PL) *est équivalent à un programme linéaire sous forme standard (ou sous forme canonique).*

Démonstration. La transformation en forme standard (canonique) se fait en appliquant successivement l'une des quatre opérations expliquées ci-dessous :
(i) Passage d'une variable réelle à des variables astreintes d'être positives : sans perte de généralité, supposons que $x_1 \in \mathbb{R}$. On fait la substitution[2] $x_1 = x_1' - x_1''$ avec $x_1', x_1'' \geq 0$.
(ii) Passage de K^+, K^- à K^0 : par l'introduction d'une variable d'écart[3]

$$\begin{aligned} A_k x \leq a_k \quad &\Longleftrightarrow \quad A_k x + y = a_k, y \geq 0, \\ A_k x \geq a_k \quad &\Longleftrightarrow \quad A_k x - y = a_k, y \geq 0. \end{aligned}$$

(iii) Passage de K^- à K^+ : par multiplication[4] par -1.
(iv) Passage de K^0 à K^+ :

$$A_k x = a_k \quad \Longleftrightarrow \quad A_k x \leq a_k, (-A_k)x \leq (-a_k).$$

□

1. Si $f \neq 0$ alors forcément $\widehat{x}$ se trouve sur le bord ∂P, car avec $\phi(x) = fx$ nous avons $\phi \in \mathcal{C}^\infty$ et $\nabla\phi(\widehat{x}) = f \neq 0$.

2. Le nouveau programme $(\widetilde{PL})$ aura par exemple la variable $\widetilde{x} = (x_1', x_1'', x_2, ..., x_n)^T$ et l'objectif $\widetilde{f} = (f^1, -f^1, f^2, ..., f^n)$. Les applications $\phi, \widetilde{\phi}$ sont

$$\widetilde{\phi}((x_1', x_1'', x_2, ..., x_n)^T) = (x_1' - x_1'', x_2, ..., x_n)^T, \quad \phi((x_1, x_2, ..., x_n)^T) = (\max(0, x_1), -\min(0, x_1), x_2, ..., x_n)^T,$$

avec $\widetilde{f}\phi(x) = f(x)$, $f\widetilde{\phi}(\widetilde{x}) = \widetilde{f}\widetilde{x}$.

3. Le nouveau programme $(\widetilde{PL})$ pour K^+ aura par exemple la variable $\widetilde{x} = (x_1, x_2, ..., x_n, y)^T$ et l'objectif $\widetilde{f} = (f^1, f^2, ..., f^n, 0)$. Les applications $\phi, \widetilde{\phi}$ sont

$$\widetilde{\phi}((x_1, x_2, ..., x_n, y)^T) = (x_1, x_2, ..., x_n)^T, \quad \phi(x) = (x^T, a_k - A_k x)^T,$$

avec $\widetilde{f}\phi(x) = f(x)$, $f\widetilde{\phi}(\widetilde{x}) = \widetilde{f}\widetilde{x}$.

4. Pour les derniers deux transformations on n'aura pas besoin d'une telle équivalence car leur système d'inégalités est équivalent au sens classique.

2.1.5. Exemples

$$\left\{\begin{array}{l}\max(fx)\\ Ax\le a\\ x\ge 0\end{array}\right\}\Longleftrightarrow\left\{\begin{array}{l}\max(fx)\\ Ax+y=a\\ x,y\ge 0\end{array}\right\}\Longleftrightarrow\left\{\begin{array}{l}\max(f,0)\binom{x}{y}\\ (A,U)\binom{x}{y}=a\\ \binom{x}{y}\ge 0\end{array}\right\}$$

et

$$\left\{\begin{array}{l}\max(fx)\\ Ax=a\\ x\ge 0\end{array}\right\}\Longleftrightarrow\left\{\begin{array}{l}\max(fx)\\ \binom{A}{-A}x\le\binom{a}{-a}\\ x\ge 0\end{array}\right\}$$

et

$$\left\{\begin{array}{l}\min(fx)\\ Ax\ge a\end{array}\right\}\Longleftrightarrow\left\{\begin{array}{l}-\max(-fx)\\ Ax\ge a\end{array}\right\}\Longleftrightarrow\left\{\begin{array}{l}-\max(f,-f)\binom{x'}{x''}\\ (A,-A)\binom{x'}{x''}\ge a\\ x',x''\ge 0\end{array}\right\}\Longleftrightarrow\left\{\begin{array}{l}-\max(f,-f,0)\binom{x'}{x''}\\ (A,-A,-U)\binom{x'}{x''}=a\\ \binom{x'}{x''}\ge 0\end{array}\right\}$$

2.1.6. Remarques

(a) L'occurrence de la contrainte $x\ge 0$ est souvent naturelle dans les applications.
(b) Le cas d'une contrainte $x\ge b$ ou $x\le c$ peut être transformé en $x'\ge 0$ par substitution.
(c) Algorithmiquement, le passage d'une variable non astreinte de signe à des variables positives n'est pas efficace. Mieux vaut travailler directement avec la forme standard généralisée (avec $b_k=-\infty$ et/ou $c_k=+\infty$).

2.1.7. Exercice

Transformer en problème d'optimisation linéaire et donner la forme standard des problèmes (non linéaires) suivants

$$(PL_1)\left\{\begin{array}{l}\min(-x_1+7x_2)\\ x_1+2x_2\le 3\\ x_1+2x_2\ge -5\\ -x_1+x_2\ge -2\\ x_1\ge 0\end{array}\right.\qquad(PL_2)\left\{\begin{array}{l}\min(|x_1-x_2-3|+x_2)\\ |x_1+x_2-2|\le 1\\ x_2\le 2\\ x_1,x_2\ge 0\end{array}\right.$$

$$(PL_3)\left\{\begin{array}{l}\min(\max\{2x_1+x_2,x_1+2x_2\})\\ \max(x_1-x_2,x_2)\le 2x_1+1\\ x_1,x_2\ge 0\end{array}\right.$$

N.B. : $\max(u,v)=\frac{1}{2}(u+v+|u-v|)$.

2.1.8. Exercice

Les données (x_k,y_k), $k=0,1,..,N$, $N\ge 2$ obtenues par une expérience suggèrent une loi de formation $y=f(x)$ avec $f(x)=ax+b$. Comment choisir a,b si on veut minimiser $\sum_{k=0}^{N}|y_k-f(x_k)|$? Transformer en problème d'optimisation linéaire.

2.2 Les polyèdres (intermezzo sur la géométrie du $\mathbb{R}^n$)

2.2.1. Définition : demi-espace, hyper-plan, droite et demi-droite
Pour un vecteur ligne $a \in (\mathbb{R}^n)^*$ et $b \in \mathbb{R}$, on définit un demi-espace (et un hyper-plan, respectivement) par

$$H^+(a,b) = \{x \in \mathbb{R}^n : ax \geq b\}$$
$$H(a,b) = \{x \in \mathbb{R}^n : ax = b\} = \partial H^+(a,b).$$

Pour $c \in \mathbb{R}^n$ et $d \in \mathbb{R}^n \setminus \{0\}$ on définit une droite (et une demi-droite, respectivement) par

$$D(c,d) = \{c + \theta d : \theta \in \mathbb{R}\},$$
$$D^+(c,d) = \{c + \theta d : \theta \in \mathbb{R}, \theta \geq 0\}.$$

2.2.2. Définition : polyèdre, support, point extrême, cône asymptotique
On appelle polyèdre *toute intersection d'un nombre fini de demi-espaces*

$$P = \bigcap_{k \in K} H^+(A_k, b_k) = \{x \in \mathbb{R}^n : Ax \geq b\} \quad \textit{avec} \quad (A,b) = (A_k, b_k)_{k \in K}.$$

Le support *d'un $x \in P$ est donné par* $\mathrm{supp}(x) = \{k \in K : A_k x \neq b_k\}$. *On appelle $x \in P$*
— *un* point extrême *de P si $x = (y+z)/2$ avec $y, z \in P$ implique que $x = y = z$;*
— *un point à* support minimal *si $y \in P$ et $\mathrm{supp}(y) \subset \mathrm{supp}(x)$ implique que $x = y$.*
Le cône asymptotique *de P est défini par $C(P) = \{x \in \mathbb{R}^n : Ax \geq 0\}$.*

Notons qu'un polyèdre P est convexe (c'est-à-dire, $x, y \in P$ implique que $[x,y] = \{\theta x + (1-\theta)y : \theta \in [0,1]\} \subset P$) et fermé. L'ensemble $C(P)$ est également un polyèdre, mais aussi un cône (c'est-à-dire, $x \in C(P)$, $\theta \in (0, +\infty)$ implique que $\theta x \in C(P)$) pointé ($0 \in C(P)$) , il peut être réduit à $\{0\}$ (voir Exercice 2.2.11). La caractérisation d'un point extrême est géométrique : si une direction $d \neq 0$ permet de rester dans P (c-à-d, $x + d \in P$) alors la direction opposée nous fait sortir du polyèdre (c-à-d, $x - d \notin P$), ce qui correspond bien à la notion d'un sommet en $\mathbb{R}^2$ ou en $\mathbb{R}^3$. La notion de support est plutôt algébrique : x se trouve dans l'intersection des hyperplans $H(A_k, b_b)$ d'indice $k \in K \setminus \mathit{supp}(x)$. Les points à support minimal sont alors déterminés par l'intersection d'un nombre maximum de ces hyperplans ; on montrera dans le lemme 2.2.4 que ceci donne également lieu à une caractérisation des sommets d'un polyèdre du $\mathbb{R}^n$.

L'importance des points extrêmes (appelés sommets au §1) devient claire à partir du résultat suivant (et de son corollaire 2.2.5).

2.2.3. Théorème
Soit $P = \{x \in \mathbb{R}^n : Ax \geq b\} \neq \emptyset$, ne contenant aucune droite, et notons par $E(P)$ l'ensemble des combinaisons convexes des points extrêmes de P. Alors $P = E(P) + C(P)$.

Le lecteur intéressé pourrait montrer le complément suivant que $P = E(P)$ si et seulement si P est compact. La preuve d'une des inclusions du théorème 2.2.3 est un peu compliquée, on raisonnera par récurrence sur la taille du support d'un élément de P. Dans cette optique, nous proposons d'abord de montrer quelques résultats techniques.

2.2.4. Lemme
(a) Soient $x, y \in P$, $x \neq y$ et $\mathrm{supp}(y) \subset \mathrm{supp}(x)$. Si $D^+(x, x-y) \subset P$ alors $x - y \in C(P)$. Sinon, nous avons $D^+(x, x-y) \cap P = [x, \widehat{x}]$, avec $\mathrm{supp}(\widehat{x}) \subsetneq \mathrm{supp}(x)$.
(b) Un point $x \in P$ à support minimal est un point extrême de P.
(c) Un point extrême de P est un point à support minimal.
(d) Soit $P \neq \emptyset$. P admet de points extrêmes si et seulement si P ne contient pas de droite.
(e) Tout polyèdre admet un nombre fini de points extrêmes.
(f) Pour tout $\theta \in [0,1]$: $\theta E(P) + (1-\theta)E(P) \subset E(P)$, et $C(P) + C(P) \subset C(P)$.

Démonstration. **(a)** Soit $\theta \in [0, +\infty)$, alors

$$t(\theta) = x + \theta(x-y) \in P \iff Ax - b \geq -\theta A(x-y) \iff \forall k : \underbrace{A_k x - b_k}_{\geq 0} \geq -\theta A_k(x-y). \quad (2.1)$$

Par conséquent, $D^+(x, x-y) \subset P$ implique que $\forall k : A_k(x-y) \geq 0$, d'où $x - y \in C(P)$. Supposons maintenant $D^+(x, x-y) \not\subset P$, c'est à dire, $L := \{k : A_k(x-y) < 0\} \neq \emptyset$. Notons que $L \subset \mathit{supp}(x)$, car sinon $\exists k \in L \setminus \mathit{supp}(x)$, c'est-à-dire, $0 = A_k x - b_k = A_k(x-y) + (A_k y - b)$, d'où $y \in \mathit{supp}(y) \setminus \mathit{supp}(x)$, en contradiction avec nos hypothèses. Nous posons

$$\theta_{\min} := \min\{-(\underbrace{A_k x - b_k}_{>0})/(\underbrace{A_k(x-y)}_{<0}) : k \in L\},$$

alors $\theta_{\min} > 0$ par construction, et comme seulement les indices $k \in L$ posent des difficultés dans (2.1), on vérifie aisément que $P \cap D^+(x, x-y) = [x, t(\theta_{\min})]$. Plus précisément, si le minimum dans la définition de $\theta_{\min}$ est atteint pour l'indice $\kappa \in L$, on obtient $A_\kappa t(\theta_{\min}) - b_\kappa = 0$, d'où $\kappa \notin \mathit{supp}(t(\theta_{\min}))$. De l'autre coté, $A_k t(\theta_{\min}) - b_k = (1 + \theta_{\min})(A_k x - b_k) - \theta_{\min}(A_k y - b_k)$, et donc $\mathit{supp}(t(\theta_{\min})) \subset \mathit{supp}(x) \cap \mathit{supp}(y) = \mathit{supp}(x)$, ce qu'il fallait démontrer.
(b) Soit $x \in P$ à support minimal, et $x = (y+z)/2$ avec $y, z \in P$, alors pour tout $k \notin \mathit{supp}(x)$

$$0 = A_k x - b_k = \tfrac{1}{2}(\underbrace{A_k y - b_k}_{\geq 0}) + \tfrac{1}{2}(\underbrace{A_k z - b_k}_{\geq 0}),$$

d'où $k \notin \mathit{supp}(y) \cup \mathit{supp}(z)$. Par conséquent, $\mathit{supp}(y) \subset \mathit{supp}(x)$, d'où $x = y (= z)$ par définition d'un point à support minimal.
(c) Soit $x \in P$, pas à support minimal. Par définition, ceci implique qu'il existe $y \in P$ avec $\mathit{supp}(y) \subset \mathit{supp}(x)$ et $y \neq x$. D'après (a), $\exists \widetilde{x} \in D^+(x, x-y) \cap P$, $\widetilde{\neq} x$, et donc par convexité $[y, \widetilde{x}] \subset P$. Un petit dessin montre que x n'est alors pas non plus un point extrême de P.
(d) Si $x \in P$ et $D(\widetilde{x}, d) \subset P$ alors $Ay = 0$, d'où $D(x, d) \subset P$. En considérant les points $x \pm d$, on conclut que x n'est pas point extrême. Réciproquement, supposons que P ne contient pas de droite. Comme P est non vide et l'inclusion est un ordre partiel, on trouve toujours un $x \in P$ avec $\forall y \in P : \mathit{supp}(y) \subset \mathit{supp}(x)$ implique que $\mathit{supp}(y) = \mathit{supp}(x)$. Il suffit de montrer qu'un tel $y \in P$ avec $\mathit{supp}(y) = \mathit{supp}(x)$ coïncide forcément avec x, car dans ce cas x serait un point à support minimal (et alors point extrême). Sinon, $x \neq y$, et d'après (a), le cas $D^+(x, x-y) \not\subset P$ est impossible car il n'y a pas de points à support strictement plus petit. Par conséquent, $D^+(x, x-y) \subset P$, et aussi $D^+(y, y-x) \subset P$ en échangeant les rôles de x et de y. Finalement, $[x, y] \subset P$ par convexité, ce qui montre que $D(x, x-y) \subset P$, une contradiction.
(e) Par définition, les supports de deux points différents à support minimal sont différents.

Comme il existe seulement au plus 2^m sous-ensembles finis pour m contraintes, on en déduit qu'il y a un nombre limité de points à support minimal, et il suffit d'appliquer (c).
(f) Laissé à titre d'exercice. □

Démonstration de 2.2.3. Pour montrer $E(P) + C(P) \subset P$, soit $y \in C(P)$ et $x \in E(P)$, c'est-à-dire,

$$x = \sum_j \lambda_j x^j,\ x^j \in P, \lambda_j \geq 0, \sum_j \lambda_j = 1$$

et donc $Ay \geq 0$, $Ax^j - b \geq 0$, et finalement

$$A(x+y) - b = Ay + \sum_j \lambda_j (Ax^j - b) \geq 0.$$

L'inclusion réciproque $P \subset E(P) + C(P)$ est montré par récurrence sur la taille du support d'un $x \in P$.
(a) Si x est à support minimal alors $x = x + 0 \in E(P) + C(P)$ d'après 2.2.4(b).
(b) Sinon, il existe un $y \in P$ avec $supp(y) \subset supp(x)$ et $y \neq x$.
(b1) Si $supp(y) \neq supp(x)$, alors par hypothèse de récurrence $y \in E(P) + C(P)$.
(b1.i) Si $D^+(x, x-y) \subset P$ alors $x - y \in C(P)$ d'après 2.2.4(a), et alors $x = y + (x-y) \in E(P) + C(P) + C(P) \subset E(P) + C(P)$ d'après 2.2.4(f).
(b1.ii) Si $D^+(x, x-y) \not\subset P$ alors $D^+(x, x-y) \cap P = [x, \widetilde{x}]$ avec $supp(\widetilde{x}) \subsetneq supp(x)$, et par hypothèse de récurrence $\widetilde{x} \in E(P) + C(P)$. Finalement, par construction $\exists \theta \in (0,1)$ avec

$$x = \theta y + (1-\theta)\widetilde{x} \in \theta(E(P) + C(P)) + (1-\theta)(E(P) + C(P)) \subset E(P) + C(P).$$

(b2) Le dernier cas à examiner est $supp(x) = supp(y)$, ici (comme dans la preuve de 2.2.4(d)) on applique 2.2.4(a) deux fois.
(b2.i) Si $D^+(x, x-y) \subset P$ et $D^+(y, y-x) \subset P$ alors aussi $D(x, x-y) \subset P$, ce qui a été exclus par hypothèse.
(b2.ii) Si $D^+(x, x-y) \subset P$ et $D^+(y, y-x) \not\subset P$ alors $x - y \in C(P)$ d'après 2.2.4(a), et avec $D^+(y, y-x) \cap P =: [y, \widetilde{y}]$ on trouve que $x - \widetilde{y} \in C(P)$ (car multiple positif de $x - y$), et $\widetilde{y} \in E(P) + C(P)$ par hypothèse de récurrence. Finalement, en observant que $x = (x - \widetilde{y}) + \widetilde{y}$ on conclut comme dans cas (b1.i).
(b2.iii) Si $D^+(x, x-y) \not\subset P$ et $D^+(y, y-x) \subset P$: voir cas (b2.ii).
(b2.iv) Si $D^+(x, x-y) \not\subset P$ et $D^+(y, y-x) \not\subset P$ alors avec $D^+(x, x-y) \cap P =: [x, \widetilde{x}]$, $D^+(y, y-x) \cap P =: [y, \widetilde{y}]$ on trouve que $\widetilde{x}$ et $\widetilde{y}$ ont un support strictement plus petit que x, et que $x \in [\widetilde{x}, \widetilde{y}]$, ce qui permet de conclure comme dans cas (b1.ii). □

Notons que le théorème 2.2.3 comporte entre autres le fait bien connu que tout point à l'intérieur d'un triangle, d'un carré, d'une pyramide, etc, admet des coordonnées baricentriques. Une conséquence d'importance majeure mentionnée déjà dans l'introduction du § 1.3 est donnée ci-dessous.

2.2.5. Corollaire
Soit $P = \{x \in \mathbb{R}^n : Ax \geq b\} \neq \emptyset$, ne contenant aucune droite. Si $\exists y \in C(P)$ avec $fy > 0$ alors

$$\sup\{fx : x \in P\} = +\infty,$$

sinon, $\sup\{fx : x \in P\}$ *est fini, et atteint en un point extrême de P.*

Démonstration. Supposons dans un premier temps que $\exists y \in C(P)$ avec $fy > 0$. P étant non vide comporte au moins un élément, noté par x_0. Par définition de $C(P)$, nous obtenons $D^+(x_0, y) \subset P$, et alors

$$\sup\{fx : x \in P\} \geq \sup\{fx : x \in D^+(x_0, y)\} = fx_0 + \sup\{\theta fy : \theta \in [0, +\infty)\} = +\infty,$$

la propriété recherchée. Dans le cas contraire, nous posons $M = \sup\{fx : x \text{ point extrême de } P\}$, le supremum étant bien entendu un maximum car il y a d'après le lemme 2.2.4(e) un nombre fini de candidats à examiner. D'après le le théorème 2.2.3

$$\begin{aligned} M &\leq \sup\{fx : x \in P\} = \sup\{fx + \underbrace{fy}_{\leq 0} : x \in E(P), y \in C(P)\} \leq \sup\{fx : x \in E(P)\} \\ &= \sup\{f \sum_{j=1}^{k} \lambda^j x^j : k \geq 0, \lambda^j \geq 0, \sum_{j=1}^{k} \lambda^j = 1, x^j \text{ points extrêmes de } P\} \\ &= \sup\{\sum_{j=1}^{k} \lambda^j \underbrace{fx^j}_{\leq M} : k \geq 0, \lambda^j \geq 0, \sum_{j=1}^{k} \lambda^j = 1, x^j \text{ points extrêmes de } P\} \leq M, \end{aligned}$$

ce qui donne donc égalité partout. □

On sait d'après 2.2.4(e) qu'un polyèdre admet un nombre fini de points extrêmes. Le corollaire 2.2.5 permet de conclure que tout problème borné d'optimisation linéaire est un problème d'optimisation combinatoire (maximum d'un ensemble fini). Il nous faut alors une façon simple de caractériser et d'énumérer les points extrêmes. Dans la suite on considère, sans perte de la généralité, les problèmes sous forme standard généralisé

$$(PL) \begin{cases} \max fx \\ Ax = a \\ x \geq b, x \leq c \end{cases} \qquad \text{avec } A = A_L^J, \text{rang}(A) = |L| = m \leq |J| = n,$$

et $\forall j \in J : \mathbb{R} \cup \{-\infty\} \ni b_j \leq c_l \in \mathbb{R} \cup \{+\infty\}$. Nous rappelons le lecteur certaines conventions concernant l'énumération des lignes et colonnes des matrices par des listes, voir le chapitre 1.4. Rappelons également que (PL) comporte la forme standard en posant $b = 0$ et $\forall j \in J : c_j = +\infty$.

2.2.6. Exercice : *Soient $P = \{x \in \mathbb{R}^n : Ax = a, x \geq 0\}$ et $x \in P$.*
(a) *Montrer que $x + \theta z \in P$ pour un $\theta \in \mathbb{R} \setminus \{0\}$ seulement si $z \in Ker\,(A)$. Cette condition, est–elle aussi suffisante ?*
(b) *Montrer que $D(x, z) \not\subset P$ pour tout $z \neq 0$.*
(d) *Montrer que P n'est pas borné ssi il existe $z \in Ker\,(A) \setminus \{0\}$ avec $z \geq 0$.*

2.2.7. Définition : base, point de base
On appelle base de (PL) *toute partition (I, B_-, B_+) de J de sorte que*

$$\begin{aligned} &\forall j \in B_- : b_j > -\infty, \quad \forall j \in B_+ : c_j < +\infty, \\ &A^I \text{ est inversible (ce qui implique que } |I| = |L| = m). \end{aligned}$$

On note $x = x(I, B_-, B_+)$ (dit point de base*) l'unique point vérifiant*

$$x_{B_-} = b_{B_-}, \quad x_{B_+} = c_{B_+}, \quad Ax = a$$

ce qui implique que $x_I = (A^I)^{-1}(a - A^{B_-}b_{B_-} - A^{B_+}c_{B_+})$. *La base* (I, B_-, B_+) *est dite* réalisable *si* $x(I, B_-, B_+) \in P$ *(c'est-à-dire,* $\forall j \in I : b_j \leq x(I, B_-, B_+)_j \leq c_j$.

On remarquera que, pour un problème sous forme standard, la définition 2.2.7 prend une forme bien plus simple : ici B_+ est toujours vide, et par conséquent $B_- = \overline{I} := J \setminus I$, le complémentaire de I dans J. On dira donc pour un problème sous forme standard que I est une base (éventuellement réalisable) si A^I inversible, et on écrira $x(I) = x(I, J \setminus I, \emptyset)$, avec $x(I)_{\overline{I}} = 0$ et $x(I)_I = (A^I)^{-1}a$.

2.2.8. Théorème
Pour le polyèdre P *de* (PL)*, l'ensemble des points extrêmes coïncide avec avec l'ensemble des points de base réalisable.*

Démonstration. Soit $x = x(I, B_-, B_+)$ un point de base réalisable, et $y, z \in P$ avec $x = (y+z)/2$. Pour tout $j \in B_-$ nous avons

$$0 = x_j - b_j = \frac{y_j - b_j}{2} + \frac{z_j - b_j}{2},$$

mais une somme de deux termes positives s'annule seulement si les deux termes s'annulent, d'où $\forall j \in B_- : y_j = b_j$. De même, on montre $\forall j \in B_+ : y_j = c_j$. Comme de plus A^I est inversible par définition d'une base, on obtient

$$y_I = (A^I)^{-1}(a - A^{B_+}c_{B_+} - A^{B_-}b_{B_-})$$

et alors $x = y$ ce qui montre que x est point extrême.
Réciproquement, soit $x \in P$ un point extrême, et

$$K_- = \{j \in J : x_j = b_j (> -\infty)\}, \quad K_+ = \{j \in J : x_j = c_j (< +\infty)\},$$
$$K_0 = \{j \in J : b_j < x_j < c_j\}.$$

Montrons par absurde que $\{A^j : j \in K_0\}$ sont libres. Sinon, $\exists d \neq 0$ avec $d_{J \setminus K_O} = 0$ et $Ad = 0$ et alors $\forall \theta \in R : A(x + \theta d) = 0$. Comme de plus par construction de K_0 et d nous avons que

$$\exists \theta_0 > 0 : \forall \theta \in [-\theta_0, \theta_0] \, \forall j \in J : \qquad (x + \theta d)_j \in [b_j, c_j]$$

nous concluons que $x \pm \theta_0 d \in P$, en contradiction avec le fait que x est point extrême.
Comme $\text{span}(\{A^j : j \in J\}) = \mathbb{R}^m$ par hypothèse, un théorème d'algèbre linéaire nous affirme que l'on peut compléter un système libre pour trouver une base : $\exists I \subset J$ avec $K_0 \subset I$ de sorte que A^I est inversible. Par conséquent, $(I, B_+, B_-) = (I, K_+ \setminus I, K_- \setminus I)$ forme une base. De plus, $x_{B_-} = b_{B_-}$ et $x_{B_+} = c_{B_+}$ et $Ax = a$, ce qui implique que $x = x(I, B_-, B_+)$ est un point de base réalisable. □

2.2.9. Remarque
On peut avoir une dégénérescence *d'un point extrême* $\hat{x}$*, ce qui signifie que* $\hat{x}$ *est point de base pour plusieurs bases distinctes. Géométriquement, ceci veut dire que* $\{\hat{x}\}$ *est l'intersection de* $|K_-| + |K_+| + |L| = |J| - |K_0| + |I|$ *hyperplans, avec* $-|K_0| + |I| > 0$ *(voir les notations de la preuve de 2.2.8).*

2.2.10. Exercice : *On rappelle que le cône asymptotique du polyèdre* $P = \{x \in \mathbb{R}^n : Ax = a, x \geq 0\}$ *est défini par* $C(P) = \{x \in \mathbb{R}^n : Ax = 0, \quad x \geq 0\}$. *Considérons l'hyperplan* $H = \{x \in \mathbb{R}^n : ex = 1\}$, $e = (1, ..., 1)$.

(a) *Montrer que les propriétés suivantes sont équivalentes*

$$(i) \quad C(P) = \{0\} \qquad (ii) \quad C(P) \cap H = \emptyset.$$

(b) *Soit* $C(P) \neq \{0\}$. *Montrer qu'il existe* $z_1, ..., z_p \in \mathbb{R}^n$ *t.q.*

$$H \cap C(P) = \Big\{ \sum_{j=1}^{p} \alpha_j z_j : \alpha_j \geq 0, \sum_{j=1}^{p} \alpha_j = 1 \Big\}.$$

En déduire que

$$C(P) = \Big\{ \sum_{j=1}^{p} \alpha_j z_j : \alpha_j \geq 0 \Big\}.$$

Quel ensemble de systèmes d'équations faut-il résoudre pour trouver ces vecteurs z_j *?*

(c) *Montrer que pour notre polyèdre* P *on trouve des entiers* $p, q \geq 0$ *et* $y_1, ..., y_{p+q} \in \mathbb{R}^n$ *t.q.*

$$P = \Big\{ \sum_{j=1}^{p+q} \alpha_j y_j : \alpha_j \geq 0, \sum_{j=1}^{q} \alpha_j = 1 \Big\}.$$

2.2.11. Exercice :

(a) Montrer que les deux ensembles

$$P = \{x : \quad Ax \geq b, \quad x \geq 0\} \quad et \quad \{x^* : \quad (A, -U)x^* = b, \quad x^* \geq 0\}$$

sont en bijection. En particulier il y a des bijections entre les points extrêmes (même résultat pour des variables doublement bornées).

(b) Déterminer le support et le cône asymptotique pour les polyèdres de la forme

$$P = \{x : \quad Ax = b, \quad x \geq 0\}$$

$$P^* = \{x : \quad Ax = a, \quad b \leq x \leq c\}, \quad b_j, c_j \in \mathbb{R}$$

(c) Soit P *le polyèdre non vide*

$$P = \{x \in \mathbb{R}^n : \quad Ax \geq b\}$$

Montrer que :

(c.1) P *admet des points extrêmes ssi* P *ne contient pas de droite ;*

(c.2) il existe un nombre fini de points à support minimal ;

(c.3) P *est compact ssi le cône asymptotique* $C(P)$ *est réduit à* $\{0\}$.

(c.4) si $f(x)$ *est une fonction convexe et* $C(P) = \{0\}$ *alors* $sup\{f(x), \quad x \in P\}$ *est atteint pour un point extrême de* P.

2.3 La méthode Simplex

Dans ce chapitre on se donne un programme linéaire sous forme standard généralisé

$$(PL)\left\{\begin{array}{l}\max fx\\ Ax=a\\ x\geq b, x\leq c\end{array}\right.$$

avec les hypothèses habituelles $A=A_L^J$, de rang $|L|=m\leq|J|=n$, et $\forall j\in J:\mathbb{R}\cup\{-\infty\}\ni b_j\leq c_l\in\mathbb{R}\cup\{+\infty\}$. On a vu au § 2.2 que, si (PL) admet un optimum fini, l'optimum est atteint en un point extrême du polyèdre (voir 2.2.5), qui peut être représenté comme point de base réalisable (voir 2.2.8). Voici un critère simple pour décider si un tel point est une solution optimale.

2.3.1. Théorème : critère suffisant d'optimalité (CSO)
Soit (I,B_-,B_+) une base réalisable, et $d(I):=f-f^I(A^I)^{-1}A$. Si

$$d(I)^j\left\{\begin{array}{ll}\leq 0 & \text{pour } j\in B_-\\ \geq 0 & \text{pour } j\in B_+\end{array}\right. \tag{2.2}$$

alors $x(I,B_-,B_+)$ est solution optimale de (PL).

Démonstration. On se donne un $x\in P$, c'est-à-dire, $Ax=a$, $x_{B_-}\geq b_{B_-}$, $x_{B_+}\leq c_{B_+}$, et alors

$$\begin{array}{rl} & fx(I,B_-,B_+)-fx\\ = & f^{B_-}\Big(b_{B_-}-x_{B_-}\Big)+f^{B_+}\Big(c_{B_+}-x_{B_+}\Big)\\ & +f^I(A^I)^{-1}\Big((a-A^{B_+}c_{B_+}-A^{B_-}b_{B_-})-(a-A^{B_+}x_{B_+}-A^{B_-}x_{B_-})\Big)\\ = & f^{B_-}\Big(b_{B_-}-x_{B_-}\Big)+f^{B_+}\Big(c_{B_+}-x_{B_+}\Big)\\ & +f^I(A^I)^{-1}A^{B_+}\Big(-c_{B_+}+x_{B_+}\Big)+f^I(A^I)^{-1}A^{B_-}\Big(-b_{B_-}+x_{B_+}\Big)\\ = & \underbrace{d(I)^{B_-}}_{\leq 0}\Big(\underbrace{b_{B_-}-x_{B_-}}_{\leq 0}\Big)+\underbrace{d(I)^{B_+}}_{\geq 0}\Big(\underbrace{c_{B_+}-x_{B_+}}_{\geq 0}\Big)\geq 0.\end{array}$$

Donc $fx(I,B_-,B_+)\geq fx$ pour tout $x\in P$. □

2.3.2. Remarques
(a) $d(I)$ s'appelle vecteur coûts réduits, *$d(I)^I=0$ par construction.*
(b) Le critère (2.3.3) est seulement suffisant et pas nécessaire pour l'optimalité (problèmes avec dégénérescences).

2.3.3. Exercice : *Montrer que si*

$$d(I)^j\left\{\begin{array}{ll}<0 & \text{pour } j\in B_-\\ >0 & \text{pour } j\in B_+\end{array}\right.$$

(inégalités strictes) alors $x(I,B_-,B_+)$ est l'unique solution optimale de (PL). La réciproque, est–elle aussi valable ?

On suppose maintenant donné $x = x(I, B_-, B_+)$, un point de base ne vérifiant pas (encore) la condition suffisante d'optimalité. Envisageons un déplacement suivant une demi-droite $D^+(x, v)$, avec comme but

— d'augmenter l'objectif ;
— de rester dans le polyèdre P de (PL) ;
— de trouver un point de base réalisable comme nouveau point,

d'où la définition suivante.

2.3.4. Définition
Soit $x \in P$. On appelle $v \in \mathbb{R}^n \setminus \{0\}$

— *direction de pente (en x) si $fv > 0$ (dérivée directionnelle).*
— *direction réalisable (en x) si $\exists \theta > 0$ avec $[x, x + \theta v] \subset P$.*

Notons qu'une direction réalisable v est un élément du noyau de A. Si elle est de plus de pente, alors le nouveau point devrait être $\hat{x}$ avec $D^+(x, v) \cap P = [x, \hat{x}]$ (l'intersection de deux convexes donne un convexe). L'idée de la méthode Simplex (proposée par Dantzig en 1949) est de choisir v pour passer d'un point extrême x à un point extrême $\hat{x}$ adjacent, en suivant une arête du polyèdre. Autrement dit, $\hat{x} = x(I', B'_-, B'_+)$, avec (I', B'_-, B'_+) obtenus à partir de (I, B_-, B_+) en échangeant deux indices r et s : on part de l'hyper-plan d'indice r pour arriver à l'hyper-plan d'indice s, tout en restant dans les autres $|J| - 1$ hyper-plans qui définissent x et $\hat{x}$. Ces directions particulières sont données comme suit.

2.3.5. Définition et Lemme : directions privilégiées
Soit (I, B_-, B_+) une base, $T(I) := (A^I)^{-1}A$ (dit tableau simplicial*), et $s \in B_- \cup B_+$, avec $\epsilon(I)_s = +1$ si $s \in B_-$, et $\epsilon(I)_s = -1$ si $s \in B_+$. La* direction privilégiée *$v(I)^s$ est définie par*

$$k \in J : v(I)^s_k = \begin{cases} \epsilon(I)_s & \text{si } k = s, \\ -\epsilon(I)_s T(I)^s_k & \text{si } k \in I, \\ 0 & \text{sinon.} \end{cases}$$

La direction privilégiée $v(I)^s$ appartient au noyau de A. Elle est de pente ssi l'indice s ne vérifie pas la CSO (2.3.3).

Démonstration. On calcule $Av(I)^s = \epsilon(I)_s(A^s - A^I T(I)^s) = 0$. De même, $fv(I)^s = \epsilon(I)_s(f^s - f^I T(I)^s) = \epsilon(I)_s d(I)^s$, ce qui ensemble avec (2.3.3) montre la dernière partie de l'enoncé. □

Rappelons que A^{I+s-r} est inversible si et seulement si $T(I)^s_r \neq 0$, voir le chapitre 1.4. Le résultat suivant discute l'intersection entre P et la demi-droite $D^+(x(I, B_-, B_+), v(I)^s)$. Nous allons d'abord donner en 2.3.6 une formulation et une preuve dans le cadre de la forme standard. Le cas de la forme standard aux variables doublement bornées traité en 2.3.7 suit le même principe (et contient comme cas particulier le théorème 2.3.6, à vérifier...), mais est plus long à décrire, car ici on doit échanger les indices s, r entre trois ensembles et pas seulement entre deux.

2.3.6. Théorème
Considérons la forme standard. Soit $x(I)$ un point de base réalisable, et $s \notin I$. Soit $\theta_{\max} =$

$\min\{\theta_s\} \cup \{\theta_k : k \in I\} \in [0, +\infty]$, *avec*

$$\theta_k = \begin{cases} \frac{x(I)_k}{T(I)^s_k} & \textit{si } k \in I \textit{ et } T(I)^s_k > 0, \\ +\infty & \textit{sinon.} \end{cases}$$

Si $\theta_{\max} = +\infty$ *et* $v(I)^s$ *est de pente alors l'optimum de* (PL) *vaut* $+\infty$. *Si au contraire* $\theta_{\max} < +\infty$, *on peut définir l'indice* $r \in I$ *par* $\theta_{\max} = \theta_r$. *Dans ce dernier cas,* $v(I)^s$ *est réalisable ssi* $\theta_{\max} > 0$. *De plus,* $I' = I + s - r$ *est une base réalisable, et* $D^+(x(I), v(I)^s) \cap P = [x(I), x(I')]$.

Démonstration. Comme dans la preuve du lemme 2.2.4(a) nous observons que, pour un $\theta \geq 0$,

$$t(\theta) = x(I) + \theta v(I)^s \in P \iff \forall k \in I : x(I)_k - \theta\, T(I)^s_k \geq 0 \iff \forall k \text{ avec } T(I)^s_k > 0 : \theta \leq \theta_k.$$

Si maintenant $\theta_{\max} = \infty$ alors la condition à droite est valable pour tout $\theta \geq 0$, c'est-à-dire, $D^+(x(I), v(I)^s) \subset P$ et alors

$$\sup\{fx : x \in P\} \geq \sup\{fx : x \in D^+(x(I), v(I)^s)\} = fx(I) + \sup\{\theta f v(I)^s : \theta \geq 0\} = +\infty,$$

car $fv(I)^s = d(I)^s > 0$ d'après 2.3.5. Si par contre $\theta_{\max} = \theta_r < \infty$ alors $P \cap D^+(x(I), v(I)^s) = [x(I), t(\theta_r)]$, et $T(I)^s_r > 0$ par définition de r. Donc $I' = I + s - r$ est une base, et $t(\theta_r) \in P$. Reste à montrer que $t(\theta_r)$ coïncide avec le point de base correspondant : effectivement, $At(\theta_r) = a$, et $\forall k \notin I, k \neq s : t(\theta_r)_k = x(I)_k + \theta 0 = 0$. Finalement, pour $k = r$ nous avons $t(\theta_r)_k = 0$ par définition de θ_r, et alors $t(\theta_r)_{J \setminus I'} = 0$, ce qu'il fallait démontrer. □

2.3.7. Théorème
Soit $x(I, B_-, B_+)$ *un point de base réalisable, et* $s \in B_- \cup B_+$. *Soit* $\theta_{\max} = \min\{\theta_s\} \cup \{\theta_k : k \in I\} \in [0, +\infty]$, *avec* $\theta_s = c_s - b_s$, *et pour* $k \in I$

$$\theta_k = \begin{cases} +\infty & \textit{si } T(I)^s_k = 0, \\ \big(x(I, B_-, B_+)_k - b_k\big) / \big(\epsilon(I)_s T(I)^s_k\big) & \textit{si } \epsilon(I)_s T(I)^s_k > 0, \\ \big(x(I, B_-, B_+)_k - c_k\big) / \big(\epsilon(I)_s T(I)^s_k\big) & \textit{si } \epsilon(I)_s T(I)^s_k < 0. \end{cases}$$

Si $\theta_{\max} = +\infty$ *et* $v(I)^s$ *est de pente alors l'optimum de* (PL) *vaut* $+\infty$. *Si au contraire* $\theta_{\max} < +\infty$, *on peut définir l'indice* $r \in I + s$ *par* $\theta_{\max} = \theta_r$. *Dans ce dernier cas,* $v(I)^s$ *est réalisable ssi* $\theta_{\max} > 0$. *De plus,* (I', B'_-, B'_+) *défini par le tableau ci-dessous est une base réalisable, et* $D^+(x(I, B_-, B_+), v(I)^s) \cap P = [x(I, B_-, B_+), x(I', B'_-, B'_+)]$.

(I', B'_-, B'_+)	$s = r$	$r \in I$ *et* $\epsilon(I)_s T(I)^s_r > 0$	$r \in I$ *et* $\epsilon(I)_s T(I)^s_r < 0$
$s \in B_+ \iff \epsilon(I)_s = -1$	$(I, B_- + s, B_+ - s)$	$(I + s - r), B_- + r, B_+ - s)$	$(I + s - r), B_-, B_+ - s + r)$
enlever s *de* B_+	*échange entre* B_-, B_+	*ajouter* r *à* B_-	*ajouter* r *à* B_+
$s \in B_- \iff \epsilon(I)_s = 1$	$(I, B_- - s, B_+ + s)$	$(I + s - r), B_- - s + r, B_+)$	$(I + s - r), B_- - s, B_+ + r)$
enlever s *de* B_-	*échange entre* B_-, B_+	*ajouter* r *à* B_-	*ajouter* r *à* B_+

Démonstration. Nous observons que, pour un $\theta \geq 0$,

$$t(\theta) = x(I) + \theta v(I)^s \in P \iff b_s \leq \theta \epsilon(I)_s \leq c_s \text{ et } \forall k \in I : b_k \leq x(I)_k - \theta\, \epsilon(I)_s\, T(I)^s_k \leq c_k$$
$$\iff \theta \in [0, \theta_s] \text{ et } \forall k \text{ avec } T(I)^s_k \neq 0 : \theta \in [0, \theta_k].$$

Pour le cas $\theta_{\max} = \infty$ on peut conclure comme dans la preuve du théorème 2.3.6. Soit maintenant $\theta_{\max} = \theta_r < \infty$ avec $r \in I+s$, alors $P \cap D^+(x(I), v(I)^s) = [x(I), t(\theta_r)]$, et $T(I)^s_r \neq 0$ par définition de r. Donc $I' = I + s - r$ est une base, et $t(\theta_r) \in P$. Reste à montrer que $t(\theta_r)$ coïncide avec le point de base correspondant. Comme $v(I)^s$ appartient au noyau de A nous gardons la propriété $At(\theta_r) = a$, et évidemment $t(\theta_r)_{B_- - s} = b_{B_- - s}$, $t(\theta_r)_{B_+ - s} = c_{B_+ - s}$, à composantes finies. Reste à considérer un certain nombre de cas : si $s = r$, alors $A^{I'} = A^I$ est inversible, et $s \in B_+$ (et $s \in B_-$) implique que $t(\theta_r)_s = b_s > -\infty$ (et $t(\theta_r)_s = c_s < +\infty$, respectivement), donc $t(\theta_r) = x(I', B'_-, B'_+)$.
Si $s \neq r$ alors $T(I)^s_r \neq 0$ par construction, montrant que $A^{I'}$ est inversible. Aussi, $\epsilon(I)_s T(I)^s_r > 0$ implique que $t(\theta_r)_r = b_r > -\infty$, et $\epsilon(I)_s T(I)^s_r < 0$ implique que $t(\theta_r)_r = c_r < +\infty$. Donc effectivement $t(\theta_r) = x(I', B'_-, B'_+)$ par construction de B'_-, B'_+ explicité dans le tableau ci-dessus. □

En résumant, si (I, B_-, B_+) ne vérifie pas encore la CSO, alors on peut obtenir une direction privilégiée de pente, et on s'arrête en un nouveau point de base.

2.3.8. Algorithme : forme primitive de la méthode Simplex
INVARIANT : $x(I, B_-, B_+)$ *point de base réalisable*
ITÉRATION :

1. *Calculer* $d(I)^{B_- \cup B_+}$
2. *Arrêt si condition suffisante d'optimalité valable, sinon*
   PRICING : choisir $s \in B_- \cup B_+$ *avec* $\epsilon(I)_s d(I)^s > 0$.
3. *Trouver direction privilégiée* $v(I)^s$, θ_j, $j \in I + s$, *et* $\theta_{\max}$.
   Arrêt si $\theta_{\max} = +\infty$ *(optimum vaut* $+\infty$*), sinon*
   choisir $r \in I + s$ *avec* $\theta_r = \theta_{\max}$.
4. *Mise à jour des invariants* I, B_-, B_+, $x(I, B_-, B_+) \leftarrow x(I, B_-, B_+) + \theta_r v(I)^s$

Il reste des libertés de choix de s et r. Les stratégies suivantes ont été proposées.

2.3.9. Définition : variantes de Pricing

— *Pivotage de Dantzig (most violating pricing, mvp)*

$$s = \min\{\tilde{s} : \epsilon(I)_{\tilde{s}} d(I)^{\tilde{s}} = \max_k \epsilon(I)_k d(I)^k\}, \quad r = \min\{\tilde{r} : \theta_{\tilde{r}} = \theta_{\max}\}.$$

— *Partial pricing : calculer seulement une partie de* $d(I)$ *et y appliquer pivotage de Dantzig.*
— *Pivotage de Bland (procédé anti-cyclage)*

$$s = \min\{\tilde{s} : \epsilon(I)_{\tilde{s}} d(I)^{\tilde{s}} > 0\}, \quad r = \min\{\tilde{r} : \theta_{\tilde{r}} = \theta_{\max}\}.$$

— *Steepest edge pricing (Goldfarb 1992, Reid 1977) : maximiser* $\epsilon(I)_k d(I)^k / ||v(I)^k||$. *Version simplifiée DEVEX proposée par Harris (1973).*

Le steepest edge pricing paraît bien le plus intéressant, car on compare les dérivées directionnelles normalisées pour les différentes directions privilégiées. Pourtant, pour comparer les différents choix de s, r et retenir le meilleur (permettant par exemple la plus grande augmentation de l'objectif) il faudrait aussi comparer les $\theta_{\max}$ correspondants, ce qui revient beaucoup trop cher en pratique.

2.3.10. Exemple
Nous souhaitons résoudre

$$\begin{cases} \max(4x_1 + 5x_2) \\ x_1 + 2x_2 \leq 7, 2x_1 + x_2 \leq 8, \\ x_1 \in [0, +\infty), x_2 \in [0, 3] \end{cases}$$

ce qui par introduction de variables d'écart x_3, x_4 devient

$$\begin{cases} \max(fx) \\ Ax = a, \\ x \geq b, x \leq c \end{cases} \quad avec \quad \begin{bmatrix} A \\ f \end{bmatrix} = \begin{bmatrix} 1 & 2 & 1 & 0 \\ 2 & 1 & 0 & 1 \\ 4 & 5 & 0 & 0 \end{bmatrix}, \quad a = \begin{bmatrix} 7 \\ 8 \end{bmatrix}, \quad [x, b, c] = \begin{bmatrix} x_1 & 0 & +\infty \\ x_2 & 0 & 3 \\ x_3 & 0 & +\infty \\ x_4 & 0 & +\infty \end{bmatrix}.$$

Vérifions que l'on peut choisir $(I, B_-, B_+) = ((3, 4), (1, 2), \emptyset)$ comme base réalisable de départ. Effectivement, A^I composée de la troisième et quatrième colonne de A est la matrice identité, donc inversible. De plus, $b_1 = 0 \neq -\infty, b_2 = 0 \neq -\infty$, donc il s'agit d'une base. Pour trouver le point de base associé $x = x(I, B_-, B_+)$, on fixe les composantes avec indices appartenant à $B_- \cup B_+$, ici $x_1 = b_1 = 0$, $x_2 = b_2 = 0$, et on résout pour les autres deux composantes :

$$A^I x_I = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_3 \\ x_4 \end{bmatrix} = a - A^1 x_1 - A^2 x_2 = \begin{bmatrix} 7 \\ 8 \end{bmatrix} - 0 - 0,$$

ce qui donne bien $x^t = [0, 0, 7, 8]$, un point de base réalisable (les premiers deux composantes donnant le point géométrique dans la résolution graphique, et les derniers deux l'écart aux droites $x_1 + 2x_2 = 7, 2x_1 + x_2 = 8$).

Vérifions la condition suffisante d'optimalité. Pour $s \in B_- \cup B_+ = (1, 2)$ on trouve les directions

$$[v(I)^1, v(I)^2] = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ -(A^I)^{-1}A^1 & -(A^I)^{-1}A^2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ -1 & -2 \\ -2 & -1 \end{bmatrix},$$

et $[d(I)^1, d(I)^2] = [fv(I)^1, fv(I)^2] = [4, 5] \not\leq 0$. Notons que $d(I)^j \not\leq 0$ pour $j \in B_-$, donc pas optimalité. Comme $d(I)^1 < d(I)^2$, on choisit pour le pivotage de Dantzig de se déplacer suivant la direction $v(I)^s$ avec $s = 2 \in B_-$, en partant de x. Reste à examiner quel est le plus grand paramètre paramètre θ de sorte que $x + \theta v(I)^s \geq b$, $x + \theta v(I)^s \leq c$ (le dernier point dans le polyèdre sur la demi-droite). Ceci donne un système de 8 inégalités (dont trois triviales correspondant aux indices j avec $c_j = +\infty$). Écrivons le explicitement :

$$0 \leq 0 + \theta 0 \leq +\infty, \quad 0 \leq 0 + \theta 1 \leq 3, \quad 0 \leq 7 + \theta(-2) \leq +\infty, \quad 0 \leq 8 + \theta(-1) \leq +\infty.$$

On trouve l'inégalité critique $r = 2$, le paramètre critique $\theta = 3$, et

$$x(I, B_-, B_+) + \theta c(s) = \begin{bmatrix} 0 \\ 0 \\ 7 \\ 8 \end{bmatrix} + 3 \begin{bmatrix} 0 \\ 1 \\ -2 \\ -1 \end{bmatrix} = \begin{bmatrix} 0 \\ 3 \\ 1 \\ 5 \end{bmatrix}.$$

Le théorème 2.3.7 nous dit qu'il s'agit aussi d'un point de base réalisable, effectivement il suffit de choisir la nouvelle partition (on la note par les mêmes lettres) $(I, B_-, B_+) = ((3, 4), (1), (2))$ (on échange $r = s = 2$ entre B_- et B_+).

Dans l'étape suivante, vérifions si on a maintenant trouvé une solution optimale. Notons que I n'a pas changé, mais $\epsilon(I)_2$ change de signe, tout comme $v(I)^2$, et par contre $v(I)^1$ reste la même direction. D'où

$$[v(I)^1, v(I)^2] = \begin{bmatrix} 1 & 0 \\ 0 & -1 \\ -1 & 2 \\ -2 & 1 \end{bmatrix}, \quad [d(I)^1, -d(I)^2] = [fv(I)^1, fv(I)^2] = [4, -5] \not\leq 0.$$

En particulier pour $j = 2 \in B_+ : d(I)^2 = 5 \geq 0$ (comme nécessaire pour la condition suffisante d'optimalité), mais pour $s = 1 \in B_- : d(I)^1 = 4 \not\leq 0$. On se déplace alors sur la demi-droite $x(I, B_-, B_+) + \theta v(I)^s$, donnant lieu aux inégalités

$$0 \leq 0 + \theta 1 \leq +\infty, \quad 0 \leq 3 + \theta 0 \leq 3, \quad 0 \leq 1 + \theta(-1) \leq +\infty, \quad 0 \leq 5 + \theta(-2) \leq +\infty.$$

On trouve l'inégalité critique $r = 3$, le paramètre critique $\theta = 1$, et

$$x(I, B_-, B_+) + \theta v(I)^s = \begin{bmatrix} 0 \\ 3 \\ 1 \\ 5 \end{bmatrix} + 1 \begin{bmatrix} 1 \\ 0 \\ -1 \\ -2 \end{bmatrix} = \begin{bmatrix} 1 \\ 3 \\ 0 \\ 3 \end{bmatrix},$$

ce qui nous donne la nouvelle base $(I, B_-, B_+) = ((1,4),(3),(2))$ (l'indice s sort de B_- pour entrer en I, et r sort de I pour entrer dans B_-).

Pour les composantes avec indices $j \in I$ des nouvelles directions, il faut calculer

$$A^I = \begin{bmatrix} 1 & 0 \\ 2 & 1 \end{bmatrix}, \quad (A^I)^{-1} = \begin{bmatrix} 1 & 0 \\ -2 & 1 \end{bmatrix}, \quad (A^I)^{-1}A^2 = \begin{bmatrix} 2 \\ -3 \end{bmatrix}, \quad (A^I)^{-1}A^3 = \begin{bmatrix} 1 \\ -2 \end{bmatrix},$$

donnant lieu a des directions

$$[v(I)^2, v(I)^3] = \begin{bmatrix} 2 & -1 \\ -1 & 0 \\ 0 & 1 \\ -3 & 2 \end{bmatrix}, \quad [-d(I)^2, d(I)^3] = [fv(I)^2, fv(I)^3] = [3, -4] \not\leq 0.$$

On a la CSO pour $j = 3 \in B_-$, mais pas pour $s = 2 \in B_+$ car $d(I)^2 \not\geq 0$, d'où le système d'inégalités

$$0 \leq 1 + \theta 2 \leq +\infty, \quad 0 \leq 3 + \theta(-1) \leq 3, \quad 0 \leq 0 + \theta 0 \leq +\infty, \quad 0 \leq 3 + \theta(-3) \leq +\infty.$$

On trouve l'inégalité critique $r = 4$, le paramètre critique $\theta = 1$, et

$$x(I, B_-, B_+) + \theta v(I)^s = \begin{bmatrix} 1 \\ 3 \\ 0 \\ 3 \end{bmatrix} + 1 \begin{bmatrix} 2 \\ -1 \\ 0 \\ -3 \end{bmatrix} = \begin{bmatrix} 3 \\ 2 \\ 0 \\ 0 \end{bmatrix},$$

ce qui nous donne la nouvelle base $(I, B_-, B_+) = ((1,2),(3,4),\emptyset)$ (l'indice s sort de B_+ pour entrer en I, et r sort de I pour entrer dans B_-).

Pour les composantes avec indices $j \in I$ des nouvelles directions, il faut calculer

$$A^I = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}, \quad (A^I)^{-1} = -\frac{1}{3}\begin{bmatrix} 1 & -2 \\ -2 & 1 \end{bmatrix}, \quad (A^I)^{-1}A^3 = \begin{bmatrix} -1/3 \\ 2/3 \end{bmatrix}, \quad (A^I)^{-1}A^4 = \begin{bmatrix} 2/3 \\ -1/3 \end{bmatrix},$$

donnant lieu a des directions

$$[v(I)^3, v(I)^4] = \begin{bmatrix} 1/3 & -2/3 \\ -2/3 & 1/3 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}, \quad [d(I)^3, d(I)^4] = [fv(I)^3, fv(I)^4] = [-2, -1] \leq 0,$$

d'où la condition suffisante d'optimalité. Par conséquent, la solution optimale est donnée par $x^t = [3, 2, 0, 0]$ pour le problème sous forme standard aux variables doublement bornées, et $x^t = [3, 2]$ pour le problème de départ.

2.3.11. Exercice : *Considérons le problème aux variables doublement bornées*

$$\max\{2x_1 - x_2 : x_2 \leq x_1 + 1, -1 \leq x_1 \leq 1, -1 \leq x_2 \leq 1\}.$$

En introduisant une variable d'écart x_3, donner la forme standard généralisé du problème. Montrer que $(I, B_-, B_+) = ((1),(3),(2))$ est une base réalisable. Appliquer la forme primitive de la méthode SIMPLEX pour trouver une solution optimale du problème. Indiquer sur un dessin les différents points extrêmes examinés.

2.3.12. Exercice : *Considérons le problème aux variables doublement bornées*

$$\begin{cases} \max(3x_1 + 2x_2 + 9x_3 + x_4 + 5x_5) \\ 7x_1 + 5x_2 + 3x_3 + x_4 + 2x_5 = 17 \\ 0 \leq x_j \leq j, \quad j = 1, 2, 3, 4, 5. \end{cases} .$$

En partant de la partition $I = (3)$, $B^- = (4, 5)$, $B^+ = (1, 2)$, résoudre ce problème. Est-ce qu'on aurait pu deviner la solution en examinant les coefficients d'utilité f^j/A_1^j pour $j = 1, ..., 5$?

Discutons la finitude de la méthode Simplex.

2.3.13. Théorème : finitude
(a) Dans la méthode Simplex, si à chaque itération on ne rencontre pas de dégénérescence ($\theta_{\max} \neq 0$) alors on obtient en un nombre fini d'itérations l'étape d'arrêt (soit solution optimale soit optimum= $+\infty$).
(b) Si on utilise des procédés anticyclage (par exemple le pivotage de Bland) alors la méthode Simplex est finie même en présence de dégénérescences.

Démonstration de 2.3.13(a). Notons par $(I^{(k)}, B_-^{(k)}, B_+^{(k)})$ la suite des bases rencontrés dans Simplex. Par le théorème 2.3.7 et sa preuve, nous avons que

$$x(I^{(k+1)}, B_-^{(k+1)}, B_+^{(k+1)}) = x(I^{(k)}, B_-^{(k)}, B_+^{(k)}) + \theta_{\max}^{(k)} v(I^{(k)})^{s_k}$$

et alors

$$fx(I^{(k+1)}, B_-^{(k+1)}, B_+^{(k+1)}) = fx(I^{(k)}, B_-^{(k)}, B_+^{(k)}) + \theta_{\max}^{(k)} \underbrace{\epsilon(I^{(k)})_{s^{(k)}}\, d(I^{(k)})^{s_k}}_{>0}.$$

En absence de dégénérescence, nous avons $\theta_{\max}^{(k)} > 0$, c'est-à-dire, la suite des valeurs est strictement croissante. En conséquence, les points de base réalisables rencontrés sont deux à deux distincts, en particulier les bases sont deux à deux distincts. Comme il y a un nombre fini de bases (voir aussi le lemme 2.2.4(e) et le théorème 2.2.8), il faudra que la suite soit finie, autrement dit, que Simplex s'arrête après un nombre fini de changements de base. □

Démonstration de 2.3.13(b). Cette preuve est assez complexe. Comme l'enoncé sera utilisé dans la suite, on propose ici des détails, mais leur lecture n'est pas indispensable. Aussi, la preuve est seulement présentée pour la forme standard.

Comme le nombre de bases est fini, il suffit de montrer que l'on ne rencontre jamais deux fois la même base. Par absurde, supposons qu'il se produit un cyclage sur les bases I_k, $k = 0, 1, ..., h$, avec $h > 1$, i.e., $I_h = I_0$, et les bases $I_0, .., I_{h-1}$ sont deux à deux différentes. On note $J := \{1, 2, .., n\}$ l'indexage de la variable x,

$$B := \bigcap_k I_k, \quad H := \bigcap_k \overline{I}_k, \quad F := J \setminus (B \cup H),$$

et $r(I_k), s(I_k)$ par $I_{k+1} = I_k - r(I_k) + s(I_k)$. De plus, on considère

$$\sigma := \max\{j : j \in F\}.$$

Notons que F est non vide (par exemple, $s(I_1) \in F$) et que

$$\sigma = s(I) = r(I')$$

pour des bases distincts $I, I' \in \{I_0, ..., I_{h-1}\}$ par construction de F.
Pour établir une contradiction, on montra d'abord quatre relations

$$\forall k = 0, .., h \quad \forall i \in I_k \setminus B : \quad t(I_k)_i = 0, \tag{2.3}$$

$$\forall j \in J \setminus (\{\sigma\} \cup H) : \quad d(I)^j \leq 0, \tag{2.4}$$

$$\forall i \in I' \setminus (\{\sigma\} \cup B) : \quad T(I')_i^{s(I')} \leq 0, \tag{2.5}$$

$$T(I')_\sigma^{s(I')} > 0. \tag{2.6}$$

Pour montrer (2.3), exprimons, en fonction de $x(I_0)$, les solutions de base $x(I_k)$ et les valeurs associées de la fonction économique. Comme $I_0 = I_h$, nous avons $x(I_0) = x(I_h)$ (définition d'un point de base) et alors

$$fx(I_0) \leq fx(I_1) \leq ... \leq fx(I_h) = fx(I_0)$$

et

$$fx(I_{k+1}) = fx(I_k) + \Theta(I_k)^{s(I_k)} \cdot d(I_k)^{s(I_k)}, \qquad x(I_{k+1}) = x(I_k) + \Theta(I_k)^{s(I_k)} \cdot C(I_k)^{s(I_k)}.$$

Par construction, $d(I_k)^{s(I_k)} > 0$. On en déduit que

$$x(I_0) = x(I_1) = ... = x(I_h)$$

et que

$$\Theta(I_k)^{s(I_k)} = 0 = t(I_k)_{r(I_k)}, \quad k = 0, ..., h-1.$$

Rappelant que $i \in I_k \setminus B$ s'exprime comme $i = r(I_j)$ pour un j, nous obtenons

$$t(I_k)_i = x(I_k)_i = x(I_j)_i = t(I_j)_{r(I_j)} = 0.$$

Passons à la démonstration de (2.4) : il n'y a rien à montrer si $j \in I$. Sinon, pour tout $j \in J \setminus (\{\sigma\} \cup H \cup I)$ nous savons que $j \in F$ et $j < \sigma = s(I)$. D'après le pivotage de Bland, on en déduit que j n'etait pas candidat pour $s(I)$, d'où $d(I)^j \leq 0$.
Pour montrer (2.5), soit $i \in I' \setminus B$, $i \neq \sigma$. Alors $i \in F$ et $i < \sigma$ par définition de F et σ. D'après (2.3), nous savons que $t(I')_i = 0 = \Theta(I')^{s(I')}$, et la propriéte $T(I')_i^{s(I')} > 0$ nous donnerait un candidat i pour $r(I')$ qui serait plus petit que $\sigma = r(I')$, en contradiction avec les règles de Bland. Finalement, propriété (2.6) provient du fait que $\sigma = r(I')$.
Comme I' est une base, un trouve un et un seul vecteur $z \in \mathbb{R}^n$ de sorte que

$$Az = 0, \quad z_{\overline{I'} - s(I')} = 0, \quad z_{s(I')} = -1.$$

Il s'agit de $z = -v(I')^s$ et alors $z_{I'} = T(I')^{s(I')}$. Notons que

$$\begin{aligned} d(I) \cdot z &= -d(I)^{s(I')} + d(I)^{I'} \cdot T(I')^{s(I')} \\ &= -d(I)^{s(I')} + [f^{I'} - f^I \cdot (A^I)^{-1} A^{I'}] \cdot (A^{I'})^{-1} \cdot A^{s(I')} \\ &= -f^{s(I')} + f^{I'} \cdot (A^{I'})^{-1} \cdot A^{s(I')} = -d(I')^{s(I')} < 0. \end{aligned}$$

Par conséquent, il existe un ℓ tel que

$$d(I)^\ell \cdot z_\ell < 0. \tag{2.7}$$

Cet indice ℓ vérifie $d(I)^\ell \neq 0 \neq z_\ell$, et alors $\ell \notin I$ et $\ell \in I' + s(I')$, ce qui signifie que $\ell \in F$.
On est alors préparé à trouver une contradiction. En effet, il reste à examiner deux cas : le cas $\ell = \sigma (= r(I') \in I')$ est exclus car $d(I)^\sigma = d(I)^{s(I)} > 0$ et $z_\sigma = T(I')_\sigma^{s(I')} > 0$ d'après (2.6). On en déduit que $\ell \in F - \sigma$. Pour ce cas on sait d'après (2.4) que $d(I)^\ell \leq 0$. Il découle de (2.7) que $z_\ell > 0$, et alors $\ell \in I'$ d'après la définition de z. D'autre part, en utilisant (2.5) on obtient

$$0 \geq T(I')_\ell^{s(I')} = z_\ell > 0,$$

une contradiction. On en déduit l'impossibilité de l'hypothèse de cyclage. □

2.3.14. Remarques

(a) Si l'expérience montre que la dégénérescence est extrêmement fréquente dans certaines

classes de problèmes pratiques, elle montre aussi que le cas de cyclage est tout a fait exceptionnel (néanmoins il existe des exemples, voir Exercice 2.4.15). Aussi, un pricing comme Bland peut être incompatible avec les stratégies de calculs en grande taille creux. C'est pour cela que généralement des procédés anticyclage ne sont pas inclus dans les codes industriels.
(b) Dans la pratique on observe également que, "en général", seulement $\mathcal{O}(m)$ itérations sont nécessaires (si $m \to \infty$ et n/m tend vers une constante non nulle) pour trouver une solution optimale de (PL). Cependant, cette observation doit être comparée avec la seule borne théorique connue $\binom{n}{m}$ pour le nombre d'itérations (le nombre de bases distincts pour des problèmes sous forme standard). Effectivement, Simplex est "non-polynomial" : on trouve des exemples avec $n = 2m$ ($\binom{2m}{m} \approx 8^m$) où Simplex ne s'arrête pas avant 2^{m-1} itérations.[5] *Malgré cela, pour sa simplicité et sa puissance, la méthode Simplex est encore largement utilisée dans le cadre industriel (même si Khachian en 1979 (méthode des ellipses, généralisation d'une méthode du mathématicien lillois Pierre Huard) et Karmarkar en 1984 (points intérieurs) ont récemment proposé des méthodes polynomiales).*

2.3.15. Exercice : *Le pivotage "steepest edge pricing" (Reid, 1977) consiste à*

$$\text{choisir } s \in \overline{I} \text{ t.q.} \qquad \frac{f \cdot v(I)^s}{||v(I)^s||} = \max\Big\{ \frac{f \cdot v(I)^k}{||v(I)^k||} \Big\},$$

d'où la necessité de calculer (recursivement et d'une manière efficace) $\rho(I)^k := ||v(I)^k||^2 = 1+||T(I)^k||^2$ pour $k \in \overline{I}$ (comme on souhaite utiliser la forme révisée de SIMPLEX, les quantités $T(I)^k$, $k \neq s$, ne sont a priori pas disponibles...).
Montrer que si $I' = I - r + s$ alors

$$\rho(I')^k = \begin{cases} \rho(I)^s/[T(I)^s_r]^2 & \text{si } k = r \in \overline{I'}, \\ \rho(I)^k + \rho(I)^s \cdot [\alpha_k]^2 - 2wA^k \cdot \alpha_k & \text{si } k \in \overline{I'} \cap \overline{I}, \end{cases}$$

où $\alpha_k := [d(I)^k - d(I')^k]/d(I)^s$, et $[T(I)^s]^T = w \cdot A^I$. Compter le nombre d'opérations élémentaires.

2.4 La mise en œuvre de Simplex pour les problèmes sous forme standard

Dans l'algorithme 2.3.8, il reste à expliciter comment, pour une base réalisable donnée, calculer les composantes du vecteur coûts réduits, de la direction privilégiée, et du point de base, ou comment les mettre à jour après un changement de base. Aussi, il n'est pas encore clair comment démarrer cet algorithme, car il nous faut un premier point de base réalisable. Par simplicité, ici nous allons seulement discuter les problèmes sous forme standard, et laissons au lecteur le soin de généraliser pour la forme standard généralisée.

Pour répondre à la première question, on discutera d'abord une façon de type Gauß-Jordan nécessitant $\mathcal{O}(mn)$ opérations arithmétiques et espace mémoire par itération de Simplex, dite méthode du tableau simplicial. Cette version peut convenir pour les problèmes de petite taille, et est généralement enseignée dans le cadre économique ou informatique. Dans les applications

5. $\max\{\sum_{j=1}^m 10^{n-j} x_j, x_j \geq 0, \forall i = 1, ..., m : \sum_{j=1}^i 10^{i-j} x_j \leq 100^i\}$

industrielles (comme par exemple les problèmes de coupe, voir Exercice 2.7.4), le nombre de contraintes m peut rester relativement petit, mais le nombre n d'inconnues peut être trop important, ce qui pose des problèmes pour stocker et modifier le tableau simplicial entier. C'est pour cela que l'on discutera ensuite la forme révisée de Simplex qui, selon la mise en œuvre, peut avoir une complexité de $\mathcal{O}(m^2)$ par itération, un chiffre bien plus intéressant.

2.4.1. Définition Tableau simplicial
Etant donnée une base réalisable I, et $A = A_L^J$, on définit

$$\begin{aligned} T(I) &= T(I)_I^J = (A^I)^{-1} \cdot A, \\ t(I) &= t(I)_I = (A^I)^{-1} \cdot a. \end{aligned}$$

2.4.2. Remarques sur le tableau simplicial
Au lieu du problème de départ $\max\{fx : Ax = a, x \geq 0\}$*, dans la méthode du tableau simplicial on discutera successivement les problèmes équivalents*

$$\max\{fx : T(I)x = t(I), x \geq 0\}, \tag{2.8}$$

d'où l'intérêt d'exprimer $v(I)^s, x(I)$ etc. en termes de $T(I)$ et $t(I)$. Clairement, $x(I)_{\overline{I}} = 0$ (on rappelle que $\overline{I} = J \setminus I$), $x(I)_I = t(I)$, et $d(I) = f - f^I T(I)$. Aussi, $v(I)_I^s = -T(I)^s$, et donc

$$\theta_{\max} = \min\{+\infty\} \cup \{\frac{t(I)_j}{T(I)_j^s} : T(I)_j^s > 0\}. \tag{2.9}$$

Finalement, on observe que $T(I)^I$ est l'identité, et que, pour $I' = I - r + s$, $T(I)^{I'}$ coïncide à une colonne près avec la matrice identité, ce qui facilite le calcul de son inverse (voir le chapitre 1.4)

$$(T(I)^{I'})^{-1} = T(I')^I \backsimeq \begin{array}{|ccc|c|c|} \hline & U_{I\cap I'}^{I\cap I'} & & -\frac{T(I)_{I\cap I'}^s}{T(I)_r^s} & I \cap I' \\ \hline 0 & \cdots & 0 & 1/T(I)_r^s & s \\ \hline & I \cap I' & & r & \\ \hline \end{array} \tag{2.10}$$

2.4.3. Exercice : Mise à jour du tableau simplicial *(prémultiplication avec $(T(I)^{I'})^{-1}$).*
Soit $T' = I - r + s$ une base réalisable adjacente. Montrer que

$$\begin{aligned} T(I')_{I\cap I'} &= T(I)_{I\cap I'} - \frac{1}{T(I)_r^s} \cdot T(I)_{I\cap I'}^s \cdot T(I)_r, \qquad T(I')_s = \frac{1}{T(I)_r^s} \cdot T(I)_r, \\ t(I')_{I\cap I'} &= t(I)_{I\cap I'} - \frac{1}{T(I)_r^s} \cdot T(I)_{I\cap I'}^s \cdot t(I)_r, \qquad t(I')_s = \frac{1}{T(I)_r^s} \cdot t(I)_r, \\ d(I') &= d(I) - \frac{1}{T(I)_r^s} \cdot d(I)^s \cdot T(I)_r. \end{aligned}$$

(NB : $t(I)$ est traitée comme une colonne de $T(I)$, $d(I)$ est traitée comme une ligne de $T(I)$.)

2.4.4. Remarques sur la manipulation du tableau simplicial

La mise à jour suivant 2.4.3 nécessite $\mathcal{O}(mn)$ opérations, et en particulier la connaissance de la ligne $T(I)_r$ et de la colonne $T(I)^s$. Le calcul peut être réalisé par des opérations élémentaires sur les lignes du schéma (dit tableau simplicial*)*

$T(I)$	$t(I)$
$d(I)$	$fx(I)$

Après avoir déterminé r et s, on créera le nouveau tableau associé à $I' = I - r + s$ en additionnant un multiple scalaire de la $r$ième ligne a toutes les autres et en divisant la $r$ième ligne par un scalaire, de sorte que la $s$ième colonne devienne un vecteur canonique (c'est-à-dire, $T(I')^{I'} = U$ et $d(I')^s = 0$).

Par une organisation différente du calcul, on obtient la *forme révisée* de Simplex. Notons que l'algorithme 2.3.8 nécessite seulement les quantités $f^I(A^I)^{-1}$ (pour le calcul de $d(I)$), $t(I)$ et $T(I)^s$, lesquelles sont solutions des systèmes

$$f^I = \lambda A^I, \quad A^I t = a, \quad A^I T = A^s, \tag{2.11}$$

respectivement. Donc au lieu de calculer et surtout stocker le tableau entier $T(I)$, il suffit de pouvoir résoudre ces systèmes, par la connaissance d'une décomposition LU de A^I (en produit d'une matrice triangulaire inférieure L et une matrice triangulaire supérieure U plus éventuellement une matrice de permutation), d'une décomposition QR de A^I (en produit d'une matrice orthogonale Q et une matrice triangulaire supérieure R plus éventuellement une matrice de permutation), ou alors [6] de l'inverse de A^I. Ensuite il faudra voir comment mettre à jour ces données après remplacement de la colonne d'indice r (qui sort de la base) par la colonne d'indice s (qui entre en base). Pour l'inverse, nous avons la formule simple de mise à jour $(A^{I'})^{-1} = T(I')^I (A^I)^{-1}$, avec $T(I')^I$ donnée en (2.10).

En utilisant l'inverse, on obtient l'implémentation suivante :

2.4.5. Algorithme : forme revisée de Simplex

Invariant : I *base réalisable,* $M = (A^I)^{-1}$ *connue*
Phase 1 : *Calculer point de base* $t \leftarrow Ma$ $[= t(I)]$
Calculer multiplicateur $\lambda \leftarrow f^I \cdot M$
Calculer vecteur coûts réduits $d \leftarrow f^{\overline{I}} - \lambda \cdot A^{\overline{I}}$ $[= d(I)^{\overline{I}}]$ *(ou une partie)*
Phases 2,3 : *test d'optimalité, pricing [ici mvp]*
$dmax \leftarrow -\infty$
pour $k \in \overline{I}$ *faire*
si $d^k > dmax$ *alors* $dmax \leftarrow d^k$, $s \leftarrow k$
STOP si $dmax \leq 0$ *[* $x(I)$ *solution optimale]*
Phases 4,5,6 : *choix de* r*, test problème non borné*
$\Theta \leftarrow +\infty$, $T \leftarrow M \cdot A^s$ $[= T(I)^s]$

6. Ouf, en analyse numérique on avait appris qu'il ne faut jamais calculer l'inverse d'une matrice ! Pour toute application de l'inverse, une décomposition LU ou QR devrait faire le travail d'une manière plus efficace et numériquement plus stable... C'est vrai ! Mais on pourrait dire beaucoup de choses sur ce sujet, voir aussi 2.4.6(d)...

pour $k \in I$ *faire*
si $((T_k > 0)$ *et* $(\Theta > t_k/T_k))$ *alors* $\Theta \leftarrow t_k/T_k$, $r \leftarrow k$
STOP si $\Theta = +\infty$ *[optimum vaut* $+\infty$ *]*

Phase 7 : *Nouvelle base :* $I \leftarrow I + s - r$ *(*s *prend la position* $rpos$ *de* r*)*
Nouvelle inverse : $M_{rpos} \leftarrow M_{rpos}/T_{rpos}$
pour $(j \neq rpos)$ *faire* $M_j \leftarrow M_j - T_j \cdot M_{rpos}$

On vérifie aisément qu'une itération nécessite $\mathcal{O}(m^2)$ opérations arithmétiques si on calcule seulement $\mathcal{O}(m)$ composantes de $d(I)$ (voir partial pricing 2.3.9), et $\mathcal{O}(mn)$ sinon. Au niveau de l'encombrement mémoire on a besoin de stocker $m^2 + \mathcal{O}(m)$ quantités (au lieu de $\mathcal{O}(mn)$ pour le tableau simplicial), plus les données A, a, f. Mais le vrai avantage est que l'on doit accéder seulement rarement à ces données : il suffit de disposer d'une boîte noire qui réalise le produit scalaire entre un vecteur ligne et une colonne de la matrice $\begin{bmatrix} A & a \\ f & 0 \end{bmatrix}$. Ceci s'avère particulièrement intéressant dans le cas où l'on dispose de données creuses (beaucoup de coefficients 0), ou dans des situations comme dans Exercice 2.7.4 où une colonne de A doit d'abord être calculée (parfois d'une manière implicite, voir la méthode de décomposition de Dantzig et Wolfe au chapitre 2.7). Rappelons que l'inverse d'une matrice creuse n'est généralement pas creuse, et donc le tableau simplicial risque de ne pas être creux du tout.

2.4.6. Remarques sur l'implémentation sur machine :
(a) Comme précisé dans la phase 7 de l'algorithme 2.4.5, le tableau d'indices I *n'est pas forcément rangé dans l'ordre croissant d'indices. Il s'avère alors utile d'introduire sur machine deux tableaux :* `base(k)=j` *si l'indice* j *est le* k*ième élément de la liste* I*, et* `posi(i)=k` *avec* $k = -1$ *si* i *est hors base, et sinon* `base(k)=i` *(la permutation inverse), en particulier* `rpos=posi(r)`.

(b) Dans l'algorithme de Gauß, on utilise pivotage (partiel) pour rendre le calcul numériquement stable en cas de précision finie, le but étant que le pivot $T(I)^s_r$ *soit le plus grand possible* $|T(I)^s_r| \approx \max_{j \in I} |T(I)^s_j|$*. Malheureusement, Simplex ne permet pas cette possibilité, car* r *a été choisi dans le but de ne pas sortir du polyèdre. On peut donc cumuler des erreurs d'arrondis dans les mises à jour successives de l'inverse. D'où le conseil de calculer l'inverse explicitement à l'aide d'une méthode numériquement stable (par exemple, l'algorithme de Gauss avec pivotage partiel) après un nombre fixe d'itérations (par exemple* m *itérations, car le calcul direct de l'inverse nécessite* $\mathcal{O}(m^3)$ *opérations).*

Bien entendu, dans un cadre de précision finie, on risque de sortir du polyèdre faute de pouvoir calculer exactement $\theta_{\max}$ *etc. On espère néanmoins rester proche des points de base réalisables. Il y a des tests heuristiques pas trop chers (*$||d(I)^I||$ *ou* $||A^I t(I) - a||$ *"petit",* $||(A^I)^{-1}||$ *pas "trop grand") qui permettent de donner une indication quand il vaut mieux recalculer explicitement l'inverse...*

(c) Au lieu de calculer explicitement la mise à jour de $(A^I)^{-1}$*, souvent on stocke* $(A^I)^{-1}$ *sous forme de la suite des colonnes non triviales des facteurs* $T(I')^I$ *de modification (forme produit de l'inverse).*

(d) Au lieu de calculer l'inverse de A^I*, on peut aussi donner une variante de 2.4.5 basée sur une décomposition* $P_1(I)A^I P_2(I) = L(I)U(I)$*, avec une matrice triangulaire inférieure* $L(I)$ *et une matrice triangulaire supérieure* $U(I)$*, et des matrices de permutation* $P_1(I), P_2(I)$*. En phase 7 on met à jour (ceci se fait en* $\mathcal{O}(m^2)$ *opérations), et on détermine la factorisation explicitement*

après un certain nombre d'itérations. Ceci ne résout pas les problèmes de pivotage et stabilité numérique, mais pour m grand et données creuses, souvent les facteurs $L(I)$ et $U(I)$ restent creux...

Par contre, les matrices dans une factorisation QR seront généralement toujours denses, mais la mise à jour de la factorisation est numériquement stable... Moralité : il n'existe pas un meilleur choix !

2.4.7. Schéma de "calcul sur papier"

Pour terminer cette partie, on propose un schéma de "calcul sur papier" pour la forme révisée de la méthode SIMPLEX qui permettra de résoudre à la main des problèmes de petite taille.

$T(I)^{I(prec)}$ *si nécessaire*	$(A^I)^{-1}$ *lignes énumérées par I*	I	$t(I)$ $= (A^I)^{-1}a$	$T(I)^s$ $= (A^I)^{-1}A^s$	*rapport (choix de r)*
f^I *(én. par I)*	$u = f^I(A^I)^{-1}$	$f - uA$ *(choix de s)*			

Ordre de calcul :

- *Point de départ pour l'itération : la colonne I est déjà remplie. De plus, soit $(A^I)^{-1}$ est connue, soit on peut calculer le produit $(A^I)^{-1}$ entre $T(I)^{I(prec)}$ et l'inverse précédante.*
- *On calcule $t(I) = (A^I)^{-1}a$.*
- *On détermine f^I (attention à l'ordre des coefficients !) et $u = f^I(A^I)^{-1}$.*
- *On calcule $d(I)$, on vérifie que $d(I)^I = 0$, et on choisit s (pricing). Solution optimale ?*
- *Disposant du s, on calcule $T(I)^s$, et les rapports $t(I)_k/T(I)^s_k$ pour tout pivot potentiel $T(I)^s_k > 0$. Pas de candidats signifie que l'optimum vaut $+\infty$.*
- *On choisit r comme indice réalisant Θ (le plus petit rapport).*
- *Dans le schéma suivant, on note le nouveau I (s prend la place de r).*
- *Suivant (2.10), la matrice $T(I)^{I(prec)}$ du schéma suivant est obtenue en partant de la matrice unité et en modifiant la colonne associée à la position du pivot ($1/T(I)^s_r$ sur diagonale, $-T(I)^s_k/T(I)^s_r$ sinon).*

Variante : mise à jour de $t(I)$ (et éventuellement de $d(I)$) suivant 2.4.3. $T(I)^{\overline{I}}_r$ peut être noté au dessous de $d(I)^{\overline{I}}$.

Exemple : résoudre $\max\{fx : Ax = a\}$, *avec*

$$A = \begin{bmatrix} 1 & -2 & 1 & 0 & 0 \\ 2 & -1 & 0 & 1 & 0 \\ -3 & 1 & 0 & 0 & 1 \end{bmatrix}, \quad a = \begin{bmatrix} 2 \\ 4 \\ 6 \end{bmatrix}, \quad f = [-1, 1, 0, 0, 0].$$

On obtient

			1	0	0	3	2	-2	/	
			0	1	0	4	4	-1	/	
			0	0	1	5	6	1	$\boxed{6}$	r=5
0	0	0	0	0	0	0,0,0,-1 ,		$\boxed{1}$		s=2
1	0	2	1	0	2	3	14	-5	/	
0	1	1	0	1	1	4	10	-1	/	
0	0	1	0	0	1	2	6	-3	/	
0	0	1	0	0	1	$\boxed{2}$, 0 , 0, 0, -1				s=1

Conclusion : l'optimum vaut $+\infty$.

2.4.8. Exercice : *Résoudre par la forme révisée du simplex les problèmes suivants*

$$(a)\ \begin{cases} \min(2x_2 - x_1) \\ 3x_1 + 4x_2 \leq 12 \\ 2x_1 + x_2 \leq 6 \\ x_1, x_2 \geq 0. \end{cases}, \quad (b)\ \begin{cases} \min(x_1 - x_2) \\ x_1 - 2x_2 \leq 2 \\ 2x_1 - x_2 \leq 4 \\ 3x_1 - x_2 \geq -6 \\ x_1, x_2 \geq 0. \end{cases}, \quad (c)\ \begin{cases} \max(x_1 + x_2 + x_3) \\ x_1 + x_2 \leq 1 \\ x_1 + x_3 \leq 1 \\ x_1, x_2, x_3 \geq 0 \end{cases}.$$

On devrait observer en (a) plusieurs solutions optimales, en (b) que l'optimum vaut $+\infty$, et en (c) une dégénerescence, avec une solution optimale unique $(0, 1, 1, 0, 0)^T$.

Toute variante de la méthode Simplex discutée pour l'instant nécessite la connaissance d'une base réalisable I de départ (et éventuellement aussi de $(A^I)^{-1}$), quelquefois obtenue à partir des variables d'écart. Dans le cas général, une telle base (et un test $P = \emptyset$?) est obtenue par la résolution d'un programme linéaire auxiliaire (PA) en phase 1 à l'aide de $\leq m$ variables artificielles qui s'annulent dans (et peuvent donc être éliminées de) la phase 2 de la résolution de (PL).

2.4.9. Théorème : calcul en deux phases
Soit à résoudre (PL) avec $a \geq 0$ (sans perte de la généralité), et $A = A^J_L$, avec (sans perte de la généralité) $L \cap J = \emptyset$. On considère avec $e = (1, 1, ..., 1) \in \mathbb{R}^{1\times m}$ le problème auxiliaire

$$(PA)\ \begin{cases} \max(-ez) \\ Ax + U^L_L z = a \\ x, z \geq 0, \end{cases}$$

et on note P_A l'ensemble des solutions réalisables pour (PA).
(a) $P_A \neq \emptyset$.
(b) L'objectif de (PA) est borné supérieurement par 0.
(c) L'optimum de (PA) vaut 0 ssi P est non vide.
(d) Soit $\widetilde{A} = \widetilde{A}^{J,L}_L = (A, U^L_L)$. L'ensemble L est une base réalisable de départ pour Simplex appliqué au problème (PA) en phase 1. Soit I la base finale (optimale) de Simplex en phase 1.
(d1) Si $I \subset J$ alors I est aussi base réalisable pour (PL), avec $(\widetilde{A}^I)^{-1} = (A^I)^{-1}$.
(d2) Si l'optimum de (PA) vaut zéro mais $I \not\subset J$ (dégénérescence), on peut continuer la phase 1, pour obtenir une autre base réalisable et optimale $I' \subset J$ (base de départ pour la phase 2).

Démonstration. Les parties (a)–(d1) sont faciles à vérifier (exercice). La preuve de (d2) se fait par récurrence sur $|L \cap I|$. Notre objectif est de remplacer successivement un $r \in I \cap L$ par $s \in J \setminus I$, de sorte que $I' = I + s - r$ reste base réalisable, avec $x(I')$ solution optimale de (PA). Soit $r \in I \cap L = I \setminus J$ quelconque. Comme *rang* $(A) = m$ et $\widetilde{A}^I$ est inversible, nous déduisons que $[(\widetilde{A}^I)^{-1}A]_r = \widetilde{T}(I)_r^J \neq 0$. Mais comme $\widetilde{T}(I)_r^{J\cap I} = U_r^{J\cap I} = 0$ par construction, nous déduisons qu'il existe un $s \in J \setminus I$ avec $\widetilde{T}(I)_r^s \neq 0$. Par conséquent, $I' = I - r + s$ est une base pour (PA). Notons que, par hypothèse,

$$\widetilde{f}\widetilde{x}(I) = 0 = -e\widetilde{x}(I)_L$$

ce qui implique que $\widetilde{x}(I)_L = 0$ et donc $\widetilde{t}(I)_r = 0$. En conséquence, $\widetilde{x}(I') = \widetilde{x}(I)$ reste solution optimale de (PA), et I' est aussi réalisable (dégénérescence) . □

2.4.10. Exercice : *Montrer que pour le problème suivant on peut utiliser le simplex sans une procédure de démarrage*

$$\begin{cases} \max(-3x_1 - 5x_2 - 2x_3 - 4x_4 - x_5) \\ 3x_1 + \frac{21}{2}x_3 + \frac{3}{2}x_5 = \frac{69}{2} \\ 20x_1 + \frac{5}{3}x_2 + 15x_3 = \frac{205}{3} \\ 5x_1 + 4x_3 + x_4 = 17 \\ x_1, ..., x_5 \geq 0. \end{cases}$$

2.4.11. Exercice : *Résoudre en deux phases*

$$(a) \begin{cases} \min(x_1 - 2x_2) \\ x_1 + x_2 \leq 3 \\ -x_1 + 3x_2 \leq -4 \\ x_1, x_2 \geq 0. \end{cases}, \quad (b) \begin{cases} \min(x_1 - 2x_2 + x_3) \\ 3x_1 + x_2 - x_3 = 1 \\ -2x_1 + x_2 - 2x_3 = 1 \\ qx_1, x_2, x_3 \geq 0. \end{cases}, \quad (c) \begin{cases} \min(x_1 - 2x_2 + 2x_3) \\ x_1 + x_2 - x_3 = 3 \\ -x_1 + 3x_2 = -4 \\ x_1, x_2, x_3 \geq 0. \end{cases}$$

2.4.12. Exercice : *Etant donné le polyèdre P suivant*

$$\begin{cases} -2x_1 + 2x_2 - 3x_3 \geq 30 \\ -x_1 + x_2 + 2x_3 \geq 8 \\ x_1 - x_2 - x_3 \geq 2 \\ x_1, x_2, x_3 \geq 0, \end{cases}$$

trouver une représentation matricielle équivalente de P (dite variante de GASS) telle que la procédure de démarrage nécessite seulement une seule variable artificielle (idée : dans le problème sous forme standard, retirer une équation des autres).

2.4.13. Exercice : *On considère le problème aux variables doublement bornées*

$$\begin{cases} \max(f \cdot x) \\ Ax = a \\ x \geq x^-, \quad x \leq x^+. \end{cases}$$

Soit (I, B^+, B^-) une partition de $\{1, ..., n\}$, avec A^I inversible. Une solution $x(I, B^+, B^-)$ de $Ax = a$ est dite point de base *(*réalisable*) si $x(I, B^+, B^-)_{B^-} = x^-_{B^-}$ et $x(I, B^+, B^-)_{B^+} = x^+_{B^+}$ (et $x(I, B^+, B^-) \in P$).*

(a) *Déterminer $x(I, B^+, B^-)_I$.*

(b) *M.q. un point de base réalisable est point extrême de P.*

(c) *Soit $x(I, B^+, B^-)$ un point de base réalisable, et $d(I, B^+, B^-) := f - f^I(A^I)^{-1}A$. M.q. $x(I, B^+, B^-)$ est une solution optimale si*

$$d(I, B^+, B^-)^{B^-} \leq 0, \quad et \quad d(I, B^+, B^-)^{B^+} \geq 0. \tag{2.12}$$

(d) *Soit s un indice pour lequel condition (2.12) n'est pas valable. M.q. la direction $C(I)^s$ du simplex est une direction de pente si $s \in B^-$, et que $-C(I)^s$ est une direction de pente si $s \in B^+$.*

(e) *Quelles sont les composantes "critiques" de $t(\Theta) := x(I, B^+, B^-) + \Theta \cdot C(I)^s$? Donner des restrictions sur Θ en fonction de ces composantes. En déduire qu'il existe un Θ_{max} t.q. $t(\Theta_{max})$ est un point de base réalisable.*

(f) *A partir des propriétés etablies en (d), (e), donner une version de SIMPLEX revisé adaptée aux variables doublement bornées.*

(g) *Si on exclut cyclage, est-ce qu'on peut montrer convergence finie ? Dans ce cas, caractériser le dernier point de base rencontré.*

(h) *Considérons le problème aux variables doublement bornées*

$$\begin{cases} \max(3x_1 + 2x_2 + 9x_3 + x_4 + 5x_5) \\ 7x_1 + 5x_2 + 3x_3 + x_4 + 2x_5 = 17, \\ 0 \leq x_j \leq j, \quad j = 1, 2, 3, 4, 5. \end{cases}$$

En partant de la partition $I = (3)$, $B^- = (4, 5)$, $B^+ = (1, 2)$, résoudre ce problème.

On peut résumer le théorème 2.3.13 (finitude de Simplex) et théorème 2.4.9 (démarrage de Simplex) comme suit (comparer avec 2.2.8) :

2.4.14. Corollaire
Etant donné un programme linéaire (PL) sous forme standard, avec P l'ensemble de points réalisables,
(a) si $P \neq \emptyset$ alors (PL) admet une solution de base réalisable.
(a) si $P \neq \emptyset$, et si l'objectif est borné supérieurement sur P alors $\exists$ solution optimale de (PL) ayant une représentation comme point de base réalisable et vérifiant la condition suffisante d'optimalité (2.3.3).

2.4.15. Exemple :
Le problème suivant a été proposé par E.M.L. BEALE

$$(PL) \begin{cases} \max(\frac{3}{4}x_1 - 20x_2 + \frac{1}{2}x_3 - 6x_4) \\ x_1/4 - 8x_2 - x_3 + 9x_4 \leq 0 \\ x_1/2 - 12x_2 - x_3/2 + 3x_4 \leq 0 \\ x_3 \leq 1 \\ x_1, x_2, x_3, x_4 \geq 0. \end{cases}$$

La méthode SIMPLEX, en appliquant le pivotage "naturel" (most violating pricing) d'après Dantzig, donne lieu à un cyclage :

			1	0	0	5	0	1/4	0	r=5
			0	1	0	6	0	1/2	0	
			0	0	1	7	1	0	/	
0	0	0	0	0	0	3/4, -20, 1/2, -6, 0,0,0				s=1
4	0	0	4	0	0	1	0	-32	/	
-2	1	0	-2	1	0	6	0	4	0	r=6
0	0	1	0	0	1	7	1	0	/	
3/4	0	0	3	0	0	0, 4,7/2, -33, -3,0,0				s=2
1	8	0	-12	8	0	1	0	8	0	r=1
0	1/4	0	-1/2	1/4	0	2	0	3/8	0	
0	0	1	0	0	1	7	1	1	1	
3/4	-20	0	1	1	0	0,0,2,-18,-1,-1,0				s=3
1/8	0	0	-3/2	1	0	3	0	-21/2	/	
-3/64	1	0	1/16	-1/8	0	2	0	3/16	0	r=2
-1/8	0	1	3/2	-1	1	7	1	21/2	2/21	
1/2	-20	0	-2	3	0	-1/4,0,0,3,2,-3,0				s=4
1	56	0	2	-6	0	3	0	2	0	r=3
0	16/3	0	1/3	-2/3	0	4	0	1/3	0	
0	-56	1	-2	6	1	7	1	-2	/	
1/2	-6	0	-1	1	0	1/2,-16,0,0,1,-1,0				s=5
1/2	0	0	1	-3	0	5	0	-3	/	
-2/3	1	0	0	1/3	0	4	0	1/3	0	r=4
1	0	1	0	0	1	7	1	0	/	
0	-6	0	0	-2	0	7/4,-44,-1/2,0,0,2,0				s=6
1	9	0	1	0	0	5	0	1/4	0	
9	3	0	0	1	0	6	0	1/2	0	**voir première étape**
0	0	1	0	0	1	7	1	0	/	
0	0	0	0	0	0	3/4, -20, 1/2, -6,0,0,0				

On trouve dans le fichier `don_ch2/beale1.bat` *une simulation sous Scilab de Simplex avec pivotage naturel de Dantzig (malgré le cyclage, on s'arrête à l'itération 21, car le nombre maximum d'itérations est dépassé), et dans le fichier* `don_ch2/beale2.bat` *une simulation montrant que le pricing de BLAND permet d'aboutir à une solution optimale sans cyclage.*

2.5 La dualité

Le but de ce chapitre est de fournir des caractérisations d'optimalité (alternatives–Farkas–Kuhn et Tucker–Dualité).

2.5.1. Théorème des alternatives *(aussi nommé Lemme de Farkas)*
Soient $A \in \mathbb{R}^{m\times n}$ (pas forcément de rang m), $a \in \mathbb{R}^m$, alors l'une et seulement une des propriétés suivantes est valable

$$(i) \qquad \exists x \in \mathbb{R}^n : Ax = a \quad et \quad x \geq 0,$$
$$(ii) \qquad \exists \lambda \in \mathbb{R}^{1\times m} : \lambda A \leq 0 \quad et \quad \lambda a > 0.$$

Démonstration. Montrons d'abord que (i) et (ii) ne peuvent pas être simultanément valables.

Sinon

$$0 \geq (\lambda A)x = \lambda(Ax) = \lambda a > 0,$$

une contradiction.
Supposons maintenant que (i) soit faux, et démontrons que (ii) est valable. Nous reprenons le Simplex en deux phases, comparer avec la preuve de 2.4.9. Soit $A = A_L^J$ avec (sans perte de la généralité) $L \cap J = \emptyset$, $B = B_L^L = diag\,(\epsilon_1, ..., \epsilon_m)$, avec $\epsilon = +1$ si $a_k \geq 0$, et $\epsilon_k = -1$ sinon, et $\widetilde{A} = \widetilde{A}_L^{J,L} = (A, B)$. Comme $rang\,(\widetilde{A}) = m$, on peut appliquer Simplex au problème

$$(PL): \quad \max\{fx : \widetilde{A}x = a, x \geq 0\}, \quad f^J = 0, \quad f^L = (-1, ..., -1)$$

avec L comme base réalisable de départ. Clairement, (PL) est borné supérieurement par 0, de valeur optimale < 0 (car (i) est faux). Le corollaire 2.4.14 nous dit qu'il existe I une base réalisable de (PL) telle que $fx(I) < 0$ et $d(I) \leq 0$. Posons $\lambda = -f^I(\widetilde{A}^I)^{-1}$ (multiplicateur avec changement de signe), alors

$$\lambda a = -f^I x(I)_I = -fx(I) > 0, \qquad \lambda A = f^J + \lambda \widetilde{A}^J = d(I)^J \leq 0.$$

□

Il existe plusieurs formes équivalentes du théorème des alternatives, evoquées dans les exercices suivants.

2.5.2. Exercice : *Soit $A \in \mathbb{R}^{m\times n}$. Montrer à l'aide du théorème des alternatives que l'un et l'un seulement des deux systèmes de contraintes suivants admet une solution*

(a) *(i)* $\exists x \in \mathbb{R}^n \quad Ax = b$ *(ii)* $\exists \lambda \in \mathbb{R}^{1\times m} \quad \lambda A = 0,\ \lambda b < 0$ *;*

(b) *(i)* $\exists x \in \mathbb{R}^n \quad Ax = 0,\ x \geq 0,\ x \neq 0$ *(ii)* $\exists \lambda \in \mathbb{R}^{1\times m} \quad \lambda A \geq e = (1, ..., 1)$.

2.5.3. Exercice : *Considérons les problèmes*

$$(P) \qquad \max\{f \cdot x : x \in P\}, \quad P = \{x : Ax = a, x \geq 0\},$$
$$(D) \qquad \min\{\lambda \cdot a : \lambda \in D\}, \quad D = \{\lambda : \lambda A \geq f\}.$$

A l'aide du théorème des alternatives, montrer que pour tout $M > \max\{f \cdot x : x \in P\}$ il existe un $\lambda \in D$ t.q. $\lambda a < M$.

2.5.4. Définition : problème primal–dual
On associe au problème (dit primal*)*

$$(P) \quad \max\{fx : x \in P\}, \quad P = \{x \in \mathbb{R}^n : Ax = a, x \geq 0\},$$

le problème (dit dual*)*

$$(D) \quad \min\{\lambda a : \lambda \in D\}, \quad D = \{\lambda \in \mathbb{R}^{1\times m} : \lambda A \geq f\}.$$

Plus généralement, le dual d'un programme pas sous forme standard est défini comme le dual du programme équivalent sous forme standard.

2.5.5. Remarque :
Au sens stricte, (D) n'est pas un programme linéaire. Néanmoins on obtient le problème équivalent

$$-\max\{(-a)^t y : A^t y \geq -f^t\}$$

qui s'écrit sous forme standard (z variable d'écart, $y = y' - y''$)

$$(D)_s \quad -\max\{\widetilde{f}\widetilde{x} : \widetilde{A}\widetilde{x} = \widetilde{a}, \widetilde{x} \geq 0\},$$

avec

$$\widetilde{x} = \begin{bmatrix} y' \\ y'' \\ z \end{bmatrix}, \quad \widetilde{A} = \begin{bmatrix} A^t, -A^t, -U \end{bmatrix}, \quad \widetilde{a} = -f^t, \quad \widetilde{f} = \begin{bmatrix} -a^t, a^t, 0 \end{bmatrix}.$$

2.5.6. Lemme :
Le dual du dual est équivalent au primal.

Démonstration. Le dual du problème (D_S) est (en posant $x = -\lambda^T$)

$$-\min\{\lambda\widetilde{a} : \lambda\widetilde{A} \geq \widetilde{f}\} = \max\{\widetilde{a}^T x : -\widetilde{A}^T x \geq \widetilde{f}^T\} = \max\{fx : \begin{bmatrix} -A \\ +A \\ U \end{bmatrix} x \geq \begin{bmatrix} -a \\ a \\ 0 \end{bmatrix}\}.$$

□

Le lien entre les problèmes (P) et (D) est discuté dans

2.5.7. Théorème de la dualité
(a) $\forall x \in P \forall \lambda \in D : fx \leq \lambda a$.
(b) Si $P = \emptyset$ alors (D) n'admet pas de valeur optimale finie (impossible ou optimum=$-\infty$).
(c) Si $D = \emptyset$ alors (P) n'admet pas de valeur optimale finie (impossible ou optimum=$+\infty$).
(c) Si $D \neq \emptyset$ et $P \neq \emptyset$ alors (P) et (D) admettent la même valeur optimale (finie).

Le résultat peut être schématisé comme suit :

	$D = \emptyset$	$D \neq \emptyset$
$P = \emptyset$	(P) et (D) impossible	(D) non borné
$P \neq \emptyset$	(P) non borné	(P) et (D) bornés, même optimum

L'exemple

$$A = \begin{bmatrix} 1 & 0 & -1 \\ 0 & -1 & 0 \end{bmatrix}, \quad a = \begin{bmatrix} 0 \\ 2 \end{bmatrix}, \quad f = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$$

montre que $P = D = \emptyset$ est possible. Comme les quatre cas s'excluent mutuellement, on obtient le corollaire suivant.

2.5.8. Corollaire
Si l'un des problèmes (P) ou (D) admet un optimum fini, alors aussi l'autre, avec la même valeur optimale.

Démonstration de 2.5.7. **(a)** Pour $x \in P$ et $\lambda \in D$ nous avons $Ax = a$, $x \geq 0$ et $\lambda A \geq f$, et alors

$$fx - \lambda a = fx - \lambda Ax = (\underbrace{f - \lambda A}_{\leq 0})\underbrace{x}_{\geq 0} \leq 0.$$

(b) Soit $P = \emptyset$ et $D \neq \emptyset$. Montrons que (D) admet la valeur optimale $-\infty$. Comme $D \neq \emptyset$, nous trouvons $\lambda_1 \in D$. D'après le théorème 2.5.1 des alternatives et $P = \emptyset$,

$$\exists \lambda_0 : \quad \lambda_0 A \leq O, \quad \lambda_0 a > 0.$$

Donc pour tout $\theta \geq 0$ nous trouvons que $A(\lambda_1 - \theta\lambda_0) \geq f$, d'où $D^+(\lambda_1, -\lambda_0) \subset D$, avec

$$\inf\{\lambda a : \lambda \in D\} \leq \inf\{\lambda a : \lambda \in D^+(\lambda_1, -\lambda_0)\} = \lambda_1 a + \inf\{-\theta\lambda_0 a : \theta \geq 0\} = -\infty.$$

(c) Voir (b) et le lemme 2.5.6. **(d)** Soient $P \neq \emptyset$ et $D \neq \emptyset$. D'après (a), l'objectif de (P) est borné supérieurement, et donc (P) admet une valeur optimale finie d'après le corollaire 2.2.5. Une propriété similaire est vrai pour (D) avec

$$\text{valeur optimale de } (P) \leq \text{valeur optimale de } (D). \tag{2.13}$$

On laissera au lecteur la tâche de montrer égalité dans (2.13) en utilisant le théorème 2.5.1. Ici on donnera une preuve directe : d'après le corollaire 2.4.14(b), il existe une solution optimale $x(I)$ avec $d(I) \leq 0$. En posant $\lambda_0 := f^I(A^I)^{-1}$ nous trouvons que $\lambda_0 A - f = -d(I) \geq 0$, et donc $\lambda_0 \in D$. De plus

$$fx(I) - \lambda_0 a = (f - \lambda_0 A)x(I) = -d(I)x(I) = 0,$$

car $x(I)_{J\setminus I} = 0$ et $d(I)^I = 0$. Donc on obtient égalité dans (2.13). □

On vient de démontrer que si un point de base réalisable $x(I)$ vérifie la CSO (2.3.3) alors le vecteur multiplicateur associé $f^I(A^I)^{-1}$ est solution optimale de (D).

Deux résultats intéressants découlent directement du théorème de la dualité.

2.5.9. Théorème de la complémentarité
Soient $\widehat{x} \in P$ et $\widehat{\lambda} \in D$. Alors nous avons l'équivalence

$$\begin{array}{ll} & \widehat{x} \text{ est solution optimale de } (P) \text{ et } \widehat{\lambda} \text{ est solution optimale de } (D) \\ \iff & (\widehat{\lambda}A - f)\widehat{x} = 0 \quad \iff \quad \forall j : (\widehat{\lambda}A - f)^j \widehat{x}_j = 0. \end{array}$$

Démonstration. D'après 2.5.7(a),(d) nous savons que, pour tout couple $(x, \lambda) \in P \times D$,

$$\lambda a - fx = (\lambda A - f)x \geq 0,$$

avec égalité si et seulement si (x, λ) est un couple optimal. □

2.5.10. Théorème de Kuhn et Tucker
Soit $\widehat{x} \in P$. Alors nous avons l'équivalence

$$\begin{array}{ll} & \widehat{x} \text{ est solution optimale de } (P) \\ \iff & \exists \lambda \in D \text{ avec } (\lambda A - f)\widehat{x} = 0. \end{array}$$

Démonstration. Voir preuve de 2.5.9. □

2.5.11. Remarques
(a) Le vecteur λ dans le théorème 2.5.10 est connu sous le nom de vecteur (ou multiplicateur) de Kuhn et Tucker.
(b) Le cas $\widehat{x}_j = 0$ (contrainte active pour (P)) n'implique pas que $(\lambda A - f)^j = 0$ (contrainte pas forcément active pour (D)).
(c) Si $\hat{x} = x(I)$ un point de base réalisable vérifiant la CSO, il n'est pas difficile de vérifier que $\lambda = f^I(A^I)^{-1}$, calculé par la forme revisée de Simplex, est effectivement un multiplicateur de Kuhn et Tucker (il peut y avoir d'autres...).
(d) Comment obtenir un théorème de Kuhn et Tucker pour un problème pas forcément sous forme standard ? A titre d'exemple, considérons

$$(PL) : \max\{fx : Ax \geq a\} \quad \Longleftrightarrow$$
$$(P) : \max\{[f, -f, 0]y : [A, -A, -U]y = a, y \geq 0\}, \quad y = \begin{bmatrix} x' \\ x'' \\ z \end{bmatrix}, \quad x = x' - x''.$$

Le problème dual est

$$\min\{\lambda a : \lambda[A, -A, -U] \geq [f, -f, 0]\}.$$

Donc pour un $\hat{x}$ réalisable pour (PL)

$$\begin{aligned}
& \hat{x} \ \textit{ solution optimale pour (PL)} \\
\Longleftrightarrow \quad & \hat{x} = \hat{x}' - \hat{x}'', \hat{x}', \hat{x}'' \geq 0, \hat{z} = A\hat{x} - a, \begin{bmatrix} \hat{x}' \\ \hat{x}'' \\ \hat{z} \end{bmatrix} \ \textit{ solution optimale pour (P)} \\
\Longleftrightarrow \quad & \exists \lambda : \lambda[A, -A, -U] \geq [f, -f, 0] \quad \textit{et} \quad (\lambda[A, -A, -U] - [f, -f, 0]) \begin{bmatrix} \hat{x}' \\ \hat{x}'' \\ \hat{z} \end{bmatrix} = 0 \\
\Longleftrightarrow \quad & \exists \lambda : \lambda A = f, -\lambda \geq 0 \quad \textit{et} \quad \lambda\hat{z} = \lambda(A\hat{x} - a) = 0.
\end{aligned}$$

Donnons encore une interprétation supplémentaire pour les multiplicateurs de Kuhn et Tucker, très importante dans les applications (voir par exemple le problème multi-période exposé dans l'annexe A.2).

2.5.12. Corollaire : coûts marginaux
Considérons le problème parametré

$$(P)_a : \max\{fx : Ax = a, x \geq 0\},$$

de valeur optimale $v(a)$, avec I une base réalisable pour $(PL)_a$ vérifiant la CSO, et $u = f^I(A^I)^{-1}$ le vecteur multiplicateur associé (=solution optimale du dual de $(PL)_a$). Si Δa est si petit que $(A^I)^{-1}(a + \Delta a) \geq 0$ alors I reste réalisable et optimale pour $(PL)_{a+\Delta a}$, et la variation de l'optimum est donné par

$$v(a + \Delta a) - v(a) = u\Delta a.$$

Si $(PL)_a$ est la forme standard d'un problème de distribution de ressources (voir 1.2.1), on a l'interprétation économique suivante : si on achète α tonnes de la matière première j alors les bénéfices changent de $u\Delta a = \alpha u_j$ (sous l'hypothèse que I reste réalisable, i.e., α "très petit"). On parle d'un *coût ou prix marginal* u_j car l'affaire sera intéressante pour l'entreprise si on peut acheter de la matière première j à un prix $\leq u_j$ E/tonne.

Démonstration de 2.5.12. D'abord notons que la matrice de coefficients ne change pas pour les problèmes $(PL)_a$, et donc I est toujours une base pour $(PL)_{a+\Delta a}$ quelque soit Δa. Par contre, elle est réalisable pour $(PL)_{a+\Delta a}$ si et seulement si $(A^I)^{-1}(a+\Delta a) \geq 0$. Egalement, $d(I)$ pour $(PL)_{a+\Delta a}$ ne dépend pas de Δa, et donc I vérifie la CSO pour $(PL)_{a+\Delta a}$ quelque soit Δa. Finalement, avec le point de base correspondant $\widetilde{x}(I)_I = (A^I)^{-1}(a+\Delta a)$, $\widetilde{x}(I)_{J\setminus I} = 0$, nous obtenons

$$\text{optimum } (PL)_{a+\Delta a} = f\widetilde{x}(I) = \text{optimum } (PL)_a + u\Delta a.$$

□

2.5.13. Exercice : *On considère les problèmes*

$$(P) \qquad \min\{fx : Ax \geq a, x \geq 0\},$$
$$(D) \qquad \max\{\lambda a : \lambda \geq 0, \lambda A \leq f\}.$$

(a) *Ecrire le problème (P) sous forme standard (noté par (P'), avec polyèdre associé P', variable d'écart $y = Ax - a$). Ecrire également le problème (D) sous forme standard (noté par (D'), avec polyèdre associé D', variable d'écart $\mu = f - \lambda A$).*
Montrer que les problèmes (P) et (D) sont en dualité (dite dualité symétrique).

(b) *Montrer que*

$$\forall \binom{x}{y} \in P' \quad \forall (\lambda, \mu) \in D' : \quad fx \geq \lambda a,$$

avec égalité ssi $\lambda \cdot y = 0$ et $\mu \cdot x = 0$. Quelle est la signification de ce résultat ? Comparer avec le Théorème de complémentarité.

(c) *Supposons que (P') admet une solution optimale $\binom{\widehat{x}}{\widehat{y}} \in P'$. A l'aide du Théorème de Kuhn et Tucker, vérifier qu'il existe des $(\widehat{\lambda}, \widehat{\mu}) \in D'$ t.q. l'on obtient égalité dans (b).*

(d) *Où est-ce qu'on retrouve $(\widehat{\lambda}, \widehat{\mu})$ dans le tableau simplicial final de (P') ? Où est-ce qu'on retrouve $\binom{\widehat{x}}{\widehat{y}}$ dans le tableau simplicial final de (D') ? Quelle est leur signification économique ?*

Pour certains problèmes, par exemple de type '*satisfaction de demande*

$$\min\{gx : Bx \geq b, x \geq 0\}, \quad g \geq 0,$$

on trouve relativement vite une base qui n'est pas réalisable mais qui vérifie la CSO (ici $A = (B^J_L, -U^L_L)$, $I = L$, $t(I) = -b$ n'est pas forcément ≥ 0, mais $d(I) = f - 0 = (-g, 0) \leq 0$). Dans ces cas, une variante de la méthode Simplex – le *Simplex dual* – est plus appropriée.

2.5.14. Algorithme : forme primitive de la méthode Simplex dual
INVARIANT : *I base vérifiant la CSO (2.3.3)*
ITÉRATION :

1. *Arrêt si $t(I) \geq 0$ (base optimale), sinon déterminer $r \in I$ avec $t(I)_r < 0$.*
2. *Calculer $d(I)^{\overline{I}}$ et $T(I)_r^{\overline{I}}$ (noté au dessous de $d(I)^{\overline{I}}$)*
3. *Arrêt si $T(I)_r^{\overline{I}} \geq 0$ ((PL) impossible), sinon chercher $s \in \overline{I}$ avec $T(I)_r^s < 0$ et*

$$\frac{d(I)^s}{T(I)_r^s} = \min\{\frac{d(I)^j}{T(I)_r^j} : j \in \overline{I}, T(I)_r^j < 0\}.$$

4. *Mise à jour des invariants $I \leftarrow I + s - r$, calculer $T(I)^s$ pour nouvelle inverse.*

On peut montrer (voir Exercice 2.5.16) que Simplex dual n'est rien que la méthode Simplex appliquée au problème dual (d'où la finitude de Simplex dual). Aussi, il existe des généralisations de Simplex dual pour le cas des variables doublement bornés, voir Exercice 2.5.17. Donnons ici une démonstration directe des conditions d'arrêt, et du fait que l'on garde les invariants après une itération.

2.5.15. Lemme
(a) $T(I)_r^{\overline{I}} \geq 0$ implique que (PL) est impossible.
(b) $I' = I + s - r$ est une base et vérifie la CSO (2.3.3).

Démonstration. **(a)** D'après le théorème 2.5.7 de la dualité, il suffit de démontrer que le problème dual (D) n'est pas borné. Soit $u := f^I(A^I)^{-1}$, alors $uA - f = -d(I) \geq 0$, et alors $u \in D$. Posons $d = [(A^I)^{-1}]_r$, alors $dA = T(I)_r \geq 0$, ce qui implique que, pour tout $\theta \geq 0$, nous avons $u + \theta d \in D$, et $(u + \theta d)a = ua + \theta t(I)_r \to -\infty$ pour $\theta \to +\infty$.
(b) D'après construction, $T(I)_r^s \neq 0$ et alors $I' = I - r + s$ est une base. D'après la formule de mise à jour de 2.4.3 nous trouvons que

$$d(I')^r = d(I)^r - \frac{d(I)^s}{T(I)_r^s} T(I)_r^r = -\frac{d(I)^s}{T(I)_r^s} \leq 0.$$

Aussi, pour $j \notin I'$, $j \neq r$ et $T(I)_r^j \geq 0$

$$d(I')^j = d(I)^j - \frac{d(I)^s}{T(I)_r^s} T(I)_r^j \leq d(I)^j \leq 0,$$

et pour $j \notin I'$, $j \neq r$ et $T(I)_r^j < 0$

$$d(I')^j = d(I)^j - \frac{d(I)^s}{T(I)_r^s} T(I)_r^j = T(I)_r^j \Big(\frac{d(I)^j}{T(I)_r^j} - \frac{d(I)^s}{T(I)_r^s} \Big)$$

ce qui est ≤ 0 par choix de s. □

2.5.16. Exercice : *(problème primal - problème dual)*
On considère le problème (PL)

$$\max\{fx, \quad x \in P\}, \quad P = \{x \in \mathbb{R}^n : \quad Ax = a, \quad x \geq 0\}$$

et son dual. On notera les quantités du dual par un "tilde"

1. *Ecrire le problème dual sous forme standard.*
2. *Soit I une base du primal et on pose $\widetilde{I} = \overline{I} \cup L$. Calculer $\left[(\widetilde{A}^{\widetilde{I}})^{-1}\right]^T$ en fonction de $(A^I)^{-1}$ et de $T(I)$.*
3. *Montrer que I est une base du primal ssi $\widetilde{I}$ est une base du dual.*
4. *Notons les quantités simpliciales du dual par $\widetilde{T}, \widetilde{t}, \widetilde{d}$. Vérifier que*

$$\begin{array}{lcl} [\widetilde{t}(\widetilde{I})]^T & = & [u(I), -d(I)^{\overline{I}}], \\ \left[\widetilde{d}(\widetilde{I})^I\right]^T & = & -t(I), \\ \left[\widetilde{T}(\widetilde{I})^r\right]^T & = & \left[-\left[(A_L^I)^{-1}\right]_r, -T(I)_r^{\overline{I}}\right]. \end{array} \tag{2.14}$$

5. *Conclure que le simplex dual pour (PL) coincide avec le simplex aux variables doublement bornées pour (DL).*

2.5.17. Exercice : Simplex dual aux variables doublement bornées
Etant donné un point de base aux variables doublement bornées pas forcément réalisable, mais vérifiant la CSO

$$x = x(I, B_-, B_+), \quad x_{B_-} = b_{B_-}, \quad x_{B_+} = c_{B_+}, \qquad d(I)^{B_-} \leq 0, \quad d(I)^{B_+} \geq 0,$$

et $r \in I$, on se demande d'abord comment choisir $s \notin I$ (et où ranger r) pour que pour la nouvelle base (I', B'_-, B'_+) on garde la CSO (NB : les changements de base d'échange d'éléments entre B_- et B_+ permis pour SIMPLEX primal ne peuvent pas préserver la CSO car on ne change pas le vecteur coûts réduits). Pour $r \in I$, soient

$$J_+(r) := \{j \in B_- \cup B_+ : T(I)_r^j \neq 0, \frac{d(I)^j}{T(I)_r^j} \geq 0\}, \; J_-(r) := \{j \in B_- \cup B_+ : T(I)_r^j \neq 0, \frac{d(I)^j}{T(I)_r^j} \leq 0\},$$

et (en cas d'ensembles non vides)

$$J_+(r) \ni s_+(r) = \arg\min\{\left|\frac{d(I)^j}{T(I)_r^j}\right| : j \in J_+(r)\}, \; J_-(r) \ni s_-(r) = \arg\min\{\left|\frac{d(I)^j}{T(I)_r^j}\right| : j \in J_-(r)\}$$

les indices ou les min sont atteints.

(a) Supposons que $x_r < b_r$ et $J_+(r) = \emptyset$. On souhaite montrer que le polyèdre $\{x : Ax = a, x \geq b, x \leq c\}$ est vide (de le même manière on déduit que si $x_r > c_r$ et $J_-(r) = \emptyset$ alors le polyèdre $\{x : Ax = a, x \geq b, x \leq c\}$ est vide).

(a1) Vérifier que le dual de notre problème aux variables doublement bornées s'écrit comme

$$fb + \min\{(\lambda, \mu) \begin{bmatrix} a - Ab \\ c - b \end{bmatrix} : (\lambda, \mu) \begin{bmatrix} A & 0 \\ U & U \end{bmatrix} \geq (f, 0)\}.$$

(a2) Avec U_r le rième vecteur canonique (ligne), soient

$$\lambda_1 = f^I(A^I)^{-1},\ \lambda_2 = U_r(A^I)^{-1},\ \mu_1^j = \begin{cases} d(I)^j & si\ j \in B_+ \\ 0 & sinon \end{cases}, \quad \mu_2^j = \begin{cases} -T(I)_r^j & si\ j \in B_+ \\ 0 & sinon \end{cases}$$

Vérifier que pour tout $\theta \in [0, +\infty)$, le point $(\lambda_1 + \theta\lambda_2, \mu_1 + \theta\mu_2)$ est réalisable pour le dual.

(a3) Montrer que

$$(\lambda_2, \mu_2) \begin{bmatrix} a - Ab \\ c - b \end{bmatrix} = x_r - b_r < 0.$$

Qu'est-ce qu'on peut déduire pour l'optimum du problème dual ?

(a4) En déduire la propriété énoncé en (a).

(b) Vérifier que si $s \in \{s_+(r), s_-(r)\}$ alors $I - r + s$ est une base.

(c) Montrer que, avec $s = s_+(r)$, la CSO est valable pour $(I - r + s, B_- - s + r, B_+ - s)$ (de la même manière, si on choisit $s = s_-(r)$, la CSO est valable pour $(I - r + s, B_- - s, B_+ - s + r)$).
Idée : formule de mise à jour de $d(I)$.

(d) Dans l'algorithme SIMPLEX dual aux variables doublement bornées, après avoir choisi $r \in I$ avec $x_r \notin [b_r, c_r]$, on envisage un déplacement

$$\hat{x} = x(I, B_-, B_+) + \theta \begin{bmatrix} -T(I)_r^s \\ U^s \end{bmatrix}, \qquad s \in \{s_+(r), s_-(r)\}.$$

Le réel θ pas forcément positif doit être ajusté de sorte que $\hat{x}_r = b_r$ si $s = s_+(r)$ (car r entrera dans B_- pour préserver la CSO) et $\hat{x}_r = c_r$ si $s = s_-(r)$ (car r entrera dans B_+ pour préserver la CSO).

(d1) Si dans le cas $x_r < b_r$ on choisit

$$\theta = \frac{x_r - b_r}{T(I)_r^s}, \quad s = s_+(r), \qquad (I', B'_-, B'_+) := (I - r + s, B_- - s + r, B_+ - s),$$

ou dans le cas $x_r > c_r$ on choisit

$$\theta = \frac{x_r - c_r}{T(I)_r^s}, \quad s = s_-(r), \qquad (I', B'_-, B'_+) := (I - r + s, B_- - s, B_+ - s + r),$$

montrer que $\hat{x} = x(I', B'_-, B'_+)$.

(d2) Vérifier que

$$fx(I', B'_-, B'_+) - fx(I, B_-, B_+) = \theta d(I)^s = -|\theta d(I)^s| \leq 0.$$

(d3) En déduire que, en absence de cyclage, l'algorithme Simplex dual s'arrête après un nombre fini de changements de base, soit en observant que le problème est impossible, soit avec une solution optimale.

2.5.18. Exercice : *On considère le problème*

$$(P)\begin{cases}\min(2x_1+3x_2)\\ 4x_1+x_2\geq 8\\ x_1+4x_2\geq 8\\ 7x_1+10x_2\geq 47\\ x_1,x_2\geq 0.\end{cases}$$

(a) *Ecrire le problème (P) sous forme standard (noté par (P^*)). Enoncer le problème dual (D) de (P^*), et montrer qu'il est équivalent au problème*

$$(D^*)\begin{cases}\max(8z_1+8z_2+47z_3)\\ 4z_1+z_2+7z_3+z_4=2\\ z_1+4z_2+10z_3+z_5=3\\ z_1,z_2,z_3,z_4,z_5\geq 0.\end{cases}$$

(b) *Préférez–vous résoudre (P^*) ou (D^*) ? Justifier (en deux phrases).*

(c) *Résoudre (D^*) à l'aide de la méthode Simplex (forme révisée) en utilisant le pivotage de "most violating pricing" (la base finale devrait être $I=(3,2)$). Indiquer clairement l'optimum et une solution optimale de (D^*) ainsi que de (D).*

(d) *Donner l'optimum des problèmes (P) et (P^*). Ecrire les conditions de complémentarité entre une solution optimale de (D) et une de (P^*). En déduire une solution optimale de (P). Est-ce que cette solution optimale apparaît déjà en partie (c) ? Pourquoi ?*

(e)* *Dans les problèmes $(P),(P^*),(D)$ et (D^*), on introduit maintenant un paramètre $\epsilon\in\mathbb{R}$, en remplaçant 47 par $47+\epsilon$. Pour quel choix de ϵ la base I finale trouvée en partie (c) pour le problème (D^*) reste–t–elle réalisable ? Et pour quel choix de ϵ vérifie–t–elle encore la condition suffisante d'optimalité ? Signification pour le problème (P) ?*

2.5.19. Exercice :

(a) *Une entreprise de construction d'automobiles possède trois usines situées respectivement à Paris, Strasbourg et Lyon. Un certain métal nécessaire à la construction des véhicules est disponible aux ports du Havre et de Marseille. Les quantités de ce métal nécessaires aux usines sont 400, 300 et 200 tonnes respectivement pour les usines de Paris, Strasbourg et Lyon chaque semaine, tandis que les quantités disponibles sont de 550 et 350 tonnes par semaine respectivement à Marseille et au Havre. Les coûts de transport sont supposés varier proportionnellement aux quantités transportées, les coûts unitaires étant (en F/tonne)*

	Paris	*Strasbourg*	*Lyon*
Marseille	*5*	*6*	*3*
Le Havre	*3*	*5*	*4*

Le problème consiste à déterminer un "plan de transport" optimal, c'est–à–dire à trouver quels sont les poids de métal à envoyer depuis chaque port à chaque usine de sorte que
(i) Les demandes des usines soient satisfaites ;
(ii) Les quantités demandées à chaque port n'excèdent pas les quantités disponibles ;
(iii) Les quantitées envoyées sont positives ou nulles ;
(iv) Le coût total du transport est rendu minimum compte tenu des contraintes ci-dessus.
Donner un modèle mathématique sous forme d'un programme linéaire.

(b) *Supposons maintenant qu'un transporteur prenne contact avec la direction de l'entreprise de construction automobile et lui propose de lui acheter le métal aux prix λ_1, λ_2 aux ports de Marseille et du Havre, de se charger du transport et de lui revendre le métal aux prix λ_3, λ_4 et λ_5 aux usines de Paris, Strasbourg et Lyon. Pour convaincre la Direction de l'entreprise de construction automobile que l'affaire ne lui sera pas défavorable, le transporteur garantit que ses prix seront compétitifs avec les coûts actuels de transport, c'est–à–dire, le coût pour une usine soit inférieur où égal à la somme du coût de transport calculé par l'entreprise plus les bénéfices de vente. La Direction de l'entreprise convient que dans ces conditions il vaut mieux laisser le transporteur se charger du travail. Le transporteur qui se voit donc confier la mission doit trouver des prix $\lambda_1, ..., \lambda_5$ satisfaisant les contraintes tout en maximisant son profit.*

Donner un modèle mathématique sous forme d'un programme linéaire.

(c) *Résoudre les deux programmes linéaires.*

(d) *La SNCF est en grève et les coûts unitaires de transport pour l'entrepise de construction automobile risquent de varier. Quelle est la signification économique pour le transporteur ?*

2.5.20. Exercice : *Problème du consommateur*

On peut acheter chez M. Dupont quatre types d'aliments dont les teneurs en calories, vitamines exprimées dans la même unité de poids u et les prix, sont donnés par le tableau suivant

	Type 1	*Type 2*	*Type 3*	*Type 4*
calories	*2*	*1*	*0*	*1*
vitamines	*3*	*4*	*3*	*5*
prix	*2*	*2*	*1*	*8*

1. *(a) Ecrire sous forme de problème de programmation linéaire les équations permettant d'obtenir au moindre coût au moins 12 calories et au moins 7 vitamines (forme canonique).*

   (b) Le polyèdre associé est-il borné ? Non vide ?

   (c) Donner la forme standard du problème associé : on obtient le problème (PL).

   (d) Calculer la solution de (PL).

2. *M. Durand souhaite s'approprier le marché alimentaire précédent avec deux nouveaux types d'aliments contenant vitamines (resp. calories) à l'état pur :*

	Type A	*Type B*
calories	*0*	*1*
vitamines	*1*	*0*

M. Durand souhaite déterminer le prix de vente λ_2 et λ_1 de A et B par unité de poids u de sorte que son gain soit maximum pour la vente de 12 unités de poids u de B et 7 unités de poids u de A et que ces prix concurrencent ceux de M. Dupont.

(a) Ecrire la forme canonique (DL) du problème d'optimisation linéaire associé.

(b) Montrer que (DL) est le dual de (PL).

(c) Résoudre (DL).

2.5.21. Exercice :

Soit A symétrique, et $\widehat{x} \geq 0$ avec $A\widehat{x} = a$. Montrer que $\widehat{x}$ est solution optimale de

$$\min\{a^T x : Ax \geq a, x \geq 0\}.$$

2.6 La post optimisation

Dans les applications on a souvent besoin de connaître la variation de l'optimum après une perturbation de l'objectif ou du second membre. Ici, contrairement au corollaire 2.5.12, on ne s'intéresse pas seulement aux petites perturbations. Considérons pour un $\alpha \in \mathbb{R}$ les deux problèmes suivants

$$(P)_\alpha : \begin{cases} \max fx \\ Ax = a(\alpha) := a(0) + \alpha\Delta a \\ x \geq 0 \end{cases}$$

avec valeur optimale $v(\alpha)$, polyèdre de solutions réalisables P_α, et

$$(\widetilde{P})_\alpha : \begin{cases} \max f(\alpha)x, \qquad f(\alpha) := f(0) + \alpha\Delta f \\ Ax = a(0) \\ x \geq 0 \end{cases}$$

avec valeur optimale $\widetilde{v}(\alpha)$, et polyèdre de solutions réalisables P_0. Qu'est-ce qu'on peut dire sur le graphe des fonctions v et $\widetilde{v}$?

Notons que, pour $(P)_\alpha$ et $(\widetilde{P})_\alpha$, la matrice de coefficients ne dépend pas de α. Par conséquent, une base pour $(P)_\alpha$ restera une base pour tout autre valeur du paramètre, mais elle ne restera peut-être pas réalisable car le point de base associé dépend de α (par contre, le vecteur coûts réduits de $(P)_\alpha$ ne dépend pas de α). Pour expliciter cette dépendance de α on écrira $x(I)(\alpha)$ pour un point de base de $(P)_\alpha$, et on observe que

$$x(I)(\alpha)_I =: t(I)(\alpha) = t(I)(\overline{\alpha}) + (\alpha - \overline{\alpha})(A^I)^{-1}\Delta a$$

et donc $x(I)(\alpha) = x(I)(\overline{\alpha}) + (\alpha - \overline{\alpha})\Delta x(I)$, avec $\Delta x(I)$ le point de base associé au second membre Δa. D'une manière similaire, pour le problème $(\widetilde{P})_\alpha$ il se peut que la CSO soit valable seulement pour certains paramètres α. on écrira plus explicitement

$$d(I)(\alpha) = d(I)(\overline{\alpha}) + (\alpha - \overline{\alpha})\Delta d(I),$$

avec $\Delta d(I)$ le vecteur coûts réduits pour l'objectif Δf.

2.6.1. Théorème de post optimisation
Soit $P_0 \neq \emptyset$. Les propriétés suivantes sont valables.
(a) $M := \{\alpha \in \mathbb{R} : P_\alpha \neq \emptyset\}$ est un intervalle fermé.
(b) $\exists \alpha \in M : v(\alpha) < \infty$ si et seulement si $\forall \alpha \in M : v(\alpha) < \infty$.
(c) Soit I une base (de $(P)_\alpha$ ou de $(\widetilde{P})_\alpha$), et

$$M_1(I) = \{\alpha \in \mathbb{R} : \quad I \text{ est réalisable pour } (P)_\alpha\},$$
$$M_2(I) = \{\alpha \in \mathbb{R} : \quad I \text{ vérifie la CSO (2.3.3) pour } (\widetilde{P})_\alpha\},$$

alors $M_1(I)$ et $M_2(I)$ sont des intervalles fermés "facilement" calculables.
(d) $v : M \mapsto \mathbb{R} \cup \{\infty\}$ est concave et affine par morceaux.
(e) $\widetilde{v} : \mathbb{R} \mapsto \mathbb{R} \cup \{\infty\}$ est convexe et affine par morceaux.

Démonstration. **(a)** Soient

$$E := \{\begin{bmatrix} x \\ \alpha \end{bmatrix} : Ax = a(\alpha), x \geq 0\}, \qquad \Pi(\begin{bmatrix} x \\ \alpha \end{bmatrix}) = \alpha,$$

alors E est un polyèdre et donc convexe. Comme Π est une application linéaire, on en déduit que $M = \Pi(E)$ est une partie convexe de $\mathbb{R}$. De plus, E est non vide car $0 \in M$ par hypothèse. Il reste seulement à montrer que M est fermé. Si $v := \sup\{\alpha : \alpha \in \Pi(E)\} < +\infty$, alors v est la valeur optimale d'un programme linéaire borné. D'après le corollaire 2.2.5, nous pouvons déduire que la valeur optimale est atteinte dans E, et donc $v \in M$. En raisonnant de la même manière pour $\inf\{\alpha : \alpha \in \Pi(E)\}$, nous déduisons que M est fermé.
(b) L'implication $\Longleftarrow$ est triviale, car $0 \in M$. Soit $\alpha' \in M$ avec $v(\alpha') < \infty$, c'est-à-dire, $(P)_{\alpha'}$ admet une valeur optimale fini. D'après le théorème 2.5.7 de dualité, nous déduisons que le polyèdre dual est non vide, mais ce polyèdre ne dépend pas de α. Aussi, d'après le théorème 2.5.7 il s'en suit que, $\forall \alpha \in M$, soit $(P)_\alpha$ est impossible ($\alpha \notin M$) soit borné ($\alpha \in M$ et $v(\alpha) < \infty$).
(c) D'après définition,

$$M_1(I) = \{\alpha \in \mathbb{R} : 0 \leq t(I)(\alpha) = t(I)(0) + \alpha(A^I)^{-1}\Delta a\},$$
$$M_2(I) = \{\alpha \in \mathbb{R} : d(I)(\alpha) = d(I)(0) + \alpha\Delta d(I) \leq 0\},$$

sont des polyèdres.
(d) D'après (b), il y a rien à démontrer si $v(\alpha) = +\infty$ pour un $\alpha \in M$. Sinon, $\forall \alpha \in M$: $v(\alpha) < \infty$, et il existe une base réalisable I_α vérifiant la CSO pour tout $\alpha \in M$, c'est-à-dire, $v(\alpha) = fx(I_\alpha)$. Soient $\theta \in (0,1)$, $\alpha, \beta \in M$, et $\overline{\alpha} := \theta\alpha + (1-\theta)\beta$, alors

$$\overline{x} := \theta x(I_\alpha) + (1-\theta)x(I_\beta) \in P_{\overline{\alpha}}$$

(car $A\overline{x} = \theta a(\alpha) + (1-\theta)a(\beta) = a(\overline{\alpha})$), et

$$v(\overline{\alpha}) \geq f\overline{x} = \theta v(\alpha) + (1-\theta)v(\beta),$$

d'où la concavité de v. Finalement, v est affine sur chaque $M_1(I_\alpha)$.
(e) Cette propriété découle de (d) et du théorème 2.5.7 de la dualité. Le lecteur intéressé pourrait aussi donner une preuve directe. □

Les graphes de v ou de $\tilde{v}$ semblent essentiels pour "vendre" un résultat mathématique auprès d'un patron d'entreprise, reste à décrire comment obtenir en pratique la forme exacte de cette courbe sans trop d'efforts.

2.6.2. Schéma de résolution de $(\widetilde{P})_\alpha$ pour $\alpha \in [\alpha_0, +\infty)$

Trouver I_0 base réalisable vérifiant la CSO pour $(\widetilde{P})_{\alpha_0}$ par Simplex
Calculer pour $k = 0, 1, 2, ...$
déterminer α_{k+1} avec $[\alpha_k, \alpha_{k+1}] = [\alpha_k, +\infty] \cap M_2(I_k)$.

STOP si $\alpha_{k+1} = \infty$ *(*$\forall \alpha \geq \alpha_k$ *:* $\widetilde{v}(\alpha) = f(\alpha)x(I_k)$*)*
sinon faire changements de base $I_k \to I_{k+1}$ *comme suit*
Soit $J = \{j : d(I)(\alpha_{k+1})^j = 0\}$*. Résoudre par Simplex le problème auxiliaire*
$(\widetilde{PA}) : \max\{\Delta f^J y : A^J y = a(0), y \geq 0\}$ *avec base initiale* I_k*.*
STOP si $(\widetilde{PA})$ *est non borné (*$\forall \alpha > \alpha_{k+1}$ *:* $\widetilde{v}(\alpha) = +\infty$*),*
sinon I_{k+1} *base finale de SIMPLEX pour* $(\widetilde{PA})$*.*

Le problème auxiliaire $(\widetilde{PA})$ a été introduit pour faire disparaître (entrer en base) les indices "gênants" s avec

$$d(I)(\alpha_{k+1})^s = 0, \qquad \text{et} \quad \Delta d(I)^s > 0,$$

car par construction ces indices empêchent I de vérifier la CSO pour $(\widetilde{P})_\alpha$ pour $\alpha > \alpha_{k+1}$. D'un autre coté, on souhaite garder la CSO pour $\alpha = \alpha_{k+1}$, d'où la restriction $I \subset J$.

Un certain nombre de propriétés du schéma 2.6.2 doivent encore être démontrées. La propriété énoncée dans le cas $\alpha_{k+1} = +\infty$ est facilement vérifiée en revenant à la définition de $M_2(I_k)$. Aussi, on vérifie aisément que $I_k \subset J$ par construction, et que I_k, étant une base réalisable pour $(\widetilde{P})_{\alpha_{k+1}}$, restera base réalisable pour $(\widetilde{PA})$. De même, I_{k+1} est également une base réalisable pour $(\widetilde{P})_{\alpha_{k+1}}$. Reste à démontrer :

2.6.3. Lemme
(a) Si $(\widetilde{PA})$ *n'est pas borné alors* $\forall \alpha > \alpha_{k+1}$ *:* $\widetilde{v}(\alpha) = +\infty$*.*
(b) $d(I_k)(\alpha_{k+1}) = d(I_{k+1})(\alpha_{k+1})$ *(on préserve la CSO).*
(c) Pour $k \geq 1$ *nous avons* $\alpha_{k+1} > \alpha_k$ *(absence d'indices "gênants" pour* $I = I_{k+1}$*).*

Démonstration. **(b)** Si $I \subset J$ est la base de départ de $(\widetilde{PA})$ et $I' \subset J$ une base intermédiaire ou la base finale, alors (on omet l'argument α_{k+1})

$$d(I') - d(I) = -f^{I'}T(I') + f^I T(I)^{I'}T(I') = -d(I)^{I'}T(I') = 0$$

par définition de J.
(a) D'après l'hypothèse, on trouve $I \subset J$ base réalisable et $s \in J \setminus I$ de sorte que, pour les quantités simpliciales du problème $(\widetilde{PA})$,

$$\Delta d(I)^s > 0, \qquad T(I)^s \leq 0.$$

Par conséquent, $d(I)(\alpha_{k+1})^s = 0$ d'après la preuve de (b), et pour $\alpha > \alpha_{k+1}$

$$f(\alpha)v(I)^s = d(I)(\alpha)^s = (\alpha - \alpha_{k+1})\Delta d(I)^s > 0.$$

Comme le polyèdre de $(\widetilde{P})_\alpha$ ne dépend pas de α, nous déduisons de $T(I)^s \leq 0$ que $D^+(x(I), v(I)^s) \subset P_0$ et alors $(\widetilde{P})_\alpha$ n'est pas borné pour $\alpha > \alpha_{k+1}$.
(c) Pour $k \geq 1$ nous avons d'après la CSO pour I_k pour $(\widetilde{PA})$ et pour $(\widetilde{P})_{\alpha_k}$ pour tout j

$$d(I_k)(\alpha_k)^j \leq 0, \quad \text{et} \quad \Big(d(I_k)(\alpha_k)^j = 0 \Longrightarrow \Delta d(I_k)^j \leq 0\Big).$$

Par conséquent, I_k vérifie la CSO pour $\alpha > \alpha_k$ tant que

$$\alpha - \alpha_k \leq \min\{-\frac{d(I)(\alpha_k)}{\Delta d(I_k)} : \Delta d(I_k) > 0\} > 0.$$

□

Le calcul décrit dans le schéma 2.6.2 peut être effectué en ajoutant quelques lignes au schéma 2.4.7

$T(I)^{I(prec)}$ si nécessaire	$(A^I)^{-1}$ lignes énumérées par I	I	$t(I)$ $= (A^I)^{-1}a$	$T(I)^s$ $= (A^I)^{-1}A^s$	rapport (choix de r)
$f(\alpha_k)^I$	$\lambda(\alpha_k) = f(\alpha_k)^I(A^I)^{-1}$	$d(I)(\alpha_k)$			
Δf^I	$\Delta f^I(A^I)^{-1}$	$\Delta d(I)$ (calcul de α_{k+1})			
		$d(I)(\alpha_{k+1})$ (choix de $s \in J$, pivotage restreint selon $\widetilde{(PA)}$)			

Généralement, il y a qu'un seul indice "gênant", et $\widetilde{(PA)}$ est résolu par 1 ou 2 itérations de Simplex.

2.6.4. Exemple : *Considérons pour $\alpha \in [0, +\infty)$ le problème*

$$(\widetilde{P})_\alpha : \left\{ \begin{array}{c} \max(x_1 + \alpha x_2) \\ 2x_1 + x_2 \leq 6, x_1 + 2x_2 \leq 6, x_1, x_2 \geq 0 \end{array} \right.$$

ce qui se résout graphiquement assez facilement : augmenter α correspond à une rotation des courbes de niveau. Par conséquent, presque sans calcul on devrait trouver la solution optimale $(3,0)^T$ pour $\alpha \in [0, 1/2]$, la solution optimale $(2,2)^T$ pour $\alpha \in [1/2, 2]$, et la solution optimale $(0,3)^T$ pour $\alpha \geq 2$. Vérifions que l'on trouve bien la même chose avec notre schéma. D'abord, en passant à la forme standard

$A =$	2	1	1	0	6	$= a$
	1	2	0	1	6	
$f(0) =$	1	0	0	0		
$\Delta f =$	0	1	0	0		

On pose $\alpha_0 = 0$ et on prend la base $I = (1,4)$ de départ (qui va vérifier la CSO pour $\alpha = \alpha_0$. Dans le premier schéma on calculera $\alpha_1 = 1/2$, car $d(I)(\alpha) = d(I)(\alpha_0) + \Delta d(I)(\alpha - \alpha_0) \not\leq 0$ pour $\alpha - \alpha_0 > 1/2$.

			1/2	*0*	*1*	*3*	*1/2*	*6*	
			-1/2	*1*	*4*	*3*	*3/2*	*2*	*r=4*
$f(\alpha_0)^I =$	*1*	*0*	*1/2*	*0*	$d(I)(\alpha_0) = (0, -1/2, -1/2, 0)$				
$\Delta f^I =$	*0*	*0*	*0*	*0*	$\Delta d(I) = (0, 1, 0, 0)$				*seul indice gênant $s = 2$*
			1/2	*0*	$d(I)(\alpha_1) = (0, 0, -1/2, 0)$				$\alpha_1 - \alpha_0 = 1/2$

Ce schéma se répète maintenant deux fois, et on note bien que, pour $\alpha = \alpha_k$, on garde le même vecteur coût réduit $d(I)(\alpha)$ après changement de base

	1	*-1/3*	*2/3*	*-1/3*	*1*	*2*	*2/3*	*3*	*r=1*
	0	*2/3*	*-1/3*	*2/3*	*2*	*2*	*-1/3*	*/*	
$f(\alpha_1)^I =$	*1*	*1/2*	*1/2*	*0*	$d(I)(\alpha_1) = (0, 0, -1/2, 0)$				
$\Delta f^I =$	*0*	*1*	*-1/3*	*2/3*	$\Delta d(I) = (0, 0, 1/3, -2/3)$				*seul indice gênant $s = 3$*
			1/2	*0*	$d(I)(\alpha_2) = (0, 0, 0, -1)$				$\alpha_2 - \alpha_1 = 3/2$
	3/2	*0*	*1*	*-1/2*	*3*	*3*			
	1/2	*1*	*0*	*1/2*	*2*	*3*			
$f(\alpha_2)^I =$	*0*	*2*	*0*	*1*	$d(I)(\alpha_2) = (0, 0, 0, -1)$				
$\Delta f^I =$	*0*	*1*	*0*	*1/2*	$\Delta d(I) = (-1/2, 0, 0, -1/2)$				*pas d'indice genant*

Comme il n'y a plus d'indices gênant, la dernière base convient bien pour $\alpha \in [\alpha_2, +\infty) = [2, +\infty)$.

Le problème $(P)_\alpha$ pour $\alpha \in [\alpha_0, \infty)$ peut être résolu de deux manières : soit on passe au problème dual (qui est du type $(\widetilde{P})_\alpha$ après écriture sous forme standard), soit on élimine les indices "gênants" $r \in I$ avec $x(I)(\alpha_{k+1})_r = 0$ et $\Delta x(I)_r < 0$ en s'inspirant de la méthode Simplex dual 2.5.14 pour préserver la CSO. Nous donnons ici un descriptif de cette deuxième approche, et laissons au lecteur le soin de vérifier les détails.

2.6.5. Schéma de résolution de $(P)_\alpha$ pour $\alpha \in [\alpha_0, +\infty)$

Trouver I_0 *base réalisable vérifiant la CSO pour* $(P)_{\alpha_0}$ *par Simplex*
Calculer pour $k = 0, 1, 2, ...$
déterminer α_{k+1} *avec* $[\alpha_k, \alpha_{k+1}] = [\alpha_k, +\infty] \cap M_1(I)$.
STOP si $\alpha_{k+1} = \infty$ *(*$\forall \alpha \geq \alpha_k$ *:* $v(\alpha) = fx(I_k)(\alpha)$*)*
sinon faire changement de base $I_k \to I_{k+1}$ *comme suit*
Poser $I \longleftarrow I_k$.
Tant qu'il existe $r \in I$ *avec* $x(I)(\alpha_{k+1})_r = 0$ *et* $\Delta x(I)_r < 0$
Calculer $T(I)_r$.
STOP si $T(I)_r^{\overline{I}} \geq 0$ *(*$\forall \alpha > \alpha_{k+1}$ *:* $(P)_\alpha$ *impossible)*
sinon trouver $s \in \overline{I}$ *avec* $T(I)_r^s < 0$ *et*
$d(I)^s / T(I)_r^s = \min\{d(I)^j / T(I)_r^j : j \in \overline{I}, T(I)_r^j < 0\}$.
$I \leftarrow I - r + s$
$I_{k+1} \leftarrow I$.

2.6.6. Exercice : *post-optimisation*

Soit $\alpha \geq 0$ *un scalaire et notons par* $v(\alpha)$ *la solution du problème*

$$(PL)_\alpha : \quad \max\{fx : Ax = a + \alpha \Delta a, \quad x \geq 0\}.$$

(a) *Justifier l'algorithme (...ref ...) pour le calcul de* $v(\alpha), \quad \alpha \geq 0$

(b) *Soient* $\alpha_1 > \alpha_0 > \alpha_{-1} := 0$, *et* $I_1 := I_0 - r + s$. *On suppose pour* $j = 0, 1$ *que* I_j *est une base réalisable pour* $(PL)_\alpha$, $\alpha_{j-1} \leq \alpha \leq \alpha_j$, *vérifiant la condition d'optimalité, et que* I_0 *n'est plus réalisable pour* $\alpha > \alpha_0$. *Montrer que (différence des dérivées directionnelles)*

$$v'(\alpha_0+) - v'(\alpha_0-) = -\frac{t_r(I_0)}{\alpha_0} \frac{d(I_0)^s}{T(I_0)_r^s} \leq 0.$$

(c) *Pour les données*

$$A = \begin{bmatrix} 1 & 1 & -2 & 0 \\ 1 & 0 & -1 & 1 \end{bmatrix}, \quad a = \begin{bmatrix} 4 \\ 3 \end{bmatrix}, \quad \Delta a = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad f = [3, 1, -7, -2],$$

calculer $v(\alpha)$ *pour* $\alpha \geq 0$, *et déterminer* $\max\{v(\alpha) : \alpha \geq 0\}$.

2.7 La décomposition de Dantzig et Wolfe

La méthode de décomposition de Dantzig-Wolfe s'applique à des problèmes de type $\max\{fx : x \in P_1 \cap P_2\}$ où il est relativement simple de résoudre le problème relâché $\max\{fx : x \in P_2\}$ ou de trouver des points extrêmes de ce problème. Un exemple est donné par la gestion simultanée de plusieurs usines, qui ont leur propre gestion de ressources traduite par les contraintes de P_2, et certaines ressources en commun à partager par ces usines (traduites par P_1), on parle des contraintes couplantes. Ici la matrice de coefficients du problème initial admet la structure (on notera par $*$ les blocs non nuls pas forcément carrés, et tout autre bloc ne contient que des 0)

$$\left[\begin{array}{cccc|l} * & * & * & * & \} \text{ contraintes } P_1 \\ * & & & & \\ & * & & & \\ & & * & & \} \text{ contraintes } P_2 \\ & & & * & \end{array}\right. \tag{2.15}$$

et le problème rélaché consiste à résoudre (par exemple en parallèle) une grande quantité de petits programmes linéaires indépendants. D'autres applications sont des problèmes de flot ou de cheminement (contraintes traduites par P_2) auxquels on ajoute un certain nombre de contraintes linéaires "compliquées" [7] traduites par P_1.

La philosophie de la méthode consiste à répartir a priori les ressources communes, en fixant un prix pour ces ressources aux différentes usines (esclaves) [8] qui en fonction de ces prix optimisent leurs plans de production séparément. Ensuite, dans un programme maître, on vérifie si on a trouvé une répartition optimale (CSO pour le programme maître) ou s'il faudra fixer d'autres prix (choix de s pour le programme maître). On consultera avec grand plaisir le livre de Dantzig [9] à ce propos, qui propose une pièce de théâtre pour décrire l'interaction entre le maître et ses esclaves...

Pour être plus précis, soit à résoudre

$$\max\{fx : Ax = a, x \in P_2\}$$

avec un polyèdre P_2. Notons $Z = \left[\ z^1, z^2, ..., z^{p+q}\ \right]$, avec $z^1, ..., z^p$ les points extrêmes de P_2, et $z^{p+1}, ..., z^{p+q}$ les rayons extrêmes [10] de P_2, alors d'après Théorème 2.2.3

$$\max\{fx : Ax = a, x \in P_2\} = \max\{fx : Ax = a, x = \sum_{j=1}^{p+q} z^j \widetilde{x}_j, \sum_{j=1}^{p} \widetilde{x}_j = 1, \forall j : \widetilde{x}_j \geq 0\}$$

$$= \max\{\widetilde{f}\widetilde{x} : \widetilde{A}\widetilde{x} = \widetilde{a}, \widetilde{x} \geq 0\}, \quad \widetilde{f} = fZ, \quad \widetilde{a} = \left[\begin{array}{c} a \\ 1 \end{array}\right], \quad \widetilde{A} = \left[\begin{array}{c} AZ \\ 1, ..., 1, 0, ..., 0 \end{array}\right],$$

7. A titre d'exemple, les problèmes de multi-flot, où chaque flot est soumis a une loi de conservation (contraintes P_2), et la somme des flots sur chaque arc ne devrait pas dépasser une certaine limite (contraintes P_1).

8. Plus précisément, on communique aux esclaves des bénéfices unitaires modifiées $f^j - \mu A^j$ ou le montant μA^j correspond a un prix unitaire, une sorte de moyenne pour les besoins au niveau des ressources communes : comme dans un problème de distribution de ressources la quantité A^j_k donne la quantité nécessaire de la ressource k pour produire une unité du produit j, la composante μ^k correspond alors à un prix unitaire pour la ressource k.

9. G.B. Dantzig, Applications et prolongements de la programmation linéaire, Dunod (1966), chapitre 14-2

10. A un multiple scalaire positif près, les points extrêmes de $P_2 \cap \{x : x_1 + ... + x_n = 1\}$ si P_2 est sous forme standard, à revoir dans un autre Exo...

et la dernière ligne de $\widetilde{A}$ comportant q zéros. On appellera ce dernier problème le programme maître. Bien entendu, généralement il coûte trop cher de pré-calculer toute la matrice Z ou $\widetilde{A}$. Mais la forme révisée 2.4.5 de Simplex pour le programme maître ne nécessite la connaissance de la matrice entière qu'au moment du calcul des composantes du vecteur coûts réduits. Plus précisément, si on dispose d'une autre procédure de pricing et de génération de la colonne $\widetilde{A}^s$ dans 2.4.5, ceci nous permettra de ne plus faire appel du tout à cette matrice $\widetilde{A}$. Le pricing mvp (recherche de la plus grande composante de $d(I)$) peut être mis en œuvre comme suit.

2.7.1. "mvp" pricing pour le mâître dans la décomposition de Dantzig et Wolfe
Soit donnée une base réalisable I pour le maître. Notons le vecteur multiplicateur par

$$(\mu, \gamma) = \widetilde{f}^I(\widetilde{A}^I)^{-1}, \qquad \textit{avec} \quad \gamma \in \mathbb{R}.$$

Résolvons le programme esclave $\max\{(f-\mu A)z : z \in P_2\}$ *par Simplex.*
(a) Si le problème esclave n'est pas borné, nous avons trouvé un rayon extrême $z^{(s)}$ de P_2 avec $(f-\mu A)z^{(s)} > 0$, et nous posons $\widetilde{A}^{(s)} = \begin{pmatrix} Az^{(s)} \\ 0 \end{pmatrix}$.
(b) Supposons que le problème esclave admet une valeur optimale finie γ' et une solution optimale $z^{(s)}$. Si $\gamma' \leq \gamma$ alors la CSO est valable pour le programme maître, sinon nous posons $\widetilde{A}^{(s)} = \begin{pmatrix} Az^{(s)} \\ 1 \end{pmatrix}$.

Démonstration. Nous notons d'abord que la composante d'indice s du vecteur coût réduit du maître prend la forme

$$\begin{aligned} d(I)^s &= \widetilde{f}^s - (\mu,\gamma)\widetilde{A}^s \\ &= (f-\mu A)z^{(s)} - \begin{cases} \gamma & \text{si } s \in \{1,...,p\} \text{ (c'est-à-dire, } z^{(s)} \text{ point extrême de } P_2,) \\ 0 & \text{si } s \in \{p+1,...,p+q\} \text{ (c'est-à-dire, } z^{(s)} \text{ rayon extrême de } P_2.) \end{cases} \end{aligned}$$

Sous l'hypothèse de la partie (a), Simplex pour l'esclave s'arrête car il a trouvé une colonne $T \leq 0$ dans son tableau simplicial. Par conséquent, la direction privilégiée associée est un rayon extrême de P_2 et une direction de pente pour l'esclave. Par conséquent, on a trouvé un indice colonne s avec $d(I)^s > 0$ pour le maître (mais on ne peut pas assurer d'avoir trouvé la plus grande composante de $d(I)^s$). Ceci démontre (a).
Si comme dans (b) l'optimum de l'esclave est fini, alors, d'après le corollaire 2.2.5, pour tout élément du cône asymptotique de l'esclave, et plus particulièrement pour les rayons extrêmes z de l'esclave nous avons $(f-\mu A)z \leq 0$, ce qui implique $d(I)^{(p+1,...,p+q)} \leq 0$. Par conséquent, comme la résolution de l'esclave revient à trouver la plus grande valeur réalisée par les points extrêmes de P_2, nous avons

$$\gamma' - \gamma = \max\{d(I)^j : j = 1,...,p\}, \qquad \max\{0, \gamma' - \gamma\} = \max\{d(I)^j : j = 1,...,p+q\},$$

d'où la conclusion de (b) (avec pricing mvp). □

Quelques commentaires sur cette méthode de décomposition de Dantzig et Wolfe : d'abord, il est nécessaire de stocker pas seulement (l'inverse ou la factorisation de) $\widetilde{A}^I$ mais aussi Z^I, pour obtenir a partir d'une solution optimale du maître une solution optimale du problème initial. La première base pour le maître nécessite déjà la connaissance d'un certain nombre de points extrêmes/rayons extrêmes de P_2, ici on applique très avantageusement la méthode de résolution

en deux phases 2.4.9, sachant que pour démarrer la phase 1 pour le maître on n'a pas besoin de connaître (une partie de) $\widetilde{A}$. Aussi, pour démarrer le calcul de l'esclave on peut se servir de la base finale de la précédente résolution de l'esclave par Simplex (sauf en prémière itération).

La décomposition de Dantzig–Wolfe rend possible la planification globale de toute une organisation sans que l'autorité centrale (le maître) ait besoin des informations technologiques complètes relatives aux différentes parties (esclaves) constituantes. Suivant Dantzig, on parle d'une "planification centrale sans informations complète du centre planificateur" : le centre propose des bénéfices unitaires $f - \mu A$ modifiés au différentes parties, et modifie le vecteur μ (prix unitaires pour les ressources communes) en fonction de la réponse des parties, pour trouver finalement la meilleure moyenne des différentes réponses.

2.7.2. Remarque : variante pour les esclaves séparables
Dans le cas (2.15) où l'esclave est séparable , c'est-à-dire, constitué de plusieurs sous-problèmes (esclaves) indépendants

$$\max\{gx : x \in P_2\} = \sum_{j=1}^{r} \max\{g^{(j)}x_{(j)} : x_{(j)} \in P_{(j)}\}$$

avec $[g^{(1)}, ..., g^{(r)}]$ *une partition du vecteur ligne* g *et des partitions similaires pour le vecteur* x*, il existe une autre variante de construction du maître : avec* $Z_{(j)}$ *la matrice de points extrêmes/rayon extrêmes de* $P_{(j)}$*, on prend comme premier bloc ligne de* $\widetilde{A}$ *la matrice* $[AZ_{(1)}, ..., AZ_{(r)}]$*, à laquelle on ajoute* r *lignes comportant des* 0 *et des* 1 *pour tenir compte des combinaisons convexes pour chaque* $P_{(j)}$*. Cette approche a l'avantage de pouvoir plus facilement mettre en œuvre un* partial pricing *: il suffit de questionner les esclaves l'un après l'autre jusqu'au moment où l'un entre eux propose une valeur optimale suffisamment élevée pour obtenir un* $d(I)^s > 0$*.*

2.7.3. Remarque : les programmes maître restreint
Dans le d'un seul ou de plusieurs esclaves, il y a une autre variante de résolution suivant Dantzig et Wolfe : au cas ou il est possible de stocker un nombre un peu plus important de colonnes, disons, $\widetilde{A}^J$ *et* Z^J *avec* $J = J(I) \supsetneq I$*, on peut également envisager de résoudre à chaque itération du maître le programme maître restreint*

$$\max\{\widetilde{f}^J\widetilde{x}_J : \widetilde{A}^J\widetilde{x}_J = \widetilde{a}, \widetilde{x}_J \geq 0\},$$

en prenant comme base initiale la base finale de l'étape précédente. Ensuite, on vérifie la CSO du problème maître non restreint en faisant appel aux esclaves. Par exemple, si on garde en mémoire toutes les réponses des esclaves, ceci revient à trouver la meilleure moyenne entre ces plans de production, et permet de communiquer un prix de mieux en mieux ajusté aux esclaves dans leur résolution suivante. L'approche du programme maître restreint peut être avantageux si le programme maître comporte peu de contraintes, car on fait moins appel aux esclaves (une sorte de partial pricing).

Notons que toute valeur optimale d'un programme maître restreint nous donne une borne inférieure pour l'optimum du problème d'origine, et cette borne croît tant qu'à chaque itération du maître on ajoute des éléments à J*. Finalement, notons que, pour le programme esclave de paramètre* μ *et de valeur optimale* γ'

$$\mu a + \gamma' = \max_{x \in P_2}(\mu a + (f - \mu A)x) = \max_{x \in P_2}(fx + \mu(a - Ax)) \geq \max_{x \in P_1 \cap P_2}(fx + \mu(a - Ax)) = \max_{x \in P_1 \cap P_2} fx,$$

et alors $\mu a + \gamma'$ est une borne supérieure[11] *pour l'optimum du problème d'origine. On dispose donc, à chaque itération, d'un encadrement de la valeur optimale cherchée et l'on pourra interrompre les calculs dès que l'on entre dans une tolerance ϵ fixée à l'avance.*[12]

Pour le problème des coupes on suit une approche similaire de génération successive de colonnes de la matrice de coefficients : la $s$ième colonne est générée seulement au moment où elle entre en base.

2.7.4. Exercice : problème de coupes
A partir d'un nombre illimité de barres de longueur L, on doit satisfaire une demande de livraison de b_j barres de longueur $\ell_j \leq L$, $j = 1, 2, ..., m$, tout en minimisant la chute. On appelle coupe admissible un vecteur $\vec{n} = (n_1, ..., n_m)^t$ d'entiers ≥ 0 de sorte que $n_1\ell_1 + ... + n_m\ell_m = L - S(\vec{n})$, avec $S(\vec{n}) \geq 0$ la chute si on coupe une barre de longueur L en n_j barres de longueur ℓ_j pour $j = 1, 2, ..., m$.

En notant par $x_{\vec{n}} \geq 0$ le nombre de coupes de type $\vec{n}$, et par $\mathcal{N}$ l'ensemble des coupes admissibles, il reste à résoudre

$$\min \sum_{\vec{n}\in\mathcal{N}} S(\vec{n})x_{\vec{n}} : Ax \geq b, x \geq 0\}, \quad avec \quad A = (\vec{n})_{\vec{n}\in\mathcal{N}}.$$

Ici encore il n'est pas envisageable de calculer a priori la matrice A, qui généralement admet trop de colonnes. Montrer qu'en résolvant un problème de type "sac-à-dos" (voir chapitre 3.7)

$$\begin{cases} \max(g^1 n_1 + ... + g^m n_m) \\ \ell_1 n_1 + ... + \ell_m n_m \leq L, \quad \forall j : n_j \in \{0, 1, 2, 3, ...\} \end{cases}$$

pour un g approprié, on peut créer une boîte noire pour le pricing et pour la génération de colonnes dans la résolution de notre problème de coupes par Simplex.

Comment change l'approche si différentes barres de longueur L_k, $k = 1, 2, ..., r$ sont à notre disposition ?

2.8 La méthode de Benders

La méthode de Benders est particulièrement intéressante pour les problèmes ayant deux groupes de variables x' et x'', qui deviennent "facile" a résoudre si on fixe x'' et on optimise par

11. En notant $w(\mu) = \max_{x\in P_2}(\mu a + (f - \mu A)x)$ on sait d'après le théorème 2.6.1 de post optimisation (variation de l'objectif) que w est convexe, et sa restriction sur un segment est linéaire par morceaux : w donc pas de classe $\mathcal{C}^1$. Ces propriétés seront d'ailleurs étudiés dans un contexte plus général dans le chapitre **??** où on montre aussi que la valeur minimale pour w pour $\mu \in \mathbb{R}^{1\times m}$ coincide avec la valeur optimale de notre problème d'origine. Malheureusement, pour les différents prix μ rencontrés dans la méthode de Dantzig-Wolfe, on ne peut pas mettre en évidence une monotonie de la borne supérieure $w(\mu)$. C'est une des raisons pourquoi au lieu d'un programme maître on applique parfois aussi des méthodes de sous-gradients, particulièrement adaptés aux fonctions convexes/concaves pas forcément de classe $\mathcal{C}^1$. On consultera pour ce sujet le livre de Michel Minoux, Programmation mathématique, Tome 2, Dunod (1983), en particulier le chapitre 8.2.4.

12. Un tel encadrement, également obtenu pour les méthodes Benders et primal-dual explicités ci-dessous, est devenu très populaire car aux applications il suffit souvent de connaître une "bonne" au lieu de la meilleure solution.

rapport à x'. On peut s'imaginer plusieurs usines qui certains objets séparément (variable x') et certains objets en commun (variable x'', on parle des variables couplantes). Un autre exemple est donné par les problèmes de type *multi-période* : on doit proposer une planification d'une usine avec entrepôt sur plusieurs années. La gestion sur chaque année (production, chiffres de vente etc) est décrit à l'aide de la variable x', mais on permet de stocker une partie de la production (décrite par la variable x'') pour la vendre à un moment ultérieur (ce qui serait intéressant si les prix fluctuent beaucoup au cours des années). Ici on se ramène à un polyèdre avec matrice de coefficients de la forme

$$\underbrace{\begin{bmatrix} * & & & & * \\ & * & & & * \\ & & * & & * \\ & & & * & * \end{bmatrix}}_{\text{gestion autonome}} \quad \underbrace{\phantom{\begin{bmatrix} * \\ * \\ * \\ * \end{bmatrix}}}_{F}$$

c'est-à-dire, essentiellement la structure de la transposé de la matrice (2.15). Suivant la philosophie du chapitre précédent, ici un centre planificateur (maître) imposera l'état de l'entrepôt x'' et les usines (l'esclave) adapte au mieux ses degrés de liberté x'. En fonction de la réponse de l'esclave, le maître adaptera x'' et on recommence.

Dans la méthode de Benders, on applique essentiellement la décomposition de Dantzig-Wolfe à la forme standard du problème dual, en gardant en mémoire toutes les réponses des esclaves (voir Remarque 2.7.3). Autrement dit, on ajoute à chaque itération une colonne au problème maître du dual, ce qui revient à générer à chaque itération une contrainte supplémentaire (une ligne) pour le maître du primal. Pour des généralisations ultérieures (voir Remarque 2.8.3), on préfère ici de donner une formulation en termes du primal.

Soit à résoudre

$$\max\{f'x' + f''x'' : Bx' + Fx'' = b, x' \geq 0, x'' \in P\} \tag{2.16}$$

avec P un polyèdre. En fixant x'', on se retrouve avec un problème esclave

$$(Q(x'')) : \qquad \max\{f'x : Bx' = b - Fx'', x' \geq 0\},$$

avec dual

$$(Q^*(x'')) : \qquad \min\{\lambda(b - Fx'') : \lambda B \geq f'\},$$

c'est à dire, le dual de l'esclave admet la même forme que l'esclave du chapitre précédent, en particulier le polyèdre P_2 formé par ses solutions réalisables ne dépend pas de x''. On va donc successivement générer des points extrêmes et rayons extrêmes de P_2 pour construire le programme maître. En convenant que la valeur optimale d'un problème de maximisation impossible est égale à $-\infty$, le théorème 2.5.7 de la dualité nous dit que $(Q(x''))$ et $(Q^*(x''))$ ont la même valeur optimale, ce qui montre le résultat suivant.

2.8.1. Lemme : la transformation de Benders
Soit $L(x, \lambda) := f''x'' + \lambda(b - Fx'')$, alors la valeur optimale du problème (2.16) peut s'écrire de la manière

$$\max_{x'' \in P} \min_{\lambda \in P_2} L(x'', \lambda). \tag{2.17}$$

De plus, si le maximum dans (2.17) est atteint pour $x''_* \in P$, alors le minimum est atteint pour λ^* solution optimale de $(Q^*)_{x''_*}$, et en notant par x'_* le vecteur multiplicateur associé (ou toute autre solution optimale de $(Q)_{x''_*}$) alors (x'_*, x''_*) est une solution optimale de (2.16).

Nous avons alors transformé le problème (2.16) à l'aide du Lagangien. Cette approche sera étudiée plus en détail dans un cadre pas forcément linéaire au chapitre **??**.

Si on note par $\lambda_1, ..., \lambda_p$ les points extrêmes de P_2 et par $\lambda_{p+1}, ..., \lambda_{p+q}$ les rayons extrêmes, alors comme dans Corollaire 2.2.5 nous pouvons écrire (en tenant compte seulement des x'' avec $Q^*(x'')$ borné)

$$\begin{aligned}
&\max_{x''\in P}\min_{\lambda\in P_2} L(x'',\lambda)\\
&= \max\left\{\min_{j=1,...,p}\Big(f''x'' + \lambda^j(b-Fx'')\Big) : x''\in P, \forall j = p+1,...,q : \lambda^j(b-Fx'')\geq 0\right\},\\
&= \begin{cases} \max\gamma\\ (\gamma, x'')\in\mathbb{R}\times P,\\ \forall j = 1,...,p : f''x'' + \lambda^j(b-Fx'')\geq\gamma,\\ \forall j = p+1,...,q : \lambda^j(b-Fx'')\geq 0. \end{cases}
\end{aligned}$$

Le dernier problème est dit *programme maître*. Un *programme maître restreint* tiendra seulement compte d'une partie de ces points/rayons extrèmes, une de plus par itération du maître (comparer avec 2.7.3), jusqu'au moment ou la nouvelle contrainte ne changera plus le maximum. Ceci donne l'algorithme suivant.

2.8.2. Méthode de Benders

Invariant : P_{2p} *ensemble de certain points extrêmes de* P_2,
P_{2r} *ensemble de certain rayons extrêmes de* P_2,
Initialisation $P_{2p} = \emptyset$, $P_{2r} = \emptyset$
Itération : Pour $k = 0, 1, 2, ...$
Résoudre le programme maître restreint aux inconnues (γ, x'')
$\max(\gamma)$ *sous les contraintes* $x''\in P$ *et*
$\forall\lambda\in P_{2p} : f''x'' + \lambda(b-Fx'')\geq\gamma$
$\forall\lambda\in P_{2r} : \lambda(b-Fx'')\geq 0$
STOP si le programme maître restreint est impossible (c.-à-d., (2.16) est impossible)
Si le programme maître restreint n'est pas borné alors
$\gamma_k = +\infty$, x''_k *réalisable pour le maître restreint*
sinon notons une solution optimale par (γ_k, x''_k).

Résoudre le programme esclave $Q(x''_k)$ *(ou son dual* $Q^*(x''_k)$*)*
Si $Q(x'')$ *est impossible (*$Q^*(x'')$ *non borné) alors*
on dispose d'un rayon extrème λ *de* P_2 *avec* $\lambda(b-Fx'') < 0$
ajouter λ *à* P_{2r}
sinon on dispose d'une solution optimale λ *de* $Q^*(x''_k)$,
d'une solution optimale x'_k *de* $Q(x''_k)$, *et de l'optimum* γ'_k.
STOP si $\gamma_k = \gamma'_k$ *:* (x'_k, x''_k) *est solution optimale de (2.16)*
sinon ajouter λ *à* P_{2p}.
Propriétés : $\gamma_0 \geq \gamma_1 \geq .... \geq \max_j \gamma'_j$

Notons que la monotonie des γ_j provient du fait que, à chaque itération, on ajoute un élément à $P_{2r} \cup P_{2p}$. Aussi, par construction, toute valeur optimale de $Q(x'')$ pour un $x'' \in P$ est une borne inférieure de la valeur optimale de (2.16), d'où $\gamma_j \geq \gamma'_j$ et on obtient l'encadrement de la valeur optimale donné à la fin de 2.8.2 (qui peut servir comme condition d'arrêt en cas de convergence lente). Cet encadrement ensemble avec Lemme 2.8.1 montre d'ailleurs aussi que l'on peut arrêter l'algorithme au cas $\gamma_k = \gamma'_k$, en disposant d'une solution optimale de (2.16). En général, on n'observe pas une monotonie pour les quantités γ'_j, d'où l'intérêt de garder en mémoire le couple (x'_k, x''_k) réalisant la plus grande des valeurs γ'_j en cas d'arrêt prématuré.

Pour la résolution (de la forme standard) du programme maître restreint, on peut se servir de la base finale du maître restreint précédent, en ajoutant a la base finale la variable d'écart de la nouvelle contrainte : on peut montrer que cette base est non réalisable et vérifie la CSO, et on utilise Simplex dual. Parfois on préfère de résoudre le dual du programme maître par Simplex (car ici on ajoute une colonne, et l'ancienne base continue d'être réalisable). Notons aussi que pour la résolution de l'esclave on peut se servir de la base finale de l'esclave précédent, cette base étant dual réalisable mais pas forcément primal réalisable (on utilise par exemple Simplex dual pour $Q(x''_k)$). Finalement, la finitude de la méthode de Benders decoule du fait qu'il existe un nombre fini de points/rayons extrèmes de P_2 (mais on espère de pouvoir arrêter l'algorithme bien avant).

2.8.3. Remarque : extensions
Dans le descriptif de la méthode nous n'avons pas explicité le rôle de $x'' \in P$. Si P est un polyèdre, alors le programme maître peut être résolu par Simplex, mais d'autres répartitions x', x'' sont imaginables.

La méthode de Benders a été décrite pour la première fois en 1962 pour résoudre des programmes en variables mixtes, c'est-à-dire, une partie des variables sont astreintes à ne prendre que des valeurs entières. En incluant ces contraintes dans (2.16) par la condition $P \subset \mathbb{Z}^p$, nous voyons que la méthode de Benders permet de transformer un programme linéaire mixte en une suite de programmes linéaires en variables exclusivement entières (le maître) et en variables exclusivement continues (l'esclave).

Finalement, prenant comme point de départ le résultat du Lemme 2.8.1, la méthode de Benders peut être étendue à tout problème du type

$$\max_{\mu \in P} \min_{y \in S} \widetilde{L}(y, \mu) = -\min_{\mu \in P} \max_{y \in S} L(y, \mu)$$

pourvu que S est fini (pour assurer la convergence finie). Ici le programme maître restreint pour un $S_0 \subset S$ prendra la forme $\max\{\gamma : (\gamma, \mu) \in \mathbb{R} \times P, \forall y \in S_0 : \widetilde{L}(y, \mu) \geq \gamma\}$, c'est-à-dire, un programme linéaire si $\widetilde{L}$ est linéaire par rapport à μ et P est un polyèdre (comme c'est le cas pour le problème dual d'un problème non linéaire formé à l'aide du Lagrangien L au chapitre **??***).*

2.9 La méthode Simplex primal-dual

Etant données $A \in \mathbb{R}^{m \times n}$, $a \in \mathbb{R}^{m \times 1}$, $f \in \mathbb{R}^{1 \times n}$, on souhaite résoudre le problème sous forme canonique

$$(P): \qquad \max\{fx : Ax \leq a, x \geq 0\}, \quad A = A_L^J.$$

Le but de la méthode Simplex primal dual est de construire une suite de $x^{(k)}$ réalisable pour (P) et une suite de $\lambda_{(k)}$ réalisables pour le problème dual (D) de sorte que

$$fx^{(0)} \leq fx^{(1)} \leq fx^{(2)} \leq ... \leq \lambda_{(2)}a \leq \lambda_{(1)}a \leq \lambda^{(0)}a, \tag{2.18}$$

avec l'idée de terminer l'algorithme dès que l'encadrement de la valeur optimale est suffisamment fine. Notons le polyèdre des solutions réalisables de (P) par P, et celui du problème dual par D.

Pour un $\lambda \in D$ on considère l'ensemble

$$M = M(\lambda) = \{j \in J : \lambda A^j - f^j = 0\},$$

et le programme restreint associé

$$(P)_M \qquad \max\{f^M x_M : A^M x_M + y = a, x_M \geq 0, y \geq 0\}.$$

Nous appelons ce programme restreint car il est obtenu à partir de (P) en ajoutant la contrainte $x_{J\setminus M} = 0$.

On donnera d'abord la formulation de la méthode Simplex primal-dual, et on montrera ensuite les différents propriétés de la méthode.

2.9.1. Algorithme : La méthode Simplex primal-dual

Invariant : $\lambda \in D$
Phase 1 : *Calculer* $M = M(\lambda)$
Phase 2 : *Résoudre le programme restreint* $(P)_M$ *par SIMPLEX*
avec solution optimale $(\hat{x}_M, \hat{y})$, *base finale* $I \subset M \cup L$ *et vecteur multiplicateur* u.
Stop si $uA - f \geq 0$.
Phase 3 : *Mise à jour de* λ *comme suit :*
trouver le plus grand $\theta \in (0,1)$ *tel que* $(1-\theta)(\lambda A - f) + \theta(uA - f) \geq 0$,
poser $\lambda \leftarrow (1-\theta)\lambda + \theta u$ *et retourner en phase 1.*

Posons $\hat{x}_{J\setminus M} = 0$. On montrera dans l'exercice suivant étape par étape que, au cas d'arrêt, on a trouvé une solution optimale $\hat{x}$ de (P) et une solution optimale $\lambda + u$ de (D), et l'encadrement (2.18).

2.9.2. Exercice : Preuve de la méthode Simplex primal-dual

(a) Ecrire la CSO pour le programme restreint. En déduire que

$$u \geq 0, \quad u\hat{y} = 0, \quad (uA - f)^M \geq 0, \quad (uA - f)^{M \cap I} = 0, \quad (uA - f)\hat{x} = 0.$$

(b) Ecrire les conditions de complémentarité entre (P) *et* (D). *En déduire que si* $uA - f \geq 0$ *alors* $\hat{x}$ *est solution de* (P) *et* u *est solution optimale de* (D).

(c) Supposons maintenant que $uA - f \ngeq 0$.

(c1) Vérifier que

$$\theta = \min\{-\frac{(\lambda A - f)^j}{(\lambda A - f)^j - (uA - f)^j} : (uA - f)^j < 0\},$$

et que $\theta \in (0,1)$.

(c2) Montrer que $\lambda' := (1-\theta)\lambda + \theta u \in D$.

(c3) Montrer que $M(\lambda) \cap I \subset M(\lambda')$. *En déduire que l'on peut utiliser la base finale du programme restreint précédent pour démarrer le nouveau programme restreint.*

(c4) Soit $j_0 \in J$ *l'indice pour lequel le minimum est atteint dans (c1). Vérifier que* $j_0 \in M(\lambda')$.

(d) Montrons maintenant la finitude de Simplex primal-dual.

(d1) Avec les notations de (c3),(c4), montrer que la composante d'indice j_0 *du premier vecteur coûts réduits du nouveau programme restreint est strictement positive.*

(d2) En déduire que, en absence de dégénérescence, il y a un nombre fini d'itérations pour Simplex primal-dual.

(e) Finalement montrons l'encadrement (2.18). Soient $(x_M^{(k)}, y^{(k)})$ *la solution obtenue par le programme restreint,* $\lambda_{(k)}$ *le nouveau élément de* D *calculé après en phase 3, et finalement* $x^{(k)} \in P$ *en complétant par* $x^{(k)}_{J\setminus M(\lambda_{(k-1)})} = 0$. *Vérifier que*

$$\lambda_{(k)} a = (1-\theta)\lambda_{(k-1)} a + \theta f x^{(k)}.$$

En déduire que

$$f x^{(k-1)} \leq f x^{(k)} \leq \lambda_{(k)} a \leq \lambda_{(k-1)} a.$$

L'exercice 2.9.2 permet de conclure que Simplex primal dual n'est rien que la résolution du programme (P) par la méthode Simplex, mais dans le pricing on donne une préférence aux indices appartenant à $L \cup M(\lambda)$. Si ces indices vérifient la CSO, mais pas encore les autres appartenant à $J \setminus M(\lambda)$, on met à jour le vecteur $\lambda \in D$, ce qui permet de continuer Simplex. Tout se passe comme si on résolvait un problème (P) comprenant toutes les variables x_j. Nous passons alors d'un point extrême à un autre et l'algorithme est fini (en cas de dégénérescence on applique la règle de Bland pour éviter le cyclage). Pourtant, le grand avantage de Simplex primal dual est que l'on obtient un encadrement de la valeur optimale, permettant de construire des critères d'arrêt comme dans les deux chapitres précédents.

Chapitre 3

Optimisation combinatoire

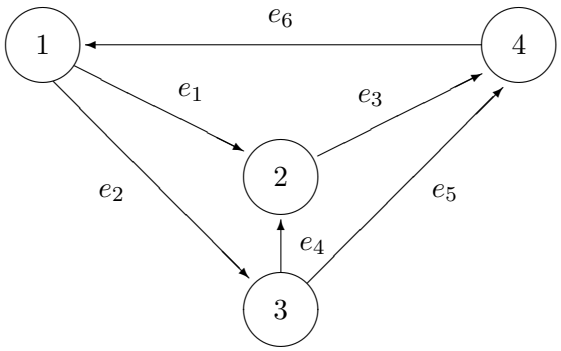
3.1 Introduction et définitions de base

3.1.1. Définition d'un graphe (orienté) et notations :

Un graphe *est un couple* $G = (L, J)$, L *ensemble de* sommets, $|L| = m$, $J \subset L \times L$ *ensemble d'*arcs *(généralement on exclut les* boucles (ℓ, ℓ)*)*, $|J| = n$.
$j = (\ell, k) \in J : \ell =$ *source*(j), $k =$ *but*(j), ℓ, k extrémités *de* j,
ℓ *(ou* k*) est* incident *à* j, ℓ *et* k *sont adjacents.*
On parle d'un graphe valué si il existe une valuation $d : J \mapsto \mathbb{R}$.

On rencontre les *graphes* (valués ou non) dans des *contextes variés*, dot voici quelques exemples :

- **Descriptif d'une séquence d'événements :** ici les sommets représentent les différents états d'un système, et les arcs représentent des possibilités de passage entre deux états (éventuellement avec un coût). On se pose la question de comment passer d'un état initial à un état final. Un exemple classique est donné par le problème du loup, de la chèvre et du chou, voir 3.2.5.
- **Réseau de transport (voiture, train, bus, ...) :** ici les arcs sont des connections (à sens unique) entre deux villes (=sommets). A partir des connections (=arcs) on construit des parcours (=chaînes/chemins). Différentes valuations possibles : distance



$L = \{1, 2, 3, 4\}$,
$J = \{e_1, e_2, e_3, e_4, e_5, e_6\}$
$= \{(1, 2), (1, 3), (2, 4), (3, 2), (3, 4), (4, 1)\}$

l'arc $e_2 = (1, 3)$ admet comme source
le sommet 1 et comme but le sommet 3

FIGURE 3.1 – EXEMPLE D'UN GRAPHE À $m = 4$ SOMMETS ET $n = 6$ ARCS

(vol d'oiseau), péage, restriction pour les poids lourds, restriction de capacité pour le trafic.

— **Réseau de conduite (eau, électricité,...) :** ici on dispose de tuyaux entre stations de pompage, avec valuation naturelle : capacité maximale, capacité minimale, coûts unitaires d'utilisation, pertes par tuyau,...

— Réseaux de télécommunications, circuits imprimés (graphes planaires), organisation d'une file d'attente, problèmes d'ordonnancement, stockage des matrices creuses (comportant peu d'éléments $\neq 0$), etc.

Dans l'introduction on a déjà vu quelques problèmes d'optimisation combinatoire qui se formulent naturellement en termes de graphes : le problème 1.2.2 de recherche d'un chemin de valeur minimale, le problème 1.2.3 du voyageur de commerce, le problème 1.2.4 de transport. D'autres classes de problèmes sont :

— **Accessibilité :** on se pose la question de savoir si on peut construire à partir d'un sommet donné des chemins allant vers un autre sommet ou vers tout autre sommet.

— **Problèmes de coloriage :** comment colorier une carte (les sommets d'un graphe planaire) sachant que deux pays adjacents devraient avoir une couleur différente ?

— **Problèmes de flot :** ici on dispose d'un réseau de conduite avec une loi de conservation sur chaque sommet (sauf la source et le puits) : la quantité entrante coïncide avec la quantité sortante. On s'intéresse aux questions suivantes :

  — Flot maximum : transporter un maximum depuis la source vers le puits.
  Exemple : Dans le réseau de métro de Paris, envoyer un maximum de personnes depuis la Place d'Italie vers la Gare du Nord sachant que chaque rame peut transporter au maximum 100 personnes.

  — Flot compatible : trouver un flot respectant les contraintes max/min de capacité.
  Exemple : ajuster le flot dans un graphe à 10 sommets.

  — Flot de coût minimal : chaque unité transportée dans un tuyau engendre un coût unitaire, trouver la façon la plus économique de transporter une quantité donnée en respectant les contraintes max/min de capacité.
  Exemple : Dans le réseau de métro de Paris, envoyer 120 personnes depuis la Place d'Italie vers la Gare du Nord dans un temps minimal.

  — Plusieurs demandes (conflictuelles) de transport : multiflots (problème actuel de recherche). [1]

— **Problème d'ordonnancement :** construire dans un minimum de temps une maison sachant que certaines tâches sont soumises à des contraintes de succession (on ne peut pas commencer à poser la toiture avant de terminer les murs), voir Exemple 3.3.4.

— **Problème de transbordement/transport :** transporter au moindre coût un certain bien depuis plusieurs usines vers plusieurs clients, en passant éventuellement par des entrepôts, voir Exemple 1.2.4 et Exemple 3.4.4 ;

— **Problème d'affectation :** affecter un certain nombre de personnes à un certain nombre de tâches tout en minimisant les coûts de formation, voir Exemple 3.4.5.

— **Arbres de valeur minimale :** étant donné un graphe non orienté, chercher à extraire un sous-graphe de valeur minimale qui soit connexe et sans cycles (=arbre).
Exemple : On souhaite construire un réseau de canaux permettant de relier 10 villes entre elles. Dans le graphe non orienté on indique les tracés potentiels des canaux.

1. ANA, ON POURRAIT EN PARLER EN REPRENANT PAR EXEMPLE LA DECOMPOSITION DE DANTZIG ET WOLFE

Sachant que le coût de réalisation d'un canal entre deux villes est proportionnel à leur distance (vol d'oiseau), quel ensemble de canaux faut-il réaliser ?

Tous ces problèmes seront illustrés dans la suite du chapitre, et traités en détail.

Comme pour toute nouvelle théorie mathématique il nous faut commencer par un certain nombre de définitions.

3.1.2. Définition : Graphes particuliers

Le graphe (L, J) *est dit* complet *si* $J = (L \times L) \setminus \{(\ell, \ell) : \ell \in L\}$. *Il est dit* planaire *si on peut le tracer sur une feuille de papier sans intersection d'arcs. Le graphe* (L, J) *est dit bi-parti s'il existe une partition* $L = L_1 \cup L_2$ *avec* $J \subset L_1 \times L_2$. *Finalement, on parle*[2] *d'un graphe* non orienté *si on oublie les orientations des arcs : ici, au lieu d'un arc* $j = (\ell, k)$, *on parle d'une* arête $j = \{\ell, k\}$ *(sans source/but).*

3.1.3. Définition : Parties d'un graphe $G = (L, J)$

Soit $L' \subset L$, *et* J' *obtenu à partir de* J *en enlevant tous les arcs incident à un sommet* $\in L \setminus L'$, *alors* (L', J') *est dit* sous-graphe *de* (L, J) *(induit par* L'*).*
Si $J' \subsetneqq J$ *alors* (L, J') *est dit* graphe partiel *de* (L, J).

3.1.4. Définition : Représentation d'un graphe $G = (L, J)$

Matrice d'adjacence $B = B_L^L$ *définie par* $B_\ell^k = 1$ *si* $(\ell, k) \in J$ *et* $B_\ell^k = 0$ *sinon.*
Matrice d'incidence *(sommets–arcs)* $A = A_L^J$ *définie par*

$$A_\ell^j = \begin{cases} 1 & \text{si } \ell = \text{source}(j), \\ -1 & \text{si } \ell = \text{but}(j), \\ 0 & \text{sinon.} \end{cases}$$

Liste des successeurs *pour* $\ell \in L$: $\Gamma^+(\ell) = \{k \in L : (\ell, k) \in J\}$.
Liste des prédécesseurs *pour* $\ell \in L$: $\Gamma^-(\ell) = \{k \in L : (k, \ell) \in J\}$.
Liste des sommets adjacents (voisins) *pour* $\ell \in L$: $\Gamma(\ell) = \Gamma^+(\ell) \cup \Gamma^-(\ell)$.

A titre d'exemple, les matrices pour le graphe de la figure 3.1 prennent la forme

$$B = B_L^L = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}, \qquad A = A_L^J = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & -1 \\ -1 & 0 & 1 & -1 & 0 & 0 \\ 0 & -1 & 0 & 1 & 1 & 0 \\ 0 & 1 & -1 & 0 & -1 & 1 \end{bmatrix},$$

et le sommet 2 a des successeurs 4 (voir ligne B_2) et les prédécesseurs $1, 3$ (voir colonne B^2).

Pour un stockage informatique, il semble être plus intéressant de travailler avec les listes de successeurs ou prédécesseurs (ce qui d'ailleurs peut être compris comme un stockage en creux de la matrice d'adjacence), nécessitant $\mathcal{O}(m+n)$ espaces mémoire. Néanmoins, la matrice d'incidence (sommets-arcs) va nous permettre de faire le lien entre l'optimisation linéaire et les problèmes formulés sur les graphes.

2. Pour ne pas trop multiplier des différents définitions, on considéra dans ce chapitre (à l'exception du sous-chapitre 3.6) seulement les graphes orientés, même si l'orientation n'est pas justifiée dans certaines modélisations.

3.1.5. Définition : chaîne, chemin, cycle

Une chaîne *est une suite finie $\gamma = (j_1, .., j_\kappa)$ avec $j_\alpha \in J$ de sorte que $j_\alpha, j_{\alpha+1}$ ont une extrémité commune $\ell_\alpha \in L$ pour $\alpha = 1, ..., \kappa - 1$; c'est à dire, il existe des sommets ℓ_0 (source), $\ell_1, ..., \ell_{\kappa-1}$ (sommets internes), et ℓ_κ (but) dans L t.q. $j_\alpha \in \{(\ell_{\alpha-1}, \ell_\alpha), (\ell_\alpha, \ell_{\alpha-1})\}$ pour $\alpha = 1, .., \kappa$ (on oublie les orientations).*
Si de plus $j_k = (\ell_{\alpha-1}, \ell_\alpha)$ pour $\alpha = 1, .., \kappa$ (on respecte les orientations) alors γ est appelé un chemin *(autre écriture d'un chemin en énumérant les sommets : $\gamma = [\ell_0, \ell_1, ..., \ell_\kappa]$).*
Un cycle *est une chaîne fermée (avec $\ell_0 = \ell_\kappa$) ; un* circuit *est un chemin fermé.*
Une chaîne simple est une chaîne sans répétition d'arcs (de même pour cycles/chemins/circuits).
Une chaîne élémentaire est une chaîne sans répétition de sommets (sauf éventuellement $\ell_0 = \ell_\kappa$, un cycle élémentaire).

A titre d'exemple, pour le graphe de la figure 3.1, $\gamma = [4, 1, 3, 4] = (e_6, e_2, e_5)$ est un chemin, qui est aussi fermé (un circuit), élémentaire et donc simple. $\gamma' = (e_5, e_4, e_3, e_6) = [4, 3, 2, 4, 1]$ n'est pas un chemin, mais une chaîne, et d'ailleurs simple mais pas élémentaire.

3.1.6. Lemme de König : *De toute chaîne (chemin/cycle/circuit) on peut extraire une chaîne (chemin/cycle/circuit) élémentaire ayant les mêmes extrémités.*

Démonstration. On donnera une preuve par construction explicite de la chaîne élémentaire extraite d'une chaîne $\gamma = [\ell_0, \ell_1, ..., \ell_\kappa]$: on parcourt dans l'ordre les sommets. Pour le sommet ℓ_k, on vérifie s'il existe parmi les sommets suivants un doublant, c'est-à-dire, un indice $k' \in \{k + 1, ..., \kappa\}$ avec $\ell_k = \ell_{k'}$. Dans ce cas, on supprime la partie $[\ell_{k+1}, \ell_{k+2}, ..., \ell_{k'}]$ de γ : le nouvel objet aussi appelé γ est une chaîne ayant les mêmes extrémités. Ensuite on cherche les doublants suivants de ℓ_k, et on passe au sommet suivant ℓ_{k+1}, en répétant la même procédure. Par construction, il en résulte une chaîne élémentaire. □

3.1.7. Exercice :
Donner des bornes en fonction de $|L| = m$ et $|J| = n$ du nombre des chaînes/chemins/cycles élémentaires/simples.

3.1.8. Exercice
Soit B la matrice d'adjacence d'un graphe (L, J), et notons par $B^\ell(j, k)$ l'élément à la position (j, k) de la puissance B^ℓ. Montrer que, pour $\ell \geq 2$,

$$B^\ell(j, k) = \sum_{i \in \Gamma^-(k)} B^{\ell-1}(j, i).$$

En déduire que $B^\ell(j, k)$ donne le nombre de chemins distincts du sommet j au sommet k et de longueur ℓ.

3.2 Problèmes d'accessibilité

Avant de se lancer dans la recherche d'un "meilleur" chemin reliant deux sommets donnés, on se pose la question plus élémentaire suivante : dans un graphe orienté $G = (L, J)$, déterminer l'ensemble de sommets $x \in L$ pour lesquels on peut construire un chemin avec source $s \in L$ et

but x : dans ce cas on dira que x est accessible depuis s. Aussi, on dira que s est *racine* de G si tout sommet est accessible depuis s.

3.2.1. Algorithme de Tarjan (1972, et de Trémeaux 1882) :

Invariant : *On marque le sommet x*
- *au crayon si on a su construire un chemin avec source s et but x ;*
- *à l'encre si de plus on a déjà examiné tous les successeurs du sommet x.*

Initialisation : *Le sommet s est marqué au crayon.*
Itération : *Tant qu'il existe encore un sommet x marqué au crayon faire*
marquer ce sommet à l'encre et tous ses successeurs non marqués au crayon
Résultat : *Tout sommet marqué à l'encre est accessible, tout autre sommet n'est pas accessible.*
Complexité : $\mathcal{O}(n)$.

Démonstration. Il y a trois choses à démontrer :
(i) A la fin il n'y a plus de sommets marqués au crayon : sinon, on aurait continué l'algorithme.
(ii) Un sommet x marqué à l'encre est accessible depuis s : preuve par récurrence sur l'ordre de marquage à l'encre. Le sommet $x = s$ est clairement accessible. Tout sommet $x \neq s$ marqué à l'encre à été marqué d'abord au crayon, parce qu'il y avait un sommet y prédécesseur de x qui était marqué à l'encre. Par hypothèse de récurrence, y est accessible depuis s, et donc x l'est aussi.
(iii) A la fin, un sommet x non marqué n'est pas accessible depuis s : par absurde, supposons qu'il existe un chemin $\gamma = [y_1, y_2, ..., y_k]$ avec source $s = y_1$ et but $x = y_k$. Le sommet s étant marqué, il existe un indice $j \in \{1, 2, ..., k-1\}$ de sorte que y_j est marqué, mais pas y_{j+1}. D'après (i) on sait que y_j est marqué à l'encre, et y_{j+1} est successeur de y_j d'après la définition d'un chemin. Par conséquent, on aurait oublié de marquer y_{j+1} au crayon au moment où on a marqué y_j à l'encre, ce qui contredit les règles de l'algorithme. □

L'algorithme 3.2.1 de Tarjan est appelé un *algorithme de marquage* : on explore un graphe en énumérant successivement (une partie de) ses sommets. Ce principe de marquage est appliqué aussi pour la recherche des plus courts chemins dans le chapitre suivant. Nous retrouvons la philosophie des algorithmes gloutons : on parcourt étape par étape les parties d'un objet en question (ici les sommets d'un graphe) en se basant sur une propriété locale (ici voisinage), pour ensuite conclure sur une propriété globale (ici accessibilité).

Pour la mise en place de l'algorithme de Tarjan sur ordinateur, il faudra savoir faire appel aux successeurs (ce qui est trivial si on stocke le graphe sous forme de listes de successeurs). Aussi, il faudra gérer une pile de sommets marqués au crayon. Ici on peut faire appel à des techniques classiques.

3.2.2. Remarque sur la gestion de la pile

On retire le sommet à marquer à l'encre au début de la pile comportant les sommets marqués au crayon. Pour ajouter on peut suivre une des deux stratégies suivantes :

— *"FIFO" (=first in first out=premier arrivé premier servi) : on ajoute les nouveaux sommets à la fin de la liste. En général, ceci donne lieu à un grand nombre de chemins relativement courts (en nombre d'arcs). Notons que l'ensemble des chemins ainsi construits forme une sorte d'arborescence avec racine s. Notre ordre de marquage*

correspond à parcourir cette arborescence en largeur. Voir l'exemple dans le fichier `don_ch3/tar_fifo.bat`. *On note que l'état de la pile à chaque itération est affichée dans le titre de la fenêtre.*

— *"FILO" (=first in last out=premier arrivé dernier servi) : on ajoute les nouveaux sommets au début de la liste, ce qui fait que l'on construit un petit nombre de chemins relativement longs : on parcourt l'arborescence en profondeur. Voir l'exemple dans le fichier* `don_ch3/tar_filo.bat`.

Il est d'ailleurs intéressant d'observer (voir Exercice 3.3.7) que, par la version FIFO de l'algorithme de Tarjan, on trouve (implicitement) le plus court chemin en nombre d'arcs de s à tout autre sommet. Pour discuter quelques variantes de l'algorithme de Tarjan, on introduit d'abord formellement la notion de connexité.

3.2.3. Définition : (forte) connexité
Composantes connexes : classes d'équivalence pour la rélation d'équivalence suivante : $\ell \sim k$ pour $\ell, k \in L$ ssi il existe une chaîne ayant les extrémités ℓ et k (inclus le cas trivial $k = \ell$).
Composantes fortement connexes : *classes d'équivalence pour la rélation d'équivalence suivante : $\ell \approx k$ pour $\ell, k \in L$ ssi il existe un chemin avec source ℓ et but k et un chemin avec source k et but ℓ (inclus le cas trivial $k = \ell$).*
Un graphe est dit (fortement) connexe *s'il existe une seule composante (fortement) connexe.*

3.2.4. Remarques : Variantes de Tarjan
(a) *Par introduction d'un tableau nommé père*

$$père(y) = x \text{ si le sommet } x \text{ a permis de marquer au crayon le sommet } y$$

on peut reconstruire explicitement le chemin ayant permis d'accéder à un sommet x_0 donné : on construit x_k =père(x_{k-1}) jusqu'au moment où $x_\ell = s$, et on obtient le chemin $[x_\ell, x_{\ell-1}, ..., x_1, x_0]$.
(b) *Pour trouver l'ensemble des sommets x de sorte que le sommet s est accessible depuis x, il suffit de remplacer, dans l'algorithme de Tarjan, le mot "successeur" par "prédécesseur".*
(c) *Finalement, si on remplace "successeurs de x" par "voisins de x" (c'est-à-dire, $\Gamma(x) = \Gamma^-(x) \cup \Gamma^+(x)$), l'ensemble des sommets marqués à la fin de l'algorithme forme la composante connexe à laquelle appartient s (aussi pour graphes non orientés).*

Discutons en détail la modélisation d'un problème de recherche d'une séquence d'événements (où d'une stratégie) où l'utilisation de la théorie des graphes est extrêmement utile (surtout pour des problèmes similaires d'une plus grande complexité, voir Exercice **??** et Exercice **??**).

3.2.5. Exemple : le loup, la chèvre et le choux
Un loup, une chèvre et un choux souhaitent traverser une rivière à l'aide d'un bateau qui en plus du batelier ne peut transporter qu'un seul objet. Sachant que tout le monde se trouve au début rive-gauche et que l'on ne peut pas laisser (pour des raisons évidentes) ni le couple choux/chèvre ni le couple loup/chèvre seul sur un des deux cotés de la rivière, dans quel ordre faut-il acheminer les trois objets ?

Dans un premier temps il faudra clairement définir les différents états (=sommets), pour ensuite discuter des possibilités de passage entre états (=arcs), et finalement résoudre le problème initial de passage d'un état initial (chez nous "tout le monde rive-gauche") à un état final (chez

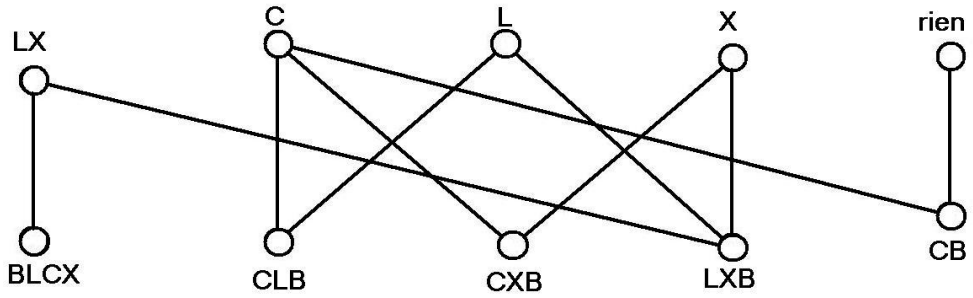


FIGURE 3.2 – L'EXEMPLE DU LOUP (L), DE LA CHÈVRE (C) ET DU CHOUX (X) QUI SOUHAITENT TRAVERSER UNE RIVIÈRE, À L'AIDE D'UN BATELIER (B). LES SOMMETS REPRÉSENTENT LES OBJETS SE TROUVANT RIVE GAUCHE, ET LES ARCS DÉCRIVENT LES POSSIBILITÉS DE PASSER D'UN ÉTAT À UN AUTRE.

nous "tout le monde rive-droite"), ce qui devient un problème de construction d'un chemin, c'est-à-dire, d'accessibilité.

Pour notre problème, un état est décrit par l'énumération des objets rive-gauche avant ou après le départ du batelier, voir la figure 3.2. Ici il faudra omettre les cas CX, LC qui posent des problèmes rive-gauche, mais aussi les cas complémentaires LB, XB posant des problèmes rive-droite. Les segments (arêtes) représentent les possibilités de passage entre un couple d'états : il est par exemple possible de passer de LX à LXB (le batelier revient seul), mais pas à CLB (le batelier peut revenir avec la chèvre, mais le choux ne disparaît pas). Notons que, en termes de graphes orientés, une arête devrait être remplacée par deux arcs (dans les deux sens). Finalement, par l'algorithme 3.2.1 de Tarjan nous trouvons que le sommet rien est accessible depuis $BLCX$ par le chemin $[BLCX, LX, LXB, L, CLB, C, CB, rien]$, ce qui peut être notre stratégie de transport. Notons qu'il y en a d'autres, et que l'on pourrait y ajouter des critères pour trouver par exemple la stratégie la moins chère (voir Exercice **??**, *Exercice* **??** *et Exercice* **??**).

3.2.6. Exercice :
Montrer que la relation $\approx$ de la définition 3.2.3 est une relation d'équivalence sur L. En déduire un algorithme de complexité $\mathcal{O}(n)$ pour trouver la composante fortement connexe à laquelle appartient un sommet $s \in L$ donné.

3.2.7. Exercice :
Soit $\Gamma^i(\ell)$ défini par $\Gamma^0(\ell) = \{\ell\}$, et pour $i \geq 1$: $\Gamma^i(\ell) = \Gamma(\Gamma^{i-1}(\{\ell\}))$, l'ensemble des voisins de sommets dans $\Gamma^{i-1}(\{\ell\})$. Montrer que la composante connexe d'un $\ell \in L$ est donnée par $\Gamma^0(\ell) \cup \Gamma^1(\ell) \cup ... \cup \Gamma^{m-1}(\ell)$.

Par un algorithme de marquage, on peut aussi décider si un graphe donné contient un circuit.

3.2.8. Lemme : détection de circuits
S'il existe un sous-ensemble $L' \subset L$ non vide de sorte que tout sommet dans L' admet un prédécesseur dans L' alors le sous-graphe engendré par L' contient un circuit.

Démonstration. Nous donnons une preuve constructive, par l'énoncé d'un algorithme de construction d'un circuit :

affecter à tout $x \in L'$ l'étiquette père$(x) = \emptyset$ et choisir un $y \in L'$
tant que père$(y) = \emptyset$ faire
    choisir comme nouveau x le sommet y
    choisir comme nouveau $y \in L'$ un prédécesseur de x
    affecter l'étiquette père$(x) = y$

L'algorithme s'arrête car chaque fois on examine un nouveau sommet y et l'ensemble des sommets est fini. À la fin on a construit un circuit car on dispose d'une suite de sommets $x_0, x_1, ..., x_k$ avec père$(x_{j-1}) = x_j$, c'est-à-dire, x_j est prédécesseur de x_{j-1}, et père$(x_k) = x_0$. Donc $[x_k, x_{k-1}, ..., x_0, x_k]$ est un circuit. □

Soit $G = (L, J)$ un graphe, et soit $L' \neq \emptyset$ l'ensemble des sommets non marqués à l'encre à un instant donné de l'algorithme de Tarjan. Alors le lemme 3.2.8 nous permet d'affirmer qu'au moins une des propriétés suivantes est vraie :

— il existe un sommet non marqué à l'encre avec tous ses prédécesseurs déjà marqués à l'encre (on y inclut aussi des sommets n'ayant pas de prédécesseurs) ;
— on peut construire dans le sous-graphe engendré par L' un circuit par l'algorithme de la preuve de 3.2.8.

Autrement dit, dans un graphe sans circuits il est possible de marquer tous les sommets en respectant l'ordre

$$\begin{array}{l}\text{on marque seulement le sommet } x \text{ à partir} \\ \text{du moment où on a marqué tous ses prédécesseurs}\end{array} \tag{3.1}$$

Cet ordre peut être réalisé en affectant à chaque sommet x une étiquette compteur(x) comptant le nombre de prédécesseurs pas encore marqués. L'ensemble de candidats pour marquage est donc l'ensemble de sommets portant l'étiquette 0. Ceci donne l'algorithme suivant.

3.2.9. Algorithme

Hypothèse : (L, J) *est un graphe sans circuits.*
Initialisation *affecter à tout* $x \in L$ *l'étiquette compteur*(x) *égale au nombre de prédécesseurs de* x
Itération : *tant qu'il existe* x *non marqué avec compteur*$(x) = 0$ *faire*
    marquer x *à l'encre*
    pour tout y *successeur de* x *faire : diminuer l'étiquette compteur*(y) *par* 1
Résultat : *on marque tous les sommets de* G *en respectant (3.1).*

Il n'est pas difficile de voir qu'un graphe G sans circuits admet une racine s si et seulement s'il existe un unique sommet s n'ayant pas de prédécesseurs. Dans ce cas, on peut alors construire par 3.2.9 successivement des chemins de s à y en notant père$(y) = x$ par exemple pour le prédécesseur x de y ayant permis de passer à compteur$(y) = 0$.

L'ordre (3.1) de marquage joue un rôle dans des diverses applications, en particulier dans le problème d'ordonnancement : dans quel ordre faut-il exécuter des différentes tâches si on veut

construire une maison ? Pour illustration, on trouve dans le fichier `don_ch3/maison.bat` un graphe associé à un problème d'ordonnancement, et le fichier `don_ch3/maison_m.bat`. permet de simuler le marquage suivant 3.2.9. On observe qu'un graphe associé à un problème d'ordonnancement ne devrait comporter aucun circuit et admet une racine, comparer avec 3.3.4.

3.2.10. Exercice
Dans un graphe avec racine s et sans circuits, montrer que l'algorithme 3.2.9 trouve (implicitement) le plus long chemin en nombre d'arcs d'un sommet sans prédécesseurs à tout autre sommet marqué, et que pour les sommets non marqués il n'existe pas un tel chemin.

La longueur d'un plus long chemin en nombre d'arcs de s à x s'appelle le rang d'un sommet x. Un dessin d'un graphe avec tous les sommets de même rang alignés au même niveau (vertical ou horizontal) s'appelle aussi la *mise à niveau* d'un graphe.

3.3 Problèmes de chemin de valeur minimale

On se donne un graphe orienté $G = (L, J)$ valué, c'est-à-dire, chaque arc $j \in J$ admet une valeur $d(j)$ (réele pas forcément positive, écrite aussi parfois d_j). Sachant que la valeur d'un chemin est la somme des valeurs des arcs qui composent ce chemin, on se pose le problème de déterminer pour tout $x \in L$ la valeur $\pi(x)$ d'un chemin de valeur minimale avec source s et but x pour un sommet fixé s. Parfois on cherche également à construire un tel chemin (nommé aussi *plus court chemin*).

3.3.1. Définition et Lemme
Un circuit de valeur < 0 est dit circuit absorbant.

Si le graphe (L, J) ne comporte pas de circuit absorbant, alors le problème du plus court chemin de s à x est bien posé pour tout $x \in L$ accessible depuis s, et on trouve un plus court chemin parmi les chemins élémentaires de s à x.

Démonstration. Comme il y a un nombre fini de chemins élémentaires, il suffit de montrer que pour tout chemin γ de s à x il existe un chemin élémentaire γ' de s à x avec $d(\gamma') \leq d(\gamma)$. Ceci est une conséquence immédiate du lemme 3.1.6 de König (et de sa preuve). On peut extraire de γ un chemin élémentaire, en supprimant un certain nombre de circuits, et chaque suppression d'un circuit donne un chemin de valeur plus petite que le précédent. □

On utilisera parfois la convention $\pi(x) = +\infty$ pour les sommets x qui ne sont pas accessibles depuis s. Notons que la réciproque du lemme 3.3.1 est aussi valable : si un circuit absorbant comporte un sommet x accessible depuis s alors, en empruntant ce circuit à plusieurs reprises, on construit des chemins de valeur non bornées inférieurement.

3.3.2. Exercice
Montrer les propriétés suivantes :
(a) Si $[x_0, x_1, x_2, ..., x_\ell]$ est un chemin de valeur minimale entre x_0 et x_ℓ alors tout sous-chemin $[x_i, x_{i+1}, ..., x_j]$ pour $0 \leq i < j \leq \ell$ est un chemin de valeur minimale entre x_i et x_j.
(b) Pour $y \neq s$ nous avons $\pi(y) = \min\{\pi(x) + d((x, y)) : x \in \Gamma^-(y)\}$, et $\pi(s) = 0$.

Par la suite on donnera trois algorithmes de recherche d'un plus court chemin, dans l'ordre croissant de complexité.

Dans un premier temps nous supposons que notre graphe admet s comme racine (tout sommet est accessible depuis s) et ne contient aucun circuit. Par conséquent, dans notre algorithme 3.2.9 de marquage, au moment où le sommet y est marqué à l'encre, tous ses prédécesseurs sont déjà marqués à l'encre. Donc $\pi(y)$ peut être calculé récursivement à l'aide de la formule 3.3.2(b). En combinant cette idée avec 3.2.9, on obtient l'algorithme suivant.

3.3.3. Algorithme de Bellman dans un graphe sans circuits

Hypothèse : $G = (L, J)$ *est un graphe valué sans circuits,* $s \in L$
Tâche : *Pour* $x \in L$*, calculer la valeur* $\pi(x)$ *d'un plus court chemin de* s *à* x*.*
Invariant : $\pi(x)$ *est la valeur d'un chemin de valeur minimale de* s *à* x *dans le sous-graphe engendré par l'ensemble des sommets marqués à l'encre plus* x*.*
Initialisation : *Pour tout* $x \in L$ *faire*
affecter étiquettes père$(x) = \emptyset$*,* $\pi(x) = +\infty$*, compteur*$(x) = |\Gamma^-(x)|$
affecter étiquette $\pi(s) = 0$*, marquer* s *au crayon.*
Itération : *tant qu'il existe* x *marqué au crayon*
marquer x *à l'encre*
pour tout y *successeur de* x *faire*
si $\pi(x) + d((x, y))$ *est* $< \pi(y)$ *alors*
poser $\pi(y) = \pi(x) + d((x, y))$*, père*$(y) = x$
diminuer l'étiquette compteur(y) *par* 1
si compteur(y) *s'annule alors marquer* y *au crayon.*
Complexité : $\mathcal{O}(n)$.
Résultat : $\pi(x)$ *pour tout sommet* x *accessible depuis* s *(= tout sommet marqué à l'encre).*

Notons que l'on a maintenu la coloration au crayon suivant l'algorithme 3.2.1 de Tarjan pour les sommets sur la pile, pour ne pas retrouver sur la pile des sommets sans prédécesseurs (qui alors ne sont pas accessibles depuis s). Un exemple du déroulement de l'algorithme de Bellman se trouve dans le fichier `don_ch3/bellma_v.bat`.

3.3.4. Remarque : Application à l'ordonnancement

Un projet (construction d'une maison) consiste à effectuer plusieurs tâches. Connaissant pour chaque tâche t *sa durée* d_t *et les tâches qui doivent la précéder, on souhaite déterminer* la date au plus tôt $\psi(t)$ *pour l'exécution de la tâche* t*, et la* durée minimale du projet*. On souhaite également indiquer les tâches dites* critiques *pour lesquelles un retard impliquera un retard pour l'achèvement du projet.*

Considérons le graphe $G = (L, J)$ *(nommé* graphe potentiel-tâches*, Roy 1960) ayant comme sommets les tâches,* $(s, t) \in J$ *de valeur* d_s *si et seulement la tâche* s *est antérieure à la tâche* t *(notre graphe ne comporte pas de circuits et tout sommet est accessible depuis le sommet début). Une* planification *consiste à déterminer pour toute tâche* x *la quantité* $\psi(x)$ *étant le début au plus tôt de la tâche* x*, avec comme début du projet* $\psi(début) = 0$ *et fin du projet la quantité recherchée* $\psi(fin)$*. Les contraintes d'antériorité sont traduites comme suit : pour toute tâche* $t \neq$*début nous avons*

$$\psi(t) = \max\{\psi(s) + d_s : \text{tâche } s \text{ est antérieure à } t\}.$$

Nom de la tâche	durée (en semaines)	tâches antérieures
Travaux de maçonnerie	7	aucune *(début)*
Charpente de la toiture	3	maçonnerie
Toiture	1	charpente
Installations sanitaires	8	maçonnerie
Façade	2	toiture, installations
Fenêtres	1	toiture, installations
Aménagement du jardin	1	toiture, installations
Travaux de plafonnage	3	fenêtres
Mise en peinture	2	plafonnage
Emménagement	1	façade, jardin, peinture
début	*0*	*aucune*
fin	*0*	tâches sans successeurs, ici *emménagement*

FIGURE 3.3 – UN PROBLÈME D'ORDONNANCEMENT.

En comparant avec formule 3.3.2(b), nous observons que

- *pour tout $t \in S$, la quantité $\psi(t)$ coïncide avec la valeur d'un chemin de valeur maximale du sommet début à t ;*
- *Les tâches critiques sont celles composant un chemin de valeur maximale entre le sommet début et le sommet fin.*

Pour le problème d'ordonnancement exposé à la figure 3.3, on peut afficher le graphe à l'aide du fichier `don_ch3/maison_a.bat` *et calculer la durée du projet par l'algorithme de Bellman à l'aide du fichier* `don_ch3/maison_o.bat`.

Généralement nos graphes comportent des circuits, mais dans une grande partie des applications on a des valeurs ≥ 0 pour les arcs. Par conséquent, tout circuit est de valeur ≥ 0, et le problème des chemins de valeur minimale est bien posé.

L'algorithme de *Dijkstra*[3] est un *algorithme de marquage* au sens de la remarque après 3.2.1. Sa particularité est le choix du sommet dans la pile : on prend *le sommet le plus prometteur.*

3.3.5. Algorithme de Dijkstra

Hypothèse : $G = (L, J)$ *est un graphe à valuation* ≥ 0, $s \in L$
Tâche : *Pour* $x \in L$, *calculer la valeur* $\pi(x)$ *d'un plus court chemin de* s *à* x.
Invariants : *Toute étiquette* $\pi(y)$ *représente la valeur d'un chemin de* s *à* y,
Une étiquette à l'encre $\pi(x)$ *représente la valeur d'un chemin de* s *à* x *de valeur minimale.*
Initialisation : écrire étiquette $\pi(s) = 0$ *au crayon.*
Itération : *tant qu'il existe encore une étiquette écrite au crayon*
parmi les sommets portant une étiquette écrite au crayon, trouver
le sommet x *avec étiquette* $\pi(x)$ *de valeur minimale*
écrire $\pi(x)$ *à l'encre*
pour tout y *successeur de* x *faire*
calculer $\Sigma = \pi(x) + d_{(x,y)}$

3. La première syllabe de "Dijkstra" se prononce approximativement comme le mot anglais "to die".

si y n'a pas encore une étiquette ou si Σ est plus petit que $\pi(y)$ alors marquer au crayon $\pi(y) = \Sigma$, père$(y) = x$

Démonstration. Nous devons démontrer trois propriétés :
(i) Toute étiquette $\pi(y)$ représente la valeur d'un chemin de s à y.
Preuve par récurrence sur l'ordre de marquage : c'est vrai au début pour $y = s$ (chemin trivial de s à s). Si $y \neq s$, toute valeur de $\pi(y)$ est égale à une somme de $\pi(x)$ plus la valeur de l'arc (x, y). Par hypothèse de récurrence, $\pi(x)$ représente la valeur d'un chemin γ de s à x. En ajoutant l'arc (x, y) à γ, on obtient bien un chemin de s à y ayant la valeur $\pi(y)$.
(ii) Une étiquette à l'encre $\pi(x)$ représente la valeur d'un chemin de s à x de valeur minimale.
Notons par $x_1, ..., x_\ell \in S$ dans l'ordre les sommets marqués à l'encre par l'algorithme de Dijkstra. Nous montrons la propriété (ii) par récurrence sur $k = 1, ..., \ell$.
Au début, s est le seul sommet avec étiquette écrite au crayon, et donc $x_1 = s$ et $\pi(s) = 0$. Comme tout chemin est de valeur ≥ 0, la propriété est bien vrai pour $k = 1$.
Soit $k > 1$, et γ un chemin de valeur minimale de s à x_k. Si la valeur de γ est $\geq \pi(x_k)$ alors, comme $\pi(x_k)$ représente la valeur d'un chemin de s à x_k d'après (i), on obtient égalité, et on a démontré notre propriété. Par absurde, supposons que la valeur de γ soit strictement plus petit que $\pi(x_k)$. La source de γ est égale à s, et le chemin γ comporte alors un arc de la forme (x_j, y) avec $1 \leq j < k$ et $y \notin \{x_1, ..., x_{k-1}\}$. Comme la valeur d'un arc est ≥ 0, la valeur de γ est minoré par la valeur d'un chemin de valeur minimale de s à x_j plus la valeur de l'arc (x_j, y). Par hypothèse de récurrence, on peut alors minorer la valeur de γ par $\pi(x_j)$ plus la valeur de l'arc (x_j, y), ce qui est $\geq \pi(y)$ au moment où l'on marque x_k à l'encre. Par conséquent, $\pi(y) < \pi(x_k)$ ce qui contredit les règles de marquage à l'encre.
(iii) A la fin de l'algorithme, tout sommet accessible depuis s porte une étiquette à l'encre, et tout autre sommet ne porte pas d'étiquette.
Il s'agit d'une variante de l'algorithme de Tarjan avec un choix particulier du sommet dans la pile. □

Dans l'algorithme 3.3.3 de Bellman, pour calculer $\pi(x)$ on avait d'abord examiné tous les prédécesseurs de x. L'élément étonnant de l'algorithme 3.3.5 de Dijkstra est que l'on arrive à conclure sur $\pi(x)$ sans avoir examiné le graphe entier, et en particulier sans avoir examiné tous les prédécesseurs de x.

Dans le fichier `don_ch3/dijk_10.bat` on propose une simulation de l'algorithme de Dijkstra pour un graphe à 10 sommets, et dans `don_ch3/dijk_100.bat` pour un graphe à 100 sommets

3.3.6. Remarques sur la complexité de l'Algorithme de Dijkstra

On vérifie aisément que l'algorithme de Dijkstra a un besoin de mémoire d'ordre de grandeur $\mathcal{O}(n)$ (tableaux de taille n comme successeurs/valeur, tableaux de taille $m \leq n$ comme sommets, marquage, compteur, père, π, pile, etc). Egalement, on compte (en dehors de la recherche de x) $\mathcal{O}(n)$ opérations élémentaires ($\mathcal{O}(1)$ opérations pour chaque sommet : marquage, gestion pile ; $\mathcal{O}(1)$ opérations pour chaque arc : test marquage, mise à jour π, pere, compteur, etc.). Il s'ajoute la tâche d'extraire, au à maximum m reprises, le plus petit élément d'une pile à au plus m éléments.

- *$\mathcal{O}(m^2)$ pour la version élémentaire : déterminer chaque fois le plus petit élément de la pile.*

— $\mathcal{O}(n\log(n))$ *pour une version plus sophistiquée : garder une pile triée par ordre croissant des éléments. Il y a au plus* $\mathcal{O}(n)$ *mises à jour des éléments : retirer l'ancien élément et insérer le nouveau par une méthode de dichotomie en complexité* $\mathcal{O}(\log(m))$.

3.3.7. Exercice :
Considérons un graphe (L, J) *valué par* $d_j = 1$ *pour tout* $j \in J$. *Soit* $s \in L$, *et considérons l'algorithme*

Initialisation : poser $\pi(s) = 0$, *pile*$:= [s]$
Itération : Tant que la pile contient encore un élément
retirer le sommet x *à gauche de la pile, marquer* x *à l'encre*
pour tout successeur y *de* x *n'étant pas marqué*
$\pi(y) = \pi(x) + 1$, *ajouter* y *à droite de la pile.*

(a) *Vérifier que l'algorithme ci-dessus coïncide avec l'Algorithme 3.2.1 de Tarjan avec organisation de pile par FIFO si on supprime toutes les instructions concernant le tableau* π.

(b) *Supposons que, au début d'une itération, la pile admet la propriété suivante*

$$(*) \qquad si \quad pile = [x_1, ..., x_k] \quad alors \quad \pi(x_1) \leq \pi(x_2) \leq ... \leq \pi(x_k) \leq \pi(x_1) + 1.$$

Montrer que cette propriété reste valable à la fin d'une itération. En déduire que $(*)$ *est un invariant de l'algorithme.*

(c) *Montrer que l'algorithme ci-dessus est mathématiquement équivalent avec l'algorithme de Dijkstra (on comparera en particulier le choix du sommet dans la pile et la mise à jour du tableau* π*).*

(d) *En déduire que, pour tout sommet* x *marqué, la valeur* $\pi(x)$ *représente la longueur d'un plus court chemin de* s *à* x *en nombre d'arcs. Comment construire explicitement ce chemin ?*

3.3.8. Exercice
M. Urgent veut vendre sa voiture le plus rapidement possible, mais la voiture est vétuste et ne passera pas le contrôle technique avant d'avoir effectué des réparations. Pour effectuer chacune de ces tâches, M. Urgent peut faire appel à des garages A ou B, sachant qu'il y a 20 *minutes de route pour joindre le garage A,* 30 *minutes de route pour joindre le garage B et* 40 *minutes de route entre les deux garages. De plus, les garages ont besoin des délais différents (voir tableau) pour effectuer une des trois tâches. Finalement, M. Urgent se trouve encore à sa maison, et souhaite y retourner avec les bénéfices de vente.*

Durée en minutes	*réparations*	*contrôle technique*	*vente de la voiture*
Garage A	*120*	*100*	*70*
Garage B	*140*	*40*	*80*

Donner une modélisation mathématique comme un problème de recherche d'un chemin de valeur minimale dans un graphe valué à préciser. À qui faut-il confier au mieux les différentes tâches ?

3.3.9. Exercice
Un jardinier doit planter au cours du mois k, $k = 1, 2, ..., s$, *une quantité de* d_k *arbres. Il peut*

s'approvisionner au début de chaque mois au prix de p_k euros par arbre. Les arbres achetés sont tous d'abord plantés temporairement dans un bout de champs derrière sa maison et ensuite retirés en fonction des besoins dans ce mois. Sur ce champs on peut planter au maximum r arbres. Au début de la période, aucun arbre se trouve sur le champs. Sachant que les prix varient fortement selon la saison, le jardinier se demande à quel moment il faut acheter quelle quantité d'arbres pour minimiser les frais d'approvisionnement.

(a) Modéliser les différentes stratégies d'approvisionnement à l'aide d'un graphe où la situation que l'on dispose (après achat) de i arbres au début du mois k est représentée par un sommet. Pourquoi ce graphe ne comporte pas de circuits ?

(b) Donner une valuation de ce graphe de sorte que la meilleure stratégie d'approvisionnement correspond au chemin de valeur minimale entre deux sommets à préciser.

(c) Résoudre le problème pour les données numériques $r = 4$, $s = 4$, et

k	1	2	3	4
d_k	3	1	4	2
p_k	1	5	2	7

3.3.10. Exercice

Donner un exemple d'un graphe valué où l'algorithme de Dijkstra ne donnera pas les valeurs des plus courts chemins d'un sommet fixe à tous les autres (mais où le problème des plus courts chemins est bien posé).

3.3.11. Exercice : Quand faut-il acheter une nouvelle voiture ?

Considérons le problème suivant de politique de remplacement de matériel :
Sur une période de T années on doit décider au début de chaque année si on achète une voiture neuve ou si on garde l'ancienne (l'achat d'une voiture d'occasion n'étant pas envisagé), sachant que

- *à tout moment on possède exactement une voiture ;*
- *au début de la période on doit acheter une voiture neuve ;*
- *la voiture au choix a un prix de vente constant de 10000E pendant la période envisagée ;*
- *au moment de l'achat d'une voiture neuve, on peut vendre l'ancienne pour un prix dépendant de son âge : pendant les premières 3 ans, une voiture perd chaque année la moitié de sa valeur, et après 10% par année ;*
- *à la fin de la période on vend sa voiture pour le prix ci-dessus,*
- *l'entretien annuel d'une voiture coûte $1000E$ si la voiture a moins que 3 ans, après il faut ajouter 500E par année supplémentaire.*

Définir un graphe valué de sorte que la stratégie la moins chère correspond à un chemin de valeur minimale.

Pour terminer ce chapitre, nous allons discuter l'algorithme de Roy-Warshall qui opère directement sur la matrice d'adjacence, voir Définition 3.1.4. Nous allons procéder en deux étapes. Dans un premier temps, nous allons déterminer la valeur $\pi(x, y)$ d'un plus court chemin de $x \in L$ à $y \in L$ dans un graphe (L, J) quelconque : cette procédure permettra aussi de détecter un circuit absorbant (voir 3.3.1). Ensuite, nous allons généraliser cette approche dans un cadre plus abstrait, pour répondre à des questions comme la recherche d'un parcours permettant de faire passer les camions les plus lourds.

Pour un chemin $\gamma = [x_0, x_1, ..., x_\ell]$, on appelle *sommet interne* les sommets $x_1, ..., x_{\ell-1}$ n'étant pas source ou but de γ. Pour un graphe (L, J) valué par d avec (sans perte de généralité)

$L = \{1, 2, ..., m\}$ et pour $\ell \in \{0, 1, ..., m\}$, soit

$$\pi^{(\ell)}(x, y) = \inf\{d(\gamma) : \ \gamma \text{ chemin de } x \text{ à } y \text{ avec sommets internes dans } \{1, ..., \ell\}\}, \qquad (3.2)$$

avec la convention que l'infimum d'un ensemble vide est $= +\infty$. Notons que la matrice recherchée π coïncide avec $\pi^{(m)}$, et que la matrice $\pi^{(0)}$ est facile à initialiser à partir des valeurs des arcs.

3.3.12. Algorithme de Roy/Warshal

Hypothèse : *L'ensemble des sommets est de la forme* $L = \{1, 2, ..., m\}$.
Invariant *(après l'itération* ℓ*)* : $\forall x, y \in L$, $\pi(x, y) = \pi^{(\ell)}(x, y)$
Initialisation : *pour* $x, y \in L$: $\pi(x, x) = 0$, $\pi(x, y) = d_{(x,y)}$ *si* $(x, y) \in J$, $x \neq y$
et $\pi(x, y) = +\infty$ *sinon.*
Calcul : *pour* $\ell = 1, 2, ..., m$ *faire*
pour $x = 1, 2, ..., m$ *faire*
pour $y = 1, 2, ..., m$ *faire*
$\pi(x, y) \leftarrow \min(\pi(x, y), \pi(x, \ell) + \pi(\ell, y))$
STOP si $\pi(x, x) \neq 0$ *(*$\Longrightarrow \exists$ *circuit absorbant)*
Complexité : $\mathcal{O}(m^3)$ *(plus cher que Bellman ou Dijkstra, mais aucune hypothèse sur le graphe)*

Démonstration. L'invariant est montré par récurrence sur ℓ : dans la définition de $\pi^{(0)}$, on prend en compte seulement les chemins sans sommets internes, c.à-d., les arcs, et alors $\pi = \pi^{(0)}$ après l'initialisation.
Supposons maintenant que l'invariant $\pi = \pi^{(\ell-1)}$ soit vrai au début de l'itération d'indice ℓ, et en particulier $\forall x \in L : \pi^{(\ell-1)}(x, x) = 0$ (sinon on aurait arrêté l'algorithme avant). Nous procédons en deux étapes.
(i) Montrons que l'algorithme s'arrête pendant l'itération ℓ *si et seulement s'il existe* $x \in L$ *de sorte que le sous-graphe induit par* $\{1, ..., \ell, x\}$ *contient un circuit absorbant.* Soit $x \in L$ avec $\pi^{(\ell-1)}(x, \ell) + \pi^{(\ell-1)}(\ell, x) < 0$. Dans ce cas, en concatenant les chemins correspondant aux deux valeurs on aura trouvé un circuit absorbant passant par les sommets $\{1, 2, ..., \ell, x\}$, comme énoncé dans l'algorithme. Réciproquement, s'il existe un circuit absorbant dans le sous-graphe induit par $\{1, 2, ..., \ell, x\}$, alors ce circuit passe forcément par ℓ et x, et $x \notin \{1, 2, ..., \ell\}$ (par hypothèse de récurrence). D'après le lemme 3.1.6 de König et sa preuve, nous pouvons supposer que ce circuit est élémentaire (ou on peut en extraire un circuit absorbant élémentaire qui doit aussi passer par ℓ et x). Par conséquent, les deux parties de ℓ à x et de x à ℓ de ce circuit ont leurs sommets internes dans $\{1, ..., \ell - 1\}$, et, par définition de $\pi^{(\ell-1)}$, la valeur du circuit est minorée par la quantité $\pi^{(\ell-1)}(x, \ell) + \pi^{(\ell-1)}(\ell, x) < 0$. Donc, par hypothèse de récurrence, l'algorithme s'arrête.
(ii) Montrons que $\pi = \pi^{(\ell)}$ *à la fin de l'itération d'indice* ℓ *:* D'après la partie (i) et le lemme 3.3.1, nous avons $\pi^{(\ell)}(x, y) = d(\gamma)$ pour un chemin γ avec sommets internes dans $\{1, 2, ..., \ell\}$ qui passe au plus une fois par le sommet ℓ. Si γ passe une fois par ℓ alors les parties de x à ℓ et de ℓ à y admettent des sommets internes dans $\{1, ..., \ell - 1\}$, et donc $d(\gamma) = \pi^{(\ell-1)}(x, \ell) + \pi^{(\ell-1)}(\ell, y)$ par l'exercice 3.3.2(a), l'inégalité $\pi^{(\ell)}(x, y) \leq \pi^{(\ell-1)}(x, y)$ étant vrai par définition. Si γ ne passe pas par ℓ alors $d(\gamma) = \pi^{(\ell-1)}(x, y)$ par définition de $\pi^{(\ell-1)}$, et $d(\gamma) \leq \pi^{(\ell-1)}(x, \ell) + \pi^{(\ell-1)}(\ell, y)$ par minimalité de γ, en concatenant les chemins correspondant aux deux termes de la somme. □

Nous allons maintenant généraliser quelques propriétés mises en évidence dans les algorithmes de plus courts chemins. Pour motiver une telle généralisation, considérons d'abord deux exemples. Le premier concerne le problème de tonnage maximum : on se donne un graphe (L, J) représentant un réseau routier, et pour chaque arc (route) $j = (\ell, k)$ une valeur $d(j)$ qui nous indique que tout camion de poids $\leq d(j)$ peut emprunter cette route. Par conséquent, faire passer un camion par le chemin $\gamma = (j_1, j_2, ..., j_\kappa)$ nécessite que son poids soit inférieur ou égal à

$$d(\gamma) = \min\{d(j_1), ..., d(j_\kappa)\}.$$

On souhaite déterminer un chemin à source et but fixé permettant de faire passer les camions les plus lourds.

Comme deuxième exemple, considérons le problème de fiabilité maximum, où on se donne un *graphe probabiliste* : ici les sommets représentent les différents états d'un système, et la valeur $d(j) \in [0, 1]$ d'un arc $j = (k, \ell)$ représente la probabilité de passer d'un état ℓ à un état k en une transition. Un graphe probabiliste admet alors la propriété [4]

$$\forall \ell \in L : \qquad \sum_{k \in \Gamma^+(\ell)} d((\ell, k)) = 1.$$

La probabilité de passer d'un état initial s_0 à un état final s_κ en passant par des états intermédiaires $s_1, ..., s_{\kappa-1}$ serait alors

$$d([s_0, ..., s_\kappa]) = d((s_0, s_1)) \times d((s_1, s_2)) \times .... \times d((s_{\kappa-1}, s_\kappa)),$$

et on s'intéresse au problème de déterminer un chemin γ à source et but fixé de fiabilité $d(\gamma)$ maximum.

Dans les deux cas, la valeur d'un chemin est obtenu par une opération abstraite

$$d([s_0, ..., s_\kappa]) = d((s_0, s_1)) \otimes d((s_1, s_2)) \otimes .... \otimes d((s_{\kappa-1}, s_\kappa)), \tag{3.3}$$

et on s'intéresse à calculer

$$x, y \in L : \qquad \pi(x, y) = d(\gamma_1) \oplus d(\gamma_2) \oplus ... \tag{3.4}$$

pour une opération abstraite $\oplus$ (ici le maximum) avec $\gamma_1, \gamma_2, ...$ décrivant tous les chemins de source x et de but y. Bien entendu, pour poser le problème correctement dans un cadre abstrait, il nous faut un certain nombre de propriétés pour les opérations $\otimes$ et $\oplus$. Par exemple, on pourrait supposer que les valeurs des arcs se trouvent dans un ensemble abstrait D, avec $(D, \oplus, 0_\oplus, \otimes, 1_\otimes)$ un anneau, c'est-à-dire, $(D, \oplus, 0_\oplus)$ un groupe abélien avec élément neutre $0_\oplus$ (fermeture, associativité, commutativité, existence d'un élément neutre $0_\oplus$ et d'un élément inverse), et $(D, \otimes, 1_\otimes)$ un monoïde (fermeture, associativité, existence d'un élément neutre $1_\otimes$, mais pas forcément commutativité), et en plus des propriétés de distributivité, ce qui implique que

$$\forall d \in D : \qquad 0_\oplus \otimes d = d \otimes 0_\oplus = 0_\oplus. \tag{3.5}$$

4. Le graphe probabiliste le plus important est peut-être le graphe du moteur de recherche Google. Ici les sommets représentent les différents pages Web dans le monde. On introduit un arc $j = (\ell, k)$ si sur la page ℓ il se trouve un lien vers la page k. Pour trouver des probabilités, on suppose que, en moyenne, un utilisateur n'a pas de préférences particulières, et alors $d((\ell, k))$ coïncide avec le nombre de liens sur la page ℓ vers la page k, divisé par le nombre de liens sur la page ℓ (pour les pages sans lien, on introduit des boucles (ℓ, ℓ) avec $d((\ell, \ell)) = 1$).

Pourtant, pour les problèmes de tonnage, les valeurs d des arcs appartiennent naturellement à $[0, +\infty]$, et donc d n'admet pas forcément un élément inverse par rapport à $\oplus$. C'est pour cela que les problèmes de cheminement sont posés dans un cadre d'une dioïde $(D, \oplus, 0_\oplus, \otimes, 1_\otimes)$ qui admet toutes les propriétés d'un anneau sauf l'existence d'un élément inverse par rapport à $\oplus$, comparer par exemple [GoMi95, Chapitre 3]. Autrement dit, nous supposons que $(D, \oplus, 0_\oplus)$ est un monoïde abélien, $(D, \otimes, 1_\otimes)$ un monoïde, avec des propriétés de distributivité

$$\forall a, b, c \in D : \qquad a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c), \quad (b \oplus c) \otimes a = (b \otimes a) \oplus (c \otimes a),$$

et on demande que (3.5) reste valable.

3.3.13. Exemples d'applications pour une dioïde
Différents problèmes de cheminement peuvent s'insérer dans notre formalisme :
(a) *Pour des problèmes de tonnage maximal, on choisit* $D = [0, +\infty]$, $\otimes = \min$, $\oplus = \max$, *et alors* $0_\oplus = +\infty$, $1_\otimes = 0$.
(b) *Pour des problèmes de chemin de fiabilité maximale on choisit* $D = [0, 1]$, $\otimes = \times$, $\oplus = \max$, *et alors* $0_\oplus = 0$, $1_\otimes = 1$.
(c) *Pour des problèmes de chemin de valeur minimale on choisit* $D = [-\infty, +\infty)$, $\otimes = +$, $\oplus = \min$, *et alors* $0_\oplus = +\infty$, $1_\otimes = 0$ *(et* $\forall j \in J : d(j) = 1$ *si on compte en nombre d'arcs).*
(d) *Pour des problèmes de chemin de valeur maximale on choisit* $D = (-\infty, +\infty]$, $\otimes = +$, $\oplus = \max$, *et alors* $0_\oplus = -\infty$, $1_\otimes = 0$ *(et* $\forall j \in J : d(j) = 1$ *si on compte en nombre d'arcs).*
(e) *Pour tester l'existence d'un chemin avec source et but fixé on choisit l'algèbre de Boole :* $D = \{0, 1\}$, $\otimes = \min$, $\oplus = \max$, *et alors* $0_\oplus = 0$, $1_\otimes = 1$, *et* $\forall j \in J : d(j) = 1$.
(f) *Pour compter le nombre de chemins avec source et but fixé on choisit* $D = \{0, 1, 2, 3, ...\}$, $\otimes = \times$, $\oplus = +$, *et alors* $0_\oplus = 0$, $1_\otimes = 1$, *et* $\forall j \in J : d(j) = 1$.

On laissera au lecteur le soin de vérifier que, pour tous ces exemples, les axiomes d'une dioïde sont satisfaits.

Pour un chemin (γ_1, γ_2) obtenu par concatenation de chemins γ_1 et γ_2 avec $but(\gamma_1) = source(\gamma_2)$, la définition (3.3) nous dit que

$$d((\gamma_1, \gamma_2)) = d(\gamma_1) \otimes d(\gamma_2).$$

En ajoutant à la définition (3.3) la convention $d([s_0]) = 1_\otimes$ pour des chemins triviaux réduits à un seul sommet (ce qui veut dire que l'on ne se déplace pas), nous concluons que cette relation de concatenation reste aussi vraie si un des chemins γ_1 ou γ_2 est trivial.

Etant donnée une dioïde $(D, \oplus, 0_\oplus, \otimes, 1_\otimes)$, on considère l'ensemble $D^{m\times m}$ de matrices carrées d'ordre m à éléments dans D, muni avec des opérations matricielles habituelles

$$(a_{j,k})_{j,k} \oplus (b_{j,k})_{j,k} := (a_{j,k} \oplus b_{j,k})_{j,k}, \quad (a_{j,k})_{j,k} \otimes (b_{j,k})_{j,k} := (a_{j,1} \otimes b_{1,k} \oplus \cdots \oplus a_{j,m} \otimes b_{m,k})_{j,k}.$$

Notons que $(D^{m\times m}, \oplus, 0_\oplus, \otimes, U)$ est également une dioïde, avec l'élément neutre de $\oplus$ donné par la matrice $0_\oplus$ comportant que des éléments $0_\oplus$, et l'élément neutre de $\otimes$ donné par la matrice U comportant sur la diagonale $1_\otimes$, et sinon que des $0_\oplus$. Nous définissons la matrice d'adjacence généralisée [5] $B = B^L_L$ par

$$B^y_x = d((x, y)) \text{ si } (x, y) \in J, \quad \text{et} \quad B^y_x = 0_\oplus \text{ si } (x, y) \notin J.$$

5. La matrice d'adjacence de 3.1.4 est obtenue pour l'algèbre de Boole

Les éléments de B peuvent être considérés comme valuation d'un graphe complet $(L, L \times L)$ obtenu en complétant le graphe valué (L, J) par $d(j) = 0_\oplus$ pour tout arc ajouté $j = (\ell, k) \in (L \times L) \setminus J$. Comme conséquence de (3.5), nous obtenons $d(\gamma) = 0_\oplus$ pour tout chemin dans le graphe complet $(L, L \times L)$ qui comporte un arc n'appartenant pas au graphe (L, J) de départ. Par conséquent, la valeur $\pi(x, y)$ définie dans (3.4) reste la même en passant de (L, J) au graphe complet $(L, L \times L)$.

Considérons également les puissances $B^{p\otimes}$ défini par $B^{0\otimes} = U$, $B^{(p+1)\otimes} = B^{p\otimes} \otimes B$. Grâce à (3.5), on conclut comme dans l'exercice **??** que l'élément à la position (x, y) de la matrice $B^{p\otimes}$ nous donne

$$[B^{p\otimes}]_x^y = d(\gamma_1) \oplus d(\gamma_2) \oplus ...$$

avec $\gamma_1, \gamma_2, ...$ chemins de longueur p de source x et de but y. Par conséquent, dans

$$B^{0\otimes} \oplus B^{1\otimes} \oplus \cdots \oplus B^{p\otimes}$$

on considère tout chemin de longueur au plus p. Pour le tableau π défini dans (3.4) il faudra alors faire tendre p vers infini, ce qui naturellement pose des problèmes de convergence, par exemple pour l'exemple 3.3.13(f) où la somme infinie diverge pour certaines composantes si le graphe comporte des circuits. [6] En généralisant 3.3.1, un chemin fermé [7] γ est dit absorbant ssi

$$d(\gamma) \oplus 1_\otimes \neq 1_\otimes. \tag{3.6}$$

Comme dans la preuve du lemme 3.3.1 on montre (en utilisant fortement la distributivité) que le problème (3.4) est bien posé quelque soit $x, y \in L$ si et seulement s'il n'existe pas un circuit absorbant, et, dans ce cas, il suffit de prendre en considération dans (3.4) seulement les chemins élémentaires. Comme un chemin élémentaire comporte au plus m arcs (au cas ou il est fermé), il en découle la propriété suivante

3.3.14. Corollaire
Soit $G = (L, J)$ valué par une dioïde $(D, \oplus, 0_\oplus, \otimes, 1_\otimes)$. Si G ne comporte pas de chemins fermés absorbants alors

$$\pi = B^{0\otimes} \oplus B^{1\otimes} \oplus \cdots \oplus B^{m\otimes}.$$

Notons que, par construction, il n'y a pas de circuit absorbant dans les exemples 3.3.13(a), (b), (e), et que tout circuit est absorbant dans l'exemple 3.3.13(f).

Le tableau π peut être calculé par une généralisation de l'algorithme 3.3.12 de Roy/Warshal donnée ci-dessous. Une preuve est omise, elle est essentiellement identique à celle donnée dans 3.3.12.

3.3.15. Algorithme de Roy/Warshal généralisé

6. Pour un graphe probabiliste, la matrice d'adjacence généralisée B peut être interprétée comme une *matrice de transition* d'une chaîne de Markov : $B_x^y \in [0, 1]$ décrit le pourcentage de l'espèce x se transformant en espèce y en une transition. De même, pour la puissance classique B^p ($\otimes = \times$, $\oplus = +$), la quantité $(B^p)_x^y$ décrit le pourcentage de l'espèce x se transformant en espèce y après p transistions. Notons que $B(1, ..., 1)^T = (1, 1, ..., 1)$ par définition d'un graphe probabiliste, et donc $\sum_{j=0}^{\infty} B^j$ ne peut pas converger.

7. Notons que, pour une opération $\otimes$ non commutative, la valeur d'un circuit n'est pas défini, car il faudra préciser le sommet source. Pour rassurer le lecteur : dans tous nos exemples l'opération $\otimes$ est commutative, et on peut alors continuer d'identifier des chemins fermés avec des circuits.

Hypothèse : $G = (L, J)$ *un graphe orienté valué par une dioïde* $(D, \oplus, 0_\oplus, \otimes, 1_\otimes)$
$L = \{1, 2, ..., m\}$ *(sans perte de généralité).*
Invariant *(après l'itération* ℓ*)* : $\forall x, y \in L : \pi(x, y) = d(\gamma_1) \oplus d(\gamma_2) \oplus ...,$
avec γ_j *parcourant l'ensemble des chemins élémentaires de* x *à* y
avec sommets internes dans $\{1, 2, ..., `\ell\}$.
Initialisation : $\pi = U \oplus B$, *c.-à-d., pour* $x, y \in L$: $\pi(x, x) = 1_\otimes$,
$\pi(x, y) = d((x, y))$ *si* $(x, y) \in J$, $x \neq y$, *et* $\pi(x, y) = 0_\oplus$ *sinon.*
Calcul : *pour* $\ell = 1, 2, ..., m$ *faire*
pour $x = 1, 2, ..., m$ *faire*
pour $y = 1, 2, ..., m$ *faire*
$\pi(x, y) \leftarrow \pi(x, y) \oplus (\pi(x, \ell) \otimes \pi(\ell, y))$
STOP si $\pi(x, x) \neq 1_\otimes$ *(*$\Longrightarrow \exists$ *chemin fermé absorbant)*
Complexité : $\mathcal{O}(m^3)$

Nous donnons dans le fichier `don_ch3/warshal.bat` une simulation pour d'un graphe à 10 sommets et l'algèbre de Boole, voir l'exemple 3.3.13(e). On cherche alors à savoir pour tout $j, k \in S$ s'il existe un chemin avec source j et but k. Pour notre graphe nous trouvons à la fin une matrice remplie par 1, autrement dit, ce graphe est fortement connexe.

3.3.16. Exercice
Dans un graphe $G = (L, J)$ *sans circuits valué par une dioïde* $(D, \oplus, 0_\oplus, \otimes, 1_\otimes)$, *donner l'équivalent de l'algorithme de Bellman.*

3.3.17. Exercice
Soit $G = (L, J)$ *valué par une dioïde* $(D, \oplus, 0_\oplus, \otimes, 1_\otimes)$.

1. *Montrer que si* G *ne comporte pas de chemins fermés absorbants alors*

$$\pi = U \oplus (\pi \otimes B) = U \oplus (B \otimes \pi).$$

2. *Considérons l'itération de Picard pour résoudre cette équation de point fixe :*

$$x^{(0)} = U, \quad \text{et pour } k = 0, 1, ... \quad x^{(k+1)} = U \oplus (x^{(k)} \otimes B)$$

   (a) *Vérifier que* $x^{(k)} = B^{0\otimes} \oplus B^{1\otimes} \oplus \cdots \oplus B^{k\otimes}$.

   (b) *Dans le cas* $x^{(m+1)} \neq x^{(m)}$, *vérifier que* G *comporte un chemin fermé absorbant.*

   (c) *Dans le cas* $x^{(k)} = x^{(k+1)}$, *montrer que* $\pi = x^{(k)}$ *(on peut arrêter l'algorithme).*

   (d) *Soit maintenant* $y^{(k)}$ *la première ligne de* $x^{(k)}$.

      i. *Donner l'initialisation et la formule de récurrence pour les composantes de* $y^{(k)}$.

      ii. *Démontrer que les deux dernières propriétés énoncées pour* $x^{(k)}$ *sont aussi valables pour* $y^{(k)}$.

   On vient d'énoncer et de démontrer l'algorithme de Bellman dans un graphe avec circuits [GoMi95, Chapitre 2.2].

3.4 Problèmes de flot

Dans ce chapitre on se donne un graphe $G = (L, J)$ orienté, sa matrice d'incidence (sommets-arcs) $A = A_L^J$, et des bornes de capacité $\forall j \in J : -\infty \leq b_j \leq c_j \leq +\infty$.

3.4.1. Définition : flot (réalisable)
On appelle flot *tout vecteur $\phi = \phi_J$ de sorte que $A\phi = 0$. Le flot ϕ est dit* réalisable *si $\phi \geq b$ et $\phi \leq c$.*

Notons que

$$(A\phi)_\ell = \underbrace{\sum_{j \in J, source(j)=\ell} \phi_j}_{\text{quantité partante du sommet } \ell} - \underbrace{\sum_{j \in J, but(j)=\ell} \phi_j}_{\text{quantité entrante au sommet } \ell} . \tag{3.7}$$

Par conséquent, la relation $A\phi = 0$ pour un flot peut être interprétée comme une loi de conservation en tout sommet : il n'y a pas de pertes.

Comme déjà mentionné dans l'introduction de ce chapitre, un flot peut représenter une quantité par tuyau à transporter dans un système de tuyaux (arcs) reliés entre eux par des stations de pompage (sommets). On peut également s'imaginer un réseau de métro, avec stations représentées par les sommets. Ici, pour un morceau $j = (\ell, k) \in J$ d'une ligne entre les deux stations consécutives ℓ et k, la composante ϕ_j d'un flot peut représenter le nombre de personnes par heure empruntant j, avec limitations $b_j \leq \phi_j \leq c_j$ au niveau des capacités (nombre limité de rames par heure). Plus précisément, pour tenir compte du flux des voyageurs dans les deux sens entre ces deux stations, on pourrait introduire des arcs (ℓ, k) et (k, ℓ) avec contraintes de la forme $0 \leq \phi_{(\ell,k)} \leq \beta$, $0 \leq \phi_{(k,\ell)} \leq \beta'$, $\beta, \beta' \geq 0$. Un modèle encore plus simple est obtenu en remplaçant ces deux arcs par un seul arc $j = (\ell, k)$, avec contrainte $-\beta' \leq \phi_j \leq \beta$, et $\phi_j < 0$ signifiant que l'on transporte des personnes dans le sens opposé par rapport à l'orientation de l'arc j. Bien entendu, dans le modèle d'un réseau de métro la loi de conservation pour le sommet ℓ correspond au fait que personne sort ou entre à la station ℓ. Pour certaines applications décrites ci-dessous, il convient alors d'ajouter d'autres sommets/arcs a (L, J) qui ne correspondent pas à des stations ou tronçons d'une ligne de metro. Aussi, notons que notre flot ne dépend pas du temps et décrit une situation d'équilibre où le temps de trajet pour se rendre d'une station àune autre est négligé. On renvoie le lecteur intéressé à l'exemple 3.4.13 pour la prise en compte du temps de trajet et une discussion des flots dynamiques.

Le problème d'optimisation linéaire

$$\min\{fx : Ax = 0, x \geq b, x \leq c\}$$

pour notre matrice d'incidence A d'un graphe orienté peut être interprété comme un problème de recherche d'un flot réalisable de coût minimal, sachant que la composante f^j nous donne le coût unitaire pour l'arc $j \in J$. Bien entendu, il est possible d'appliquer Simplex pour résoudre un tel problème, mais la matrice de coefficients admet une structure tout à fait particulière : ceci nous permettra de simplifier Simplex pour les problèmes sur les graphes au chapitre 3.5. Dans le présent chapitre nous allons d'abord évoquer quelques problèmes qui se modélisent à l'aide des flots. Ensuite nous proposons des algorithmes pour résoudre deux problèmes particuliers : la recherche d'un flot réalisable (algorithme de Hoffman) et le problème du flot maximum (algorithme de Ford-Fulkerson), les deux problèmes étant reliés entre eux.

3.4.2. Définition : problème du flot maximum
Pour le problème du flot maximum, on dispose de deux sommets distincts $\overline{s} \in L$ (la source*) et $\overline{p} \in L$ (le* puits*), et on suppose que l'on a déjà rajouté au graphe un* arc de retour *$\overline{j} = (\overline{p}, \overline{s})$, avec bornes de capacité $b_{\overline{j}} = -\infty$ et $c_{\overline{j}} = +\infty$. On se pose le problème de trouver un flot réalisable ϕ avec $\phi_{\overline{j}}$ le plus grand possible.*

Mathématiquement, on peut écrire le problème du flot maximum comme

$$\max\{fx : Ax = 0, x \geq b, x \leq c\}, \quad f^{(\overline{p},\overline{s})} = 1, \quad f^{J-(\overline{p},\overline{s})} = 0.$$

Notons que la loi de conservation $A\phi = 0$ est seulement valable après ajout de l'arc de retour (qui dans un réseau de métro est tout à fait artificiel car il ne représente pas un tronçon d'une ligne de métro).

3.4.3. Remarque : quelques généralisations du problème du flot maximum
(a) *Si on on souhaite transporter le même bien depuis plusieurs sources $s_1, ..., s_r$ vers plusieurs puits $p_1, ..., p_q$, on peut toujours modifier le graphe de sorte à se ramener à un flot (maximum) : on ajoute deux sommets $\overline{s}$ (la super-source) et $\overline{p}$ (le super-puits), ainsi que les arcs*

$$j = (\overline{s}, s_i) \quad \textit{pour } i = 1, ..., r$$

avec b_j (et c_j) désignant la quantité minimale (et maximale, respectivement) partante de la source s_i, ainsi que les arcs

$$j = (p_i, \overline{p}) \quad \textit{pour } i = 1, ..., q$$

avec b_j (et c_j) désignant la quantité minimale (et maximale, respectivement) qui arrive au puits p_i, et finalement l'arc de retour $(\overline{p}, \overline{s})$.
(b) *Dans les applications comme le réseau de métro, il n'y a pas seulement des restrictions de capacité au niveau des arcs mais aussi parfois au niveau des sommets. L'idée suivante permet de se ramener à un problème de flot : si on souhaite que la quantité passant par le sommet ℓ soit comprise entre les bornes b_ℓ et c_ℓ, on ajoute pour un tel sommet ℓ un sommet supplémentaire ℓ' (une copie), et les arcs dans le graphe d'origine avec source ℓ auront la nouvelle source ℓ'. Finalement, on introduit le nouvel arc $j' = (\ell, \ell')$ avec bornes de capacité $b_{j'} = b_\ell$, $c_{j'} = c_\ell$, car toute quantité passant par l'ancien sommet ℓ doit maintenant passer par l'arc $j' = (\ell, \ell')$.*

Discutons plus en détail deux problèmes classiques d'optimisation combinatoire où l'on peut se ramener à un problème de flot (généralement de coût minimal).

3.4.4. Exemple : problème de transport/transbordement
Dans le problème de transbordement on dispose d'un graphe orienté $G = (L, J)$ où on partitionne l'ensemble de sommets en trois parties : les usines représentées par $L_1 \subset L$, les clients représentés par $L_2 \subset L$, et finalement des entrepôts représentés par $L_3 = L \setminus (L_1 \cup L_2)$. Ici l'arc $j = (\ell, k)$ modélise la possibilité de transporter un bien d'un partenaire $\ell \in L_1 \cup L_3$ à $k \in L_2 \cup L_3$, avec contraintes naturelles de capacité b_j,c_j pour chaque route j. Pour chaque usine ℓ on connaît la quantité disponible a_ℓ et pour chaque client ℓ on connaît la quantité demandée a_ℓ (on retrouve notre problème 1.2.4 de transport pour $L_3 = \emptyset$ et $c_j = +\infty$). Pour un problème de transbordement, l'objectif est de minimiser le coût de transport sachant les coûts unitaires de transport f^j sur chaque route j.

Nous considérons les usines comme sources et les clients comme puits, et posons $b_j = 0$, $c_j = a_\ell$ pour tous les arcs $j = (\overline{s}, \ell)$ ajoutés en 3.4.3(a) entre super-source $\overline{s}$ et sources (pour de pas dépasser la quantité disponible à l'usine ℓ), et finalement nous posons $b_j = a_\ell$, $c_j = +\infty$ pour tous les arcs ajoutés en 3.4.3(a) entre puits et super-puits (pour livrer au moins ce qui a été demandé). Par conséquent, la recherche d'un plan de transport admissible revient à la recherche d'un flot réalisable. De même, la recherche d'un plan de transport minimisant le coût devient un problème de flot de coût minimal.

Finalement, notons que l'on peut traduire des éventuelles contraintes de capacité au niveau des entrepôts par l'introduction d'une copie de L_3 comme décrit dans 3.4.3(b).

3.4.5. Exemple : problème d'affectation
Ici on dispose d'une liste L_1 de personnes et d'une liste L_2 de tâches. Pour chaque personne $\ell \in L_1$, on connaît la liste de tâches $\Gamma^+(\ell) \subset L_2$ que ℓ saurait effectuer (après avoir suivi une formation). Sachant que

— *une personne effectue au plus une tâche,*

— *une tâche est faite au plus une fois,*

— *on connaît le coût de formation $f^{(\ell,k)}$ si la personne ℓ est choisie pour faire la tâche k,*

le problème d'affectation consiste à faire travailler N personnes au moindre coût. Un problème voisin est de faire travailler simultanément un maximum de personnes (sans faire attention au coût engendré).

Les listes $\Gamma^+()$ nous permettent d'introduire un premier graphe bi-parti (voir 3.1.2) avec comme sommets $L = L_1 \cup L_2$, les arcs dans $J \subset L_1 \times L_2$ représentant la liste des affectations possibles. En introduisant pour $j \in J$ les bornes $b_j = 0$, $c_j = 1$, on sait que tout vecteur ϕ respectant ces bornes et à coefficients entiers aura des composantes $\in \{0, 1\}$, avec comme interprétation

$$(\ell, k) \in L_1 \times L_2 : \quad \phi_{(\ell,k)} = \begin{cases} 1 & \text{si la personne } \ell \text{ effectue la tâche } k, \\ 0 & \text{sinon.} \end{cases}$$

Nous complétons notre graphe suivant 3.4.3(a) en considérant les personnes comme sources et les tâches comme puits. A condition que l'on considère seulement les flots à coefficients entiers (comparer avec Exercice 3.4.14), les bornes $b_j = 0$, $c_j = 1$ pour un arc entre super-source et source signifient qu'une personne effectue au plus une tâche, et les mêmes bornes pour les arcs entre puits et super-puits nous permettent de vérifier qu'une tâche est faite au plus une fois. Finalement, le flot sur l'arc de retour nous indique le nombre de tâches effectués (ou, suivant la loi de conservation, le nombre de personnes qui travaillent). Donc le problème de faire travailler un maximum de personnes revient à trouver un flot maximum à coefficients entiers. Par contre, pour trouver l'affectation au moindre coût on impose $b_{\overline{p},\overline{s}} = c_{\overline{p},\overline{s}} = N$ et on retrouve un problème de flot à coefficients entiers de coût minimal.

Pour un flot donné ϕ, le graphe d'écart $G(\phi)$ permet de visualiser les possibilités de changement de flot tout en respectant les bornes.

3.4.6. Définition : graphe d'écart
Pour un graphe $G = (L, J)$ avec flot ϕ et arc de retour $(\overline{p}, \overline{s})$, le graphe d'écart *$G(\phi) = (L, J(\phi))$ est donné par*

$$J(\phi) = \{(\ell, k) : (\ell, k) \in J \setminus \{(\overline{p}, \overline{s})\}, \phi_{(\ell,k)} < c_{(\ell,k)}\} \cup \{(k, \ell) : (\ell, k) \in J \setminus \{(\overline{p}, \overline{s})\}, \phi_{(\ell,k)} > b_{(\ell,k)}\},$$

c'est-à-dire, pour chaque arc $j \in J$ différent de l'arc de retour, on le garde dans le graphe d'écart si on peut augmenter le flot sur j, et on garde son opposé dans le graphe d'écart si on peut diminuer le flot sur j.

Notons qu'un chemin (élémentaire) γ_0 de $\overline{s}$ à $\overline{p}$ dans le graphe d'écart correspond à une chaîne (élémentaire) dans le graphe de départ (car l'orientation d'un certain nombre d'arcs risque d'avoir été changée), qui devient un cycle (élémentaire) γ si on ajoute l'arc de retour. Aussi, à chaque arc dans γ_0 on peut associer une marge $\in \{\phi_{(\ell,k)} - b_{(\ell,k)}, c_{(\ell,k)} - \phi_{(\ell,k)}\}$ strictement positive indiquant le maximum de changement du flot en respectant l'orientation de l'arc dans le graphe d'écart. La plus petite de ces marges est appelée la capacité du cycle γ. Voici une définition plus formelle. [8]

3.4.7. Définition : vecteur/capacité d'une chaîne élémentaire
Soit $\gamma = (j_1, ..., j_\beta) = [\ell_0, \ell_1, ..., \ell_\beta]$ une chaîne (ou un cycle) élémentaire dans (L, J) (et donc $j_\alpha \in \{(\ell_{\alpha-1}, \ell_\alpha), (\ell_\alpha, \ell_{\alpha-1})\}$). Alors son vecteur $\vec{\gamma} \in \mathbb{Z}^{|J|}$ est défini par

$$\vec{\gamma}_j = \begin{cases} 1 & \text{si } j = j_\alpha \text{ et but}(j) = \ell_\alpha \quad \text{(on emprunte } j \text{ en respectant l'orientation),} \\ -1 & \text{si } j = j_\alpha \text{ et but}(j) = \ell_{\alpha-1} \quad \text{(on emprunte } j \text{ dans le sens opposé),} \\ 0 & \text{sinon} \quad \text{(on n'emprunte pas } j\text{).} \end{cases}$$

Pour un $x \in \mathbb{R}^J$, la capacité d'une chaîne (d'un cycle) élémentaire est définie par

$$cap_x(\gamma) := \inf\{x_j - b_j : \vec{\gamma}_j < 0\} \cup \{c_j - x_j : \vec{\gamma}_j > 0\}. \tag{3.8}$$

Nous sommes maintenant préparés pour énoncer et démontrer l'algorithme de Ford–Fulkerson.

3.4.8. Algorithme de Ford-Fulkerson

Hypothèse : $G = (L, J)$ *un graphe comportant un arc de retour* $(\overline{p}, \overline{s})$,
$b, c \in \mathbb{R}^J$ *des bornes,* $\overline{s} \in L$ *la source,* $\overline{p} \in L$ *le puits.*
Invariant : $\phi \in \mathbb{R}^J$ *un flot réalisable.*
Itération : *tant qu'il existe un chemin élémentaire γ_0 de $\overline{s}$ à $\overline{p}$ dans le graphe d'écart $G(\phi)$*
construire γ, le cycle élémentaire dans G associé à γ_0
Mettre à jour le flot : $\phi \leftarrow \phi + cap_\phi(\gamma)\,\vec{\gamma}$
Résultat : *une solution ϕ du problème de flot maximum.*

Démonstration. Montrons d'abord qu'une itération de l'algorithme de Ford-Fulkerson produit un flot réalisable, avec le flot $\phi_{(\overline{p},\overline{s})}$ sur l'arc de retour strictement croissant. On observe sans grandes difficultés que le vecteur d'un cycle élémentaire est un flot (voir le lemme 3.5.4(b) ci-dessous), et alors $\phi + \theta\vec{\gamma}$ est un flot pour tout $\theta \geq 0$. Egalement, pour $j = (\ell, k) \in J$

$$(\phi + \theta\vec{\gamma})_j = \begin{cases} \phi_j + \theta & \text{si } j = (\overline{p}, \overline{s}), \text{ l'arc de retour,} \\ \phi_j + \theta & \text{si } j \neq (\overline{p}, \overline{s}), \text{ et l'arc } (\ell, k) \text{ est emprunté dans } \gamma_0 \\ \phi_j - \theta & \text{si } j \neq (\overline{p}, \overline{s}), \text{ et l'arc } (k, \ell) \text{ est emprunté dans } \gamma_0 \end{cases}$$

8. Pour être plus précis, tant que (L, J) ne comporte pas simultanément un arc (ℓ, k) et son opposé (k, ℓ), il y a une bijection entre chemins de $\overline{s}$ à $\overline{p}$ dans le graphe d'écart et cycles de capacité > 0 comportant l'arc de retour $(\overline{p}, \overline{s})$ dans (L, J). Dans le cas général, il se peut que $j \in J(\phi)$ car $\phi_j < c_j$ et simultanément $\phi_{j'} > b_{j'}$ pour l'arc j' opposé à j : à ce moment, on pourrait construire la chaîne dans (L, J) en prénant un de ces deux arcs j, j', en donnant par exemple préférence à celui avec la plus grande marge.

Dans le premier cas on n'atteint jamais les bornes, car $c_{(\overline{p},\overline{s})} = +\infty$ par définition d'un arc de retour. Par la définition de la capacité d'un cycle nous savons que $\phi+\theta\vec{\gamma}$ est réalisable pour $\theta \geq 0$ si et seulement si $\theta \in [0, \mathrm{cap}_\phi(\gamma)]$. Finalement, la relation $\mathrm{cap}_\phi(\gamma) > 0$ provient de la définition du graphe d'écart : pour chaque arc de γ_0, il y avait une marge non-triviale pour changer le flot. Il reste à montrer que l'on a trouvé un flot optimum ϕ si on ne peut plus construire un chemin élémentaire de $\overline{s}$ à $\overline{p}$ dans le graphe d'écart $G(\phi)$. Notons par $M \subset L$ l'ensemble des sommets accessibles depuis $\overline{s}$ dans le graphe d'écart $G(\phi)$. Par construction, $\overline{s} \in M$, et par hypothèse (plus éventuellement le lemme 3.1.6 de König) nous savons que $\overline{p} \notin M$. En sommant la loi de conservation d'un flot pour $\ell \in M$ nous obtenons

$$\sum_{j\in J, source(j)\in M, but(j)\notin M} \phi_j = \sum_{j\in J, source(j)\notin M, but(j)\in M} \phi_j.$$

Notons que le flot sur l'arc de retour $(\overline{p},\overline{s})$ se trouve à droite de cette égalité. En notant $J' := J \setminus \{(\overline{p},\overline{s})\}$, nous obtenons

$$\phi_{(\overline{p},\overline{s})} = \sum_{j\in J', source(j)\in M, but(j)\notin M} \phi_j - \sum_{j\in J', source(j)\notin M, but(j)\in M} \phi_j. \tag{3.9}$$

Par définition, un arc $j = (k,\ell) \in J'$ avec $k \in M$ et $\ell \notin M$, ne peut pas appartenir au graphe d'écart, et alors $\phi_j = c_j$. De même, pour un arc $j = (k,\ell) \in J'$ avec $k \notin M$ et $\ell \in M$, l'arc opposé (ℓ,k) ne peut pas appartenir au graphe d'écart, et alors $\phi_j = b_j$. Nous venons alors de démontrer que

$$\phi_{(\overline{p},\overline{s})} = \sum_{j\in J', source(j)\in M, but(j)\notin M} c_j - \sum_{j\in J', source(j)\notin M, but(j)\in M} b_j.$$

En écrivant (3.9) pour un autre flot réalisable ψ, nous concluons que

$$\phi_{(\overline{p},\overline{s})} - \psi_{(\overline{p},\overline{s})} = \sum_{j\in J', source(j)\in M, but(j)\notin M} (c_j - \psi_j) - \sum_{j\in J', source(j)\notin M, but(j)\in M} (b_j - \psi_j) \geq 0,$$

d'où l'optimalité du flot ϕ. □

Dans le fichier `don_ch3/fordfulk.bat` on trouve une simulation de l'algorithme de Ford-Fulkerson. L'arc de retour est en rouge, et les différents chemins dans le graphe d'écart (pas affiché) sont dessinés en bleu. Sur chaque arc j on affiche le triplet (b_j, ϕ_j, c_j).

3.4.9. Exercice : Théorème de la coupe
Pour un graphe $G = (L, J)$ avec arc de retour $(\overline{p},\overline{s})$ associons le graphe partiel $G' = (L, J')$ sans arc de retour $(\overline{p},\overline{s})$, c'est-à-dire, $J' = J \setminus \{(\overline{p},\overline{s})\}$. Une coupe dans G' est un ensemble $M \subset L$ avec $\overline{s} \in M$ et $\overline{p} \notin M$, sa capacité étant définie par

$$C(M) = \sum_{j\in J', source(j)\in M, but(j)\notin M} c_j - \sum_{j\in J', source(j)\notin M, but(j)\in M} b_j.$$

Montrer le théorème de la coupe minimale

$$\min\{C(M) : \ M \text{ coupe de } G'\} = \max\{\phi_{(\overline{p},\overline{s})} : \ \phi \text{ flot réalisable dans } G\}$$

(on cherchera une preuve directe en termes de graphes, et éventuellement un lien avec le théorème 2.5.7 de dualité). [9]

3.4.10. Exercice : Ford-Fulkerson peut se planter
Construire un exemple (avec un malheureux choix des chemins dans les graphes d'écart) où l'algorithme de Ford-Fulkerson n'est pas fini, et où en plus la valeur sur l'arc de retour converge, mais pas vers la solution optimale du problème de flot maximum. [10]

3.4.11. Remarques sur la complexité de l'algorithme de Ford-Fulkerson
En prenant le plus court chemin de $\overline{s}$ à $\overline{p}$ en nombre d'arcs dans le graphe d'écart (construit par Tarjan en $\mathcal{O}(n)$ opérations, voir Exercice 3.3.7), on peut montrer que le nombre d'itérations de l'algorithme de Ford–Fulkerson est borné par $\frac{mn}{2}$ [Sak84b, p.107]. De plus, Edmonds et Karp (1972) ont montré [GoMi95, p.197] que le nombre d'opérations arithmétiques est borné par $\mathcal{O}(m^2n)$.

Finalement on donnera sans preuve l'énonce de l'algorithme de Hoffman pour la recherche d'un flot réalisable.

3.4.12. Algorithme de Hoffman

Hypothèse : $G = (L, J)$ *un graphe,* $b, c \in \mathbb{R}^J$ *des bornes.*
Invariant : $\phi \in \mathbb{R}^J$ *un flot.*
Itération : tant qu'il existe un arc j *avec* $\phi_j < b_j$ *ou* $\phi_j > c_j$
si $\phi_j < b_j$ *alors (on considère* j *comme arc de retour)*
construire un cycle élémentaire γ *dans* G *avec* $\vec{\gamma}_j = 1$, $cap_\phi(\gamma) > 0$
STOP si échoué, sinon $\phi \leftarrow \phi + cap_\phi(\gamma)\,\vec{\gamma}$
sinon (ici $\phi_j > c_j$ *et on considère l'opposé de* j *comme arc de retour)*
construire un cycle élémentaire γ *dans* G *avec* $\vec{\gamma}_j = -1$, $cap_\phi(\gamma) > 0$
STOP si échoué, sinon $\phi \leftarrow \phi + cap_\phi(\gamma)\,\vec{\gamma}$
Résultat : un flot réalisable ϕ *où une preuve qu'un tel flot n'existe pas.*

Démonstration. Laissé à titre d'exercice, on s'inspirera des idées de la preuve de l'algorithme de Ford-Fulkerson. □

Dans le fichier `don_ch3/hoffmann.bat` on trouve une simulation de l'algorithme de Hoffman. Les arcs j avec $\phi_j < b_j$ sont tracés en rouge et ceux avec $\phi_j > c_j$ en vert. Le point de départ est un flot aléatoire. Sur chaque arc j on affiche le triplet (b_j, ϕ_j, c_j). A chaque itération, on choisi un arc rouge ou vert, qui ensemble avec le chemin (bleu) dans le graphe d'écart donne lieu à un cycle dans le graphe de départ sur lequel on change le flot, pour se rapprocher des bornes de capacité.

Pour terminer ce chapitre montrons que l'on peut également traiter des flots dynamiques où on tient compte de la durée d'un trajet.

9. Ici on n'a pas besoin de la finitude de Ford-Fulkersson, on supposera tout simplement qu'on a démarré avec une solution optimale

10. A développer, voir Sakarovitch Tome 2, pages 107-108. Je laisse ça à toi, Ana ?

3.4.13. Exemple d'un flot dynamique : la bataille de la Marne

Nous allons étudier le problème de flot dynamique à l'aide d'un exemple tout a fait classique : la bataille de la Marne (ici réduit à un problème de 4 *villes). De trois différentes villes nommées* $1, 2, 3$ *partent des véhicules pour se rendre à une ville unique* 4*. Pour chaque route* (ℓ, k) *(à sens unique) on se donne le nombre maximum de véhicules* $C_{\ell,k}$ *pouvant partir de la ville* ℓ *à* k *par heure, et le temps de trajet* $t_{\ell,k}$*. Pour* $\ell \in \{1, 2, 3\}$ *on se donne aussi le nombre de voitures* s_ℓ *prêtes à partir (ici* $s_1 = 500, s_2 = 350, s_3 = 400$*) à l'instant* 0*, ainsi que le nombre maximal* $C_{\ell,\ell}$ *de voitures pouvant stationner en ville* ℓ*. On cherche à organiser le trafic routier de sorte que, dans un intervalle* $[0, T]$ *donné, un maximum de véhicules puissent arriver en ville* 4*.*

ville départ	*ville arrivé*	*temps de trajet sur la route*	*nb de voitures pouvant simultanément emprunter la route*
1	2	$t_{1,2} = 1$	$C_{1,2} = 250$
1	3	$t_{1,3} = 2$	$C_{1,3} = 220$
2	3	$t_{2,3} = 2$	$C_{2,3} = 300$
2	4	$t_{2,4} = 3$	$C_{2,4} = 190$
3	4	$t_{3,4} = 1$	$C_{3,4} = 270$
1	1	$t_{1,1} = 1$	$C_{1,1} = 400$
2	2	$t_{2,2} = 1$	$C_{2,2} = 600$
3	3	$t_{3,3} = 1$	$C_{3,3} = 800$

A priori, ce problème est de nature continue car les véhicules peuvent potentiellement partir à tout moment. Ici, les temps de trajet sont des multiples d'une unité de temps ($\Delta t = 1$)*, et on conviendra que l'on permet seulement un départ aux instants de temps* $t_k = k\Delta t$*, ici* $t_0 = 0, t_1 = 1, ...$ *Aussi, pour simplifier, le stationnement peut être remplacé par des routes d'une ville* ℓ *vers elle-même (des boucles), avec temps de trajet* $t_{\ell,\ell} = 1$*. Ceci nous permet d'écrire une loi de conservation : notant par* $z_{i,j,t} \in [0, C_{i,j}]$ *le trafic partant de la ville* i *à l'instant* t *et arrivant à la ville* j *à l'instant* $t + t_{i,j}$*, on devrait avoir*

$$\forall\, j, t = t_k : \qquad \sum_{i:(i,j)\ route} z_{i,j,t-t_{i,j}} = \sum_{\ell:(j,\ell)\ route} z_{j,\ell,t},$$

(sauf pour les instants $t = 0$ *où il s'ajoutent des voitures aux villes* $1, 2, 3$ *considérés comme sources, et le sommet* $j = 4$ *qui est considéré comme puits en tout moment* $t = t_k \in [0, T]$*), une loi de conservation qui fait intervenir le temps.*

Une possibilité de se ramener à un problème de flot maximum consiste à considérer le graphe développé *: on considère des copies* (ℓ, t) *du sommet* $\ell \in \{1, 2, 3, 4\}$ *pour les instants* $t = 0, 1, 2, ..., T$*. Pour une route* (i, j) *entre ville* i *et* j *(*$i = j$ *si stationnement), on introduit les arcs*

$$\text{pour } t = 0, 1, ..., T - t_{i,j} : \qquad \text{du sommet } (i, t) \text{ au sommet } (j, t + t_{i,j})$$

avec borne inf $= 0$ *et borne sup* $= C_{i,j}$*. Il s'agit alors de transporter un maximum depuis les sources* $(\ell, 0)$*,* $\ell = 1, 2, 3$ *(dans la limite* s_ℓ *pour la ville* ℓ*) vers les puits* $(4, t)$*,* $t = 0, 1, ..., T$*. En appliquant le principe de 3.4.3(a), on se ramène à un problème classique de flot maximum (statique). Notons que la limitation* s_ℓ *pour la source* ℓ *est utilisée comme borne supérieure pour l'arc de la super-source vers la source* $(\ell, 0)$*. On trouve dans le fichier* `don_ch3/marne3.bat` *une simulation du problème de la bataille de la Marne pour* $T = 3$ *(seulement* 860 *sur* 1250 *voitures arrivent), et dans* `don_ch3/marne4.bat` *pour* $T = 4$ *(où toutes les voitures arrivent).*

3.4.14. Exercice (Théorème des valeurs entières) :
Si les capacités des arcs sont des nombres entiers et s'il existe un flot réalisable $\phi^{(0)}$ à composantes entières, montrer qu'il existe un flot maximum de $\overline{s}$ à $\overline{p}$ avec composantes toutes entières. De plus, montrer que Ford-Fulkerson avec flot initial $\phi^{(0)}$ nécessite au plus $v - \phi^{(0)}_{(\overline{p},\overline{s})}$ itérations, avec v la valeur optimale du problème de flot maximum.

3.4.15. Exercice
Pour se rendre en voiture en vacances à Montpellier début août, les Parisiens peuvent emprunter des autoroutes (ici en sens unique) énumérées dans le tableau ci-dessous : un chiffre d dans ce tableau correspond à une route existante entre deux villes, avec d le nombre de voitures (en millier) pouvant emprunter cette route par heure. On se propose d'organiser le trafic de sorte qu'un maximum de personnes puissent partir en vacances, en supposant que les pauses et les durées de trajets sont négligées.

ville départ\ville d'arrivé	*2=Or.*	*3=C.F.*	*4=Bo.*	*5=Ly.*	*6=Mo.*
1=Paris	*4*			*3*	
2=Orléans		*2*	*2*		
3=Clermont Ferrand				*3*	*1*
4=Bordeaux					*1*
5=Lyon					*7*
6=Montpellier					

(a) Modéliser ce problème comme un problème de flot maximum sur un graphe avec capacités à préciser.
(b) Résoudre par l'algorithme de Ford-Fulkerson, partant d'un flot zéro (on donnera les graphes d'écart des flots intermédiaires).
(c) Expliquer comment mettre à jour d'une manière efficace la solution trouvée en partie (b) si à cause d'un accident on doit fermer l'autoroute entre Paris et Lyon.

3.5 Simplex en termes de graphes

L'objectif de ce chapitre est de donner une interprétation et une simplification importante pour quelques programmes linéaires liés aux flôts, aux réseaux de conduite (tuyaux, câbles éléctriques) et problèmes de transport/transbordement (voir Exemple 1.2.4 et Exemple 3.4.4) qui se modélisent par

$$(PL): \quad \max\{fx : Ax = a, b \leq x \leq c\},$$

avec a, b, c à coefficients entiers (inclus les contraintes triviales $c_j = +\infty$, $b_j = -\infty$), et $A = A_L^J$ la matrice d'incidence d'un graphe $G = (L, J)$ connexe valué par $f = f^J$.

On commencera par démontrer dans 3.5.1 que la matrice d'incidence d'un graphe orienté est toujours unimodulaire, ce qui expliquera que, pour a, b, c à coefficients entiers, tout point de base est à coefficients entiers. Ensuite on donnera un lien entre bases et sous-graphes connexes sans cycles dans 3.5.7, une interprétation des directions privilégiées (cycles) en termes de graphes dans **??**, l'interprétation et le calcul du vecteur coûts marginaux dans 3.5.9, et l'énoncé de Simplex en termes de graphes dans 3.5.10. L'idée conductrice dans cette version de Simplex est que l'on n'aura pas besoin de connaître l'inverse de la matrice A^I, car les systèmes (2.11) peuvent être résolus en complexité $\mathcal{O}(m)$ par des algorithmes d'accessibilité du chapitre 3.2. L'étude de deux exemples terminera ce chapitre.

3.5.1. Théorème :
Une matrice d'incidence A_L^J d'un graphe orienté (L, J) est totalement unimodulaire : pour tout $K \subset L$ et $I \subset J$ avec $|I| = |k|$ nous avons dét$(A_K^I) \in \{0, \pm 1\}$. Plus précisément, toute sous-matrice carrée inversible est triangulaire à une permutation près.

Démonstration. Nous montrons par récurrence sur ℓ qu'une sous-matrice inversible d'ordre ℓ d'une matrice d'incidence est triangulaire à une permutation près. La propriété est triviale pour $\ell = 1$. Soit maintenant A_K^I une sous-matrice inversible d'ordre $\ell \geq 2$. Alors chaque colonne de A_K^I comporte au moins un élément non nul.
Montrons par l'absurde qu'il existe au moins une colonne d'indice $i \in I$ de sorte que A_K^i comporte exactement un élément A_k^i non nul. Sinon, comme A_K^I est une sous-matrice d'une matrice d'incidence d'un graphe orienté, chaque colonne comportera exactement 2 éléments non nuls, l'un de valeur 1 et l'autre de valeur -1, ce qui implique que $(1, ..., 1)A_K^I = 0$, en contradiction avec l'hypothèse d'inversibilité.
Par conséquent, la matrice permutée

$$A_{(k,K-k)}^{(i,I-i)} = \begin{bmatrix} \pm 1 & * \\ 0 & A_{K-k}^{I-i} \end{bmatrix}$$

est triangulaire par blocs, et l'hypothèse de récurrence, appliquée au bloc A_{K-k}^{I-i}, nous permet d'affirmer que A_K^I est triangulaire à une permutation près.
Finalement, l'unimodularité découle de l'observation qu'une sous-matrice carrée d'une matrice d'incidence d'un graphe orienté est soit non inversible (son déterminant vaut 0), soit (à une permutation près) une matrice triangulaire supérieure avec ± 1 sur la diagonale, et admet donc un déterminant de valeur ± 1. □

3.5.2. Lemme :
Supposons que $a \in \mathbb{Z}^{|L|}$, et que $Ax = a$ admette une solution. Alors $Ax = a$ admet une solution dans $\mathbb{Z}^{|J|}$. Si de plus $a = A^{\widehat{j}}$ pour un arc $\widehat{j}$ (pas forcément dans J) alors on peut choisir $x \in \{0, \pm 1\}^{|J|}$.

Démonstration. Soit A_K^I une sous-matrice carrée inversible de taille maximale de A, alors par hypothèse

$$rang\ (A) = rang\ (A, a) = rang\ (A_K^I) = rang\ (A_K^I, a_K).$$

Alors le point $y \in \mathbb{R}^{|J|}$ avec $y_{J \setminus I} = 0$, $A_K^I y_I = a_K$ est solution de $Ax = a$. D'après la règle de Cramer,

$$\forall j \in I : \qquad y_j = \pm \frac{det\ ((A_K^{I-j}, a_K))}{det\ (A_K^I)}.$$

Dans cette dernière formule, le dénominateur est un élément de $\{\pm 1\}$ d'après le théorème 3.5.1, et le numérateur peut être développé par rapport à la dernière colonne, ce qui par le théorème 3.5.1 donnera une somme des composantes de a_K, où chaque terme est pondéré par 0, $+1$ ou -1. Par hypothèse sur a, nous pouvons conclure que $x_j \in \mathbb{Z}$, ce qu'il fallait démontrer. Si maintenant de plus a est le vecteur $A^{\hat{j}}$ d'un arc $\hat{j}$ alors (A_K^{I-j}, a_K) est une sous-matrice carrée de la matrice d'incidence du graphe $(L, I - i + \hat{j})$, et donc $x_j \in \{0, \pm 1\}$ d'après le théorème 3.5.1. □

Une conséquence immédiate des deux résultats précédents est donnée par

3.5.3. Corollaire :
Sous les hypothèses décrites en préambule du chapitre 3.5, et A une sous-matrice de rang maximal d'une matrice d'incidence d'un graphe, tout point de base $x(I, B_-, B_+)$ admet des coefficients entiers.

Comme on sait que l'optimum de (PL) est atteint en un point de base (si fini), on déduit du corollaire précédent que l'on trouve par SIMPLEX des solutions optimales qui sont de plus à coefficients entiers (on transporte seulement des quantités entières). De plus, toute solution intermédiaire sera à coefficients entiers, ce qui nous permettra de montrer des résultats de complexité intéressants.

Dans le lemme et l'exercice suivant on regroupe un certain nombre de liens entre le vecteur d'une chaîne élémentaire (voir la définition 3.4.7) et la matrice d'incidence d'un graphe orienté.

3.5.4. Lemme :
(a) *Soit γ une chaîne élémentaire, avec $\ell_0 = source(\gamma) \neq \ell_\kappa = but(\gamma)$, alors $A \cdot \vec{\gamma} = A^{\widehat{j}}$, avec $\widehat{j} = (\ell_0, \ell_\kappa)$.*
(b) *Soit γ un cycle élémentaire, alors $A \cdot \vec{\gamma} = 0$.*
(c) *Soit $\widehat{j}$ un arc (pas forcément dans J), avec $A^{\widehat{j}} \in span\{A^j : j \in J\}$. Alors on trouve une chaîne élémentaire γ dans (L, J) avec les mêmes extrémités que $\widehat{j}$.*

Démonstration. **(a),(b)** Soit $\gamma = [\ell_0, ..., \ell_\kappa]$, la liste des sommets rencontrés, et $\gamma = (j_1, ..., j_\kappa)$, la liste des arcs utilisés. En utilisant (3.7) et la définition d'un vecteur d'une chaîne, nous obtenons pour un sommet $\ell \in L$

$$(A\vec{\gamma})_\ell = \sum_{j \in J, source(j)=\ell} \vec{\gamma}_j - \sum_{j \in J, but(j)=\ell} \vec{\gamma}_j = \sum_{\alpha=1,...,\kappa, source(j_\alpha)=\ell} \vec{\gamma}_{j_\alpha} - \sum_{\alpha=1,...,\kappa, but(j_\alpha)=\ell} \vec{\gamma}_{j_\alpha}$$
$$= \sum_{\alpha=1,...,\kappa, \ell_{\alpha-1}=\ell} 1 - \sum_{\alpha=1,...,\kappa, \ell_\alpha=\ell} 1$$

ce qui vaut toujours 0 sauf si $\ell_0 \neq \ell_\kappa$, et $\ell = \ell_0$ (ici $(A\vec{\gamma})_\ell = 1$) ou $\ell = \ell_\kappa$ (ici $(A\vec{\gamma})_\ell = -1$).
(c) En enlevant un certain nombre d'arcs de J, on peut supposer sans perte de généralité que $\text{rang}(A^J) = |J|$. Soit $Ax = A^{\widehat{j}}$, alors on peut encore supposer sans perte de généralité que $x_j \neq 0$ pour $j \in J$, et alors $x_j = \pm 1$ d'après le lemme 3.5.2.
Montrons par récurrence sur $|J|$ que x est le vecteur d'une chaîne élémentaire avec les mêmes extrémités que $\widehat{j} =: (\ell, k)$. Le cas $|J| = 1$ est trivial. Comme $(Ax)_\ell = 1$, et pour tout $j \in J$ nous avons la relation $(A^j x_j)_\ell \in \{0, \pm 1\}$, on trouve un arc dans j avec $(A^j x_j)_\ell = 1$, plus précisément $j = (\ell, \ell')$ si $x_j = 1$, et $j = (\ell', \ell)$ si $x_j = -1$. Dans les deux cas, $A^{J-j} x_{J-j} = A^{\widehat{j}} - x_j A^j = A^{(\ell', k)}$, et par hypothèse de récurrence nous savons que x_{J-j} est le vecteur d'une chaîne élémentaire γ' avec source ℓ' et but k. Nous concluons que $\gamma = (j, \gamma')$ est une chaîne avec extrémités k et ℓ. Par construction, $k \neq \ell$ et $\ell \neq \ell'$. Si ℓ était un sommet interne de γ', alors γ comporterait un cycle passant par ℓ, en contradiction avec la partie (b) et l'hypothèse *rang* $(A^J) = |J|$. Donc γ est bien une chaîne élémentaire avec extrémités ℓ et k. □

3.5.5. Exercice
Montrer qu'une chaîne élémentaire peut être reconstruite à partir de son vecteur.

Après avoir interprété des solutions de $Ax = a$ en termes de graphes, passons maintenant à une interprétation des bases. Rappelons que $(1, ..., 1)\, A_L^J = 0$ pour tout graphe orienté (L, J). Il faudra donc enlever au moins une ligne de A_ℓ^J pour trouver des bases.

3.5.6. Définition : Arbre
Un graphe partiel (L, I) de (L, J) est dit arbre si il est connexe et sans cycles.

3.5.7. Théorème :
(a) *(L, J) contient un cycle si et seulement si rang$(A_L^J) < |J|$.*
(b) *(L, J) est connexe si et seulement si rang$(A_L^J) = |L| - 1$ si et seulement si pour tout $\ell \in L$: rang$(A_{L-\ell}^J) = |L - \ell|$.*
(c) *Le graphe partiel (L, I) est un arbre si et seulement si I est une base pour $A_{L-\ell}^J$ pour tout $\ell \in L$.*

Démonstration. **(a)** Si (L, J) contient un cycle alors, d'après le lemme 3.1.6 de König, aussi un cycle élémentaire γ, avec $A_L^J \vec{\gamma} = 0$ d'après 3.5.4(b) et $\vec{\gamma} \neq 0$, d'où rang$(A_L^J) < |J|$. Réciproquement, si rang$(A_L^J) < |J|$ alors on trouve un $j \in J$ de sorte que avec $J' = J - j$ nous avons rang$(A^{J'}) =$ rang$(A^{J'}, A^j)$. D'après 3.5.4(c), on trouve une chaîne dans (L, J') ayant les mêmes extremités que j, et en ajoutant j on obtient un cycle.
(b) Soit $J^* := \{(\ell, k) : \ell, k \in L, \ell \neq k\}$. Le graphe (L, J) est connexe ssi pour tout $(\ell, k) \in J^*$ il existe une chaîne ayant les extrémités ℓ et k. D'après 3.5.4(a),(c), ceci est équivalent au fait que $A^{j^*} \in \text{span}\{A^j : j \in J\}$ pour tout $j^* \in J^*$, où encore rang$(A_L^J) =$ rang$(A_L^{J^*})$ (car $J \subset J^*$). Mais $A_L^{J^*}$ admet le rang $|L| - 1$, car $(1, ..., 1) \cdot A_L^{J^*} = 0$, et la sous-matrice $A_{L-\ell}^{J^*}$ contient l'identité d'ordre $|L| - 1$ pour tout $\ell \in L$ (choisir les arcs (k, ℓ), $k \in L - \ell$).
(c) Voir 3.5.7(a),(b) pour $I = J$. □

Rappelons que l'on suppose $G = (L, J)$ connexe. Le théorème 3.5.7(b) implique que le problème (PL) donné en préambule du chapitre 3.5 est soit impossible (si $\sum a_j \neq 0$), soit, pour tout $\ell \in L$, on peut supprimer la ℓième contrainte (redondante), pour ensuite obtenir un problème vérifiant la condition habituelle de regularité *rang* $(A_{L-\ell}^J) = |L - \ell|$ de SIMPLEX, et une base d'après 3.5.7(c).

Avant de donner SIMPLEX en termes de graphes, terminons par une interprétation des directions privilégiées et des vecteurs coûts réduits/coûts marginaux en termes de graphes.

3.5.8. Exercice :
En ajoutant un arc s à un arbre (L, I) on créera un et un seul cycle γ élémentaire (unique à l'orientation près). Si $\vec{\gamma}_s = \epsilon(I)_s$ alors $\vec{\gamma}$ est la direction privilégiée $v(I)^s$. Finalement, si $b \leq x \leq c$ alors nous obtenons pour la capacité (voir 3.4.7) de γ la relation

$$cap_x(\gamma) = \sup\{\theta \geq 0 : b \leq x + \theta\vec{\gamma} \leq c\}.$$

3.5.9. Définition et Lemme :
Soit (L, I) un arbre. On appelle potentiel tout vecteur $\lambda = \lambda^L$ de sorte que $\lambda \cdot A^I = f^I$.

Un potentiel est unique après avoir fixé une composante $\lambda^{\widehat{\ell}} = 0$. De plus, pour tout potentiel, $d(I) = f - \lambda \cdot A$.

Démonstration. Si $\lambda^{\widehat{\ell}} = 0$ alors $\lambda \cdot A^I = \lambda^{L-\widehat{\ell}} \cdot A^I_{L-\widehat{\ell}} = f^I$, où $A^I_{L-\widehat{\ell}}$ est inversible d'après 3.5.7(c). Comme $(1, ..., 1) \cdot A^J_L = 0$, on peut conclure que l'ensemble de potentiels est donné par

$$\lambda^{L-\widehat{\ell}} = \lambda^{\widehat{\ell}} \cdot (1, ..., 1) + f^I [A^I_{L-\widehat{\ell}}]^{-1}, \quad \lambda^{\widehat{\ell}} \in \mathbb{R}.$$

en particulier $f - \lambda \cdot A$ coïncide avec le vecteur coûts réduits du problème obtenu après suppression de la contrainte d'indice $\widehat{\ell}$. □

On peut interpréter un $\lambda = \lambda^L$ comme une valuation de L (on affecte une valeur à chaque sommet, par contre, une composante $d(I)^j$ du vecteur coûts réduits est affectée à un arc j).

Comment mettre en œuvre le calcul d'un potentiel sans la connaissance de l'inverse d'une sous-matrice de A ? Notons que

$$\lambda A^I = f^I \quad ssi \quad \text{pour tout } (\ell, k) \in I : \ \lambda^\ell - \lambda^k = f^{(\ell,k)}. \tag{3.10}$$

D'après le théorème 3.5.1 et le théorème 3.5.7(c), le système obtenu en posant $\lambda^{\widehat{\ell}} = 0$ est triangulaire à une permutation près, et en plus très creux, ce qui semble être une piste prometteuse pour une résolution efficace. Ici on préfère de raisonner directement en termes de graphes : d'après le le théorème 3.5.7(c), le graphe (L, I) est connexe et sans cycle, ce qui signifie que, pour tout $\ell \in L$ il y a une et une seule chaîne γ_ℓ avec source $\widehat{\ell}$ et but ℓ. Par conséquent, en ajoutant à la variante 3.2.4(c) de l'algorithme 3.2.1 de Tarjan l'instruction $\lambda^y = \lambda^x + f^{(x,y)}$ si $y \in \Gamma^+(x)$ et $\lambda^y = \lambda^x - f^{(y,x)}$ si $y \in \Gamma^-(x)$, on calcule le vecteur potentiel en complexité $\mathcal{O}(|I|) = \mathcal{O}(m)$.

D'ailleurs, si on garde en mémoire les chaînes d'accessibilité γ_ℓ par un tableau `père` comme expliqué dans 3.2.4(a) alors le calcul la direction privilégiée $v(I)^s$ (voir **??**) devient simple : si $s = (\ell, k)$, en concatenant les chaînes γ_ℓ et γ_k et en ajoutant l'arc s on obtient l'unique cycle dans $(L, I + s)$.

Finalement, connaissant le potentiel, nous pouvons calculer les composantes hors base du vecteur coûts réduits par

$$\text{pour tout } (\ell, k) \in J \setminus I : \ d(I)^{(\ell,k)} = f^{(\ell,k)} - \lambda^\ell + \lambda^k \tag{3.11}$$

en complexité $\mathcal{O}(n)$ d'après le lemme 3.5.9.

3.5.10. Algorithme : SIMPLEX en termes de graphes

Invariants : $I \subset J$ *t.q.* (L, I) *est arbre, partition* B_+, B_- *de* $J \setminus I$, $x(I, B_-, B_+)$ *point de base réalisable, i.e.,* $Ax(I) = a, x(I, B_-, B_+)_{B_-} = b_{B_-}, x(I, B_-, B_+)_{B_+} = c_{B_+}, b_I \le x(I, B_-, B_+)_I \le c_I$.

Itération : *déterminer* potentiel (coûts marginaux) $\lambda = \lambda^L$ *par (3.10) (Tarjan modifié)*
déterminer coûts réduits $d(I)^{B_+ \cup B_-}$ *par (3.11)*
arrêt si condition suffisante d'optimalité, sinon
chercher ($s \in B_-$ *avec* $d(I)^s > 0$) *ou* ($s \in B_+$ *avec* $d(I)^s < 0$)
chercher l'unique chaîne dans (L, I) *ayant les mêmes extrémités que* s*, ajouter* s *pour former l'unique cycle dans* $(L, I + s)$ *avec orientation* $\vec{\gamma}_s = 1$ *si* $s \in B_-$ *et* $\vec{\gamma}_s = -1$ *si* $s \in B_+$
calculer $\theta = cap_{x(I,B_-,B_+)}(\gamma)$ *par (3.8), arrêt si* $\theta = +\infty$*, sinon*

soit $r \in I + s$ un indice réalisant le minimum dans (3.8)
Mise à jour I, B_-, B_+ comme dans le théorème 2.3.7, $x(I, B_-, B_+) \leftarrow x(I, B_-, B_+) + \theta \cdot \vec{\gamma}$.
Complexité (d'une itération) : $\mathcal{O}(n)$ si on calcule le vecteur $d(I)$ entier, $\mathcal{O}(m)$ sinon.

Pour générer une base de départ, nous avons proposé au théorème 2.4.9 un calcul en deux phases. Notons que l'on peut également traduire la phase 1 en termes de graphes : ajouter une matrice *diag* $(\pm 1, ..., \pm 1)$ à la matrice $A^J_{L-\hat{\ell}}$ s'interprète comme l'ajout de $m-1$ arcs "artificiels", avec une extrémité donnée par $\hat{\ell}$, et l'autre par un élément quelconque dans $L-\hat{\ell}$, et l'orientation choisie en fonction du signe de a_ℓ. En première phase on choisit alors un objectif pour d'abord éliminer ces arcs "artificiels", ce qui nous permet d'obtenir à la fin une base réalisable de départ pour la résolution de (PL) en phase 2. On laissera au lecteur le soin de spécifier les détails.

3.5.11. Théorème : Complexité
Supposons que $a_\ell, f^j \in \mathbb{Z}$, et $b_j \in \mathbb{Z} \cup \{-\infty\}$, $c_j \in \mathbb{Z} \cup \{+\infty\}$ pour tout $\ell \in L$ et $j \in J$. Si on dispose d'une base réalisable de départ I_0 alors le nombre d'itérations **non dégénérées** *de SIMPLEX peut être borné a priori par*

$$||f^T||_1 \max\Big\{\sum_{\ell\in L} |a_\ell|, \sum_{j\in J, b_j\neq-\infty} |b_j|, \sum_{j\in J, c_j\neq+\infty} |c_j|\Big\} - f\,x(I_0).$$

Si de plus $b_j, c_j \in \mathbb{Z}$ pour tout $j \in J$ alors on obtient la borne $||f^T||_1\,||c-b||_\infty$.

Démonstration. Rappelons que l'augmentation de l'objectif dans une itération est donné par $\theta \cdot |d(I)^s| = \theta \cdot f \cdot \vec{\gamma}$. D'après le Corollaire 3.5.3, les quantités $x(I)_j$ sont dans $\mathbb{Z}$, et alors θ est un entier ≥ 0. De plus, les composantes de $\vec{\gamma}$ ainsi que de f sont entières, et alors $f \cdot \vec{\gamma}$ est un entier coïncidant avec $|d(I)^s| > 0$. Par conséquent, dans chaque itération non dégénérée on augmente l'objectif par au moins 1, le plus petit entier positif. Notant par I la base finale (optimale ou celui de détection d'un cycle absorbant), il en résulte que le nombre N de changements de base non dégénérées est borné par $N \leq fx(I) - f\,x(I_0)$.
Pour le premier cas evoqué dans l'enoncé il reste à vérifier la borne pour $|fx(I)|$. Effectivement, d'après l'inégalité de Hölder nous avons pour un $\ell \in L$

$$\begin{aligned}
|fx(I)| &\leq |f^I\,(A^I_{L-\ell})^{-1}\,a_{L-\ell}| + |f^{B_-}\,(A^I_{L-\ell})^{-1}A^{B_-}_{L-\ell}b_{B_-}| + |f^{B_+}\,(A^I_{L-\ell})^{-1}A^{B_+}_{L-\ell}c_{B_+}| \\
&\leq ||f^T||_1 \max\Big\{||(A^I_{L-\ell})^{-1}\cdot a_{L-\ell}||_\infty, ||(A^I_{L-\ell})^{-1}A^{B_-}_{L-\ell}b_{B_-}||_\infty, ||(A^I_{L-\ell})^{-1}A^{B_+}_{L-\ell}c_{B_+}||_\infty\Big\} \\
&\leq ||f^T||_1 \max_j ||u_j||_\infty \max\{||a||_1, ||b_{B_-}||_1, ||c_{B_+}||_1\},
\end{aligned}$$

où u_j^T est une ligne soit de $(A^I_{L-\ell})^{-1}$, soit de $(A^I_{L-\ell})^{-1}A^{B_-}_{L-\ell}$ ou de $(A^I_{L-\ell})^{-1}A^{B_+}_{L-\ell}$. D'après la deuxième partie du lemme 3.5.2, une telle ligne u_j ne contient que des éléments 0 ou ± 1, d'où notre estimation.
Concernant le deuxième cas, notons que

$$fx(I) - fx(I_0) = |f[x(I) - x(I_0)]| \leq ||f||_1 \cdot ||x(I) - x(I_0)||_\infty,$$

où pour tout j nous avons $x(I)_j - x(I_0)_j \leq c_j - b_j$. □

Malheureusement, due à la structure particulière du problème (PL), il est assez probable de rencontrer des dégénérescences, et donc le Théorème 3.5.11 ne permet pas de borner la complexité de l'algorithme 3.5.10 dans le cas général (il existe des sous-classes de problèmes comme par exemple des problèmes d'affectation où on peut montrer que toute base réalisable est dégénérée). Effectivement, on peut donner un exemple avec une base où tout choix admissible d'un cycle (associé à un $s \in B^+ \cup B^-$) nous donne $\theta = 0$, voir en bas. Effectivement, pour le problème de flot maximum, l'algorithme 3.4.8 de Ford-Fulkerson basé sur un principe similaire permet d'examiner une classe plus importante de cycles, et ici on peut affirmer que l'on trouve au moins un cycle élémentaire de pente avec $\theta > 0$ (à condition de ne pas encore avoir trouvé une solution optimale).

Pour terminer ce chapitre, discutons quelques applications pour Simplex en termes de graphes.

3.5.12. Exemple de recherche d'un chemin de valeur maximale
Étant donné un graphe $G = (L, J)$ de valuation $f = f^J$, $L = \{1, 2, ..., m\}$, on cherche un chemin de valeur maximale allant de 1 à m (par exemple). Montrons que ce problème est modélisé par le programme linéaire

$$(PL) \quad \max\{fx : Ax = a = (1, 0, ..., 0, -1)^T = A^{(1,m)}, x \geq b = 0\}$$

et $c = (+\infty, ..., +\infty)^T$. Effectivement, le vecteur d'un chemin élémentaire γ de 1 à m vérifie par définition $\vec{\gamma} \geq 0$, et $A\vec{\gamma} = A^{(1,m)}$ par le lemme 3.5.4(a). Réciproquement, on sait d'après le chapitre 2 que (PL) n'est pas borné si et seulement si Simplex s'arrête avec une direction privilégiée de pente, qui d'après l'exercice **??** *et la définition de la capacité représente un cycle élémentaire γ absorbant avec $\vec{\gamma} \geq 0$ (et donc γ est un circuit élémentaire absorbant). Dans le cas contraire, Simplex s'arrête avec une solution optimale $x = x(I) \geq 0$ ayant une représentation comme point de base réalisable, et donc $x \in \mathbb{Z}^n$ d'après le corollaire 3.5.3. D'autre part, l'arbre (L, I) comporte une chaîne élémentaire γ de 1 à m, et $A^I\vec{\gamma} = A^{(1,m)}$ par le lemme 3.5.4(a). Comme* rang $(A^I) = |I|$, *nous déduisons que $x = \vec{\gamma}$, et donc γ est un chemin élémentaire de 1 à m de valeur maximale.*

Pour cette application, Simplex en termes de graphes doit être comparé avec l'algorithme 3.3.12 de Roy–Warshal qui (après changement de signe pour la valuation) trouve aussi des circuits absorbants, et (dans le cas échéant) un chemin de valeur maximale de 1 à m. Comme il est assez probable de rencontrer des dégénérescences pour Simplex et comme il faudra en plus rechercher une base réalisable de départ, l'application de l'algorithme 3.3.12 de Roy–Warshal est généralement préférable.

Nous concluons qu'il n'existe pas forcément un chemin de valeur maximale de 1 à m, mais par contre toujours un chemin simple de valeur maximale de 1 à m (sans aucune hypothèse sur la valuation), car il existe un nombre fini de tels chemins.

3.5.13. Exercice
Montrer qu'un chemin simple de valeur maximale de 1 à m peut être trouvé en résolvant le problème modifié aux variables doublement bornées

$$(PL') \quad \max\{fx : Ax = a := A^{(1,m)}, x \geq 0 =: b, x \leq (1, 1, ..., 1)^T =: c\}.$$

On introduira d'abord la notion d'un vecteur d'une chaîne simple, et on généralisera le lemme 3.5.4 pour des telles chaînes.

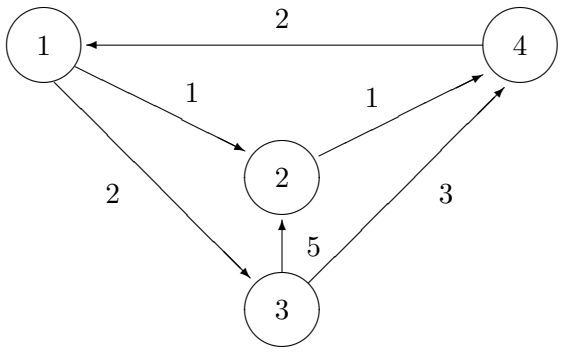


FIGURE 3.4 – LE GRAPHE POUR L'EXEMPLE 3.5.15 D'UN FLOT DE VALEUR MAXIMALE. ON NOTE SUR CHAQUE ARC j SA VALEUR f^j, ET ON CONSIDÈRE LES BORNES $b_j = 0$, $c_j = 4$.

3.5.14. Exercice
Ecrire le problème 3.4.2 de flot maximum pour le graphe $G = (L, J)$ avec arc de retour $\hat{j} = (\overline{p}, \overline{s})$ comme un programme linéaire aux variables doublement bornées. Appliquons Simplex en termes de graphes à ce problème : Montrer que l'arc de retour $\hat{j}$ appartient à I pour toute base réalisable (I, B_-, B_+). En déduire qu'un potentiel λ avec $\lambda^{\overline{s}} = 0$ admet des composantes dans $\{0, 1\}$, et que, plus précisément, le graphe $(L, I - \hat{j})$ admet les deux composantes connexes L_0 et L_1, avec $L_p := \{\ell \in L : \lambda^\ell = p\}$. Pour la base finale de Simplex, déduire un lien avec le théorème 3.4.9 de la coupe.

3.5.15. Exemple de recherche d'un flot de valeur maximale
Pour le graphe donné en Figure 3.4, nous allons résoudre le problème de flot de valeur maximale

$$(PL) \quad \max\{fx : Ax = a := 0, x \geq b := 0, x \leq c := (4, 4, ..., 4)^T\},$$

avec pour chaque arc un flot dans l'intervalle $[0, 4]$. Notons que le flot $x = 0$ est réalisable, et chaque arbre (L, I) nous donnera un point de base réalisable $x(I, B_-, B_+) = 0$ si $(I, B_-, B_+) = (I, J \setminus I, \emptyset)$. Dans le présent exemple, on décide de faire entrer en base I les indices j avec f^j maximum, car ceci devrait faciliter l'augmentation des composantes correspondants du point de base. Notre base initiale sera alors

$$(I, B_-, B_+) = (\{(3, 2), (3, 4), (4, 1)\}, \{(1, 2), (1, 3), (2, 4)\}, \emptyset)$$

ce qui donne effectivement une partition de J, et un graphe (L, I) connexe et sans cycles, c'est-à-dire, un arbre.

Les différents itérations de Simplex en termes de graphes sont affichés dans la figure 3.5, où on note sur le sommet ℓ le potentiel λ^ℓ, et sur l'arc j le couple $I|x(I)_j$ si j est en base, ou le couple $B_-|d(I)^j$ ou $B_+|d(I)^j$.

Expliquons la première itération en détail : Connaissant I, on note $x(I)_j$ pour $j \in I$ sur le graphe. Ensuite on calcule le potentiel suivant (3.10)

$$\lambda^0 = 0, \quad \lambda^4 = f^{(4,1)} + \lambda^1 = 2, \quad \lambda^3 = f^{(3,4)} + \lambda^4 = 5, \quad \lambda^2 = -f^{(3,2)} + \lambda^3 = 0,$$

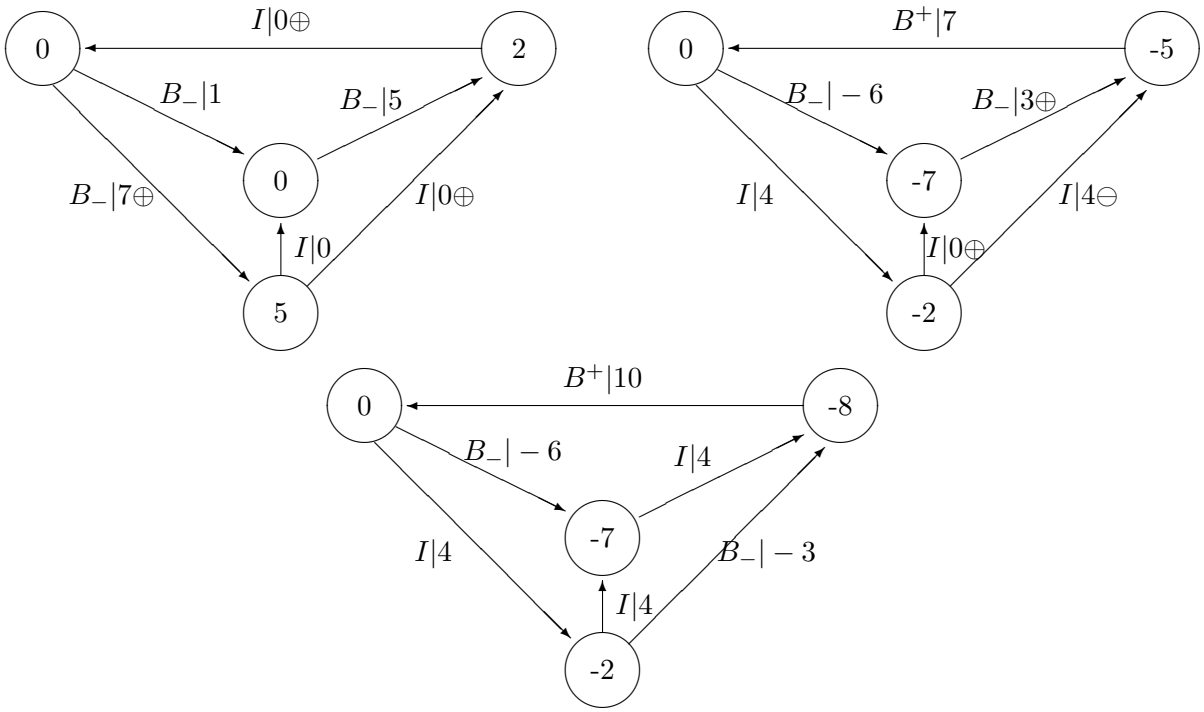


FIGURE 3.5 – LES ITÉRATIONS SIMPLEX POUR L'EXEMPLE 3.5.15 D'UN FLOT DE VALEUR MAXIMALE, DU GAUCHE À DROITE. ON NOTE SUR CHAQUE ARC j LE COUPLE $(I|x(I)_j)$ OU $(B_-|d(I)^j)$ OU $(B_+|d(I)^j)$ SUIVANT L'APPARTENANCE DE j À UN DES ENSEMBLES DE LA PARTITION (I, B_-, B_+) DE J. SUR CHAQUE SOMMET ℓ ON NOTE LE POTENTIEL λ^ℓ, AVEC LA NORMALISATION $\lambda^1 = 0$. PAR LES SIGNES $\oplus, \ominus$ ON MARQUE LE CYCLE DANS $(L, I + s)$, AVEC LE SIGNE $\oplus$ SI L'ORIENTATION DE L'ARC EST RESPECTÉE, ET $\ominus$ SINON.

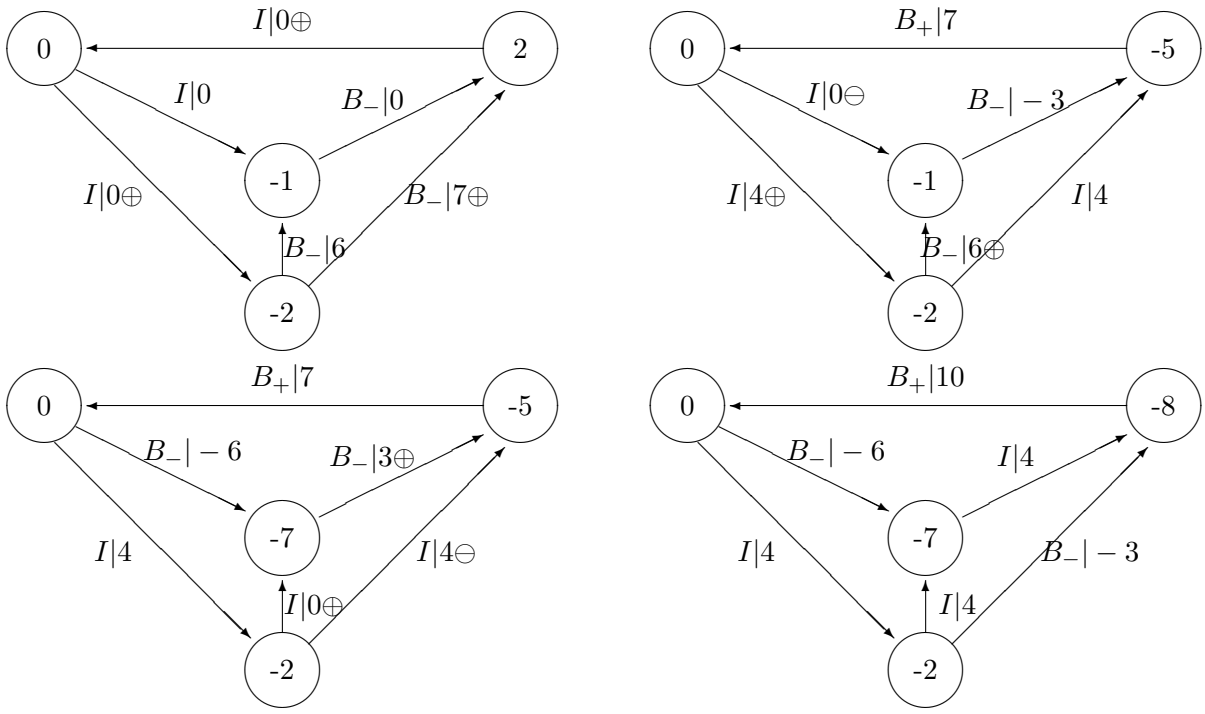


FIGURE 3.6 – LES ITÉRATIONS SIMPLEX POUR L'EXEMPLE 3.5.15 D'UN FLOT DE VALEUR MAXIMALE, DU GAUCHE À DROITE. ON NOTE SUR CHAQUE ARC j LE COUPLE $(I|x(I)_j)$ OU $(B_-|d(I)^j)$ OU $(B_+|d(I)^j)$ SUIVANT L'APPARTENANCE DE j À UN DES ENSEMBLES DE LA PARTITION (I, B_-, B_+) DE J. SUR CHAQUE SOMMET ℓ ON NOTE LE POTENTIEL λ^ℓ, AVEC LA NORMALISATION $\lambda^1 = 0$. PAR LES SIGNES $\oplus, \ominus$ ON MARQUE LE CYCLE DANS $(L, I + s)$, AVEC LE SIGNE $\oplus$ SI L'ORIENTATION DE L'ARC EST RESPECTÉE, ET $\ominus$ SINON.

suivi par les coûts réduits suivant (3.11)

$d^{(1,2)}(I) = f^{(1,2)} - \lambda^1 + \lambda^2 = 1$, $d^{(1,3)}(I) = f^{(1,3)} - \lambda^1 + \lambda^3 = 7$, $d^{(2,4)}(I) = f^{(2,4)} - \lambda^2 + \lambda^4 = 3$.

Toutes ces composantes ayant le faux signe, le pricing mvp nous dit de choisir $s = (1,3)$, et effectivement $(L, I + s)$ contient le cycle $\gamma = [3,4,1,3]$. Le calcul de $\theta = cap_{x(I,B_-,B_+)}(\gamma)$ par (3.8) donne $\theta = 4$, et la borne sup sera atteinte après adaptation du flot par $\theta\vec{\gamma}$ par exemple par l'arc $r = (4,1)$. Le théorème 2.3.7 nous dit de faire entrer s dans I et faire entrer r dans B_+, donnant lieu au dessin suivant. Dans l'itération suivante nous trouvons le choix unique $s = (2,4)$ (qui entre dans I), $\theta = 4$ et par exemple $r = (3,4)$ (qui entre en B_-). Finalement, à la troisième itération nous obtenons la condition suffisante d'optimalité (avec inégalités strictes, et donc une solution optimale unique donnée par le point de base correspondant). On remarque qu'il y a plusieurs composantes en bases pour les points de base ou les bornes sont atteintes (des dégénérescences), mais $\theta > 0$ à chaque itération, un cas chanceux.

Dans la figure 3.6, nous donnons les détails si on part d'une base différente

$$(I, B_-, B_+) = (\{(1,2),(1,3),(4,1)\}, \{(2,4),(3,2),(3,4)\}, \emptyset).$$

En première itération $s = (3,4)$ (qui entre dans I) par pricing mvp, $\theta = 4$, et par exemple $r = (4,1)$ (qui entre dans B_+). En deuxième itération $s = (3,2)$ (qui entre dans I), le seul indice candidat, $\theta = 0$, et par exemple $r = (1,2)$ (qui entre dans B_-). Ici il n'y a pas de direction privilégiée qui est de pente et réalisable (c'est-à-dire, $\theta > 0$). En troisième itération $s = (2,4)$ (qui entre dans I), le seul indice candidat, $\theta = 4$, et par exemple $r = (3,4)$ (qui entre dans B_-). En quatrième itération nous obtenons la CSO comme avant. Nous notons que ces dégénérescences avec $\theta = 0$ apparaissent naturellement, quelque soit le pricing.

3.5.16. Exemple d'un problème de transport

Considérons un problème de transport à coût minimum (voir 3.4.4) à trois usines et quatre clients, sans limitation de capacité pour les routes, qui est modélisé par le programme linéaire

$$\min\{fx : Ax = a = (6,8,10,-4,-6,-8,-6)^T, x \geq b = 0\},$$

et $c = (+\infty, ..., +\infty)^T$, avec A la matrice d'incidence d'un graphe orienté bi-parti complet $G = (L, L_1 \times L_2)$, avec $L_1 = \{1,2,3\}$ (les usines, avec stock $6,8,10$, respectivement), $L_2 = L \setminus L_1 = \{4,5,6,7\}$ (les clients, avec demande $4,6,8,6$, respectivement). Les coefficients du vecteur coût (indexé par des arcs) sont donnés dans le tableau suivant (changement de signe pour obtenir un problème de maximisation)

-1	-2	-3	-4	6
-4	-3	-2	0	8
0	-2	-2	-1	10
4	6	8	6	

avec dans la marge en bas les quantités $-a_{p+1}, ..., -a_m$ demandées, et à droite les quantités $a_1, ..., a_p$ disponibles (ici $p = 3, m = 7$). Notons que $\sum_j a_j = 0$, comme pour tout problème de transport avec égalités.[11] *Comme dans l'exemple précédent on pourrait continuer à noter*

11. Le problème de transport de l'exemple 1.2.4 comporte des inégalités, mais les variables d'écart peuvent être interprétés comme une usine ou un client supplémentaire, au moins dans le cas des coûts unitaires strictement positifs.

les quantités de Simplex directement sur le graphe, mais ce graphe devient assez important, et l'écriture moins suggestive. On préfère noter ces quantités dans des tableaux avec en ligne les usines et en colonne les clients : chaque case d'un tel tableau est en bijection avec un arc du graphe bi-parti $(L_1 \cup L_2, L_1 \times L_2)$*. Représentons dans ce tableau une solution de base réalisable. Rappelons que pour un problème de transport à* m *sommets une base de la matrice des contraintes a dimension* $m-1$ *et elle correspond à un arbre dans le graphe associé* I*. Si on définit le degré d'un sommet de* (L, I) *par*

$$deg\ (l) = card\{j : \quad but(j) = l\ ou\ source(j) = l\}$$

on conclut que dans un arbre il y a toujours au moins deux sommets de degré 1*. En effet,* $\sum_{\ell \in L} deg(l) = 2\times$ *nombre d'arcs* $= 2(m-1)$ *et tous les sommets de l'arbre ont un degré* ≥ 1*. Ceci implique que dans un tel tableau, les cases correspondantes aux arcs de l'arbre (variables de base) vérifient les propriétés :*

— *il y a au moins une variable dans la base par ligne et par colonne du tableau ;*

— *il y a au moins une ligne ou une colonne du tableau qui ne contiennent qu'un élément.*

On vérifie aisément que le tableau suivant contient un point de base réalisable

$I\|4$	$I\|2$	B_-	B_-	
B_-	$I\|4$	$I\|4$	B_-	
B_-	B_-	$I\|4$	$I\|6$	

Avant d'utiliser ces remarques pour proposer une procédure qui permet la construction d'une base réalisable de départ regardons plus en détail dans cet exemple comment se déroule la méthode du simplex et comment noter dans le tableau les quantités necessaires à chaque étape Pour les cases $j \in B_- \cup B_+$ *hors base (ici* $B_+ = \emptyset$*), on note les coûts réduits* $d(I)^j$*, et dans les marges en bas les potentiels pour les clients (les buts des arcs), et à droite les potentiels pour les usines (les sources de nos arcs). Avec la normalisation* $\lambda^4 = 0$*, ces potentiels pour notre base sont calculés par (3.10) dans l'ordre* $\lambda^4 = 0$ *(donné),* $\lambda^1, \lambda^5, \lambda^2, \lambda^6, \lambda^3, \lambda^7$*, et ensuite les coûts réduits par (3.11), ce qui donne le tableau*

$I\|4\ominus$	$I\|2\oplus$	$B_-\|-2$	$B_-\|-4$	-1
$B_-\|-2$	$I\|4\ominus$	$I\|4\oplus$	$B_-\|1$	-2
$B_-\|2\oplus$	$B_-\|1$	$I\|4\ominus$	$I\|6$	-2
0	1	0	-1	

Par mvp on trouve $s = (3,4)$*, et il faut maintenant trouver l'unique cycle élémentaire* γ *dans* $(L, I+s)$ *avec* $\vec{\gamma}_s = 1$ *(car* $s \in B_-$*). Un petit dessin du graphe* $(L, I+s)$ *donne* $\gamma = [3,4,1,5,2,6,3]$*, mais ce cycle peut également être détecté dans le tableau : on marque* s *par* $\oplus$ *et en partant de* s *on choisi successivement des cases dans* $I+s$ *de sorte que deux cases consécutives soient alternativement dans la même colonne et dans la même ligne avec signe opposé (comme* G *est bi-parti, on alterne dans ce cycle des arcs* j *empruntés dans le bon sens (symbole* $\oplus$*,* $\vec{\gamma}_j = 1$*) avec des arcs* j *empruntés au sens opposé (symbole* $\ominus$*,* $\vec{\gamma}_j = -1$*). Comme le cycle est élémentaire on a au plus deux cases par ligne ou colonne. Connaissant* γ*, nous obtenons la capacité* $\theta = 4$*, et nous trouvons trois choix différents pour* r*. Le choix* $r = (2,5)$ *donne le*

schéma suivant

$I\|0$	$I\|6$	$B_-\|0$	$B_-\|-2$	-1
$B_-\|-4$	$B_-\|-2$	$I\|8\ominus$	$B_-\|1\oplus$	0
$I\|4$	$B_-\|-1$	$I\|0\oplus$	$I\|6\ominus$	0
0	1	2	1	

avec $s = (2, 7)$ *(unique),* $\gamma = [2, 7, 3, 6, 2]$, $\theta = 6$, $r = (3, 7)$ *(unique), et*

$I\|0$	$I\|6$	$B_-\|0$	$B_-\|-3$	-1
$B_-\|-4$	$B_-\|-2$	$I\|2$	$I\|6$	0
$I\|4$	$B_-\|-1$	$I\|6$	$B_-\|-1$	0
0	1	2	0	

c'est à dire, on obtient la condition suffisante d'optimalité et une solution optimale.

Reste maintenant le problème de la construction d'une base réalisable de départ. En nous basant sur les remarques ci-dessus, nous proposons la procédure suivante basée sur une heuristique de côut minimum :

On initialise le tableau

$$\mathcal{T} = \begin{array}{|c|c|c|c|c|} \hline & & & & a_1 \\ \hline & & & & \vdots \\ \hline & & & & a_p \\ \hline a_{p+1} & \cdots & \cdots & a_m & \\ \hline \end{array}$$

Tant que $\mathcal{T}$ *n'est pas vide faire*

- *choisir une case* (l, k) *de* $\mathcal{T}$ *telle que* $f_{(l,k)} = \max_{(i,j)\in\mathcal{T}} f_{(i,j)}$
- *si* $\min(a_l, a_k) = a_l$ *faire*
  - $x_{(l,k)} = a_l$, $a_k \leftarrow a_k - a_l$
  - *mettre à jour le tableau* $\mathcal{T}$ *en supprimant sa ligne* l *;*
- *si* $\min(a_l, a_k) = a_k$ *faire*
  - $x_{(l,k)} = a_k$, $a_l \leftarrow a_l - a_k$
  - *mettre à jour le tableau* $\mathcal{T}$ *en supprimant sa colonne* k *.*

Cette boucle est effectuée $m - 1$ *fois et met en évidence un arbre dans le graphe ; chaque étape élimine un sommet et permet de sélectionner un arc qui le relie à l'ensemble de ceux qui restent. A la fin on obtient l'arborescence d'accessibilité. Si on applique cette procédure à l'exemple précédent on obtient*

	$I\|6$	$I\|0$		6
		$I\|2$	$I\|6$	8
$I\|4$		$I\|6$		10
4	6	8	6	

En choisissant $\lambda_1 = 0$, *on calcule par (ref...) les coûts marginaux suivis des coûts réduits*

0	$I\|6$	$I\|0$	-3	0
-4	-4	$I\|2$	$I\|6$	1
$I\|4$	-1	$I\|6$	-1	1
1	2	3	1	

et on conclut que la solution est optimale.

D'autres heuristiques de construction de la solution de base de départ et et un exemple avec des variables born 'ees seront traités en exercices.

3.6 Quelques autres problèmes classiques sur les graphes

Le but de ce chapitre est d'évoquer brièvement quelques autres problèmes classiques d'optimisation sur les graphes.

Dans un premier temps on se pose la question si on peut tracer l'ensemble des arêtes d'un graphe non orienté sans lever le crayon. C'est Euler qui avait donné une réponse à cette question en cherchant un parcours pour sa promenade matinale qui lui permettait de passer une et une seule fois par chaque pont dans sa ville natale de Königsberg.

3.6.1. Définition : degré, cycle eulerien
Le degré *d'un sommet s d'un graphe non orienté $G = (L, J)$ est le nombre d'arêtes dont ce sommet est une extrémité, noté par* $\deg_G(x)$.

Un cycle eulerien *est une chaîne fermée d'un graphe non orienté qui emprunte chaque arête une et une seule fois.*

3.6.2. Théorème d'Euler
Soit $G = (L, J)$ un graphe non orienté sans sommets de degré zéro (sommets isolés). G contient un cycle eulerien si et seulement si G est connexe et chaque sommet est de degré pair.

Pour la preuve du théorème 3.6.2 de Euler nous aurons besoin d'un algorithme de construction de chaînes maximales.

3.6.3. Lemme : Construction d'une chaîne maximale
Pour un graphe $G' = (J', L')$ et un sommet $s \in L'$ avec $\deg_{G'}(s) > 0$, considérons la fonction $\gamma' = maxi(s, G')$ suivante de construction d'une chaîne maximale dans G' partant de s, avec interdiction de répétition d'arêtes

Initialisation : poser $y_1 \leftarrow s$, $\gamma' \leftarrow \emptyset$, $\ell \leftarrow 1$
Itération : *Tant qu'il existe* $y_{\ell+1} \in L'$ *avec* $a := \{y_\ell, y_{\ell+1}\} \in J' \setminus \gamma'$ *faire*
$\gamma' \leftarrow (\gamma', a)$, $\ell \leftarrow \ell + 1$.

Notons par x le dernier sommet rencontré par γ'.
(a) Le sommet x admet un degré $\deg_{G'}(x)$ *pair si et seulement si $s = x$.*
(b) Dans le graphe résiduel $G'' = (L', J' \setminus \gamma')$ nous avons $\deg_{G''}(x) = 0$.
(c) Si on applique $\gamma' = maxi(s, G')$ à un graphe où les sommets sont tous de degré pair alors γ' est un cycle.

Démonstration. Pour un sommet y rencontré par γ', notons par $k(y)$ le nombre de fois que l'on rencontre y en parcourant la chaîne γ', et par $deg_{(L',\gamma')}(y)$ le nombre d'arêtes dans γ' avec extrémité y. Pour un $y \neq x, s$, chaque fois que l'on arrive à y, on repart. Sachant que les arêtes composant γ' sont disjoints, nous obtenons $deg_{(L',\gamma')}(y) = 2k(y)$. Par le même argument, nous obtenons dans le cas $y \in \{s, x\}$, $s \neq x$ d'une extrémité simple de γ' la relation $deg_{(L',\gamma')}(y) =$

$2k(y)-1$, et dans le cas $s = x = y$ d'une extrémité double de γ' la relation $deg_{(L',\gamma')}(y) = 2k(y)-2$. Finalement, le fait d'être bloqué au sommet x signifie que l'on a déjà parcouru toutes les arêtes dans G' avec extrémité x, d'ou

$$deg_{(L',\gamma')}(x) = deg_{G'}(x),$$

ce qui démontre (a), et la propriété (b) provient du fait que $deg_{G''}(x) = deg_{G'}(x) - deg_{(L',\gamma')}(x)$. Finalement, la propriété (c) est une conséquence immédiate de la propriété (a). □

Démonstration de 3.6.2. Soit γ un cycle eulerien dans G. Montrons d'abord que G est connexe. En effet, comme tout sommet de G est l'extrémité d'une arête au moins, un cycle eulerien doit passer par tous les sommets de G. Donc deux sommets quelconques dans G sont reliés par une sous-chaîne de γ. Montrons maintenant que le degré de chaque sommet doit être pair : Pour un sommet $x \in S$, notons par $k(x)$ le nombre de fois que l'on rencontre le sommet x en parcourant une fois le cycle γ (sans compter le sommet d'arrivé qui coïncide avec le sommet de départ). Chaque fois que l'on arrive à x, on repart. Sachant que les arêtes composant un cycle eulerien sont disjoints, le nombre d'arêtes dans γ (et donc dans G) ayant l'extrémité x est égal à $2k(x)$, un nombre pair.
Pour montrer l'autre implication du théorème, on donnera un algorithme de construction d'un cycle eulerien dans $G = (L, J)$, partant d'un sommet quelconque $s \in L$:

Initialisation : Construire $\gamma \leftarrow maxi(s, G)$
Construire graphe résiduel $J' \leftarrow J \setminus \gamma$, $G' \leftarrow (L, J')$, $k \leftarrow 1$
Itération : Tant que $k <$ longueur de γ faire
$k \leftarrow k + 1$. Soit x_k le k-ième sommet rencontré dans γ.
Si $deg_{G'}(x_k) > 0$ alors faire
Construire $\gamma' = [y_1, ..., y_\ell] \leftarrow maxi(x_k, G')$
Arrêt si γ' n'est pas fermé (un sommet est de degré impair)
sinon ajouter le bout $[y_2, ..., y_\ell]$ derrière x_k dans γ
Mettre à jour graphe résiduel $J' \leftarrow J' \setminus \gamma'$, $G' \leftarrow (L, J')$
Invariants : (i) $G' = (L, J') = (L, J \setminus \gamma)$
(ii) Avant chaque itération, le degré de chaque sommet dans G' est pair.
(iii) Après chaque itération nous avons $deg_{G'}(x_k) = 0$.
(iv) Après chaque itération, γ est un cycle.

On remarque que la longueur de γ est variable (et donc aussi l'énumération des sommets dans γ) : si $deg_{G'}(x_k) > 0$ alors il existe la possibilité d'inclure une nouvelle chaîne dans γ partant de x_k, avec arêtes dans G', c'est-à-dire, des arêtes pas encore utilisées dans γ. D'ailleurs, dans l'itération suivante, on continue de parcourir cette nouvelle chaîne.
L'invariant (i) est évident. Comme chaque sommet dans G admet un degré pair, on déduit par récurrence en utilisant le lemme 3.6.3(c) que tout γ' est un cycle (inclus la chaîne γ construite à l'initialisation). Donc l'invariant (ii) est une conséquence du fait que le nouveau J' est obtenu à partir de l'ancien en enlevant un cycle (en passant notons que la condition d'arrêt incluse dans l'algorithme permet de vérifier l'hypothèse sur le degré des sommets dans G). L'invariant (iii) découle du lemme 3.6.3(b), et l'invariant (iv) est une conséquence de l'observation que, à chaque itération, on insère un cycle dans un cycle. Finalement, l'algorithme est fini, car d'après (iii) on ajoute seulement des cycles pour des x_k distincts.

Notons par γ le résultat de l'algorithme (qui par (iv) est un cycle), et par $\widetilde{L} \subset L$ l'ensemble des sommets rencontrés par γ. D'après (iii) nous avons pour tout $x \in \widetilde{S}$ la relation $deg_{G'}(x) = 0$. Par conséquent, les arêtes dans $J \setminus \gamma$ ont comme extrémités seulement des éléments de $L \setminus \widetilde{L}$, et les arêtes dans γ ont comme extrémités seulement des éléments de $\widetilde{L}$.
En conséquence, aucun sommet dans $L \setminus \widetilde{L}$ n'est accessible par une chaîne depuis $s \in \widetilde{L}$, mais G est supposé d'être connexe. Donc $L = \widetilde{L}$, $J \setminus \gamma$ est vide et par conséquent γ est eulerien. □

Comme illustration nous présentons dans le fichier `don_ch3/euler0.bat` une simulation ou le premier cycle γ construit dans l'initialisation est déjà eulerien. Dans les fichiers `don_ch3/euler1.bat` et `don_ch3/euler2.bat` on trouve des exemples où il est nécessaire d'insérer un autre cycle (et deux, respectivement).

L'algorithme donné dans la preuve admet une complexité de $\mathcal{O}(n)$ car toute arête est examinée au plus deux fois. Notons également que l'on montre de la même manière qu'un graphe non orienté comporte une chaîne eulerienne (pas forcément fermée) si et seulement si il existent soit aucun soit deux sommets de degré impair (dans le dernier cas, il suffit de choisir dans l'algorithme comme s un sommet de degré impair). Notre algorithme fonctionne aussi si on ne sait pas d'avance si le graphe contient une chaîne eulerienne ou un cycle eulerien : si le graphe comporte au moins 3 sommets de degré impair alors, après initialisation, G' comporte encore au moins un sommet de degré impair. D'après le lemme 3.6.3(a), l'algorithme va s'arrêter au plus tard au moment où on rencontre comme x_k ce sommet de degré impair. A titre d'exemple, nous donnons dans les fichiers `don_ch3/euler_n1.bat` et `don_ch3/euler_n4.bat` des simulations dans un graphe (la maison du St. Nicolas) avec exactement deux sommets de degré impair, où pour la première simulation on ne part pas du sommet correct.

Aussi, on peut donner une extension au cas d'un multi-graphe (ici on permet plusieurs arêtes ayant les mêmes extrémités). Les notions de degré, chaîne, cycle (eulerien), connexité se généralisent sans difficultés. L'algorithme et le théorème d'Euler restent valables, voir l'exemple des ponts de Königsberg dans `don_ch3/pont_koe.bat`.

Généralement, un graphe $G = (L, J)$ ne permet pas la construction d'un cycle eulerien. Pour le *problème du postier chinois* on se permet d'ajouter des arcs. Pour simplifier, on supposera que les hypothèses de base du théorème 3.6.2 d'Euler sont déjà valables.

3.6.4. Définition : Problème du postier chinois
Soit $G = (L, J)$ un graphe non orienté connexe et sans sommets isolés. Etant donné un ensemble K d'arêtes supplémentaires ayant leur extrémités dans L (par exemple une copie de J), et pour chaque $j \in K$ une valeur (un coût) $d(j) > 0$, on souhaite trouver un sous-ensemble $K' \subset K$ au moindre coût de sorte que le nouveau graphe complété $(L, J \cup K')$ permet la construction d'un cycle eulerien. Mathématiquement parlant, on doit résoudre

$$\min\Big\{ \sum_{j \in K'} d(j) \quad : \quad K' \subset K, \text{ et le graphe } (L, J \cup K') \text{ contient un cycle eulerien} \Big\}.$$

Voici deux exemples d'applications.

3.6.5. Exemple : Ramassage scolaire
Un bus doit passer au moins une fois par toutes les rues d'un village pour chercher des enfants.

Il sera généralement obligé d'emprunter certaines rues à plusieurs reprises pour construire un tel parcours. Chercher à trouver ces rues tout en minimisant la longueur du parcours (on supposera que le bus part de l'école). On retrouve le problème du postier chinois pour le graphe (L, J) ou les arêtes correspondent aux rues, les sommets aux carrefours, et K est une copie de J valué par les longueurs de chaque rue.

3.6.6. Exemple : Tracer un graphe/maillage EDP

On cherche à traçer au mieux l'ensemble des arêtes d'un graphe non orienté (connexe et sans sommets isolés) sur une table traçante. On a deux modes opératoires :

- *plume basse : on trace une arête (une et une seule fois), sa forme étant connue à l'ordinateur,*
- *plume haute : on passe d'un sommet à un autre sans tracer pour positionner la plume.*

On cherche une meilleure stratégie minimisant les mouvements de la plume. Le graphe (L, J) étant donné, ici (L, K) est un graphe complet (voir définition 3.1.2) non orienté, avec arête $j = \{\ell, \ell'\} \in K$ valuée par la distance euclidienne entre les sommets ℓ et ℓ'.

Pour se ramener à un problème de *couplage parfait de valeur minimale* (voir 3.6.8 ci-dessous), nous avons besoin de quelques propriétés.

3.6.7. Lemme :

Soit $G = (L, J)$ comme dans 3.6.4, et $L^* \subset L$ l'ensemble des sommets de degré impair.

(a) *L'ensemble L^* est de cardinalité paire.*

(b) *Supposons que l'on peut construire un cycle eulerien dans $G' = (L, J \cup K')$, et que $|L^*| = 2p$. Alors on peut construire p chaînes simples dans $H = (L, K')$ n'ayant pas d'arêtes en commun, et ayant L^* comme ensemble d'extrémités (et donc aucune chaîne est fermée, et deux chaînes ont des extrémités distincts).*

(c) *Réciproquement, si $|L^*| = 2p$ et si on dispose de p chaînes simples dans (L, K) n'ayant pas d'arête en commun et ayant L^* comme ensemble d'extrémités, et finalement si on définit K' étant l'ensemble des arêtes utilisées pour ces chaînes, alors $(L, J \cup K')$ permet la construction d'un cycle eulerien.*

Démonstration. Partie (a) est basée sur l'observation que la somme des degrés des sommets dans L donne deux fois le nombre d'arêtes.

Pour démontrer (b), notons que, pour $y \in L$,

$$deg_H(y) = deg_{G'}(y) - deg_G(y) \quad \text{impair si et seulement si } y \in L^*.$$

Nous appliquons à p reprises le lemme 3.6.3(a) : soit $x_1 \in L^*$, $\gamma_1 = maxi(x_1, H)$ (qui par construction est simple), et soit x_1' le dernier sommet rencontré par γ_1. Comme $deg_H(x_1)$ est impair, nous déduisons de 3.6.3(a) que $x_1 \neq x_1'$, et que $deg_H(x_1')$ est aussi impair, d'où $x_1' \in L^*$. Pour le graphe résiduel $(L, K' \setminus \gamma_1')$ nommé aussi H, on a maintenant $2p - 2$ sommets de degré impair, car x_1 et x_1' ont un degré pair dans ce nouveau graphe, et la parité des degrés des autres sommets rencontrés par γ_1 ne change pas. En répétant la même procédure $p - 1$ fois (si $p > 1$), on obtient les chaînes désirées (notons que l'on n'utilise pas forcément toutes les arêtes dans K').

Pour une preuve de (c), nous notons $H = (L, K')$, $G' = (L, J \cup K')$, et observons que, par construction,

$$deg_{G'}(y) = deg_H(y) + deg_G(y), \qquad \text{et } deg_H(y) \text{ est impair si et seulement si } y \in L^*.$$

Donc tout sommet dans G' est de degré pair. Comme avec G aussi G' est connexe et sans sommets isolés, le théorème 3.6.2 d'Euler nous affirme que l'on peut construire un cycle eulerien dans G'. □

3.6.8. Définition : Problème de couplage parfait de valeur minimale
Etant donné un graphe non orienté $\overline{G} = (\overline{L}, \overline{J})$ valué par $\overline{d}(\cdot)$, le problème de couplage parfait de valeur minimale *consiste à extraire un sous-ensemble $\overline{K} \subset \overline{J}$ de sorte que tout sommet $x \in \overline{L}$ soit extrémité d'une et d'une seule arête dans $\overline{K}$, tout en minimisant*

$$d(\overline{K}) = \sum_{j \in \overline{K}} \overline{d}(j).$$

Autrement dit, on cherche à former des couples (sans faire attention au sexe) tout en minimisant le coût des mariages. Mathématiquement parlant, il est utile d'introduire la matrice d'incidence d'un graphe non orienté, définie par

$$\overline{A} = \overline{A}^{\overline{J}}_{\overline{L}}, \qquad \overline{A}^j_\ell = \begin{cases} 1 & \textit{si } \ell \textit{ est une extrémité de } j, \\ 0 & \textit{sinon.} \end{cases} \tag{3.12}$$

En introduisant le vecteur caractéristique $x = (x_j)_{j \in \overline{J}}$ avec $x_j = 1$ si $j \in \overline{K}$ et sinon $x_j = 0$, le problème de couplage parfait de valeur minimale devient alors

$$\min\Big\{ \sum_{j \in \overline{J}} \overline{d}(j)\, x_j : \begin{array}{l} \textit{Pour tout } j \in \overline{J},\ x_j \in \{0,1\},\ \textit{et} \\ \textit{pour tout } \ell \in \overline{L},\ \textit{la somme des } x_j \\ \textit{avec } j \textit{ ayant l'extrémité } \ell \textit{ vaut } 1 \end{array} \Big\}. \tag{3.13}$$

$$= \min\Big\{ (\overline{d}(j))^{j \in \overline{J}} x : \ \overline{A}x = (1,1,....,1)^T,\ x \in \{0,1\}^{|\overline{J}|} \Big\}. \tag{3.14}$$

Montrons maintenant le lien entre les deux problèmes 3.6.4 et 3.6.8.

3.6.9. Théorème
Soit $G = (L, J)$ comme dans 3.6.4, et $L^* \subset L$ l'ensemble des sommets dans G de degré impair. Considérons le graphe complet $(\overline{L}, \overline{J})$ ayant $\overline{L} = L^*$ comme ensemble de sommets, la valeur d'une arête $(s,t) \in \overline{J}$ étant égale à la valeur d'une chaîne de valeur minimale de s à t dans (L, K). Alors une solution optimale K^* du problème du postier chinois induit une solution optimale $\overline{K}$ du problème de couplage parfait de valeur minimale dans $(L^*, \overline{J})$, et réciproquement.

Démonstration. (i) Soit $K^* \subset K$ une solution optimale du problème du postier chinois, de valeur $d(K^*)$. Le lemme 3.6.7(b) nous donne un ensemble de $p = |L^*|/2$ chaînes $\gamma_1, ...,_g amma_p$ dans (L, K) avec extrémités x_ℓ, x'_ℓ, et $L^* = \{x_\ell, x'_\ell : \ell = 1, 2, ..., p\}$. Comme ces chaînes sont simples et n'ont pas d'arêtes en commun, nous obtenons

$$d(K^*) \geq d(\gamma_1) + ... + d(\gamma_p) \geq \overline{d}((x_1, x'_1)) + ... + \overline{d}((x_p, x'_p)), \tag{3.15}$$

la dernière inégalité provenant du fait que γ_ℓ n'est pas forcément une plus courte chaîne de x_ℓ à x'_ℓ dans (L, K). Par conséquent, $\overline{K} = \{(x_\ell, x'_\ell) : \ell = 1, ..., p\}$ est un couplage parfait, de valeur $\overline{d}(\overline{K}) \leq d(K^*)$.
(ii) Soit maintenant $\overline{K} = \{(x_\ell, x'_\ell) : \ell = 1, ..., p\}$ une solution optimale du problème de couplage

parfait de valeur minimale, et γ_ℓ une plus courte chaîne de x_ℓ à x'_ℓ dans (L, K). Comme la valuation de (L, K) est strictement positive par hypothèse, la chaîne γ_ℓ est élémentaire (comparer avec le lemme 3.3.1) et en particulier simple. Montrons que deux chaînes n'ont pas d'arête en commun. Par absurde, supposons que

$$\gamma_1 = [y_1, ..., y_{p_1}], \quad \gamma_2 = [z_1, ..., z_{p_2}], \quad \{y_{q_1}, y_{q_1+1}\} = \{z_{q_2}, z_{q_2+1}\}$$

(et donc $1 < q_1 < p_1 - 1$, $1 < q_2 < p_2 - 1$ car les extrémités sont distincts). Dans le cas $y_{q_1} = z_{q_2}$,

$$d(\gamma_1) + d(\gamma_2) > d([y_1, ..y_{q_1}, z_{q_1-1}, ..., z_1]) + d([y_{p_1}, ..y_{q_1+1}, z_{q_1+2}, ..., z_{q_2}])$$

montrant que les couples (x_1, x_2), (x'_1, x'_2) donnent un couplage parfait de valeur strictement plus petite, en contradiction avec l'hypothèse sur $\overline{K}$. De la même manière on montre une contradiction dans le cas $y_{q_1} = z_{q_2+1}$.
Par conséquent, le lemme 3.6.7(c) nous dit que K' étant l'ensemble des arêtes utilisées pour $\gamma_1, ..., \gamma_\ell$ est une solution réalisable pour le problème du postier chinois, et $\overline{d}(\overline{K}) = d(K')$ par construction.
(iii) Les parties (i) et (ii) nous permettent d'affirmer que les deux problèmes 3.6.4 et 3.6.8 ont la même valeur optimale, et que chaque fois, partant d'une solution optimale d'un des deux problèmes, on a trouvé une solution optimale de l'autre. □

La précédente preuve nous donne un certain nombre d'indications sur la manière comment le conducteur de bus dans l'exemple 3.6.5 devrait choisir son parcours : le conducteur devrait essayer de construire un parcours eulerien dans $G = (L, J)$, (prendre des rues une et une seule fois), mais il sera bloqué en un sommet (carrefour) de degré impair. Dans ce cas, il devrait faire un détour en se rendant le plus vite possible à un autre sommet de degré impair, sachant que les rues/arêtes pour ce détour (dans (L, K), avec K une copie de J) seront empruntés plusieurs fois. Plus précisément, avec un choix optimum du nouveau sommet de degré impair (problème du couplage parfait), le conducteur n'empruntera jamais une rue à plus que deux reprises (car les chaînes dans la partie (ii) de la preuve n'ont pas d'arête en commun). Le seul problème restant est de trouver la solution optimale du problème de couplage parfait de valeur minimale (ce qui est assez simple s'il y a seulement deux ou quatre sommets de degré impair).

En supposant que $|L^*| = 2p$, le calcul des plus courtes chaînes demande $\mathcal{O}(p^3)$ opérations (en utilisant par exemple l'algorithme 3.3.12 de Roy–Warshal). La résolution du problème de couplage parfait de valeur minimale dans un graphe à m sommets et n arêtes (chez nous $m = 2p$, $n = p(2p-1)$) demande $\mathcal{O}(mn^2)$ *ou* $\mathcal{O}(m^3)$ opérations suivant l'approche d'Edmonds (1965), peaufinée par Gabow et Lawner (1976), voir [GoMi95, pp.293–297].

Vu que cette approche est efficace mais assez long à décrire, dans nos simulations numériques nous allons directement résoudre le problème (3.14), un programme linéaire en variables astreintes d'être entières, ayant comme matrice de coefficients la matrice d'incidence (3.12) d'un graphe non orienté. Contrairement au cas des graphes orientés (voir le théorème 3.5.1), la matrice d'incidence d'un graphe non orienté n'est généralement pas unimodulaire. Donc remplacer les contraintes $x_j \in \{0, 1\}$ par $x_j \in [0, 1]$ nous ne donnera plus forcément des solutions à coefficients entiers. [12]

12. D'ailleurs, on peut montrer qu'une matrice d'incidence d'un graphe non orienté est unimodulaire si et

Pour les simulations ci-dessous on propose une résolution de (3.14) par l'approche Branch and Bound décrite au chapitre 3.7, qui peut être bien plus coûteuse. Dans le fichier `don_ch3/post_m10.bat` on trouve une simulation pour un graphe où il faut rajouter deux arêtes. L'exemple classique d'Euler (les ponts de Königsberg) se trouve dans le fichier don_ch3/post_koe.bat. Dans `don_ch3/post_4co.bat` et dans don_ch3/post_bat.bat on trouve des exemples ou il faut rajouter plus que deux arêtes, et où les plus courtes chaînes ne sont pas réduites à une arête.

Terminons ce chapitre en évoquant deux autres problèmes classiques sur les graphes, le voyageur de commerce (voir l'exemple 1.2.3) et le problème de coloration d'arêtes ou de sommets.

3.6.10. Quelques rémarques sur le problème du voyageur de commerce
Un circuit/cycle hamiltonien *est un chemin (pour les graphes orientés) fermé ou une chaîne (sans orientations) fermée qui passe une et une seule fois par tous les sommets. Le* voyageur de commerce *cherche à faire un parcours permettant de passer une et une seule fois par chaque ville tout en minimisant la longueur du trajet ; on cherche alors un circuit (cycle) hamiltonien de valeur minimale.*

On ne connaît aucun algorithme de complexité polynômiale $\mathcal{O}(m^p n^q)$ pour trouver un circuit hamiltonian.

Un théorème de type Euler pour la caractérisation d'un graphe comportant un circuit/cycle hamiltonien n'est pas connu. Certains auteurs ont donné des conditions suffisantes d'existence ou des conditions nécessaires d'existence, par exemple Ore (1960) : Dans un graphe non orienté G à m sommets, si pour tout couple de sommets non reliés par une arête la somme des degrés est $\geq m$ alors G permet la construction d'un cycle hamiltonien.

Pour la recherche d'un circuit hamiltonien (de valeur minimale), SCILAB comporte des commandes `hamilton` et `salesman`. Les algorithmes de recherche d'un circuit/cycle hamiltonien (de valeur minimale) sont soit de nature heuristique, soit soit ils passent par une résolution d'un programme linéaire en nombres entiers, par exemple celui donné ci-dessous.

3.6.11. Modélisation du problème du voyageur de commerce orienté
Soit donné un graphe orienté $G = (L, J)$, (L=villes, J="routes" disponibles) valué par $d(\cdot)$, $m := |L|$, $n := |J|$. Pour une modélisation d'un problème de voyageur de commerce, on introduira les variables

$$j \in J : \qquad x_j \in \{0,1\}, \qquad \ell \in L : \qquad u_\ell \in \{1,2,...,m\}. \tag{3.16}$$

La variable bivalente x_j attribuée à chaque arc a comme objectif que $x_j = 1$ si et seulement si on emprunt la route j ; la variable entière u_ℓ pour un sommet ℓ aura la valeur $u_\ell = k$ si la ville ℓ est visitée à la k-ième place dans une tournée commençant à la ville 1, et donc $u_1 = 1$ (la ville de départ est arbitraire : de toute façon, on y revient). La contrainte

$$\forall j = (\ell, k) \in J,\, k \neq 1 : \quad u_\ell - u_k + m x_j \leq (m-1) \tag{3.17}$$

seulement si le graphe est bi-parti. Il est alors plus simple de former des couple hommes-femmes (graphe bi-parti) au moindre coût que des couples quelconque (graphe complet). Bien entendu, ceci ne devrait pas être interprété comme une remarque sexiste. D'ailleurs, former des couples hommes-femmes au moindre coût est équivalent au problème d'affectation discuté déjà au 3.4.5 (à l'aide des graphes orientés).

est trivialement valable si $x_j = 0$; dans le cas $x_j = 1$ elle traduit le fait que k est visité après ℓ si on utilise la route (ℓ, k) (ce qui est vrai pour un circuit hamiltonien sauf pour l'arc revenant à la ville $k = 1$). Il s'y ajoutent des contraintes de type transport : on arrive à chaque ville exactement une fois, et on part de chaque ville exactement une fois

$$\forall \ell \in L : \qquad \sum_{k \in L, (\ell,k) \in J} x_{(\ell,k)} = 1, \qquad \sum_{k \in L, (k,\ell) \in J} x_{(k,\ell)} = 1. \tag{3.18}$$

Ces contraintes sont évidemment aussi valables pour un circuit hamiltonien.

Réciproquement, nous devons montrer que tout (x, u) vérifiant (3.16), (3.17) et (3.18) nous donne un circuit hamiltonien. D'abord, $x \in \{0,1\}^J$ vérifiant la contrainte (3.18) nous donne un sous-ensemble d'arcs $J' := \{j \in J : x_j = 1\}$ qui est formé d'un ou de plusieurs circuits rencontrant des ensembles de sommets distincts : pour construire ces circuits il suffit d'appliquer la fonction maxi du lemme 3.6.3 au graphe (L, J') car (3.17) implique que chaque sommet dans ce graphe a un degré égal à 2. La contrainte supplémentaire (3.17) nous assure qu'il existe exactement un circuit : sinon, il existe un circuit $[\ell_0, \ell_1, ..., \ell_p]$ ne passant pas par le sommet 1, $\ell_0 = \ell_p$, et alors

$$0 = \sum_{i=1}^{p} (u_{\ell_{i-1}} - u_{\ell_i}) \leq \sum_{i=1}^{p} (m - 1 - m x_{(\ell_{i-1}, \ell_i)}) = \sum_{i=1}^{p} (-1) = -p,$$

une contradiction.

Pour résoudre le problème du voyageur de commerce, nous nous retrouvons alors avec un programme linéaire (PL) formé par (3.16), (3.17), (3.18), et l'objectif

$$\min(\sum_{j \in J} d(j) x_j).$$

Ce programme comporte des variables x_j bivalentes et des variables u_ℓ astreintes d'être entières, plus éventuellement des variables d'écart (qui a priori peuvent être des réels).

Dans les fichiers ci-joints on donne quelques simulations à l'aide de la boîte noire AMPL (d'après l'affichage, AMPL utilise la méthode "Branch & Bound" pour la résolution, voir aussi 3.7.11). La valuation d'un arc dans ces exemples est donné par la distance euclidienne des deux extrémités.

- *Dans* `don_ch3/voyag0.bat` *on résout un programme pour un graphe complet à 4 sommets où on a oublié la contrainte (3.17) : on se retrouve effectivement avec deux circuits. La simulation correspondante avec la contrainte (3.17) est donnée dans le fichier* `don_ch3/voyag1.bat`*. On observe en particulier qu'il n'y a pas d'intersection d'arcs dans le plus court circuit hamiltonien.*
- *Un deuxième exemple d'un graphe orienté mais pas complet à 7 sommets est donné dans* `don_ch3/voyag2.bat`*. On n'aurait probablement pas deviné cette solution optimale...*

Le dernier problème évoqué dans ce chapitre est celui de la coloration d'un graphe.

3.6.12. Définition : Problèmes de coloration
On se pose le problème de colorier les sommets d'un graphe non orienté $G = (S, A)$ de sorte que

deux sommets adjacents (reliés par une arête) n'aient pas la même couleur. On appelle nombre chromatique *de G, noté par $\gamma(G)$, le nombre minimal de couleurs nécessaires.*

*L'*indice chromatique *$q(G)$ est le nombre minimal de couleurs pour la coloration d'arêtes, de sorte que deux arêtes ayant une extrémité en commun n'aient pas la même couleur.*

Pour colorier les arêtes, il suffit de colorier les sommets du graphe $L(G) = (A, A')$ (nommé *line graph*) avec $(\ell, k) \in A'$ si les arêtes ℓ, k ont une extrémité commune. Regardons quelques exemples d'application pratique de ce problème.

3.6.13. Exemple : problème d'une flotte d'avions
Une compagnie aérienne doit satisfaire des demandes de disponibilité d'avions dans des intervalles de temps $I_j = [a_j, b_j]$ données, pour $j = 1, ..., m$. On se demande combien d'avions seront nécessaires.

Pour répondre à cette question, on colorie les sommets du graphe d'intersection *$G = (S, A)$ avec $S = \{1, 2, ..., m\}$, et $\{x, y\} \in A$ si $I_x \cap I_y$ est non vide.*

3.6.14. Exemple : problème de l'organisation d'examens
Un certain nombre d'étudiants doit passer des examens. Pour chaque étudiant $\ell \in \{1, ..., s\}$ on connaît la liste des examens E_ℓ qu'il doit passer.

Pour les Examens écrits *il faut tenir de la disponibilité des étudiants : sous l'hypothèse que deux examens sont organisés simultanément (dans la même salle) seulement si aucun étudiant ne passe les deux épreuves, combien de séances d'examen faut-il organiser au moins ? Pour trouver une solution, on colorie les sommets du graphe $G = (S, A)$ avec $S = E_1 \cup E_2 \cup ... \cup E_s$, et $(x, y) \in A$ s'il existe un ℓ avec $x, y \in E_\ell$.*

Pour les Examens oraux *il faut maintenant tenir compte de la disponibilité des étudiants et des professeurs : supposons que chaque étudiant doit être interrogé pendant une heure par le professeur du module suivi. Combien d'heures au moins sont-elles nécessaires ? Pour trouver une solution, on colorie les arêtes du graphe $G = (S_1 \cup S_2, A)$ avec $S_1 = \{1, 2, ..., s\}$, $S_2 = E_1 \cup E_2 \cup ... \cup E_s$, et $\{x, y\} \in A \subset S_1 \times S_2$ si $y \in E_x$.*

3.6.15. Remarques sur l'encadrement du nombre/indice chromatique
Soit $G = (S, A)$ un graphe non orienté, et $d(G)$ le plus grand des degrés des sommets de G. On appelle clique *tout ensemble $C \subset S$ de sorte que le sous-graphe de G engendré par C est complet, c'est-à-dire, $\{x, y\} \in A$ pour tout $x, y \in C$ disjoints. Soit $\omega(G)$ taille maximale d'une clique de G.*

On peut montrer [GoMi95, pp.410-411] que $q(G) \in \{d(G), d(G)+1\}$. Aussi[GoMi95, pp.407-409], nous avons que $\gamma(G) \geq \omega(G)$, et $\gamma(G) \geq m/\omega((S, (S \times S) \setminus A)$.

Le fameux théorème de Appel et Haken (1976) affirme que $\gamma(G) \leq 4$ pour un graphe planaire G (c'est-à-dire, un graphe qui peut être tracé sur une feuille de papier sans intersections d'arêtes).

Trouver le nombre minimal $\gamma(G)$ de couleurs pour colorier des sommets est un problème d'une difficulté comparable au problème du voyageur de commerce. Parfois, par des colorations concrètes on obtient une borne supérieure pour $\gamma(G)$, et les résultats de la remarque 3.6.15 permettent d'affirmer qu'il s'agit bien d'une meilleure coloration.

3.6.16. Théorème
Supposons que l'on dispose d'une énumération $\Sigma = [x_1, ..., x_m]$ des sommets et d'un entier k de sorte que

$$\deg_G(x_1) \geq \deg_G(x_2) \geq ... \geq \deg_G(x_m), \qquad et \quad \forall j \geq k+1 : \quad \deg_G(x_j) \leq k.$$

Alors $\gamma(G) \leq k+1$. En particulier, $\gamma(G) \leq d(G)+1$.

Démonstration. L'algorithme suivant donne un coloriage admissible de sommets et donc une borne supérieure pour $\gamma(G)$.

Initialisation : Ordonner la liste $\Sigma = [x_1, ..., x_m]$ des sommets
de sorte que $deg_G(x_1) \geq deg_G(x_2) \geq ... \geq deg_G(x_m)$
pour tout $x \in L$ faire : $couleur(x) \leftarrow 0$
$nb\_couleur \leftarrow 0$
Itération : Tant qu'il existe encore un sommet x avec $couleur(x) = 0$
$nb\_couleur \leftarrow nb\_couleur + 1$ (nouvelle couleur d'usage)
pour tout $x \in L$ avec couleur$(x) = 0$ faire : $adja(x) \leftarrow faux$
pour $j = 1, 2, ..., m$ faire
Si $couleur(x_j) = 0$ et $adja(x_j) = faux$ alors faire
$couleur(x_j) \leftarrow nb\_couleur$
Pour tout y voisin de x_j faire : $adja(y) \leftarrow vrai$
Invariants : Pendant une itération, les sommets y avec $adja(y) = vrai$ et $couleur(y) = 0$ sont des voisins d'un sommet de couleur $nb\_couleur$, et ne peuvent pas avoir la même couleur.

Montrons que le nombre de couleurs utilisées par notre algorithme est $\leq k+1$. Supposons qu'un sommet y n'était pas colorié par une des premières k couleurs. Comme on commence au début de la liste Σ avec une nouvelle couleur, les sommets $x_1, ..., x_k$ seront coloriés par les premiers k couleurs, et donc $deg_G(y) \leq k$ par hypothèse du théorème. Aussi, comme y n'était pas colorié à l'itération $\ell = 1, ..., k$, il existait un $y_\ell \in L$ voisin de y qui était colorié à l'étape ℓ. Comme on colorie seulement des sommets pas encore coloriés, les $y_1, ..., y_k$ sont distincts. Par conséquent, au début de l'itération d'indice $\ell = k+1$, tous les voisins de y sont déjà coloriés, ce qui implique que, au cours de cette itération, la variable $adja(y)$ ne change pas de valeur. Donc y sera colorié à l'itération $k+1$, d'où $\gamma(G) \leq k+1$. □

3.6.17. Exercice
Montrer que que le nombre chromatique d'un graphe non orienté est égal à deux si et seulement si le graphe est bi-parti.

Donnons finalement une modélisation sous forme d'un programme linéaire en variables entières.

3.6.18. Modélisation du problème de coloration de sommets
Pour écrire le problème de coloration de sommets d'un graphe non orienté $G = (L, J)$, on introduira les variables

$$\forall \ell \in L : \qquad u_\ell \in \{1, ..., m\}, \tag{3.19}$$

avec u_ℓ le numéro de la couleur pour le sommet ℓ. Notre objectif est de minimiser le nombre de couleurs nécessaires, d'où l'objectif classique d'un problème du type minmax

$$\min z, \quad z \in \mathbb{R}, \quad \forall \ell \in L: \quad u_\ell \leq z. \tag{3.20}$$

Il s'y ajoute la contrainte pour deux sommets adjacents

$$\forall j = \{\ell, k\} \in J: \qquad u_\ell \neq u_k$$

qui peut s'écrire comme alternative (sachant que les u_ℓ sont des entiers)

$$\forall j = \{\ell, k\} \in J: \qquad \textit{soit } u_\ell - u_k \leq -1 \textit{ soit } u_k - u_\ell \leq -1,$$

c'est à dire, on obtient une alternative de deux inégalités pour chaque arête. La façon classique de traduire une telle alternative en un système d'inégalités est d'introduire une variable bivalente

$$\forall j = \{\ell, k\} \in J: \quad x_j \in \{0,1\} \quad \textit{et} \quad u_\ell - u_k + (x_j - 1)m \leq -1 \quad \textit{et} \quad u_k - u_\ell - x_j m \leq -1. \tag{3.21}$$

Effectivement, si $x_j = 1$ alors la deuxième contrainte prend la forme souhaitée, et la troisième est trivialement valable, tandis que dans le cas $x_j = 0$ la deuxième contrainte est trivialement valable, et la troisième prend la forme souhaitée. Nous nous retrouvons alors avec un programme linéaire (3.19), (3.20), (3.21) avec m variables astreintes d'être entières, et n variables bivalentes.

Dans la simulation AMPL du fichier `don_ch3/color_f.bat`. nous colorions une carte de la France montrant les différents régions, de sorte que deux régions adjacents n'aient pas la même couleur. Le graphe sous-jacent (L, J) (avec L l'ensemble des régions et $(\ell, k) \in J$ si les régions ℓ et k sont adjacents) est un graphe planaire, et nous savons déja que 4 couleurs sont suffisant. La simulation montre que 4 couleurs sont aussi nécessaire.

3.7 Programmes linéaires en nombres entiers

On se pose le problème de résoudre

$$v(P \cap Q) := \max\{f\, x : x \in P \cap Q\} \tag{3.22}$$

avec $v(P)$ calculable pour une certaine classe $\mathcal{P}$ de domaines P (par exemple les polyèdres où on peut utiliser Simplex), et l'ensemble Q traduisant des contraintes "compliquées", voir les exemples ci-dessous. On supposera également que l'on dispose d'un sur-estimateur

$$\forall P \in \mathcal{P}: \quad g(P) \geq v(P), \tag{3.23}$$

dont l'évaluation de g ne demande pas beaucoup d'effort (par exemple prédire $v(P)$ à l'aide des variables marginales, voir 2.5.12). Finalement, pour simplifier on supposera que $P \cap Q$ est fini mais de très grande taille, ce qui rend impossible l'idée d'énumérer les éléments de $P \cap Q$ et de comparer leur valeur. Donnons deux exemples.

3.7.1. Exemple : Ordonnancement avec contraintes disjonctives
Pour un problème d'ordonnancement comme au 3.3.4, l'ensemble P peut représenter un ensemble de planifications respectant les contraintes de succession comme décrites au 3.3.4, avec $-v(P)$ représentant la longueur du projet calculé par un algorithme de plus longue chemin, voir 3.3.4. L'ensemble Q peut traduire des contraintes disjonctives : certaines tâches sont faites par la même personne, et donc ne peuvent pas être effectuées simultanément, mais on n'impose pas l'ordre dans lequel ces tâches doivent être effectuées. [13].

3.7.2. Exemple : PLNE, PLNB
Dans des applications, certaines variables dans des programmes linéaires peuvent représenter un nombre d'objets indivisibles, et devraient donc être astreintes d'appartenir à $\mathbb{Z}$. Dans ce cas, pour A, a fixé, considérons comme $\mathcal{P}$ la classe des polyèdres à bornes variables

$$P = P(b, c) = \{x \in \mathbb{R}^n : Ax = a, x \geq b, x \leq c\}.$$

Les contraintes d'intégrité seront représentées par $Q = \mathbb{Z}^n$ (ici on parle des programmes linéaires en variables entières ou abrégé PLNE) où par $Q = \{0, 1\}^n$ (dans ce cas on parle des programmes linéaires en variables bivalentes ou abrégé PLNB) ou par une combinaison de la forme $Q = \mathbb{Z}^p \times \mathbb{R}^q$ (où on parle des programmes linéaires mixtes).

Nous avons déjà vu dans l'exemple 3.6.8 de couplage parfait de valeur minimale, dans l'exemple 3.6.11 du voyageur de commerce et dans l'exemple 3.6.18 de coloration de sommets que l'utilisation des variables bivalentes peut être extrêmement utile dans la modélisation, aussi dans le contexte suivant : si la contrainte $A_\ell x \leq a_\ell$ traduit le fait que le temps de production sur une machine est limité par a_ℓ, la contrainte $A_\ell x \leq a_\ell + \gamma z$ avec $z \in \{0, 1\}$ permet d'envisager d'acheter une deuxième machine, ce qui au cas $z = 1$ augmente le temps disponible. Les variables bivalentes traduisent donc souvent une décision "oui/non".

Un autre exemple d'un PLNE/PLNB est donné par

3.7.3. Exemple : le problème du sac-à-dos
On cherche à remplir son sac-à-dos de taille limité $a \in \mathbb{R}$, disposant des objets $j = 1, ..., n$, et en sachant que l'objet j admet un poids $A^j > 0$ et est d'une utilité $f^j \in \mathbb{R}$. On se ramène au problème (dit problème du sac-à-dos en variables bivalentes)

$$\max\{f^1 x_1 + ... + f^n x_n : A^1 x_1 + ... + A^n x_n \leq a, \forall j : x_j \in \{0, 1\}\}.$$

Dans le cas $x_j \in \{0, 1, 2, ...\}$ on parle d'un problème du sac-à-dos en variables entières.

Considérons de plus près l'exemple

$$\max\{10x_1 + 11x_2 : 10x_1 + 12x_2 \leq 59, x_1, x_2 \geq 0 \text{ entiers } \}$$

pour mieux cerner la difficulté de la contrainte $x \in \mathbb{Z}^n$. Nous avons tracé le polyèdre ainsi que les 21 éléments à coefficients entiers dans la Figure 3.7. Si on néglige les contraintes d'intégrité, la valeur optimale est 59, atteinte au point $x = (5.9, 0)^t \notin \mathbb{Z}^2$. Le point le plus proche dans $\mathbb{Z}^2$

13. Ces contraintes disjonctives donnent lieu à une alternative entre deux inégalités, qui comme dans (3.21) peuvent être transformés en un système de deux inégalités à l'aide d'une variable bivalente

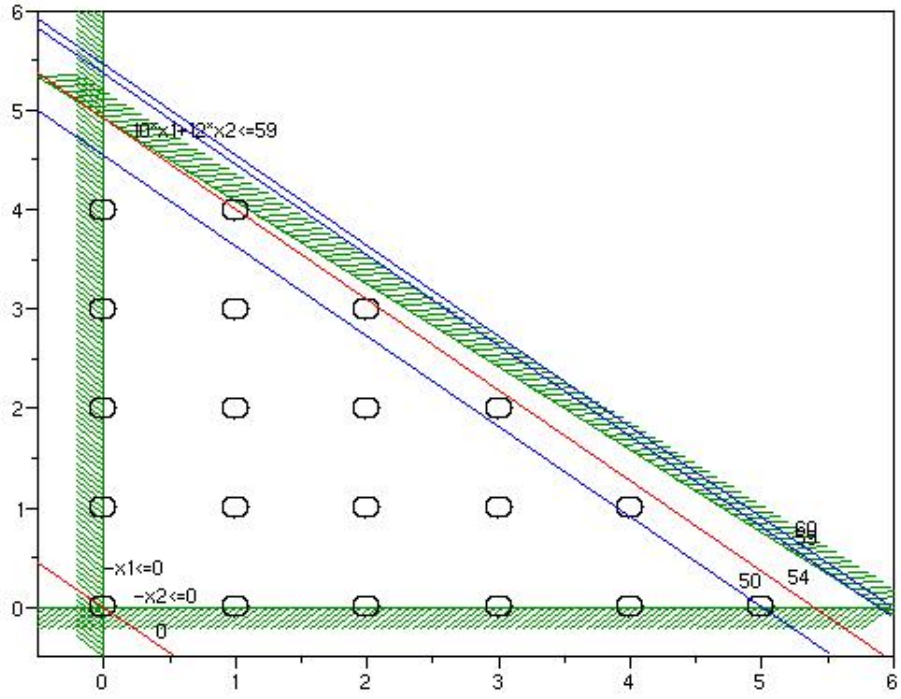
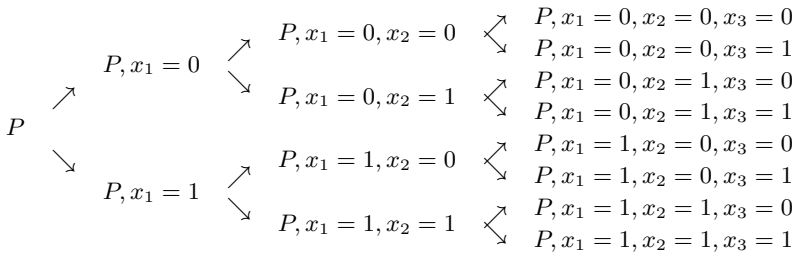


FIGURE 3.7 – RÉSOLUTION GRAPHIQUE POUR L'EXEMPLE DU SAC-À-DOS

obtenu par arrondi n'est pas réalisable. Le point réalisable le plus proche de x avec coordonnées $(5,0)^t$ nous donne une valeur 50, mais la solution optimale de notre problème est donnée par $x' = (1,4)^t$ (de valeur 54), ce qui montre clairement que la solution du problème continu peut être "loin" de la solution recherchée.

La résolution efficace des PLNE/PLNB ou des programmes en variables mixtes est sujet d'un nombre important de méthodes, voir par exemple [GoMi95, Annexe 2] pour un bref résumé. Ici on exposera seulement la méthode "Séparation par évaluation" (en anglais "Branch & Bound"), une approche assez générale qui se généralise facilement aux problèmes du type (3.22), par exemple aux problèmes d'ordonnancement comme dans 3.7.1.

Une idée peut consister à fixer quelques variables (cas des variables bivalentes) et d'adapter les autres au mieux. Si on utilise cette idée récursivement, on obtient le schéma suivant pour trois variables bivalentes



Un schéma similaire est obtenu pour les variables astreintes d'être entières, si on sépare suivant les inégalités du type $x_j \leq \nu$, $x_j \geq \nu + 1$. Il est évident que, pour un nombre plus important de variables, il est généralement impossible de construire toute l'arborescence.

Dans le "Branch & Bound" on explore l'arborescence ci-dessus, mais on adapte une stratégie qui devrait permettre l'élaguer un grand nombre de branches (ce qui constitue effectivement le pari du Branch & Bound). En chaque itération

- une *fonction de choix* nous dit quel sous-domaine P' il faut traiter ;
- une *procédure d'évaluation* nous dit quoi faire :
  - si le sous-problème n'est pas prometteur on peut définitivement *élaguer* la branche de l'aborescence ;
  - certains sous-problèmes nous donnent des candidats pour la solution optimale du problème initial, sans devoir explorer de plus près cette branche. Par conséquent, on peut également *élaguer* la branche de l'aborescence ;
  - pour les sous-problèmes restants, une *fonction de séparation* nous dit comment couper P' en deux : on créera deux nouvelles feuilles P_1 et P_2 dans l'arborescence.

Cependant, il faudra bien savoir que nos stratégies développées ci-dessous seront en grande partie de nature heuristique, et s'adaptent plus au moins bien selon la nature du problème...

3.7.4. La forme primitive de la méthode BB

Invariants : *on dispose d'un candidat $x^c \in Q \cap P$ et de sa valeur $F = f\,x^c$;*
on dispose d'une pile $J = \{P_1, P_2 ..., P_\ell\}$ de polyèdres $P_j \subset P$ pas encore évalués.
Initialisation : $J \leftarrow \{P\}$, $F \leftarrow -\infty$
Itération : *on retire de la pile $P' \in J$ (fonction de* choix*, peut utiliser g)*

evaluer $v(P') = f\, x^{P'}$.
Si $v(P') \leq F$ *(inclus* $P' = \emptyset$*) alors*
ici $\forall\, x \in P' \cap Q : f\, x \leq v(P') \leq F$. *Branche pas prometteuse, on peut l'élaguer.*
Si $v(P') > F$ *et* $x^{P'} \in Q$ *(par* chance *ou parce que* P' *est "petit") alors*
ici $v(P') = v(Q \cap P')$. *On pose* $x^c \leftarrow x^{P'}$, $F \leftarrow v(P')$.
Branche complètement explorée, on peut l'élaguer.
Si $v(P') > F$ *et* $x^{P'} \notin Q$ *alors*
création de deux fils P'_1, P'_2 *dans l'arborescence :*
La fonction de séparation *nous donne un indice* t *(en fonction de* $x^{P'}$
et l'évaluation rapide g*) et on range dans la pile les deux fils*
$$P'_1 = \{x \in P' : x_t \leq \lfloor x_t^{P'} \rfloor\},\ P'_2 = \{x \in P' : x_t \geq \lfloor x_t^{P'} \rfloor + 1\}.$$

Rappelons que, pour les PLNE et PLNB, l'étape d'évaluation (le calcul de $v(P')$) consiste à résoudre par Simplex le programme associé au polyèdre P' obtenu en supprimant les contraintes d'intégrité. Plus précisément, en passant du père au fils dans l'aborescence, on a seulement changé quelques bornes pour les variables, et c'est pour cela que l'algorithme Simplex dual est particulièrement adapté.

Dans le cas des variables bivalentes on risque de créer beaucoup de fils non réalisables. Ici on peut parfois accélérer l'algorithme en ajoutant avant l'évaluation de $v(P')$ un test basé sur (3.23) : si $g(P') \leq F$ alors la branche n'est pas prometteuse et on l'élague. Notons également que, pour les variables bivalentes, on fixe la composante x_t pour les fils.

Nous revenons ultérieurement en détail sur les procédures de *choix* et de *séparation*, considérons d'abord un exemple où comme t on prendra un indice quelconque ou $x_t^{P'}$ est non entier mais dans le problème d'origine la variable x_t est soumise à une contrainte d'intégrité. Aussi, on retira de la pile un élément quelconque à notre convenance.

3.7.5. Exemple

Considérons l'exemple

$$\begin{cases} \max(2x_1 + x_2) & \\ x_1 - 4x_2 \leq 0, 3x_1 + 4x_2 \leq 15, x_1, x_2 \geq 0 & \textit{(polyèdre } P\textit{)} \\ x_1, x_2 \in \mathbb{Z} & \textit{(contraintes } Q = \mathbb{Z}^2\textit{).} \end{cases}$$

On trouve la résolution graphique de chacun des sous-problèmes en Figure 3.8 : la première itération consiste à retirer de la pile le polyèdre $P = P_1 = P((0,0)^t, (+\infty, +\infty))$ *de départ, on trouve comme solution* $x^{P_1} = (\frac{15}{4}, \frac{15}{16})^t$, *et valeur optimale* $v(P_1) = \frac{135}{16} \approx 8.4$. *Comme actuellement* $F = -\infty$ *et* $x_1^{P_1} \notin \mathbb{Z}$, *le choix* $t = 1$ *(fonction de séparation) donne lieu aux fils*

$$P_2 = \{x \in P_1 : x_1 \leq 3\} = P(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 3 \\ +\infty \end{bmatrix}), \quad P_3 = \{x \in P_1 : x_1 \geq 4\} = P(\begin{bmatrix} 4 \\ 0 \end{bmatrix}, \begin{bmatrix} +\infty \\ +\infty \end{bmatrix}).$$

Poursuivons avec le polyèdre P_2 *(fonction de choix), avec* $x^{P_2} = (3, \frac{3}{2})^t$, *et valeur optimale* $v(P_2) = \frac{15}{2} = 7.5$. *Ici* $x_1^{P_2} \in \mathbb{Z}$ *mais* $x_2^{P_2} \notin \mathbb{Z}$, *d'où le choix* $t = 2$, *ce qui donne lieu aux fils*

$$P_4 = \{x \in P_2 : x_2 \leq 2\} = P(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 3 \\ 2 \end{bmatrix}), \quad P_5 = \{x \in P_2 : x_2 \geq 3\} = P(\begin{bmatrix} 0 \\ 3 \end{bmatrix}, \begin{bmatrix} 3 \\ +\infty \end{bmatrix}).$$

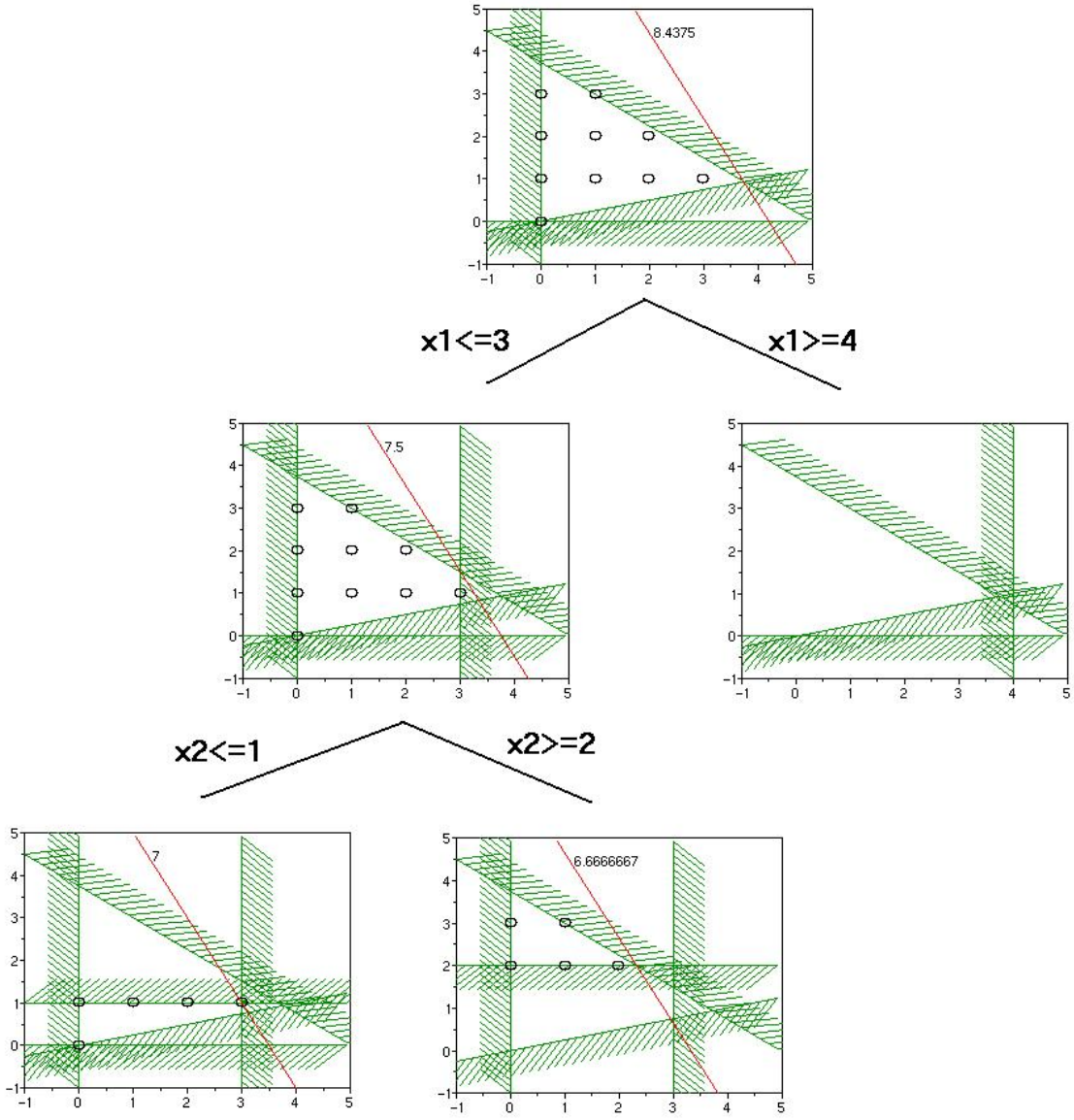


FIGURE 3.8 – RÉSOLUTION GRAPHIQUE POUR L'EXEMPLE 3.7.5

Ensuite traitons le polyèdre P_4, avec $x^{P_4} = (3,1)^t \in \mathbb{Z}^2$, et valeur optimale $v(P_4) = 7$. On mettra à jour $F = 7$ et $x^c = x^{P_4}$, et on élague. On poursuit avec le polyèdre P_5, avec $x^{P_5} = (\frac{7}{3}, 2)^t$, et valeur optimale $v(P_5) = \frac{20}{3} \approx 6.6$. Comme heureusement $v(P_5) \leq F = 7$, cette branche n'est pas prometteuse et on peut l'élaguer.

Le seul polyèdre restant sur la pile est le polyèdre P_3, et nous trouvons que P_3 est vide. Par conséquent, cette branche n'est pas prometteuse et on peut l'élaguer. Comme la pile est maintenant vide, on peut arrêter l'algorithme et trouve comme solution optimale du problème initial la dernière solution candidate : $x^C = x^{P_4} = (3,1)^t$.

Dans l'exemple 3.7.5 nous sommes particulièrement chanceux, car on a eu besoin de créer des fils seulement à deux reprises. Notons que la complexité de la méthode Branch & Bound dépend largement de la fonction de choix et de la fonction de séparation : par exemple, si on avait pris dans l'exemple 3.7.5 l'indice $t = 2$ au moment de la séparation de P_1 en deux sous-problèmes, on aurait du traiter en total 7 au lieu de 5 sous-problèmes (on aurait tranché deux fois sur la variable x_2).

3.7.6. Les fonctions de choix

On retire le candidat P' au début de la pile, qui est organisée comme suit :

- *"FIFO" (=first in first out=premier arrivé premier servi) : on ajoute les nouveaux polyèdres à la fin de la liste. Ceci signifie de parcourir l'arborescence en largeur, une approche peut-être plus facilement parallelisable (il suffit d'échanger de temps en temps les valeurs actuelles de F).*
- *"FILO" (=first in last out=premier arrivé dernier servi) : on ajoute les nouveaux polyèdres au début de la liste, ce qui fait que l'on parcourt l'arborescence en profondeur. Ceci présente l'avantage de créer relativement vite une valeur F fini, en espérant que l'on puisse élaguer les autres branches si on les examine "tard" (ce qui a été le cas dans l'exemple 3.7.5 où on avait appliqué la fonction de choix "FILO").*
- *"best" : on trie les polyèdres en attente, par ordre décroissant suivant leur évaluation rapide g, avec comme idée (heuristique) qu'un sous-problème P' avec $g(P')$ élevé est particulièrement prometteur pour obtenir une valeur F élevée (ce qui permet d'élaguer les autres branches en attente).*

Bien entendu, les approches décrites ci-dessus peuvent être combinées. La dernière approche nécessite de spécifier notre évaluation rapide. Notons que pour les PLNE il est trop coûteux de résoudre explicitement chaque fils avant de le ranger dans la pile (choix $g(P) = v(P)$). Un premier choix pour g respectant (3.23) peut être de prendre la valeur optimale du père (car pour créer un fils on a ajouté une contrainte). Un choix un peu plus sophistiqué basé sur la post-optimisation ainsi que les différents fonctions de séparation sont exposés ci-dessous.

3.7.7. Les fonctions de séparation en base

Notons par $\mathcal{T}$ l'ensemble des indices des variables astreintes d'être entières. Par construction, les bornes $b_{\mathcal{T}}$, $c_{\mathcal{T}}$ des différents sous-problèmes sont à coefficients entiers, et on dispose d'une solution optimale du père P' ayant une représentation comme point de base réalisable vérifiant la CSO

$$x^{P'} = x(I, B_-, B_+).$$

On cherchera alors l'indice de séparation t dans $I \cap \mathcal{T}$. Suivant le pricing classique de Simplex,

dans la stratégie "bland" on prendra comme t le plus petit indice dans $I \cap \mathcal{T}$ avec $x_t^{P'} \notin \mathbb{Z}$, et pour "mvp" on prendra comme t l'indice dans $I \cap \mathcal{T}$ avec $x_t^{P'}$ le plus éloigné de $\mathbb{Z}$.

Ensuite, pour le premier fils P_1' on changera $c_t \leftarrow \lfloor x_t^{P'} \rfloor$, et pour le deuxième fils P_2' on changera $b_t \leftarrow \lfloor x_t^{P'} \rfloor + 1$. Dans les deux cas, la base finale du père continue de vérifier la CSO, mais n'est plus réalisable. Il est alors intéressant de ranger ensemble avec les fils certaines informations sur le père dans la pile (comme la base et l'inverse $(A^I)^{-1}$) pour pouvoir utiliser la méthode 2.5.17 de Simplex dual aux variables doublement bornées dans l'évaluation des fils (c'est-à-dire, le calcul de $v(P_1')$ ou $v(P_2')$), voir l'exemple 3.7.8 ci-dessous. Avant de ranger les fils dans la pile, il est même envisageable d'obtenir une première évaluation rapide des fils par une itération de Simplex dual (rappelons que (3.23) sera valable car les valeurs des itérés successifs dans Simplex dual sont décroissantes). Avec les notations de 2.5.17 et $t = r$, nous obtenons

$$g(P_1') = \begin{cases} -\infty & \text{si } J_-(r) = \emptyset \text{ (car le polyèdre du fils est vide),} \\ v(P') + [x_r^{P'} - c_r]\frac{d(I)^s}{T(I)_r^s} & \text{avec } s = s_-(r) \in J_-(r) \text{ sinon,} \end{cases} \tag{3.24}$$

et

$$g(P_2') = \begin{cases} -\infty & \text{si } J_+(r) = \emptyset \text{ (car le polyèdre du fils est vide),} \\ v(P') + [x_r^{P'} - b_r]\frac{d(I)^s}{T(I)_r^s} & \text{avec } s = s_+(r) \in J_+(r) \text{ sinon,} \end{cases} \tag{3.25}$$

où nous rappelons que pour cette évaluation rapide nous avons seulement besoin de calculer en plus la ligne $T(I)_r$ pour la dernière base du père.

Pour la fonction de séparation "peb" des pénalités en base, on cherche l'indice $t \in \mathcal{T} \cap I$ avec $x_t^{P'} \notin \mathbb{Z}$ maximisant $|v(P_1') - v(P_2')|$, pour pouvoir vite élaguer le fils P_j' avec "petit" $v(P_j')$. L'écart entre les valeurs optimales des fils potentiels étant trop coûteux à calculer, on se contente de maximiser l'écart entre les sur-estimations données ci-dessus : d'abord on cherche un $t = r \in I \cap \mathcal{T}$ avec $x_r^{P'} \notin \mathbb{Z}$ et $J_+(r) = \emptyset$ ou $J_-(r) = \emptyset$ (ce qui nécessite de calculer au préalable la ligne $T(I)_r$ pour tout indice candidat r). Si un tel indice n'existe pas, on prend l'indice $t = r$ maximisant

$$|[x_r^{P'} + 1 - \lfloor x_r^{P'} \rfloor]\frac{d(I)^{s_+(r)}}{T(I)_r^{s_+(r)}} - [x_r^{P'} - \lfloor x_r^{P'} \rfloor]\frac{d(I)^{s_-(r)}}{T(I)_r^{s_-(r)}}|.$$

Cette quantité est une sorte de pénalité pour tout indice potentiel r : on prend l'indice le plus prometteur.

3.7.8. Suite de l'exemple 3.7.5
Pour l'exemple 3.7.5

$$\max\{2x_1 + x_2 : x_1 - 4x_2 \leq 0, 3x_1 + 4x_2 \leq 15, x_1, x_2 \geq 0, x_1, x_2 \in \mathbb{Z}\},$$

illustrons l'utilisation de l'algorithme 2.5.17 de Simplex dual aux variables doublement bornées. Ici

$$(A|a) = \left[\begin{array}{cccc|c} 1 & -4 & 1 & 0 & 0 \\ 3 & 4 & 0 & 1 & 15 \end{array}\right], \quad f = (2, 1, 0, 0),$$

et on démarre avec $b = (0, 0, 0, 0)^t$, $c = (+\infty, +\infty, +\infty, +\infty)$. La base finale du Simplex ordinaire aux variables doublement bornées pour P_1 est

$$(I, B_-, B_+) = (\{1, 2\}, \{3, 4\}, \{\}),\ x(I, B_-, B_+) = (\frac{15}{4}, \frac{15}{16}, 0, 0)^t,\ d(I) = (0, 0, -\frac{5}{16}, -\frac{9}{16}).$$

Pour le premier fils P_2 on a décidé d'ajouter la contrainte $x_1 \leq 3$ (fonction de séparation "bland" ou "mvp"), avec évaluation rapide

$$g(P_2) = v(P_1) + (\frac{15}{4} - 3)\frac{-5/16}{1/4} = \frac{135}{16} - \frac{15}{16} = \frac{15}{2}$$

obtenue par (3.24). Pour ce fils P_2 nous obtenons la nouvelle borne sup $c = (3, +\infty, +\infty, +\infty)^t$. On choisit $r = 1$ car $x_r > c_r$, et alors

$$T(I)_r = (1, 0, \frac{1}{4}, \frac{1}{4}).$$

Notons que $J_-(r) = \{3, 4\}$, le plus petit des rapports (en module) étant atteint pour $s_-(r) = 3$. Une itération de SIMPLEX dual donne

$$(I, B_-, B_+) = (\{2, 3\}, \{4\}, \{1\}),\ x(I, B_-, B_+) = (3, \frac{3}{2}, 3, 0)^t,\ d(I) = (\frac{5}{4}, 0, 0, -\frac{1}{4}),$$

vérifiant bien la CSO, mais étant aussi réalisable, donc on connaît une solution optimale de ce fils P_2. Nous observons pour ce polyèdre que $g(P_2) = v(P_2)$, car une itération de Simplex dual a été suffisante.

Au lieu de continuer notre algorithme avec les petits fils P_4, P_5, régardons le frère P_3 avec bornes $b = (4, 0, 0, 0)^t$, $c = (+\infty, +\infty, +\infty, +\infty)$. On reprend avec la base finale de la racine P_1

$$(I, B_-, B_+) = (\{1, 2\}, \{3, 4\}, \{\}),\ x(I, B_-, B_+) = (\frac{15}{4}, \frac{15}{16}, 0, 0)^t,\ d(I) = (0, 0, -\frac{5}{16}, -\frac{9}{16}),$$

et alors $r = 1$ car $x_r < b_r$. Pourtant, $J_+(r) = \{\}$, et donc on ne peut pas choisir $s_+(r)$: le polyèdre de ce fils est vide (et d'ailleurs aussi $g(P_3) = -\infty$ d'après (3.25)) !

La fonction de séparation "peb" semble être prometteuse mais trop coûteuse, car elle nécessite le calcul de la ligne $T(I)_r$ du tableau simplicial du père pour tout indice r candidat. Un bon compromis peut être la stratégie suivante.

3.7.9. Séparation par pénalités hors base ("phb")
Ici on supposera que (avec $\mathcal{T}$ comme dans 3.7.7 désignant l'ensemble des indices avec contrainte d'intégrité)

$$x^{P'} = x(I, B_-, B_+), \quad \textit{avec}\ \ x^{P'}_{\mathcal{T}\cap B_-} = b_{\mathcal{T}\cap B_-},\ x^{P'}_{\mathcal{T}\cap B_+} = c_{\mathcal{T}\cap B_+}\ \textit{à composantes entières.}$$

On envisage de créer les fils par le choix $t \in (B_- \cup B_+) \cap \mathcal{T}$, c'est-à-dire, $x^{P'}_t$ est déjà entier et un des fils aura la même solution optimale que le père. Plus précisément, les fils dans le cas $t \in B_- \cap \mathcal{T}$ auront la forme

$$P'_1 = \{x \in P' : x_t \leq b_t (= x^{P'}_t)\}, \quad P'_2 = \{x \in P' : x_t \geq b_t + 1 (= x^{P'}_t + 1)\},$$

et dans le cas $t \in B_+ \cap \mathcal{T}$

$$P'_1 = \{x \in P' : x_t \geq c_t (= x^{P'}_t)\}, \quad P'_2 = \{x \in P' : x_t \leq c_t - 1 (= x^{P'}_t - 1)\},$$

ce qui implique que P'_1 est la partie du polyèdre père P' où on fixe la valeur de la variable x_t, et la base finale du père est une base pour P'_2 qui vérifie la CSO mais n'est probablement pas

réalisable. Effectivement, on obtient une formule simple pour le point de base de ce fils P_2 en fonction du point de base du père : dans le cas $t \in B_-$ il faudra augmenter la t-ième composante du point de base par 1 sans changer les autres composantes hors base, et dans le cas $t \in B_+$ il faudra diminuer la t-ième composante par 1 sans changer les autres composantes hors base. Des tels changements du point de base sont réalisés par la direction privilégiée $v(I)^t$ d'indice t (voir 2.3.5), d'où la formule $x^{P'} + v(I)^t$ pour le point de base du fils P_2. Nous obtenons les changements de valeur

$$v(P'_1) = g(P'_1) := v(P'), \quad v(P'_2) \leq g(P'_2) := v(P') + fv(I)^t = v(P') - |d(I)^t|.$$

Le choix $t = s \in B_- \cup B_+$ maximisant $|d(I)^s|$ parmi les indices dans $\mathcal{T}$ vérifiant $b_s < c_s$ nous donnera alors une possibilité très peu coûteuse de créer un écart important entre les deux fils, sachant que l'on créera dans l'arborescence du Branch & Bound deux fils dont un qui est évalué sans aucun effort.

3.7.10. Exemple : Simulations pour un problème de sac-à-dos
Considérons le problème de sac-à-dos en variables entières

$$\begin{cases} \max(8x_1 + 12x_2 + 3x_3 + 6x_4) \\ 6x_1 + x_2 + 2x_3 + 3x_4 \leq 8.5, x_1, x_2, x_3, x_4 \in [0, 4], \\ x_1, x_2, x_3, x_4 \in \mathbb{Z} \end{cases}$$

On introduira une variable d'écart $x_5 \in [0, +\infty)$ pas soumise à une contrainte d'intégrité, de sorte que, pour le problème initial,

$$\mathcal{T} = \{1, 2, 3, 4\}, \quad b = (0, 0, 0, 0, 0)^t, \quad c = (4, 4, 4, 4, +\infty).$$

Pour obtenir une première base pour le problème de départ sans contraintes d'intégrité, on a utilisé Simplex en deux phases, qui donne lieu à

$$x^P = x(I, B_-, B_+) = (0, 4, 0, \frac{3}{2}, 0)^t.$$

Sur les arborescences ci-dessous on note sur les sommets les valeurs $-v(P')$ d'un sous-problème P', et sur les arcs un triplet, dont le premier élément nous donne la nouvelle contrainte, le deuxième l'indice de création du sous-problème associé au sommet but, et le troisième l'ordre dans lequel on a traité ce sous-problème. On effectuera des simulations à l'aide des fichiers

- `don_ch3/7_10a.bat` *créant l'arborescence pour fonction de choix "fifo" et fonction de séparation "mvp". On trouve une arborescence assez importante, et comme solution optimale $(x_1, x_2, x_3, x_4) = (0, 4, 0, 1)$ (ce qui est normal : les objets 2 et 4 ont clairement un plus grand interêt exprimé par le rapport f^j/A_1^j).*
- `don_ch3/7_10b.bat` *créant l'arborescence pour fonction de choix "filo" et fonction de séparation "mvp". On note que "bland"ou "peb" aurait donné le même résultat, car il y a un seul indice en base. On trouve déjà en 4ième itération une valeur référence finie $F = 54$ (et, par chance, il s'agit de la solution optimale).*
- `don_ch3/7_10c.bat` *créant l'arborescence pour fonction de choix "filo" et fonction de séparation "phb". On trouve à l'itération 8 un premier candidat à coefficients entiers, mais sa valeur n'est pas très intéressante. La solution optimale est trouvée à l'itération 52.*

— `don_ch3/7_10d.bat` *créant l'arborescence pour fonction de choix "fifo" et fonction de séparation "phb". Contrairement à "mvp", pour cet exemple "fifo" est plus performant que "filo", mais c'est peut-être un hasard.*
— `don_ch3/7_10e.bat` *créant l'arborescence pour fonction de choix "best" et fonction de séparation "phb". L'ordre décroissant de la pile suivant l'évaluation rapide semble être le plus prometteur, surtout si on tient compte du fait que, pour "phb", l'evaluation d'une grande partie des fils ne demande aucun effort.*

3.7.11. Exemple : retour au problème du voyageur de commerce

Le problème du voyageur de commerce orienté pour le graphe (L, J) orienté à m sommets et n arcs (voir 1.2.3 et 3.6.10) à été écrit dans 3.6.11 comme un programme linéaire mixte formé par les contraintes (3.16), (3.17) et (3.18). Après ajout d'un certain nombre de variables d'écart non négatives mais pas forcément entiers pour (3.17) on obtient un programme mixte sous forme standard à $\approx 2m+n$ contraintes et $\approx m+2n$ variables (dont n bivalentes et m astreintes d'être entières). Pour compléter le programme linéaire, il nous faut encore les coûts unitaires f^j dans l'objectif

$$\min(\sum_{j\in J} f^j x_j).$$

Dans les simulations ci-dessous on prendra pour f^j la distance euclidienne entre les extrémités d'un arc j, et un graphe complet sans boucles $J = \{(\ell, k) : \ell, k \in L, \ell \neq k\}$, à cinq sommets. On effectuera des simulations à l'aide des fichiers

— `don_ch3/7_11a.bat` *créant l'aborescence pour fonction de choix "fifo" et fonction de séparation "mvp". On trouve une arborescence assez importante (49 itérations), et la solution optimale à la 7ième itération (on est chanceux, car ceci permet d'élaguer). D'ailleurs, la simulation* `don_ch3/7_11b.bat` *en "filo" donne la même arborescence, parcouru dans un autre ordre.*
— `don_ch3/7_11c.bat` *créant l'aborescence pour fonction de choix "filo" et fonction de séparation "peb". On trouve la solution optimale comme deuxième solution à coefficients entiers à la 10ième itération (sur 13). D'ailleurs, la simulation* `don_ch3/7_11d.bat` *en "fifo" donne la même arborescence (aussi pour "best"), parcouru dans un autre ordre.*
— *Finalement, pour comparaison, nous donnons dans* `don_ch3/7_11e.bat` *une simulation pour fonction de choix "best" et fonction de séparation "phb". Ici il nous faut 31 itérations, mais pour 12 polyèdres on peut évaluer sans aucun effort (c'est-à-dire, sans application de Simplex dual) vu que la solution optimale du père et du fils coïncident.*

On observe que, même si un problème de voyageur de commerce à cinq villes ne paraît pas particulièrement difficile à résoudre, notre calcul devient déjà assez complexe. Les boîtes noires industrielles comme CPLEX ont d'ailleurs d'autres stratégies plus performantes dans leur Branch & Bound, comme on peut observer sous AMPL (7 villes) avec le fichier `don_ch3/voyag2.bat`.

3.7.12. Exercice : Problème du sac à dos ("knapsack") en variables doublement bornées

On considère le problème suivant :

$$\begin{cases} \max f.x \\ Ax \leq a, b \leq x \leq c \end{cases}$$

avec $f = (f^1, \cdots, f^n)$, $a = (a_1)$, $A = (A_1^1, \cdots, A_1^n)$, $b = \begin{pmatrix} b_1 \\ \vdots \\ b_n \end{pmatrix}, c = \begin{pmatrix} c_1 \\ \vdots \\ c_n \end{pmatrix}$.

On suppose, sans perte de généralité, que

$$A_1^k > 0 \quad \forall k \text{ et } \frac{f^1}{A_1^1} \geq \frac{f^2}{A_2^1} \geq \cdots \geq \frac{f^n}{A_n^1}.$$

1. *Donner un algorithme de résolution de ce problème.*
2. *Application numérique : On désire stocker dans une cuve un mélange de 4 pétroles bruts à teneurs diverses en composés azotés et en soufre :*

	P_1	P_2	P_3	P_4
soufre	0.3	4	3	1
comp. azotés	0.1	0.9	0.6	0.3

(quantités en pourcentage)

L'utilisation prévue pour ce stock nécessite un mélange le plus riche possible en azote mais ne contenant pas plus de 1.5 tonnes de soufre. D'autre part on ne dispose que de 30 tonnes de P_1 et de 35 tonnes de P_4.

Quelle quantité de chaque pétrole doit-on charger dans la cuve ?

3. *Utiliser la méthode branch and bound pour la résolution du problème du sac à dos en variables bivalentes suivant*

$$\max\{(20x_1 + 16x_2 + 11x_3 + 9x_4 + 7x_5 + x_6), x \in S\} \text{ avec}$$

$$S = \{9x_1 + 8x_2 + 6x_3 + 5x_4 + 4x_5 + x_6 \leq 12, x_i \in \{0,1\}, i = 1, \cdots, 6\}$$

3.7.13. Exercice : Transport d'appareils photographiques

La compagnie japonaise FOTOKA exporte par avion des appareils photo. Le chargement considéré ici ne peut peser plus de 330 kilogrammes et comporte quatre types de caisses. Dans les plus grosses, on peut ranger 180 appareils ; le poids total de la caisse est alors de 150 kilogrammes. Dans les autres on peut placer 140, 80 ou 40 appareils pour des poids respectifs de 120, 70 et 40 Kg. On n'expédie que des caisses pleines.
Il faut étudier le nombre de caisses de chaque type à expédier pour que le nombre total d'appareils photos livrés soit maximal.

1. *Formaliser mathématiquement le problème. Donner une borne supérieure de chacune des variables.*
2. *Résoudre le problème par une méthode de "Branch and Bound".*

3.7.14. Exercice : Problème d'affectation

n personnes doivent être affectées à n travaux, chaque personne devant effectuer une et une seule tâche. Le coût de formation de la k-ème personne lorsqu'elle est affectée au l-ème travail est d_{kl}. Il s'agit d'affecter les personnes aux travaux de sorte que la somme des coûts de formation soit minimum.

1. *Formuler ce problème comme un programme linéaire en variables bivalentes.*

2. On considère le cas particulier suivant :

ouvriers	*Tâches*			
	1	*2*	*3*	*4*
1	*8*	*3*	*1*	*5*
2	*11*	*7*	*1*	*6*
3	*7*	*8*	*6*	*8*
4	*11*	*6*	*4*	*9*

Déterminer l'affectation des 4 ouvriers aux 4 tâches qui minimise le coût de formation, en utilisant une méthode d'exploration arborescente.

3.7.15. Exercice : Choix de sites pour des entrepôts

Une entreprise de distribution envisage l'implantation de m nouveaux entrepôts. n sites ($n > m$) ont été envisagés pour leur implantation et, pour chaque site s_i, on a estimé la capacité de stockage C_i de l'entrepôt que l'on pourrait construire sur ce site.

L'entreprise cherche à déterminer les n sites sur lesquels elle doit construire ses nouveaux entrepôts de façon à maximiser la capacité totale de stockage. Mais elle désire éviter à tout prix d'avoir deux entrepôts situés à plus de 30 Km l'un de l'autre car elle effectue de très nombreux transports entre tous ses entrepôts . (Les n sites qui ont été envisagés sont toujours situés à moins de 30 Km de tous les entrepôts déjà construits.)

Formuler, par un programme linéaires en variables booléennes, et de trois façons différentes, le problème du choix des sites.

3.7.16. *Soit à résoudre par une méthode de "Branch and Bound" le programme linéraire en nombres entiers*

$$\max\{c.x, \quad x \in S\} \text{ avec } S = \{x : Ax = b, x_j \in \mathbb{N}, j = 1, \cdots, n\}$$

On suppose que S est fini. On considère une méthode arborescente dont les sommets sont les sous-ensembles de S de la forme

$$S_{\alpha,\beta} = \{x \in S : \quad \alpha \leq x \leq \beta\}$$

auxquels on associe les problèmes $(\mathcal{P}_{\alpha,\beta})$ et $(\mathcal{P}'_{\alpha,\beta})$ suivants :

$$(\mathcal{P}_{\alpha,\beta}) \quad \max\{c.x, \quad \alpha \leq x \leq \beta, \quad x_j \in \mathbb{N}, j = 1, \cdots n\}$$

$$(\mathcal{P}'_{\alpha,\beta}) \quad \max\{c.x, \quad \alpha \leq x \leq \beta\}$$

On initialise $\alpha_j = 0, \beta_j = +\infty$, $j = 1, \cdots, n$.

— *Procédure de séparation : soit $\overline{x}$ la solution optimale de $(\mathcal{P}'_{\alpha,\beta})$ et r un indice tel que $\overline{x}_r$ n'est pas entier. Alors*

$$S_{\alpha,\beta} = S_{\alpha',\beta} \cup S_{\alpha,\beta'}, \quad S_{\alpha',\beta} \cap S_{\alpha,\beta'} = \oslash$$

avec

$$\alpha'_j = \begin{cases} [\overline{x}_r] + 1 & si\ j = r \\ \alpha_j & si\ j \neq r \end{cases}, \beta'_j = \begin{cases} [\overline{x}_r] & si\ j = r \\ \beta_j & si\ j \neq r \end{cases}$$

— *Procédure d'évaluation* : *résolution de* $(\mathcal{P}'_{\alpha,\beta'})$ *et* $(\mathcal{P}'_{\alpha',\beta})$ *par l'algorithme dual du simplexe en variables bornées.*

1. *Montrer que cette méthode est finie*
2. *Résoudre par cette méthode le programme linéaire ne nombres entiers suivant :*

$$(\mathcal{P}) \max\{(2x_1 + x_2) \text{ sous } x_1 - 4x_2 \leq 0, 3x_1 + 4x_2 \leq 15, x_1, x_2 \in \mathbb{N}\}$$

3.7.17. *Résoudre par la méthode de séparation et évaluation le problème d'optimisation linéaire en nombres entiers*

$$\max\{(3x_1 + 2x_2), \quad x \in S\} \text{ avec}$$

$$S = \{-x_1 + 2x_2 \leq 7, 2x_1 + 3x_2 \leq 18, 9x_1 - 2x_2 \leq 36, x_1 \in \mathbb{N}, x_2 \in \mathbb{N}\}$$

A chaque étape on calcule l'optimum du problème relâché graphiquement et on arbitre une variable fractionnaire.

Bibliographie

[GoMi95] Michel Gondran, Michel Minoux, Graphes et Algorithmes, Editions Eyrolles (1995).

[Min83a] Michel Minoux, Programmation mathématique, Tome 1, Dunod (1983).

[Min83b] Michel Minoux, Programmation mathématique, Tome 2, Dunod (1983).

[Sak84a] Michel Sakarovitch, Optimisation combinatoire, Tome 1 : Graphes et programmation linéaire, Hermann (1984) ;

[Sak84b] Michel Sakarovitch, Optimisation combinatoire, Tome 2 : Programmation discrète, Hermann (1984) ;

[Dan66] G.B. Dantzig, Applications et prolongements de la programmation linéaire, Dunod (1966).

[GoTo] D. Goldfarb, M.J. Todd, Linear programming, dans : *Handbooks in Operations research and management sience*, Vol 1 : Optimization, G.L. Nemhauser et al., Chapitre II. North Holland (1989).

Index

Annexe A

AMPL

A.1 L'interface SCI2AMPL

Interface écrit en *scilab/TclTk/DOS* pour accéder à AMPL.

Objectif :
— Editer/afficher facilement des fichiers **.mod* et **.dat*
— Création automatique des fichiers **.run* (batchfile pour AMPL)
— Création de fichiers auxiliaires nécessaires pour afficher des graphes par SCILAB (graphe).
Comment résoudre un problème ?
— Ouvrir/editer/sauver les fichiers **.mod et *.dat* ;
— Choisir un solveur : (CPLEX=entiers, minos=non linéaire) dans le menue ;
— *Make :* créer un fichier exécutable AMPL **.run*, choisir les variables à afficher ;
— *Run :* lancer AMPL, afficher les résultats ;
— Pour afficher les résultats à l'aide d'un graphe, ajouter quelques lignes dans **.mod* (et **.dat*).
Exemples :
— Le boulanger
— steel (optimisation linéaire simple)
— hamilton2 (voyageur de commerce avec un circuit)
— netmax3 (problème de transporter une quantité maximale d'un bien du sommet a vers g)

A.1.1 Le menue de SCI2AMPL

— *File/New [button]* remplace *.mod,*.dat et *.run par un nouveau projet
— *File/Open [buttons]* ouvre des fichiers *.mod,*.dat et *.run déjà existants
— *File/Save [button]* sauvegarde les fichiers *.mod,*.dat et *.run à l'endroit actuel
— *File/Save as* sauvegarde les fichiers *.mod,*.dat et *.run à un nouvel endroit
— *File/Quit [button]* sauvegarde et sort du programme
— *Edit [button]* les commandes classiques d'édition

— *Config/Change* permet de changer répertoires, taille de caractères, kestrel
— *Config/View* (des)active l'affichage des fenêtres *.dat, *.run
— *Config/Show* (des)active l'affichage des graphes
— *AMPL/Display [button]* affiche les inconnues/contraintes du modèle actuel
— *AMPL/Graph ...* voir la page suivante
— *AMPL/Make [button]* crée fichier executable *.run, demande de fixer les inconnues à afficher
— *AMPL/Run [button]* exécute le fichier *.run, affiche les résultats
— *AMPL/Solver* choisit la boîte noire AMPL (*cplex* : entiers, *minos* : nonlineaire, *kestrel* : WEB)
— *Help* quelques pages d'aide sur AMPL et sur l'interface.

Dans les fenêtres on peut utiliser les commandes classiques de Windows pour éditer : *ctrl d = shift suppr* pour couper, *ctrl v = shift inser* pour coller, essayez aussi *ctrl* →, *shift* → *ctrl shift* → etc.

A.1.2 Comment créer un graphe ?

Si le modèle correspond à un problème d'optimisation sur un graphe, il est possible d'afficher le graphe. Pour cela on doit fixer

— les ensembles correspondant aux sommets (ici *set INTER*) et aux arcs (ici *set ROADS*) ;
— *optionnel :* les valeurs/couleurs pour les sommets (variables/paramètres indicés par *INTER*) :
— *optionnel :* les valeurs/couleurs pour les arcs (variables/paramètres indicés par *ROADS*) ;
— *optionnel :* les coordonnées des sommets (sinon positions aléatoires).

Ces données sont déclarées dans le fichier **.mod* correspondant, ou toute ligne doit commencer par une chaîne particuliere de caractères et un espace. Voici l'exemple netmax3 :

```
###graph: vertexset  INTER  # obl: set
###graph: vertexxpos  xpos # obl: unknown index vertexset
###graph: vertexypos  ypos # obl: unknown index vertexset
###graph: vertexcolor  # opt: unknown, index vertexset
###graph: vertexvalue  # opt: unknown, index vertexset
###graph: edgeset   ROADS   # obl: set subset vertexset x vertexset
###graph: edgecolor  cap  # opt: unknown, index edgeset
###graph: edgevalue  traffic  # opt: unknown, index edgeset
```

Par la commande *AMPL/Add...* on peut ajouter un fragment de ces lignes à la fin du fichier **.mod*. N'oubliez pas de déclarer dans **.mod* et définir dans **.dat* ces ensembles/variables/paramètres AMPL.

A.2 Multi-période : L'usine sur plusieurs semaines

Dans notre usine sidérurgique, on se permet de produire sur plusieurs semaines (entre semaine 1 et semaine T), mais au lieu de vendre la production immédiatement, on dispose de la

possibilité de mettre le produit dans un entrepôt.

unités produites au cours de la semaine t — unités vendues au cours de la semaine t

unités en stock à la fin de la semaine $t-1$ — unités en stock à la fin de la semaine t

Considérons les variables
- *Make[p,t]* la production en tonnes du produit p au cours de la semaine t
- *Sell[p,t]* la vente en tonnes du produit p au cours de la semaine t
- *Inv[p,t]* la quantité en tonnes du produit p à la fin de la semaine t dans l'entrepôt.
- *Inv[p,0]* est donné (par paramètre *inv0[p]*).

On obtient alors le bilan

$$\forall p \in PROD\ \forall t \in \{1...T\} : \quad Make[p,t] + Inv[p,t-1] = Sell[p,t] + Inv[p,t]$$

Pour la modélisation nous allons introduire trois groupes de paramètres
- *rate, inv0, prodcost, invcost* dépendant seulement du produit : la production par heure (en t/h), le stock au début (en t), les frais unitaires de production (en F/t) et les frais unitaires de stockage (en F/t)
- *market, revenue* dépendant du produit et de la semaine : les bornes supérieures pour la production (en t), les bénéfices unitaires du produit vendu (en F/t)
- *avail* dépendant de la semaine : le temps de disponibilité du laminoir (en h).

Ceci donne lieu au modèle donné en Figure A.1 (voir steelT.mod).

Il en résulte un problème d'optimisation aux variables doublement bornées de la forme

$$\begin{cases} \min fx \\ Bx = b,\ Cx \le c,\ x \ge \underline{x},\ x \le \overline{x}. \end{cases}$$

Supposons qu'une solution optimale est donnée par le point de base $x(I, B^+, B^-)$, avec valeur optimale $V = V(b, c, \underline{x}, \overline{x})$. D'après la théorie des coûts marginaux (variables duales), à condition que la base (I, B^+, B^-) reste aussi réalisable après une "petite" perturbation $\Delta b, \Delta c, \Delta\underline{x}, \Delta\overline{x}$, nous avons

$$V(b+\Delta b, c+\Delta c, \underline{x}+\Delta\underline{x}, \overline{x}+\Delta\overline{x}) - V(b,c,\underline{x},\overline{x}) = u\Delta b - v\Delta c + \underline{w}\,\Delta\underline{x} - \overline{w}\,\Delta\overline{x}$$

pour certains variables u, v, $\underline{w}$ et $\overline{w}$. Ces variables duales sont disponibles sous AMPL...

On a légèrement reformulé le problème de l'usine sidérurgique, ici sur *deux* semaines, avec deux produits *bands, coils*, voir steelTbis L'objectif est de minimiser la perte totale

```
minimize total_cost:
  sum {p in PROD, t in 1..T} (- revenue[p,t]*Sell[p,t] +
    prodcost[p]*Make[p,t] + invcost[p]*Inv[p,t]);
```

On obtient une valeur optimale de -248633 (et donc des bénéfices de 248633 unités).

Contraintes $Cx \le c$ et v.

```
subject to time {t in 1..T}: sum {p in PROD} (1/rate[p]) * Make[p,t] <= avail[t];
```

```
set PROD;                       # products
param T > 0;                    # number of weeks
param rate {PROD} > 0;          # tons per hour produced
param inv0 {PROD} >= 0;         # initial inventory
param avail {1..T} >= 0;        # hours available in week
param market {PROD,1..T} >=0;   # limit on tons sold in week
param prodcost {PROD} >= 0;     # cost per ton produced
param invcost {PROD} >= 0;      # carrying cost/ton of inventory
param revenue {PROD,1..T} >= 0; # revenue per ton sold

var Make {PROD,1..T} >= 0;      # tons produced
var Inv {PROD,0..T} >= 0;       # tons inventoried
var Sell {p in PROD, t in 1..T} >= 0, <= market[p,t]; # tons sold

maximize total_profit:
  sum {p in PROD, t in 1..T} (revenue[p,t]*Sell[p,t] -
     prodcost[p]*Make[p,t] - invcost[p]*Inv[p,t]);
              # Total revenue less costs in all weeks

subject to time {t in 1..T}:
  sum {p in PROD} (1/rate[p]) * Make[p,t] <= avail[t];
              # Total of hours used by all products
              # may not exceed hours available, in each week
subject to initial {p in PROD}:  Inv[p,0] = inv0[p];
              # Initial inventory must equal given value
subject to balance {p in PROD, t in 1..T}:
  Make[p,t] + Inv[p,t-1] = Sell[p,t] + Inv[p,t];
              # Tons produced and taken from inventory
              # must equal tons sold and put into inventory
```

FIGURE A.1 – LE FICHIER MODÈLE POUR L'USINE SUR PLUSIEURS SEMAINES

donne lieu à des informations suivantes (voir steelTbis)

```
: time.lslack time.ldual time.uslack   time.udual    :=
 1   Infinity       0           0        -2660
 2   Infinity       0     7.10543e-15    -3080 ;
```

Interprétation : `time.uslack[2]` est proche de la précision machine, il s'agit d'une petite erreur d'arrondi. Donc la variable d'écart $c - C\widehat{x}$ s'annule, et on obtient complémentarité avec la variable duale $v = [2660, 3080]$. Attention au changement de signe !

Augmenter par exemple `avail[1]` par une unité (petite pour préserver la réalisibilité) donnera un changement d'objectif de $-v\Delta c = -2660$ unités (on y gagne).

Les bornes sup et inf.

Les contraintes $x \leq \overline{x}$ et $x \geq \underline{x}$ ont été modélisées sous la forme (par exemple)

```
subject to sell_max {p in PROD, t in 1..T} : Sell[p,t]<=market[p,t];
subject to sell_min {p in PROD, t in 1..T} : Sell[p,t] >= 0;
```

La post optimisation donne lieu à des informations suivantes (voir steelTbis)

```
:       Sell.lb   Sell Sell.ub Sell.lslack Sell.uslack    :=
 bands 1     0     6000   6000     6000          0
 bands 2     0     6000   6000     6000          0
 coils 1     0      307   4000      307       3693
 coils 2     0     2500   2500     2500          0 ;
```

```
:       sell_min sell_min.slack :=
 bands 1     0          6000
 bands 2     0          6000
 coils 1     0           307
 coils 2     0          2500 ;
```

```
:       sell_max sell_max.slack    :=
 bands 1    -1.7           0
 bands 2    -0.6           0
 coils 1     0          3693
 coils 2    -2             0 ;
```

Interprétation : Comme on voit dans la colonne `Sell.lb`, la partie correspondante de $\widehat{x} - \underline{x}$ n'a aucune composante zéro. Donc, par complémentarité, la partie correspondante de $\underline{w}$ s'annule. Ceci est assez logique : un changement "petit" de la borne inférieure ne changera pas la perte totale de l'entreprise car on produit des quantités non nuls.

Par contre, $\overline{w} = [1.7, 0.6, 0, 2]$ (encore attention au changement de signe). Le zéro s'explique par le fait que la restriction du marché `market[coils,1]` n'est pas atteint. Par contre, si une autre restriction augmente, alors $-\overline{w}\Delta\overline{x}$ diminue, on y gagne.

Contraintes $Bx = b$ et u.

```
subject to initial {p in PROD}:  Inv[p,0] = inv0[p];
```

donne lieu à des informations suivantes (voir steelTbis)

```
:     initial initial.slack    :=
 bands   -23.3        0
 coils   -30          0 ;
```

Interprétation : Ici on a une contrainte d'égalité et donc "la variable d'écart" s'annule. La variable duale vaut $u = [-23.3, -30]$. Si par exemple on aura un stock initial `inv0[bands]` d'une unité de moins (il n'y avait pas de `coils` au départ, comparer avec le fichier de données), on y perdra $u\Delta b = u[0, -1]^t = 23.3$ unités.

Exercice : donner l'interprétation en termes de pertes en stock de

```
:       balance  balance.slack    :=
 bands 1   23.3    -9.09495e-13
 bands 2   25.4    -3.81988e-10
 coils 1   30      -2.27374e-13
 coils 2   33       0 ;
```

A.3 Astuces sous AMPL

A.3.1 Quelques compléments sur le langage AMPL

Nommer des quantités au moyen de variables supplémentaires :
Le soucis premier dans l'écriture d'un modèle AMPL doit être la lisibilité (les boîtes noires prévoient une optimisation du code). Par conséquent, une quantité souvent utilisée comme la production totale mérite parfois un propre nom (aussi variable).

Paramètres calculés :
Parfois certaines quantités sont calculés en fonction des données, à titre d'exemple les distances (vol d'oiseau) entre villes sachant leur coordonnées sur un plan

```
set VILLES; param xpos{VILLES} >=0, <= 10; param ypos{VILLES} >=0, <= 10;
param distance{(i,j) in VILLES cross VILLES} :=
   sqrt((xpos[i]-xpos[j])*(xpos[i]-xpos[j])+(ypos[i]-ypos[j])*(ypos[i]-ypos[j]));
```

Ici il s'agit de paramètres complètement déterminés, et ne pas de variables.

Nombres entiers/binaires et non linéarité :
Par défaut, les variables AMPL sont de type *réel*. On peut imposer le type *entier*

```
var jours >=0, integer;
```

(ou *binaire* par le qualificatif `binary`). Pour traiter des tels variables, if faudra choisir le solver

```
option solver cplex;
```

Le solveur par défaut, *minos*, ne résout pas les programmes linéaires en nombres entiers (mais il sait traiter certaines problèmes non linéaires contrairement à *cplex*).

nombre d'éléments :

```
set PROD; param nb_prod := card(PROD);
```

les intervalles :
ensembles de nombres ordonnés (on peut spécifier *by "pas"* si on le souhaite)

```
param N >= 1; set ANNEES := 1 .. N;
```

Opérations sur les ensembles :

```
set A; set B;
set C:= A union B; set D := A inter B;  # aussi diff, symdiff
```

Désigner un élément ou une partie d'un ensemble :

```
set NOEUDS;
param racine symbolic in NOEUDS;   # racine n'aura pas de valeur numerique
set FEUILLES within NOEUDS := NOEUDS diff { racine };
```

L'opérateur " : " : appartenance conditionnelle, p.e., matrice triangulaire supérieure

```
 param N integer >= 0;
 param matrice {j in 1 .. N, k in 1 .. N : j <= k};
```

Fixer un ordre dans un ensemble :

```
set GRADES ordered;
var anciens{GRADES} >= 0;
var nouveaux{GRADES} >= 0;
# promus [g] = ceux qui passent de g \`a g+1
var promus {g in GRADES : g <> last (GRADES)} >= 0;
```

```
subject to calcule_nouveaux {g in GRADES} :
   nouveaux [g] = anciens [g] +
       (if g <> first (GRADES) then promus [prev (g)] else 0) -
       (if g <> last (GRADES) then promus [g] else 0);
```

A.3.2 Quelques compléments sur les fichiers de données

Paramètres indicés par un même ensemble :

Plusieurs tableaux avec même indexage peuvent être définis simultanément (voir gauche), même avec leur ensemble d'indices (voir droite).

```
set PROD := A B F;
param : prix_achat prix_vente :=
   A        10        12
   B        37        29
   F        17        20;
```

```
param : PROD : prix_achat prix_vente :=
   A        10        12
   B        37        29
   F        17        20;
```

Laisser des valeurs indéfinies :

Il suffit de mettre un " . " à la place de la valeur indéfinie.

Paramètres à trois dimensions :

```
# dans fichier.mod: param cube {INDICES,INDICES, INDICES};
set INDICES := 1 .. 2;
param cube :=
  [*, *, 1] : 1    2 :=
          1   3    5
          2   4    6
```

```
[*, *, 2] : 1    2 :=
          1   2   11
          2   0    7;
```