OPTIMISATION CONVEXE DANS LES RESEAUX AVEC APPLICATIONS AU TRAFIC ROUTIER ET A L'ENERGIE ELECTRIQUE

THESE No 669 (1987)

PRESENTEE AU DEPARTEMENT DE MATHEMATIQUES

ECOLE POLYTECHNIQUE FEDERALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ES SCIENCES

PAR

CLAUDE PASCHE

Ingénieur mathématicien diplômé EPFL
originaire de Servion (VD)

acceptée sur proposition du jury :

Prof. D. de Werra, rapporteur
Prof. A. Germond, corapporteur
Prof. F. Perret, corapporteur
Prof. J.-P. Vial, corapporteur

Lausanne, EPFL
1987

à mes parents,

à Isa et Delphine.

MERCI

à **Monsieur le Professeur D. de Werra** qui m'a intéressé aux problèmes de flot et qui m'a guidé dans ces recherches ,

à **Messieurs les Professeurs A. Germond , F.-L. Perret et J.-P. Vial** qui ont accepté de faire partie du jury ,

à **mes collègues N. Chahal , S. Alec et J. Barras** qui ont créé une ambiance de travail chaleureuse et productive durant le projet "application de méthodes de recherche opérationnelle aux problèmes de la sécurité des réseaux d'énergie électrique" d'où le chapitre 5 est tiré ,

à **Monsieur le Professeur R. Dembo , à Messieurs R. Fridel et J.-P. Leyvraz** qui m'ont prêté et commenté leurs programmes ,

à tous ceux ou celles qui , sous une forme ou une autre , m'ont apporté leur appui .

RESUME .

Ce travail , réalisé dans le domaine des flots et des réseaux , traite deux thèmes à savoir :

- les algorithmes de calcul d'un flot optimal ou d'un flot d'équilibre ,
- l'étude de problèmes réels de réseaux en utilisant le modèle des flots .

Mathématiquement , un problème de flot est un problème d'optimisation sous un ensemble de contraintes linéaires . Trois types de fonctions objectives vont être traitées :

- les coûts linéaires :
  Le problème est un cas particulier de programmation linéaire pour lequel il existe une version spécialisée très performante de l'algorithme du simplexe .

- les coûts linéaires par morceaux :
  La variante de l'algorithme du simplexe présentée est caractérisée , d'une part par un traitement implicite des morceaux linéaires et d'autre part par une opération de pivotage combinant plusieurs modifications de flot .

- les coûts convexes :
  L'algorithme proposé est du type convexe simplexe et se base sur une approximation du coût associé à chaque arc par une fonction linéaire formée de plusieurs milliers de segments . Contrairement à un algorithme de descente basé sur le gradient , la différentiabilité des fonctions coûts n'est pas requise .

L'étude de ces algorithmes inclut :

- une analyse des propriétés théoriques (finitude , optimalité , précision ...) ,
- une description de l'implantation sur ordinateur (les structures de données et leur manipulation) ,
- une analyse comparative basée sur une série de tests (pour des réseaux réels ou générés aléatoirement) .

Le champ d'application d'un algorithme d'optimisation convexe dans les réseaux est illustré par trois problèmes issus de la pratique :

- un nouveau modèle pour l'ordonnancement de projets composés de tâches de durées aléatoires ,

- le problème d'affectation du trafic qui peut être modélisé comme un problème de flot avec des coûts partiellement séparables ,

- les modèles électriques du DC-flow et du load-flow .

Le dernier aspect abordé est l'analyse de sécurité des réseaux . Deux méthodes sont proposées pour étudier les ruptures de liaisons potentielles :

- l'analyse locale :
  Une méthode heuristique permet de déterminer quelles sont les liaisons les plus perturbées par le déclenchement d'une liaison .

- l'analyse topologique :
  L'algorithme présenté énumère de manière efficace tous les couples de liaisons dont la rupture disconnecte le réseau .

TABLE DES MATIERES

Chapitre 2 : FLOT A COUT CONVEXE LINEAIRE PAR MORCEAUX

Chapitre 3 : FLOT A COUT CONVEXE

Chapitre 4 : CALCUL DE L'EQUILIBRE DANS UN RESEAU DE TRANSPORT .

Chapitre 5 : PROBLEME DE SECURITE DES RESEAUX D'ENERGIE ELECTRIQUE .

NOTATIONS

En plus des notations standards , voici les principales conventions typographiques et notations qui ont été adoptées.

V	désigne un ensemble .
$V - v$	désigne l'ensemble formé des éléments de V hormis v .
$V \times E$	désigne l'ensemble produit de V et E .
$\mathbf{R}(n)$	désigne l'ensemble des vecteurs réels à n composantes ; ces composantes sont indicées par $\{ 1,...,n \}$.
$\mathbf{R}(V)$	désigne l'ensemble des vecteurs réels dont les composantes sont indicées par les éléments de l'ensemble V .
W'	désigne la transposée d'un vecteur W.
W_v	désigne la composante du vecteur $W \in \mathbf{R}(V)$ indicée par $v \in V$.
W_B	désigne le vecteur de $\mathbf{R}(B)$ formé des composantes de $W \in \mathbf{R}(V)$ indicées par les éléments de $B \subset V$.
$\mathbf{R}(n \times m)$	désigne l'ensemble des matrices réelles à n lignes et m colonnes ; les lignes sont indicées par $\{ 1,..,n \}$ et les colonnes par $\{ 1,..,m \}$.
$\mathbf{R}(V \times E)$	désigne l'ensemble des matrices réelles dont les lignes sont indicées par les éléments de V et les colonnes par ceux de E .
A'	désigne la transposée de la matrice A .
$A_{v,e}$	désigne la composante de la matrice $A \in \mathbf{R}(V \times E)$ à l'intersection de la ligne indicée par $v \in V$ et de la colonne indicée par $e \in E$.
$A_{\bullet e}$	désigne la colonne de $A \in \mathbf{R}(V \times E)$ indicée par $e \in E$; c'est un vecteur de $\mathbf{R}(V)$.
$A_{v \bullet}$	désigne la ligne de $A \subset \mathbf{R}(V \times E)$ indicée par $v \in V$; c'est la transposée d'un vecteur de $\mathbf{R}(E)$.
A_B	désigne la matrice de $\mathbf{R}(V \times B)$ formée des colonnes de $A \in \mathbf{R}(V \times E)$ indicées par les éléments de $B \subset E$.
$\mathbf{F}: V \to \mathbf{R}$	désigne une fonction réelle .
$\mathbf{F}(v)$	désigne l'image par $\mathbf{F}$ d'un élément $v \in V$.

INTRODUCTION

De nombreux problèmes d'ingénierie possèdent un réseau comme structure sous-jacente . Ce réseau peut être une réalité physique (réseau de distribution d'énergie , réseau de télécommunication ou réseau routier) ou n'être qu'un modèle (représentation des termes non nuls d'une matrice creuse , ensembles des possibilités d'affectation des ouvriers aux machines ou possibilités d'approvisionnement de magasins à partir d'entrepôts). La recherche opérationnelle regroupe tous ces problèmes en un problème théorique unique : la recherche d'un flot dans un réseau . De manière similaire , le flot va correspondre , soit à un fluide (courant électrique ou transit de voiture) , soit à la solution du problème (attribution des machines aux ouvriers ou plan d'approvisionnement optimal) .

Les contraintes sur la propagation du flot dans le réseau sont de nature très diverses :

- contraintes de conservation : le système est fermé et il n'y a aucune perte de flot ,
- contraintes de capacité : les éléments constituant le réseau ne doivent pas être surchargés ,
- contraintes de demande : les consommations de flot de certains éléments doivent être satisfaites ,
- contraintes économiques : le coût de transport du flot doit être minimum ,
- contraintes d'équilibre : le flot doit se répartir sur les différents chemins possibles .

Ce travail présente la théorie des flots dans les réseaux sous trois aspects , à savoir : le développement d'algorithmes , leur implantation sur ordinateur et la modélisation de problèmes réels .

Du point de vue mathématique , un problème de flot peut être défini comme un problème d'optimisation sous un ensemble de contraintes linéaires . Tous les outils de la recherche opérationnelle sont alors à disposition pour développer

des algorithmes de résolution et pour étudier leurs propriétés théoriques telles la finitude , la précision ou la complexité .

Les problèmes à résoudre étant , en général , de grande dimension , il faut avoir recours à l'ordinateur . Mais le passage d'un algorithme , énoncé de manière purement formelle , à un logiciel efficace n'est pas chose aisée . L'implantation consiste à mettre au point des structures de données pour représenter les objets mathématiques utilisés et des procédures pour les manipuler .

Dans la pratique , toute étude débute par une phase de modélisation . Un modèle est obtenu en formulant des hypothèses simplificatrices , puis il est comparé avec les problèmes bien connus dans le but de trouver un algorithme de résolution . C'est la raison pour laquelle , il est important de faire un inventaire des modèles équivalents au problème de flot .

Ce travail se compose de cinq chapitres conçus pour pouvoir être lus indépendamment . Chacun d'eux comporte , en introduction , un bref résumé de l'état des recherches et une bibliographie . Les trois premiers chapitres présentent et analysent des algorithmes de résolution ; les deux derniers chapitres utilisent ces algorithmes pour traiter deux modèles issus des sciences de l'ingénieur .

Le chapitre 1 traite du problème classique de flot (ou transbordement) à coût linéaire minimum . L'algorithme du simplexe dans les réseaux et son implantation sur ordinateur sont présentés en détail ; il s'agit en effet de la base de départ des algorithmes présentés dans les chapitres ultérieurs . D'autre part des tests sur des réseaux de grandes tailles ont été effectués pour comparer des stratégies de choix de pivot présentées par différents auteurs . La fin de ce chapitre est consacrée à des méthodes de post-optimisation qui ont été développées pour les applications des deux derniers chapitres .

Le chapitre 2 étudie le problème de flot à coût convexe linéaire par morceaux . Si , en théorie , ce problème peut se reformuler avec des coûts linéaires , seul un algorithme spécialisé permet de le résoudre efficacement . Une variante de l'algorithme du simplexe est présentée ; elle est basée sur une nouvelle méthode de pivotage permettant de combiner plusieurs améliorations du coût pour un seul pivot . Pour les tests sur deux types de problèmes , le temps calcul diminue de 50 à 70 % .

Le chapitre 3 , extension naturelle des précédents , est consacré au problème de flot à coût convexe . L'algorithme proposé est du type convexe-simplexe et se base sur une approximation du coût associé à chaque arc par une fonction linéaire formée de plusieurs dizaines de milliers de segments . L'analyse théorique et les tests effectués montrent sa compétitivité tant du point de vue de la précision que de celui du temps calcul .

Pour illustrer le champ d'application d'un tel algorithme , la seconde partie de ce chapitre classifie les modèles équivalents à un problème d'optimisation convexe dans un réseau . En particulier , un nouveau modèle d'ordonnancement est présenté ; il est caractérisé par la durée aléatoire des taches composant le projet . On montre alors comment utiliser l'algorithme de flot pour déterminer un ordonnancement qui minimise la probabilité d'avoir un retard .

Le chapitre 4 présente une application en technique des transports : il s'agit de la prévision grossière de la circulation dans un réseau routier . Un des modèles les plus utilisés reformule ce problème en termes de recherche d'un ensemble de flots interdépendants . La méthode de résolution proposée est basée sur l'algorithme du chapitre précédent .

Le chapitre 5 aborde le problème de la sécurité des réseaux de distribution d'énergie . En effet , la répartition des puissances actives et réactives dans un réseau électrique peut être modélisée par deux flots fortement liés et peut , par conséquent , être calculée par un processus itératif basé sur l'algorithme de flot

La première approche de l'analyse de sécurité est dérivée de la notion de calcul local . Cette méthode dérive de l'interprétation de l'approximation utilisée par l'algorithme de flot et permet de déterminer quelles sont les lignes les plus perturbées par un déclenchement .

La seconde approche , elle , est une analyse topologique du réseau , consistant en un algorithme d'énumération des paires de lignes qui disconnectent le réseau et mettent , par conséquent , en danger l'approvisionnement en énergie .

Chapitre 1

FLOT A COUT LINEAIRE

1.0. INTRODUCTION

Le problème de flot à coût linéaire , sujet de ce premier chapitre , est un problème de programmation linéaire muni d'une structure très particulière . Des études ont montré qu'un des algorithmes les plus performants pour résoudre ce problème est une version spécialisée du simplexe ([2],[8],[13]) .

Après les définitions formelles des notions de réseau et de flot ainsi que quelques rappels de théorie des graphes , ce chapitre présente l'algorithme du simplexe pour les réseaux . L'accent est mis sur les spécificités qui le rendent aussi efficace , à savoir :

- la forme particulière des contraintes :
  Toutes les opérations algébriques à la base du simplexe sont grandement simplifiées et c'est pourquoi le paragraphe 1.2 présente la variante pour les réseaux comme une "traduction" graphique de résultats algébriques .

- de nouvelles techniques de mémorisation et de manipulation d'arbres :
  Les implantations du simplexe qui en résultent , telle celle présentée dans le paragraphe 1.3 , permettent la résolution de problèmes de grandes tailles .

- des stratégies de choix du pivot :
  Les performances de trois stratégies proposées dans la littérature sont comparées dans le paragraphes 1.4 .

Le logiciel , résultat de cette étude , a été utilisé pour des travaux de diplôme dans des domaines aussi variés que l'élaboration d'horaires scolaire , la gestion de barrages d'accumulation et la planification de la production d'énergie . Ces expériences ont montré que souvent , en recherche opérationnelle (voir [6] dans le cas des horaires) , l'algorithme de flot fait partie d'un processus itératif et qu'il résout des problèmes très voisins . C'est la raison pour laquelle les méthodes de post-optimisation , décrites dans le paragraphe 1.5 , ont été développées .

1.1. PRESENTATION DU PROBLEME

Le support d'un problème de flot est un réseau . Un ***réseau*** , noté R = (V,E) est déterminé par la donnée

- d'un ensemble V de n éléments appelés ***sommets*** .

- d'un ensemble E de m éléments appelés ***arcs*** , qui sont des couples ordonnés de sommets .

L'arc (i,j) est représenté de la façon suivante :

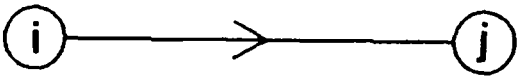


figure 1.1

Le sommet i est nommé ***extrémité initiale*** et le sommet j ***extrémité terminale*** de l' arc .

A cette connaissance topologique du réseau viennent s'ajouter des données quantitatives qui sont :

- la ***capacité*** $\mathbf{U} : E \rightarrow \mathbf{R}_+$ qui associe à chaque arc (i,j) la quantité maximale $\mathbf{U}((i,j))$ de flot qui peut transiter de i vers j le long de cet arc .

- le ***coût*** $\mathbf{C} : E \rightarrow \mathbf{R}$ qui associe à chaque arc (i,j) le coût de passage $\mathbf{C}((i,j))$ d'une unité de flot le long de cet arc .

- le ***bilan*** $\mathbf{B} : V \rightarrow \mathbf{R}$ qui associe à chaque sommet i la différence entre le flot quittant i et celui arrivant en i .
  si $\mathbf{B}(i) > 0$ on injecte $\mathbf{B}(i)$ unités de flot en i (i est dit sommet ***producteur***)
  si $\mathbf{B}(i) = 0$ il y a conservation du flot en i (i est dit sommet ***de transit***)
  si $\mathbf{B}(i) < 0$ on retire $-\mathbf{B}(i)$ unités de flot en i (i est dit sommet ***consommateur***)

Pour assurer l'équilibre du réseau , il faut que la somme des injections soit égale à celle des retraits ; c'est-à-dire que le bilan total doit être nul :

$$\sum_{i \in V} B(i) = 0 \qquad (1.1)$$

Un ***flot*** sur un réseau $R = (V,E)$ avec capacité U et bilan B peut être défini comme une fonction $F : E \to \mathbf{R}_+$ satisfaisant :

- m ***contraintes de capacité*** :

$$F((i,j)) \leq U((i,j)) \qquad \forall\ (i,j) \in E \quad (1.2)$$

Le flot $F((i,j))$ transitant sur l'arc (i,j) doit être inférieur à la capacité de l'arc .

- n ***équations de bilan*** :

$$\sum_{(i,j) \in E} F((i,j)) - \sum_{(j,i) \in E} F((j,i)) = B(i) \qquad \forall\ i \in V \quad (1.3)$$

On impose ainsi la conservation du flot en chaque sommet . Pour le sommet i , cette équation indique que la différence entre le total des flots sur les arcs admettant i comme extrémité initiale et le total des flots sur ceux l'admettant comme extrémité terminale doit être égal au bilan .

Le ***problème du flot à coût minimum*** , ou ***problème de transbordement*** , dans un réseau $R = (V,E)$ avec capacité U , coût C et bilan B peut être posé comme la recherche d'un flot F à coût minimum , c'est-à-dire :

$$\text{Minimiser}_{F \text{ : flot}} \sum_{(i,j) \in E} C((i,j))F((i,j)) \qquad (1.4)$$

La fonction à optimiser représente le "prix" qu'il faut payer pour faire passer le flot des sommets producteurs aux sommets consommateurs à travers le réseau .

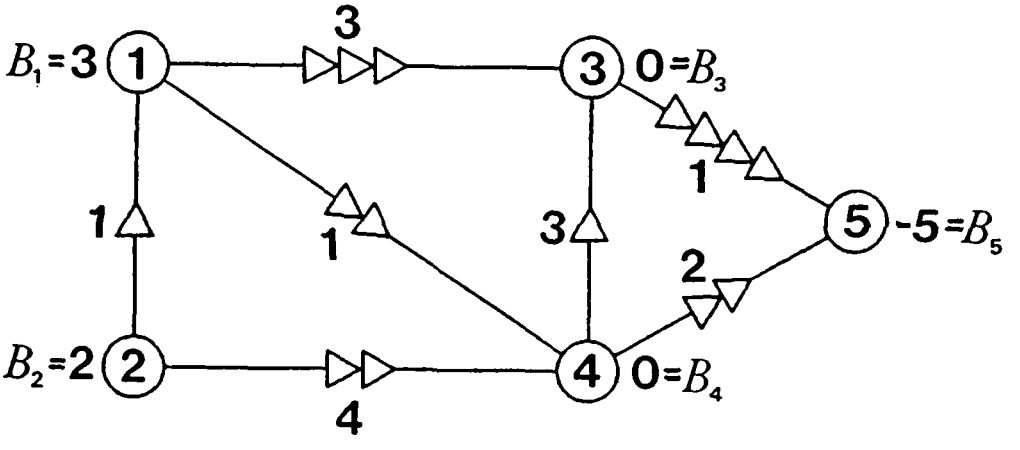
Ce problème peut aisément être formulé en termes de programmation linéaire . Pour ce faire il faut associer au réseau R = (V,E) sa matrice d'incidence $A \in \mathbf{R}(VxE)$ définie comme suit :

$$A_{v,(i,j)} = \begin{cases} 0 & v \neq i \; v \neq j \\ 1 & v = i \\ -1 & v = j \end{cases} \qquad \forall\, v \in V\,,\ (i,j) \in E \quad (1,5)$$

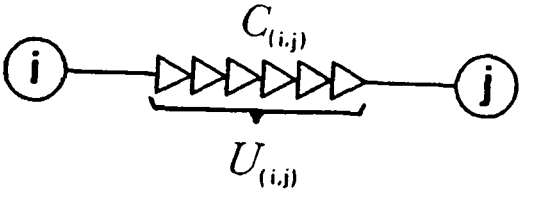
Les fonctions **U , C , B** sont représentées de façon naturelle par le vecteur des capacités $U \in \mathbf{R}(E)$, celui des coûts $C \in \mathbf{R}(E)$ et celui des bilans $B \in \mathbf{R}(V)$. La variable du problème $X \in \mathbf{R}(E)$ caractérise la fonction de flot ; la composante $X_{(i,j)}$ est égale au flot sur l'arc (i,j) . Le problème du flot à coût minimum devient :

$$\begin{aligned} &\text{Minimiser} \quad C'X \\ &\quad X \\ &\text{sc} \quad AX = B \\ &\quad 0 \leq X \leq U \end{aligned} \qquad (1.6)$$

La figure 1.2 illustre ces définitions sur un petit exemple .



notation :



Réseau

figure1.2

arcs: (1,3)(1,4)(2,1)(2,4)(3,5)(4,3)(4,5)

Minimiser C'X = (3 1 1 4 1 3 2) X

sc

sommets: 1 2 3 4 5 $AX = \begin{bmatrix} 1 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & -1 & 0 & -1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & -1 & 0 & -1 \end{bmatrix} \quad X = \begin{bmatrix} 3 \\ 2 \\ 0 \\ 0 \\ -5 \end{bmatrix} = B$

Programme linéaire

Figure 1.2 (suite)

Le problème du flot à coût minimum est très général ; il permet de traiter également :

- *Le problème du flot maximum*

  Pour maximiser le flot qui transite d'un sommet *source* s vers un sommet *puits* t à travers un réseau R = (V,E) muni de capacités **U** , il suffit de donner des coûts nuls aux arcs du réseau et d'ajouter un arc de retour (t,s) avec un coût **C**((t,s)) = -1 et une capacité **U**((t,s)) = ∞ . De plus on ne considère aucun sommet producteur ou consommateur ; c'est-à-dire que tous les bilans sont nuls **B**(i) = 0 ∀ i ∈ V .

- *Le problème du flot maximum à coût minimum*

  Pour calculer un flot maximum entre une source s et un puits t , tout en minimisant le coût , il faut ajouter un arc retour (t,s) avec une capacité **U**((t,s)) = ∞ et un coût très négatif . Par exemple :

$$\mathbf{C}((t,s)) = -\, n^{*}\Big(\max_{(i,j)\in E} \mathbf{C}((i,j)) \Big)$$

- *Le problème de transport*

  Tout sommet est soit producteur soit consommateur . Le réseau est biparti complet et les coûts **C** sont donnés par la matrice coût du problème . Les capacités sont infinies .

- *Les bornes inférieures de flot*

  Dans certains problèmes on veut imposer une quantité minimum de flot transitant dans les arcs , cette *borne inférieure* est donnée par $\mathbf{L} : E \rightarrow \mathbf{R}$. Les m contraintes de capacité deviennent :

$$\mathbf{L}((i,j)) \leqslant \mathbf{F}((i,j)) \leqslant \mathbf{U}((i,j)) \qquad \forall\ (i,j) \in E$$

  On modélise un arc (i,j) avec borne inférieure $\mathbf{L}((i,j)) \neq 0$ en diminuant le bilan $\mathbf{B}(i)$ de la quantité $\mathbf{L}((i,j))$, en augmentant le bilan $\mathbf{B}(j)$ de $\mathbf{L}((i,j))$ et en fixant la capacité à $\mathbf{C}((i,j)) - \mathbf{L}((i,j))$. Le flot circulant sur cet arc modifié correspond , pour l'arc original , à la quantité de flot en plus du minimum $\mathbf{L}((i,j))$.

1.2.ALGORITHME DU SIMPLEXE POUR LES RESEAUX

Le problème du flot à coût minimum se formulant en termes de programmation linéaire , il peut être résolu par le simplexe . Une utilisation directe de cet algorithme n'est pas judicieuse pour deux raisons :

- le programme linéaire associé à un problème de flot est de grandes dimensions (m variables bornées et n contraintes)
- la matrice considérée est très creuse (seulement deux éléments non nuls par colonne : un 1 et un -1).

Le but principal de ce paragraphe est de montrer comment adapter l'algorithme général du simplexe pour traiter le problème très particulier du flot à coût minimum .

1.2.1. Rappel de l'algorithme du simplexe

On va présenter l'algorithme primal du simplexe pour la résolution de programmes linéaires à variables bornées . Cet exposé , destiné à introduire quelques définitions et notations , n'est pas exhaustif ; le lecteur intéressé peut se référer à l'abondante littérature sur le sujet (par exemple [3],[12],[15]).

La formulation standard d'un problème de programmation linéaire est la suivante :

$$\begin{aligned} &\text{Minimiser}_{X} \quad z = C'X \\ &\text{sc} \quad AX = B \\ &\qquad 0 \leqslant X \leqslant U \end{aligned} \tag{1.7}$$

où

- $A \in \mathbb{R}(n \times m)$ et $\text{rang}(A) = n \quad n \leqslant m$
  (la j-ème colonne de cette matrice est notée $A_{.j}$)
- $B \in \mathbb{R}(n)$ et X , U , $C \in \mathbb{R}(m)$

Une ***base*** est un ensemble $B \subset \{1,...,m\}$ de n indices de colonne , tel que la sous-matrice A_B formée par ces n colonnes de A soit de rang n . On notera Z_B l'inverse de A_B . On définit $N = \{1,...,m\} - B$ l'ensemble des indices hors base et A_N la sous-matrice correspondante . On a ainsi scindé en deux la matrice :

$$A = (A_B, A_N)$$

On procède de même pour les vecteurs de $\mathbf{R}(m)$ du problème :

$$X = (X_B, X_N)' \quad U = (U_B, U_N)' \quad C = (C_B, C_N)'$$

Les variables d'indices contenus dans B (ie X_B) sont nommées ***variables de base*** , les autres (ie X_N) ***variables hors base*** .

Relativement à cette base B le problème se reformule de la manière suivante :

$$\begin{aligned} &\text{Minimiser} \quad z = C'_B X_B + C'_N X_N = C'_B Z_B B + (C'_N - C'_B Z_B A_N) X_N \\ &X_B, X_N \\ &\text{sc} \qquad X_B = Z_B B - Z_B A_N X_N \qquad (1.8) \\ &\qquad\quad 0 \leq X_B \leq U_B \\ &\qquad\quad 0 \leq X_N \leq U_N \end{aligned}$$

Une ***solution de base admissible*** (pour la base B) est un vecteur $X = (X_B, X_N)'$ satisfaisant les contraintes du problème et tel que les variables hors base soient nulles ou maximales (ie $X_j = 0$ ou $U_j \ \forall\ j \in N$). On peut montrer qu'il existe une solution optimale au problème si et seulement si il existe une solution de base admissible optimale . Dès lors l'algorithme du simplexe va cheminer de solution de base en solution de base jusqu'à l'optimum (s'il existe).

Le changement de base (ou ***pivotage***) s'effectue soit en augmentant une variable hors base $X_j = 0$ soit en diminuant une variable hors base $X_j = U_j$ et ceci jusqu'à ce qu'une variable de base atteigne une de ses bornes ou que X_j change de borne . Pour respecter les contraintes de (1.8) , la modification de X_j se répercute sur les variables de base X_B proportionnellement au vecteur $D = Z_B A_{.j}$; plus précisément :

$$X_j := X_j + \Delta$$
$$X_B := X_B - \Delta D \qquad (1.9)$$

De plus cette variation de X_j modifie le coût associé $z = C'_B X_B + C'_N X_N$ de la manière suivante :

$$z := z + (C_j - C'_B Z_B A_{.j})\Delta \qquad (1.10)$$

Une solution de base est optimale si et seulement si aucune solution de base voisine ne possède un coût inférieur ; c'est-à-dire que $X = (X_B\ X_N)'$ est optimale si et seulement si :

$$\begin{array}{ll} C_j - C'_B Z_B A_{.j} \geq 0 & \forall\ j \in N \text{ avec } X_j = 0 \\ \text{ou} & \\ C_j - C'_B Z_B A_{.j} \leq 0 & \forall\ j \in N \text{ avec } X_j = U_j \end{array} \qquad (1.11)$$

(cette valeur est le *coût réduit* de la variable X_j)

Les principes fondamentaux de l'algorithme primal du simplexe étant posés , on peut l'énoncer de façon plus précise .

Algorithme 1.1. ALGORITHME DU SIMPLEXE

0. Initialisation.

Trouver une base B et une solution de base admissible $X = (X_B\ X_N)'$.

1. Choix de la variable entrante.

1.1. Résoudre le *système dual* : $W'A_B = C'_B$ (les W sont appelés *variables duales*)

1.2. Choisir une variable hors base X_j tel que

- $C_j - W'A_{.j} < 0$ si $X_j = 0$

- $C_j - W'A_{.j} > 0$ si $X_j = U_j$

Si il n'existe pas de telle variable alors X est solution optimale

sinon X_j est la variable entrant dans la base .

2. Calcul de la direction de descente.

Résoudre le *système primal* : $A_B D = A_{.j}$

Si $X_j = U_j$ alors $D = -D$

3. Calcul de la longueur du pas.

Calculer

- $\Delta_1 = U_j$ (pour que X_j reste entre ses bornes)
- $\Delta_2 = min \{ X_i/D_i \mid D_i > 0 \}$ (pour que X_B reste $\geqslant 0$)
- $\Delta_3 = \min \{ (X_i - U_i)/D_i \mid D_i < 0 \}$ (pour que X_B reste $\leqslant U_B$)

On va faire un pas de longueur $\Delta = \min\{ \Delta_1 , \Delta_2 , \Delta_3 \}$ le long de D pour arriver à une solution de base voisine .

4. Mise-à-jour de la solution et de la base.

4.1. Mise-à-jour de la solution

$X_B := X_B - \Delta D$

Si ($X_j = U_j$) alors $X_j := X_j - \Delta$ sinon $X_j := X_j + \Delta$.

4.2. Mise-à-jour de la base

On choisit X_i l'une des variables pour lesquelles le minimum calculé en **3.** est atteint ; c'est la variable sortant de la base .

B := B + j - i

(remarque : si $\Delta = \Delta_1$ on peut choisir i = j et ainsi la base ne change pas)

Retour en **1**.

1.2.2. Algorithme particularisé aux réseaux

Après avoir introduit quelques définitions de la théorie des graphes , nous allons reprendre une à une les différentes notions présentées en 1.2.1 pour les interpréter en termes de réseau de manière à obtenir un algorithme du simplexe non plus matriciel mais graphique . Le problème de flot dans le réseau de cinq sommets et sept arcs de la figure 1.2 est utilisé pour illustrer les résultats présentés .

Quelques définitions de la théorie des graphes

Etant donné un réseau $R = (V,E)$ on peut définir les notions suivantes :

- une ***chaîne*** du sommet i vers le sommet j est un réseau avec un ensemble de sommets $\{ w_1=i , w_2 , \dots , w_{k-1} , w_k=j \} \subset V$ et un ensemble d'arcs $\{ e_1 , \dots , e_{k-1} \} \subset E$ tels que l'arc e_l ait pour extrémités w_l et w_{l+1} . Si $e_l = (w_l , w_{l+1})$ l'arc est dit ***orienté positivement*** , si $e_l = (w_{l+1} , w_l)$ il est ***orienté négativement*** . Un ***cycle*** est une chaîne de i vers i .

- un réseau est ***connexe*** s'il existe une chaîne entre toute paire de sommets .

- un ***arbre*** est un réseau (V,T) ayant le même ensemble de sommets que R et un sous-ensemble $T \subset E$ de n-1 arcs , qui est sans cycle . Un arbre est connexe et de plus il existe une chaîne unique allant d'un sommet à un autre .

Description des bases

A la fin du paragraphe 1.1 , les équations de bilan d'un problème de flot dans un réseau supposé connexe $R = (V,E)$ ont été exprimées sous la forme :

$$AX = B$$

où A est la matrice d'incidence de R et B le vecteur des bilans . La somme de ces équations donne :

$$\sum_{(i,j)\in E} \sum_{k\in V} A_{k,(i,j)} X_{(i,j)} = \sum_{k\in V} B_k = 0$$

Ce système étant redondant , la première équation (bilan au sommet 1) peut être supprimée . Pour la suite le système

$$AX = B \qquad (1.12)$$

où $A \in \mathbb{R}((V-1)\times E)$ et $B \in \mathbb{R}(V-1)$

désignera les équations de bilan des sommets de V - 1 . Cette matrice $\hat{A}$ est de plein rang (rang $\hat{A}$ = n-1) comme le montre le théorème caractérisant les bases :

Théorème 1.1.

$T \subset E$ **est une base de** $\hat{A}$ $\Leftrightarrow$ (V,T) **est un arbre de** R .

Démonstration :

$\Leftarrow$ Il s'agit de montrer que la matrice $\hat{A}$ (qui est la matrice d'incidence de (V,T) tronquée de la première ligne) est de rang n-1 . Ceci peut être fait en exhibant l'inverse de la matrice , ce qui sera fait ultérieurement .

$\Rightarrow$ La matrice A_C engendrée par l'ensemble C des arcs d'un cycle n'est pas de plein rang ; en effet , la combinaison linéaire des colonnes de cette matrice avec coefficient 1 pour les arcs orientés positivement et -1 pour ceux orientés négativement est égale au vecteur nul . Ainsi une base est un ensemble de n-1 arcs sans cycle , donc l'ensemble des arcs d'un arbre .

Q.E.D.

La figure 1.3 illustre la notion de base sur le réseau exemple .

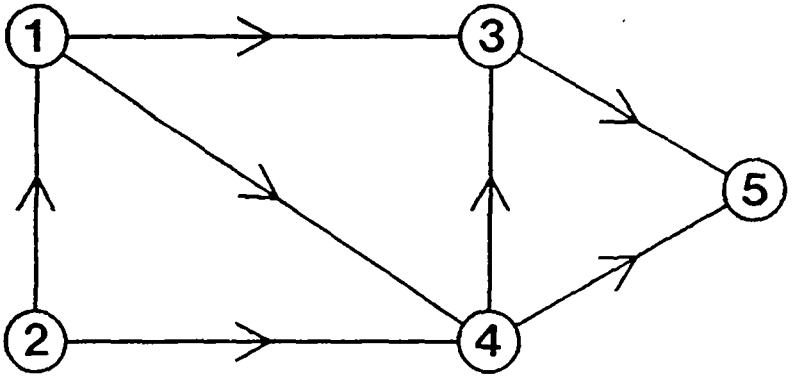
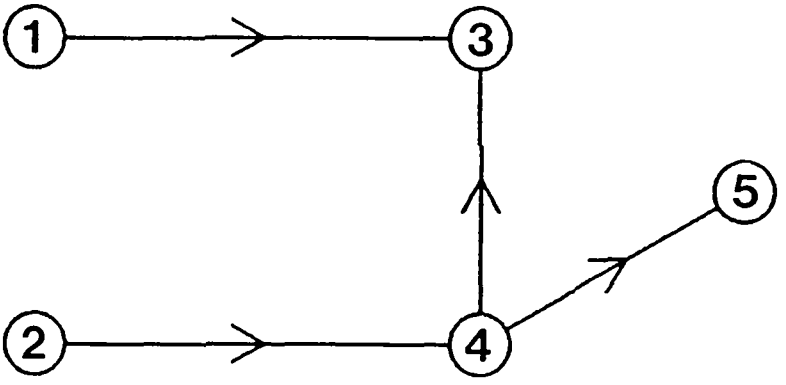


figure 1.3

arcs :	(1,3)	(1,4)	(2,1)	(2,4)	(3,5)	(4,3)	(4,5)	
sommets : 2	0	0	1	1	0	0	0	
3	- 1	0	0	0	1	- 1	0	= A
4	0	- 1	0	- 1	0	1	1	
5	0	0	0	0	- 1	0	- 1	

Une base de A : T = {(1,3) , (2,4) , (4,3) , (4,5) }



L'arbre associé à T

arcs :	(1,3)	(2,4)	(4,3)	(4,5)	
sommets : 2	0	1	0	0	
3	- 1	0	- 1	0	
4	0	- 1	1	1	= A_T
5	0	0	0	- 1	

Matrice associée à la base

figure 1.3(suite)

Etant donné un arbre (V,T)

- la ***racine*** de l'arbre est le sommet 1 qui correspond à la ligne supprimée .

- la ***profondeur*** d'un sommet i (notée d(i)) est le nombre d'arcs composant la chaîne de l'arbre allant de i à la racine 1 .

- le ***prédécesseur*** d'un sommet i (noté p(i)) est le deuxième sommet de la chaîne allant de i à 1 .

- un sommet j est dit ***descendant*** de i si la chaîne de l'arbre allant de j à la racine passe par i .

Dans le cas de l'arbre associé à la base présentée ci-dessus nous avons :

sommet i	1	2	3	4	5
profondeur d(i)	0	3	1	2	3
prédécesseur p(i)	-	4	1	3	4
descendant de i	{2,3,4,5}	-	{2,4,5}	{2,5}	-

Inversion d'une base

Un ***ordre transversal*** dans un arbre (V,T) est une numérotation des sommets obtenue par une exploration en profondeur de l'arbre à partir de la racine . Une des propriétés fondamentales de cet ordre est que chaque sommet et ses descendants ont des numéros consécutifs . Tout arc de l'arbre étant de la forme (i, p(i)) ou (p(i),i) , on peut associer à l'ordre transversal des sommets un ordre des arcs en donnant à l'arc d'extrémités i et p(i) le numéro du sommet i . Ces deux ordres jouent un rôle fondamental pour la résolution des systèmes d'équations liés à la base .

Propriété 1.2.

La matrice A_T associée à une base $T \subset E$ peut être triangulée en réordonnant les lignes selon un ordre transversal et les colonnes selon l'ordre associé .

Démonstration :

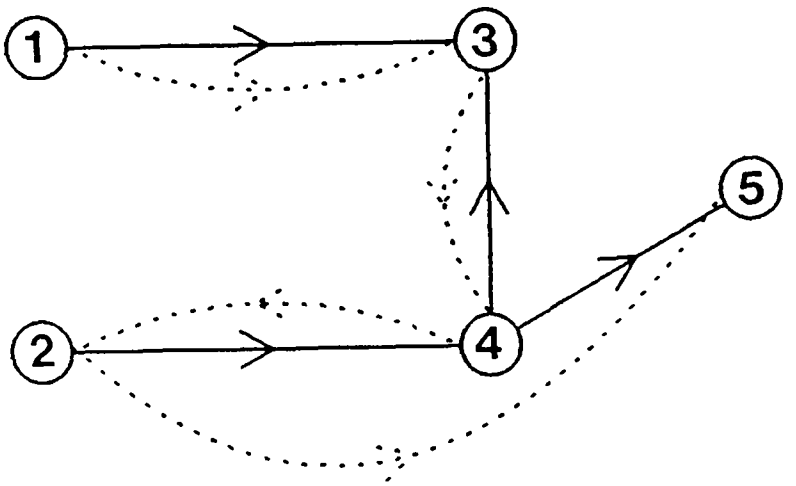
$\forall\ e \in T$

- soit e = (i,p(i)) alors A_T possède un 1 sur la diagonale en (i,e) et un -1 au-dessus de la diagonale en (p(i),e) (sauf si p(i) = 1) .
- soit e = (p(i),i) alors A_T possède un -1 sur la diagonale en (i,e) et un 1 au-dessus de la diagonale en (p(i),e) (sauf si p(i) = 1) .

Q.E.D.

La matrice engendrée par une base T peut être triangulée avec des +1 et des -1 sur la diagonale ; elle est donc inversible . Il faut remarquer qu'il n'est pas nécessaire de choisir un ordre transversal pour trianguler la matrice ; tout ordre où un sommet reçoit un numéro supérieur à celui de son prédécesseur est possible . En particulier on aurait pu choisir un ordre selon lequel les profondeurs des sommets sont décroissantes .

La figure 1.4 montre comment trianguler la base de la figure 1.3.



ordres transversaux : 1,3,4,2,5 et (1,3),(4,3),(2,4),(4,5)

figure 1.4

arcs :	(1,3)	(4,3)	(2,4)	(4,5)
sommets : 3	- 1	- 1	0	0
4	0	1	- 1	1
2	0	0	1	0
5	0	0	0	- 1

$= A_T$

figure 1.4 (suite)

On associe à la base T une ***matrice des chaînes*** notée $Z_T \in \mathbb{R}$ (T×(V-1)) définie par

$$(Z_T)_{(i,j),v} = \begin{cases} 0 \text{ si } (i,j) \text{ n'est pas sur la chaîne de l'arbre allant de } v \text{ à la racine .} \\ +1 \text{ si } (i,j) \text{ est un arc de cette chaîne orienté positivement .} \\ -1 \text{ si } (i,j) \text{ est un arc de cette chaîne orienté négativement .} \end{cases}$$

Cette matrice possède une interprétation graphique :

- La colonne de Z_T indicée par le sommet v caractérise la chaîne de v à la racine ; en effet ses composantes valent 1 si elles correspondent à des arcs orientés positivement sur cette chaîne , -1 si elles correspondent à des arcs orientés négativement et 0 sinon .

- La ligne de Z_T indicée par un arc de la forme (i,p(i)) (resp (p(i),i)) caractérise les descendants de i . En effet les composantes correspondant à i et à ses descendants valent toutes +1 (resp. -1) , les autres étant nulles .

Propriété 1.3.

La matrice des chaînes Z_T est l'inverse de la matrice A_T associée à la base T .

($Z_T * A_T = A_T * Z_T = I_{n-1}$ où I_{n-1} identité de $\mathbb{R}((n-1)\times(n-1))$)

Démonstration :

Considérons la multiplication de la ligne de Z_T indicée par l'arc (i,p(i)) (resp. (p(i),i)) et de la colonne de A_T indicée par l'arc (u,v) . Les deux termes 1 et -1 de la colonne sont multipliés par :

- 0 et 0 si u et v ne sont pas descendants de i
- 1 et 1 (resp. -1 et -1) si u et v sont descendants de i ou i lui-même
- 1 et 0 (resp. 0 et -1) si l'arc (u,v) et l'arc (p(i),i) sont les mêmes .

Ainsi le produit d'une ligne de Z_T et d'une colonne de A_T vaut 1 si elles sont indicées par le même arc et 0 sinon .

Q.E.D.

Pour la base des figures 1.3 et 1.4 , l'inverse est le suivant :

sommets :		2	3	4	5
arcs :	(1,3)	- 1	- 1	- 1	- 1
	(2,4)	1	0	0	0
	(4,3)	1	0	1	1
	(4,5)	0	0	0	- 1

$= Z_T$

arcs :		(1,3)	(2,4)	(4,3)	(4,5)
sommets :	2	0	1	0	0
	3	- 1	0	- 1	0
	4	0	- 1	1	1
	5	0	0	0	-1

$= A_T$

et $Z_T * A_T = I_4$

Dans la suite , la matrice triangulée est utilisée pour résoudre les systèmes ; tandis que la matrice inverse l'est pour l'interprétation graphique des solutions .

Résolution d'un système dual

Pour une base T donnée , le système dual est de la forme :

$$W'A_T = C'_T \qquad (1.13)$$

où

$C_T \in \mathbb{R}(T)$: vecteur des coûts des arcs de T

$W \in \mathbb{R}(V-1)$: n-1 inconnues (variables duales)

Algorithme 1.2 ALGORITHME DE RESOLUTION DU DUAL

0. Triangulation

Renuméroter les sommets de V selon un ordre transversal .

$W_1 := 0$ (W_1 est une variable artificielle)

i := 2

1. Calcul de W_i

Considérer l'arc e ∈ T d'extrémités i et p(i)

si (e = (p(i),i)) alors $W_i := W_{p(i)} - C_{(p(i),i)}$

sinon $W_i := W_{p(i)} + C_{(i,p(i))}$

2. si (i = n) alors W est la solution

sinon i := i + 1 retour en 1

La justification de cet algorithme découle immédiatement de la forme triangulée de A_T .

Cette solution aurait aussi pu être obtenue comme :

$$W' = C'_T Z_T$$

On remarque que la variable W_i est le résultat du produit scalaire entre le vecteur coût et la colonne indicée par i de Z_T caractérisant la chaîne de i à la racine . La variable W_i peut donc s'interpréter comme le coût de la chaîne de i

à la racine . (Le ***coût d'une chaîne*** se calculant comme la somme des coûts des arcs orientés positivement sur cette chaîne moins la somme des coûts des arcs orientés négativement .) Pour choisir parmi toutes les variables hors base celle qui va entrer dans la base , il est nécessaire de savoir évaluer pour tout arc (i,j) ∈ E - T l'expression suivante :

$$C_{(i,j)} - W'A_{.(i,j)} = C_{(i,j)} + W_j - W_i \qquad (1.14)$$

(notation : $A_{.(i,j)}$ est la colonne de A indicée par l'arc (i,j))

Cette valeur représente le coût de l'unique cycle créé par l'arc (i,j) et l'arbre (V,T) , ce cycle étant tel que (i,j) soit orienté positivement .

La figure 1.5 résout le système dual associé à la base de l'exemple .

$$\begin{array}{lcc} \text{arcs} & (1,3)\ (4,3) & (2,4)\ (4,5) \\ (W_3\ W_4\ W_2\ W_5) * & \begin{bmatrix} -1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} & = (3\ 3\ 4\ 2) \end{array}$$

solution algébrique :

$$(W_3\ W_4\ W_2\ W_5) = (-3\ 0\ 4\ -2)$$

solution graphique :

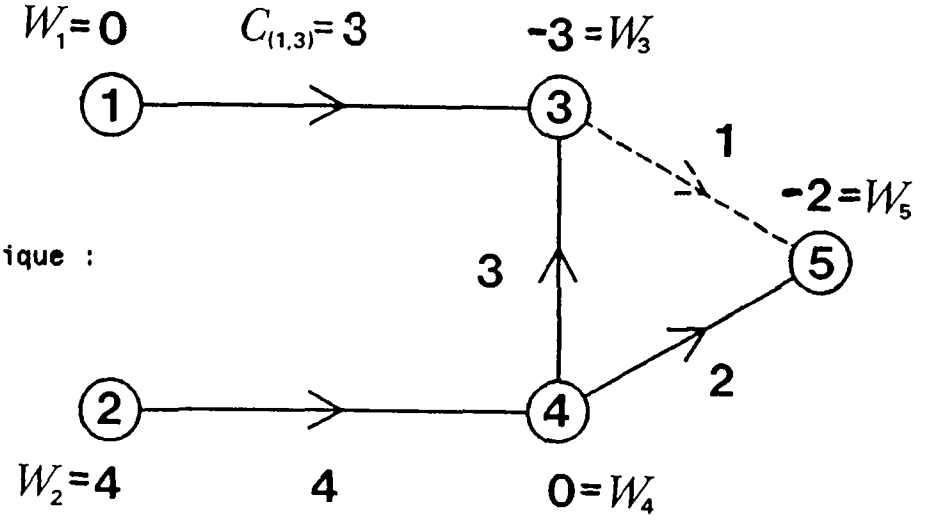


figure 1.5

Le coût du cycle {(4,3) , (4,5) , (3,5)} est : $C_{(3,5)} + W_5 - W_3 = 1 - 2 + 3 = 2$.

Résolution d'un système primal

Pour une base T donnée , un système primal est de la forme

$$A_T Y = Q \qquad (1.15)$$

où

$Q \in \mathbb{R}(V\text{-}1)$: second membre

$Y \in \mathbb{R}(T)$: n-1 inconnues

Algorithme 1.3. ALGORITHME DE RESOLUTION DU PRIMAL

0. Triangulation

Renuméroter les sommets de V selon un ordre transversal

i := n

1. Calcul de $Y_{(i,p(i))}$ ou $Y_{(p(i),i)}$

$Q_{p(i)} := Q_{p(i)} + Q_i$

Considérer l'arc e ∈ T d'extrémités i et p(i)

si (e = (p(i),i)) alors $Y_{(p(i),i)} = - Q_i$

sinon $Y_{(i,p(i))} = Q_i$

2. si (i = 2) alors Y est la solution

sinon i := i - 1 retour en 1

Cet algorithme peut être justifié soit en utilisant la forme triangulée de A_T , soit en considérant l'interprétation de la solution donnée ci-dessous .

Cette solution aurait pu être obtenue par :

$$Y = Z_T Q$$

La variable $Y_{(i,p(i))}$ (resp. $Y_{(p(i),i)}$) est le résultat du produit scalaire entre la ligne indicée par l'arc (i,p(i)) (resp. (p(i),i)) et le second membre . Cette variable est donc égale à la somme (resp. à l'opposé de la somme) des composantes du second membre correspondant à i et à ses descendants .

Dans l'exemple considérons le système primal suivant :

arcs : (1,3) (4,3) (2,4) (4,5)

sommets : 3, 4, 2, 5

$$\begin{bmatrix} -1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} Y_{(1,3)} \\ Y_{(4,3)} \\ Y_{(2,4)} \\ Y_{(4,5)} \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 2 \\ -5 \end{bmatrix}$$

solution algébrique :

$$\begin{bmatrix} Y_{(1,3)} \\ Y_{(4,3)} \\ Y_{(2,4)} \\ Y_{(4,5)} \end{bmatrix} = \begin{bmatrix} +2 \\ -3 \\ 2 \\ 5 \end{bmatrix}$$

solution graphique :

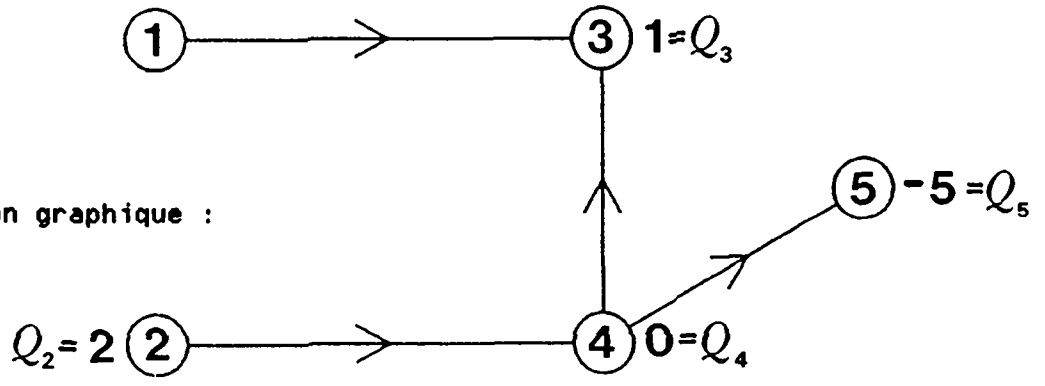


figure 1.6

Pour calculer la direction de descente $D \in \mathbf{R}(T)$ associée à un arc hors base $(i,j) \in E - T$, on considère un système primal particulier où le second membre est la colonne de A indicée par (i,j) :

$$A_T D = A_{(i,j)} \qquad (1.16)$$

donc $D = Z_T A_{(i,j)} = (Z_T)_{.i} - (Z_T)_{.j}$

Cette direction D est donc la différence des vecteurs caractérisant les chaînes de i et de j à la racine . Ainsi D caractérise l'unique chaîne de l'arbre allant de i à j .

La direction de descente associée à l'arc (3,5) de l'exemple est calculée dans la figure 1.7.

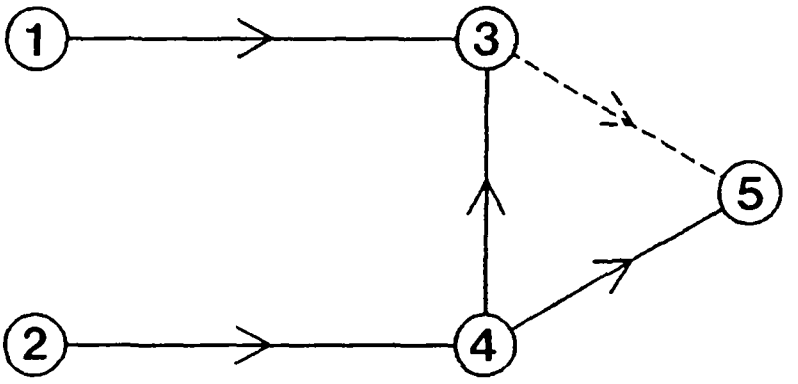
arcs : (1,3) (4,3) (2,4) (4,5)

sommets : 3, 4, 2, 5

$$\begin{bmatrix} -1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} D(1,3) \\ D(4,3) \\ D(2,4) \\ D(4,5) \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ -1 \end{bmatrix}$$

solution algébrique :

$$\begin{bmatrix} D(1,3) \\ D(4,3) \\ D(2,4) \\ D(4,5) \end{bmatrix} = \begin{bmatrix} 0 \\ -1 \\ 0 \\ +1 \end{bmatrix}$$



solution graphique :

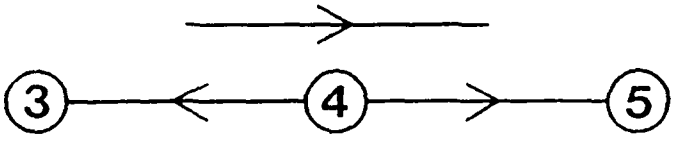


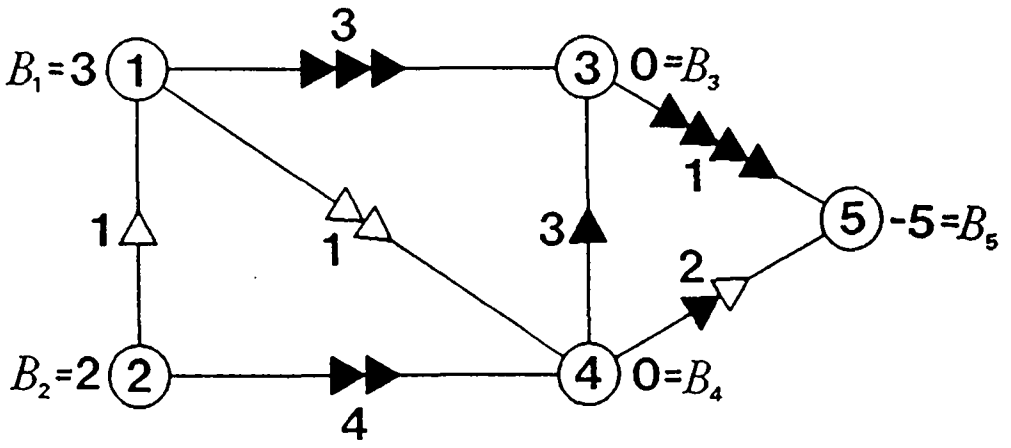
figure 1.7

Solution de base

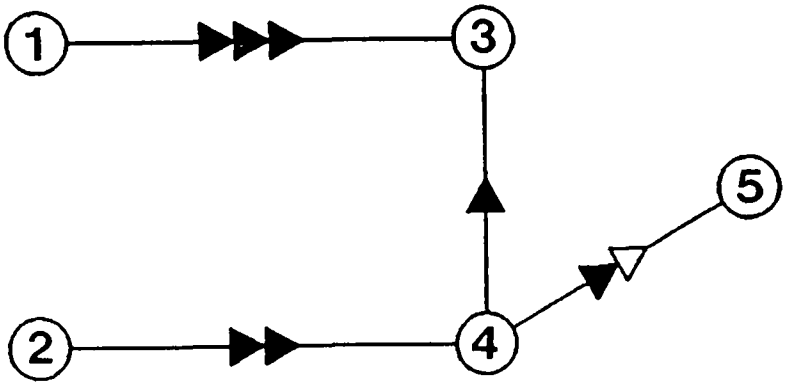
Pour un réseau R = (V,E) avec bilan $B \in \mathbf{R}(V)$, capacité $U \in \mathbf{R}_+(E)$ et une base $T \subset E$, un ***flot de base*** $X \in \mathbf{R}(E)$ est un flot avec

$$\forall\ (i,j) \in E - T \qquad X_{(i,j)} = \begin{cases} 0 & \text{l'arc est dit } \textit{vide} \\ U_{(i,j)} & \text{l'arc est dit } \textit{saturé} \end{cases}$$

La figure 1.8 donne une base et un flot de base pour l'exemple .



flot de base



flot sur la base T

notation :

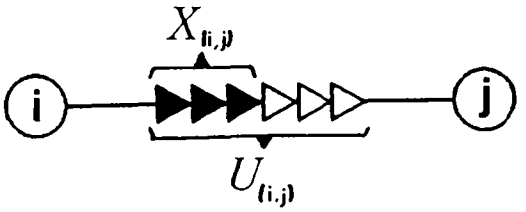


figure 1.8

Algorithme du simplexe

Les composants fondamentaux de l'algorithme du simplexe ayant pu être traduits en termes de réseau , on peut les combiner en un algorithme graphique . Le problème du flot à coût minimum défini par la donnée du réseau R = (V,E) , du vecteur des bilans B , du vecteur des capacités U et du vecteur des coûts C peut être résolu par l'algorithme suivant :

Algorithme 1.4. ALGORITHME DU SIMPLEXE

0. Initialisation

Trouver une base T ⊂ E et un flot de base X

1. Choix de l'arc entrant

1.1. Calculer les variables duales

W_i = coût de la chaîne de i à la racine 1 . $\forall$ i ∈ V ($W_1 = 0$)

1.2. Choisir un arc hors base (p,q) ∈ T tel que

- $C_{(pq)} + W_q - W_p < 0$ si (p,q) est vide
- $C_{(pq)} + W_q - W_p > 0$ si (p,q) est saturé

S'il n'existe pas un tel arc alors X est un flot optimum .

2. Calcul du cycle de descente

Trouver le cycle C formé de l'arc (p,q) et de la chaîne de p à q .

Si l'arc (p,q) est vide , l'orientation donnée à C est telle que (p,q) soit orienté positivement sinon elle est telle que (p,q) soit orienté négativement .

3. Calcul de la modification du flot

$\Delta_- = \min \{ X_{(i,j)} \mid (i,j)$ est orienté négativement sur C $\}$

$\Delta_+ = \min \{ U_{(i,j} - X_{(i,j)} \mid (i,j)$ est orienté positivement sur C $\}$

On va faire avancer $\Delta = \min (\Delta_- , \Delta_+)$ unités de flot le long de C pour arriver à un flot de base voisin .

4. Mise à jour du flot et de la base (pivotage)

4.1. Mise à jour du flot

On modifie le flot des arcs de G :

$X_{(i,j)} := X_{(i,j)} + \Delta$ si (i,j) est orienté positivement sur C

$X_{(i,j)} := X_{(i,j)} - \Delta$ si (i,j) est orienté négativement sur C

4.2. Mise à jour de la base

On choisit un arc (k,l) de C qui est :

- soit vide et orienté négativement
- soit saturé et orienté positivement

C'est l'arc sortant de la base .

T := T + (p,q) - (k,l)

retour à 1

(remarque : si $\Delta = U_{(p,q)}$ l'arc sortant peut être (p,q) et la base ne change pas)

Remarques :

1) Si l'on autorise certains arcs à ne pas avoir de capacité ($U \in (\mathbb{R} +\infty)(E)$) , il se peut que la valeur Δ , calculée dans le pas 3 , soit infinie ; dans ce cas le problème ne possède pas de solution optimale finie .

2) L'efficacité de cet algorithme peut être fortement influencée par les stratégies adoptées pour le choix de l'arc entrant et de l'arc sortant (lorsque ces choix ne sont pas uniques) . Certaines règles de choix seront analysées ultérieurement .

3) Cet algorithme peut être interprété comme une recherche de cycles de coûts négatifs . L'optimum est obtenu lorsqu'il n'existe plus de cycle de coût négatif le long duquel on peut faire circuler du flot .

4) Il n'est pas nécessaire de recalculer toutes les variables duales lors d'un pivot qui rentre l'arc (p,q) dans la base T et qui en sort (k,l) . Les seules variables duales à mettre à jour sont celles associées aux sommets déconnectés de la racine par la suppression de (k,l) ; cet ensemble de sommets est noté

$$T_{(k,l)} = \{ i \in V \mid \text{le chemin de } i \text{ à } 1 \text{ dans } T \text{ contient l'arc } (k,l) \}$$

De plus , les variables duales peuvent être modifiées par la procédure suivante :

Algorithme 1.5. MISE A JOUR DES VARIABLES DUALES

1. **Recherche des variables à modifier**

   Déterminer $T_{(k,l)}$

2. **Calcul de l'amplitude Δw de la modification**

   $\Delta w := C_{(p,q)} + W_q - W_p$ si $p \in T_{(k,l)}$

   $\Delta w := - (C_{(p,q)} + W_q - W_p)$ si $q \in T_{(k,l)}$

3. **Mise à jour**

   Pour tout $i \in T_{(k,l)}$ poser

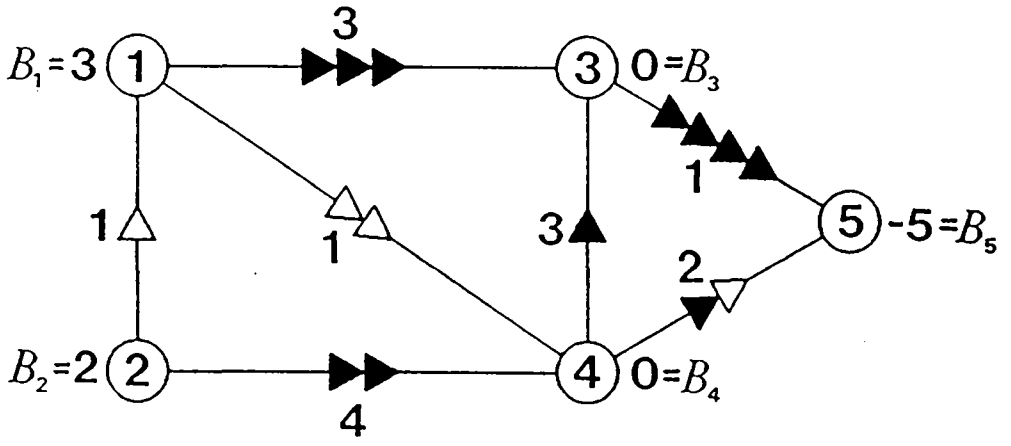
   $$W_i := W_i + \Delta w$$

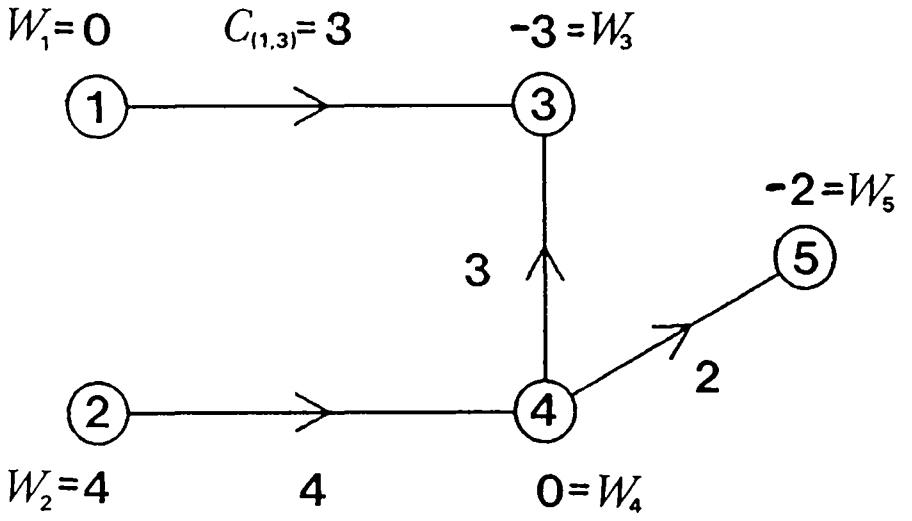
Cette procédure s'insère dans le pas 4.2) avant la modification de T et rend superflu le pas 1.1) .

5) Lors de chaque pivot le coût du flot de base diminue de

$$\Delta^* (\mid C_{(p,q)} + W_q - W_p \mid)$$

Dans la figure 1.9 , cet algorithme est appliqué au problème de flot énoncé dans le paragraphe 1.1 , en partant du flot de base de la figure 1.8 .



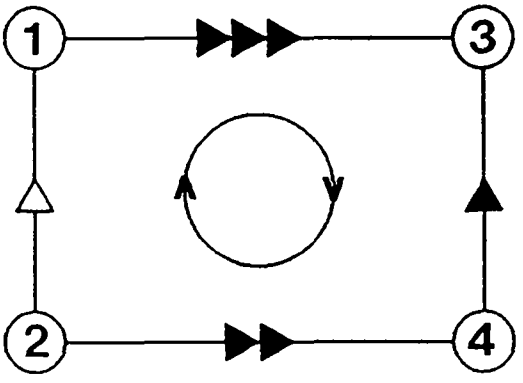


(i,j)	état	$C_{(i,j)} + W_j - W_i$
(1,4)	v	1
(2,1)	v	- 3
(3,5)	s	2

On choisit (2,1) comme arc entrant

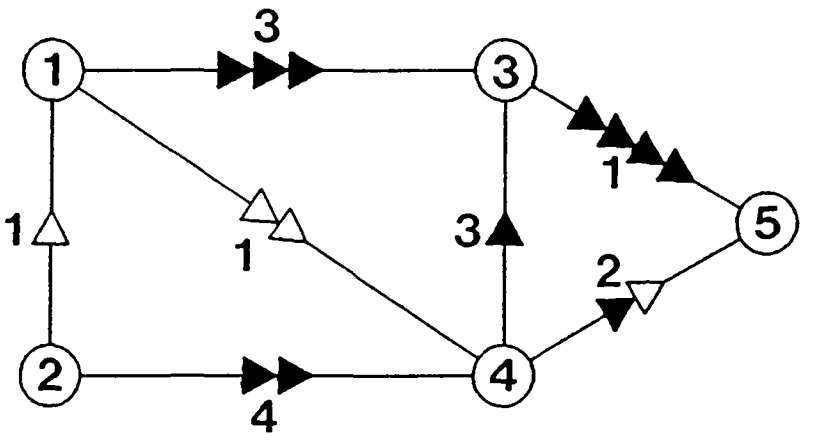
$\Delta = 0$

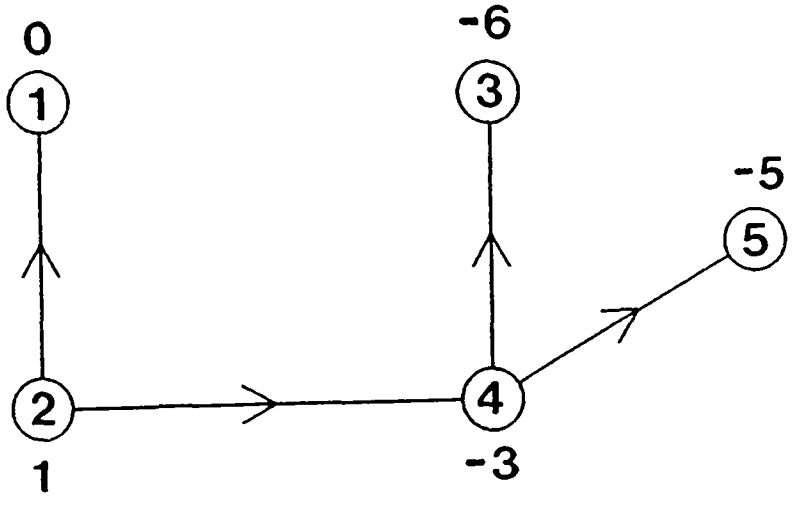
Le seul arc sortant possible est (1,3)



Pivot 1

figure 1.9



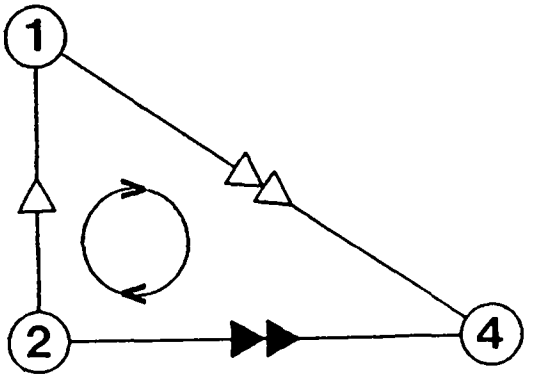


(i,j)	état	$C_{(i,j)} + W_j - W_i$
(1,4)	v	- 2
(3,5)	s	2
(1,3)	s	- 3

On choisit (1,4) comme arc entrant

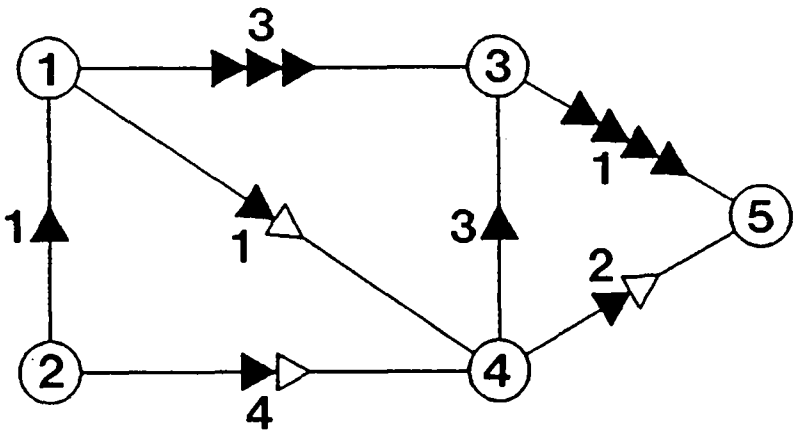
$\Delta = 1$

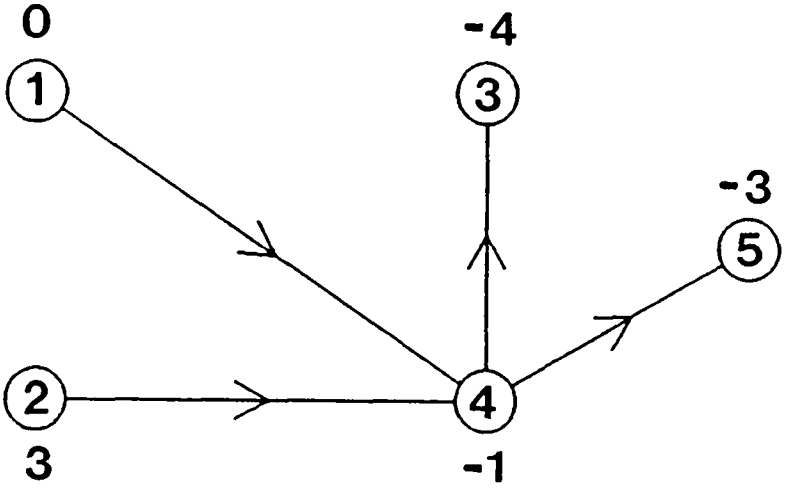
Le seul arc sortant possible est (2,1)



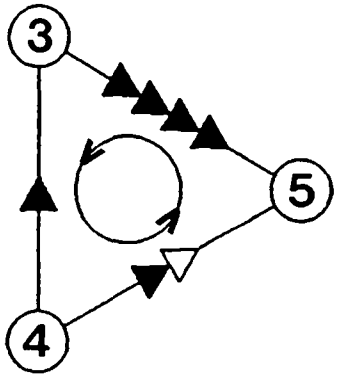
Pivot 2

figure 1.9 (suite)





(i,j)	état	$C_{(i,j)} + \Psi_j - \Psi_i$
(1,3)	s	- 1
(2,1)	s	- 2
(3,5)	s	2



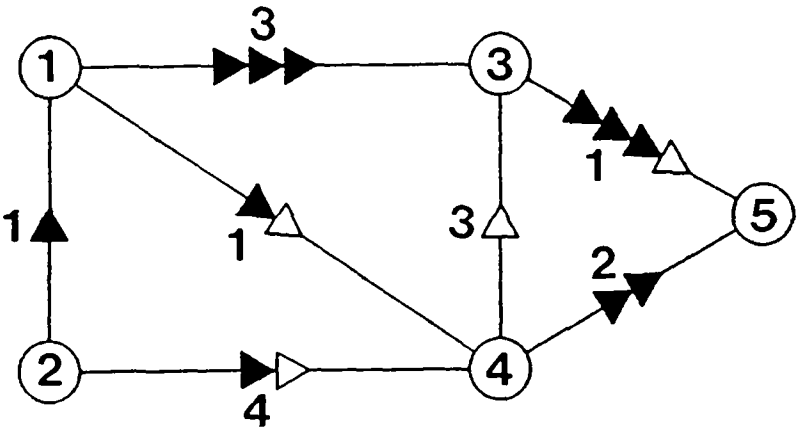
L'unique arc entrant possible est (3,5)

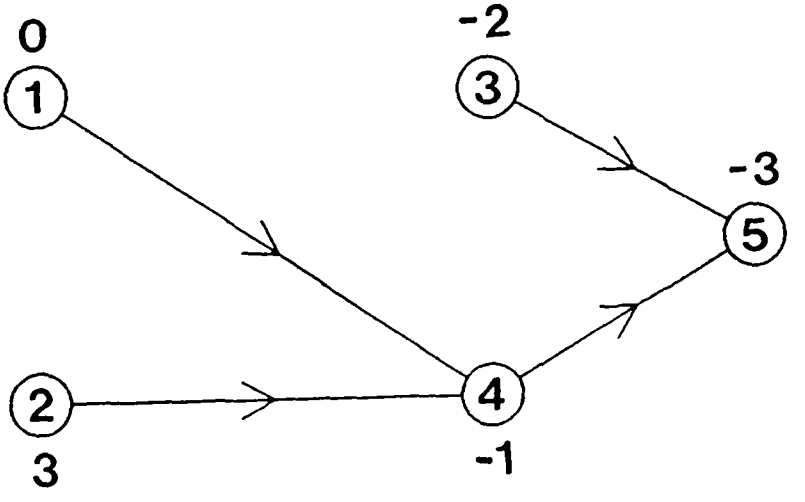
$\Delta = 1$

(4,3) est vidé et (4,5) est saturé par cette modification ;
on choisit (4,3) comme arc sortant

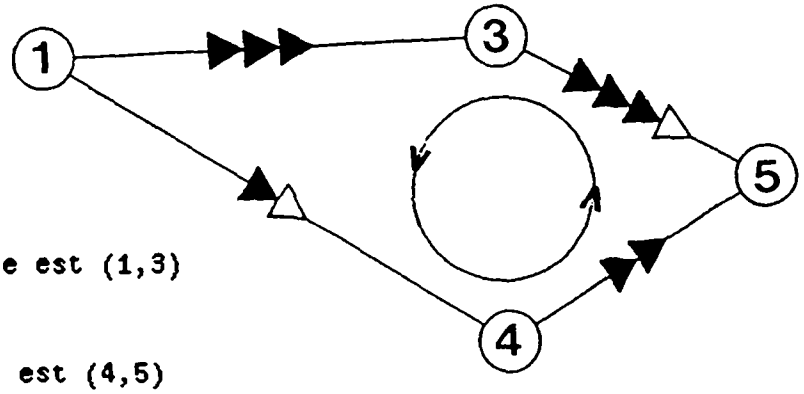
Pivot 3

figure 1.9 (suite)





(i,j)	état	$C_{(i,j)} + \Psi_j - \Psi_i$
(1,3)	s	1
(2,1)	s	- 2
(4,3)	v	2



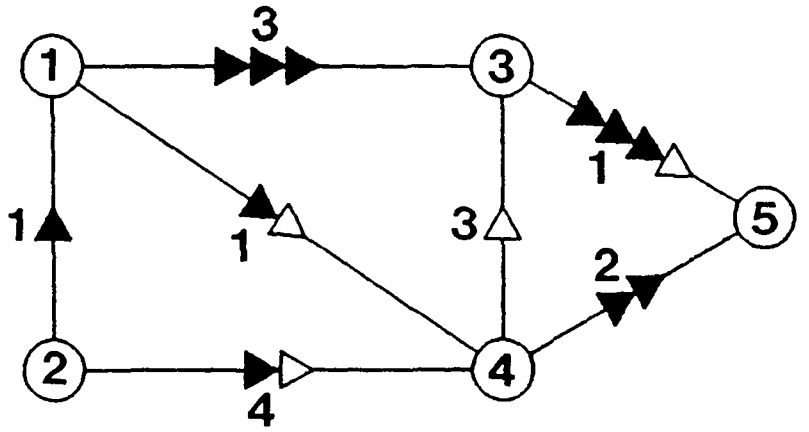
L'unique arc entrant possible est (1,3)

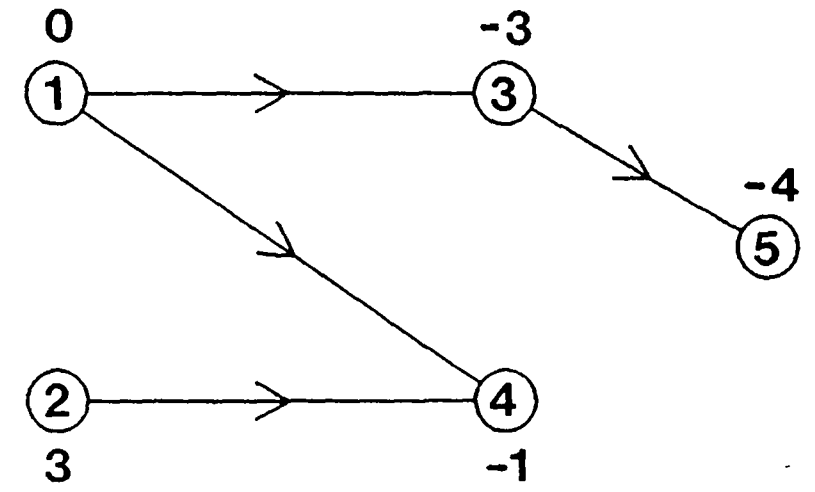
$\Delta = \theta$

Le seul arc sortant possible est (4,5)

Pivot 4

figure 1.9 (suite)



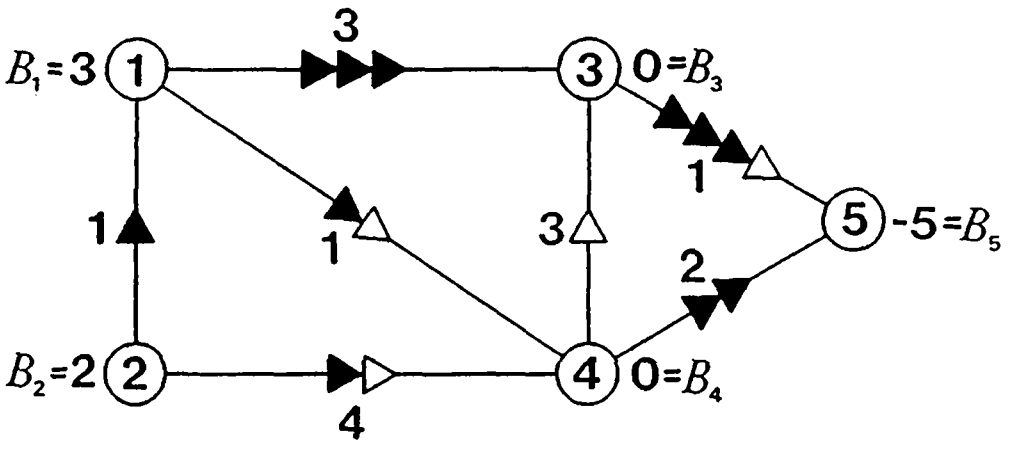


(i,j)	état	$C_{(i,j)} + W_j - W_i$
(2,1)	s	- 2
(4,3)	v	1
(4,5)	s	- 1

Il n'existe plus de candidat pour entrer dans la base.

figure 1.9 (suite)

La solution du problème est :



Son coût est de 22 .

figure 1.9 (fin)

1.2.3. Obtention d'un flot de base

Pour résoudre par l'algorithme du simplexe un problème de flot donné par R = (V,E) , B , U et C il est nécessaire de connaître une solution de base ; or ceci n'est pas toujours évident . La méthode appelée " ***big - M*** "(voir [2],[13]) considère un problème modifié , possédant un flot de base trivial et fournissant la même solution optimale . Ce nouveau réseau est formé :

- du réseau R

- d'un sommet fictif numéroté 1 qui sera la racine . Son bilan est nul .

- de n arcs fictifs ainsi définis :
  - pour tout i ∈ V avec $B_i > 0$, on ajoute un arc fictif (i,1) de capacité infinie et de coût M .
  - pour tout i ∈ V avec $B_i \leq 0$, on ajoute un arc fictif (1,i) de capacité infinie et de coût M .

La base T formée des arcs fictifs admet le flot de base défini comme suit :

- les producteurs ($B_i > 0$) livrent leur flot à la racine le long des arcs fictifs .

- les consommateurs ($B_i < 0$) reçoivent le flot de la racine par les arcs fictifs .

- les autres arcs sont vides .

En choisissant M suffisamment grand , les arcs fictifs seront vides dans la solution optimale , pour autant que le problème original possède une solution . Toute borne supérieure de la longueur des chaînes de R peut fournir une valeur pour M .
Par exemple :

$$M = n * \left(\max_{(i,j)\in E} |C_{(i,j)}| \right) + 1$$

La solution optimale du problème modifié s'obtient par l'algorithme du simplexe ; deux cas peuvent se présenter :

- soit tous les arcs fictifs sont vides et alors la solution restreinte du réseau R est également optimale pour le réseau original .

- soit un ou plusieurs arcs fictifs ont un flot non nul et alors le problème original ne possède pas de solution .

Dans le second cas , les variables duales permettent d'exhiber un ensemble de sommets O pour lequel toutes les équations de bilan (1.3) ne peuvent être respectées ; en effet considérons

$$O = \{ i \in V \mid W_i < 0 \}$$

la solution étant optimale pour le problème modifié , on vérifie aisément que les arcs sortant de O sont saturés et que ceux entrant en O sont vides . Le ou les arcs fictifs restant permettent ainsi de faire transiter tout le surplus de flot produit par les sommets de O ; ce surplus peut être calculé comme :

$$\sum_{i \in O} B_i - \sum_{(i,j) \in E \; i \in O} U_{(i,j)} > 0$$

On associe à O une équation de bilan en sommant celles des sommets de O :

$$\sum_{(i,j) \in E \; i \in O} X_{(i,j)} - \sum_{(j,i) \in E \; i \in O} X_{(j,i)} = \sum_{i \in O} B_i$$

Etant donné l'inégalité précédente , cette équation ne peut être vérifiée . La recherche d'un tel ensemble O peut être utilisée pour modifier les données du problème de manière à ce qu'il admette une solution .

1.2.4. Cyclage

Les pivots de l'algorithme du simplexe sont de deux types :

- les *pivots dégénérés* qui ne modifient que la base T et les variables duales W mais qui laissent le flot inchangé (cas où $\Delta = 0$)

- les *pivots non-dégénérés* qui modifient également le flot ($\Delta > 0$) .

L'algorithme présenté peut produire une suite infinie de pivots dégénérés , résultat de la rencontre cyclique d'une même séquence de solutions de base . Ce phénomène est connu sous le nom de ***cyclage*** . On va présenter une règle pour le choix de l'arc sortant due à Cunningham ([4],[5]) qui évite le cyclage et assure ainsi que l'algorithme s'achève en un nombre fini de pivots .

Un premier théorème caractérise les suites de pivots qui ne peuvent pas cycler :

Proposition 1.4.

Si pour tout pivot dégénéré faisant passer de la base T à la base $S = T + (p,q) - (k,l)$ **, l'arc entrant** (p,q) **est tel que :**

- $q \in T_{(k,l)}$ **si** (p,q) **est vide**
- $p \in T_{(k,l)}$ **si** (p,q) **est saturé**

alors l'algorithme du simplexe ne cycle pas .

Démonstration :

A toute base T et tout flot de base X on associe deux valeurs :

$$G(T,X) = C'X \qquad H(T,X) = \sum_{i=2}^{n} W_i$$

où W représentent les variables duales correspondant à T .

Observons le comportement de ces fonctions lors d'un pivot qui fait passer de la base T à la base $S = T + (p,q) - (k,l)$ et le flot de base X au flot Y:

- si le pivot est non dégénéré ($X \neq Y$) le principe de l'algorithme nous assure que

$$\mathbf{G}(T,X) = C'X > C'Y = \mathbf{G}(S,Y)$$

- si le pivot est dégénéré ($X = Y$) on a

$$\mathbf{G}(S,Y) = \mathbf{G}(T,X)$$

et par la remarque 4 page 30

$$\mathbf{H}(S,Y) = \mathbf{H}(T,X) + \Delta w * |T_{(k,l)}|$$

les propriétés de l'arc entrant (p,q) impliquent que $\Delta w > 0$; en conséquence

$$\mathbf{H}(S,Y) > \mathbf{H}(T,X)$$

Ainsi si l'on considère la suite (T_1 X_1) , (T_2 X_2) , (T_3 X_3) ,... des bases et des flots de base associés produite par l'algorithme du simplexe , on ne peut pas trouver de cycle ; une séquence de pivots non-dégénérés diminuant **G** alors qu'une séquence de pivots dégénérés laisse **G** invariant mais augmente **H**.

Q.E.D.

Pour satisfaire les hypothèses de ce théorème , il est nécessaire de pouvoir exercer un contrôle sur les bases et flots de base parcourus . Une base T et un flot de base X sont dits satisfaire la ***propriété de Cunningham*** si

- tout arc vide de T est de la forme (p(i),i) . (On dit que cet arc est dirigé ***loin*** de la racine) .

- tout arc saturé de T est de la forme (i,p(i)) . (On dit que cet arc est dirigé ***vers*** la racine) .

Lorsqu'on utilise la méthode " big - M " présentée dans le paragraphe 1.2.3 , on part d'une base et d'un flot de base qui vérifient cette propriété . Pour qu'elle soit vérifiée tout au long de l'algorithme , il faut choisir judicieusement l'arc sortant pour chaque pivot . La procédure de choix de l'arc sortant (k,l) pour un pivot entrant l'arc (p,q) dans la base T est décrite par l'algorithme 1.6 .

Algorithme 1.6. CHOIX DE L'ARC SORTANT

1. Calcul de la jointure

On détermine la *jointure* j qui est le premier sommet commun des chaînes de p à la racine et de q à la racine .

2. Choix de l'arc sortant

A partir de j on parcourt le cycle C (voir algorithme du simplexe alg. 1.4) et on choisit comme arc sortant (k,l) le premier arc rencontré qui est vidé ou saturé par la modification du flot .

L'application de l'algorithme à l'exemple avait abouti lors du 3ème pivot à la situation suivante (figure 1.5) :

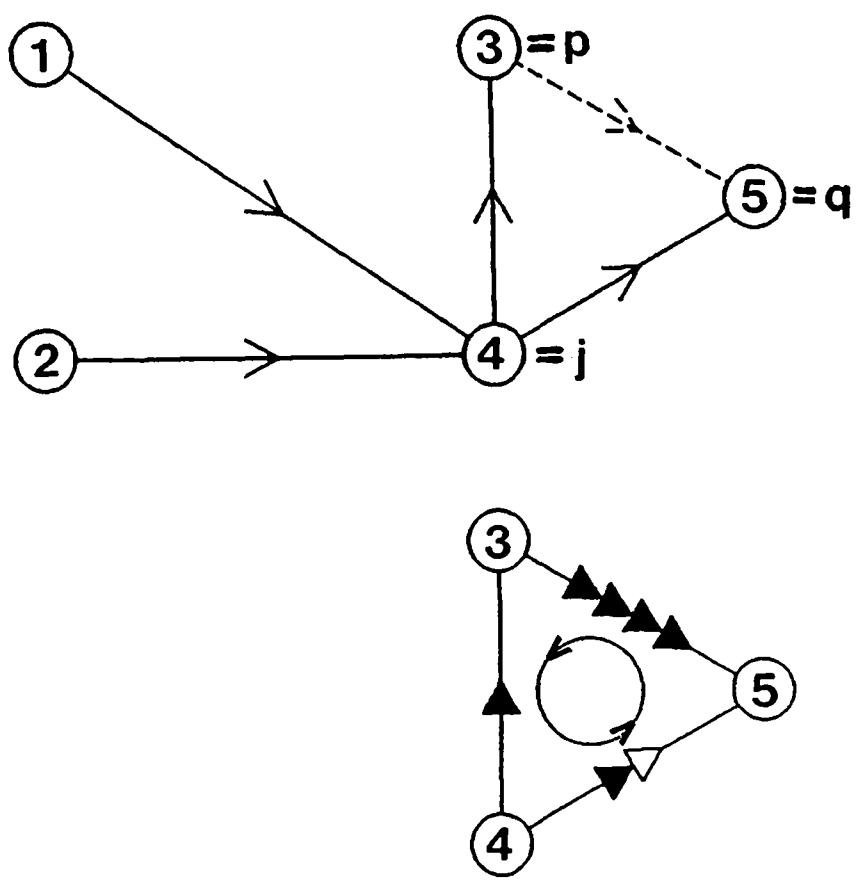


figure 1.10

L'arc entrant est l'arc (3,5) ; la jointure est le sommet 4 qui est le sommet le plus profond en commun sur les chemins de 3 à 1 et de 5 à 1 . En faisant avancer $\Delta = 1$ unité de flot le long de C en partant du sommet 4 on sature l'arc (4,5) avant de vider l'arc (4,3) . L'arc sortant sera par conséquent l'arc (4,5) .

Proposition 1.5.

Un pivot où cette procédure de choix est appliquée préserve la propriété de Cunningham .

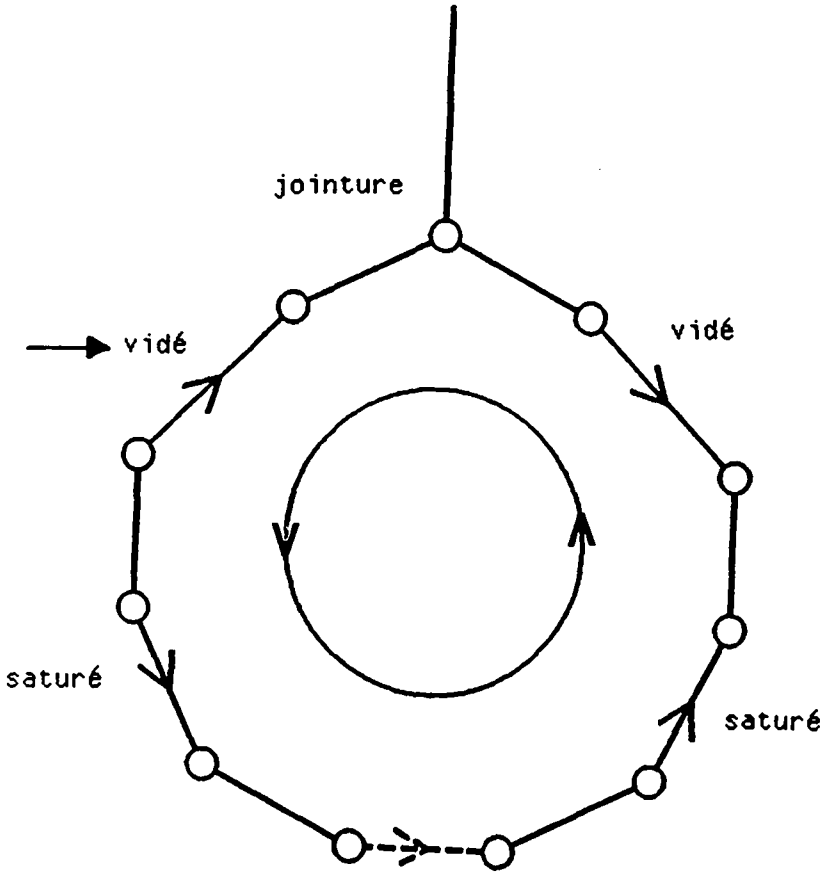
Démonstration :

On part d'une base T et d'un flot X qui ont la propriété ; on entre l'arc (p,q) dans T et on en sort l'arc (k,l) désigné par la procédure ; il faut montrer que la base T' = T + (p,q) - (k,l) et le flot Y ainsi obtenus conservent la propriété (voir figure 1.11) .

- si le pivot est non-dégénéré , les arcs qui ont été vidés sont orientés négativement sur C et ceux saturés sont orientés positivement ; on vérifie qu'en éliminant le premier rencontré depuis la jointure on conserve la propriété .

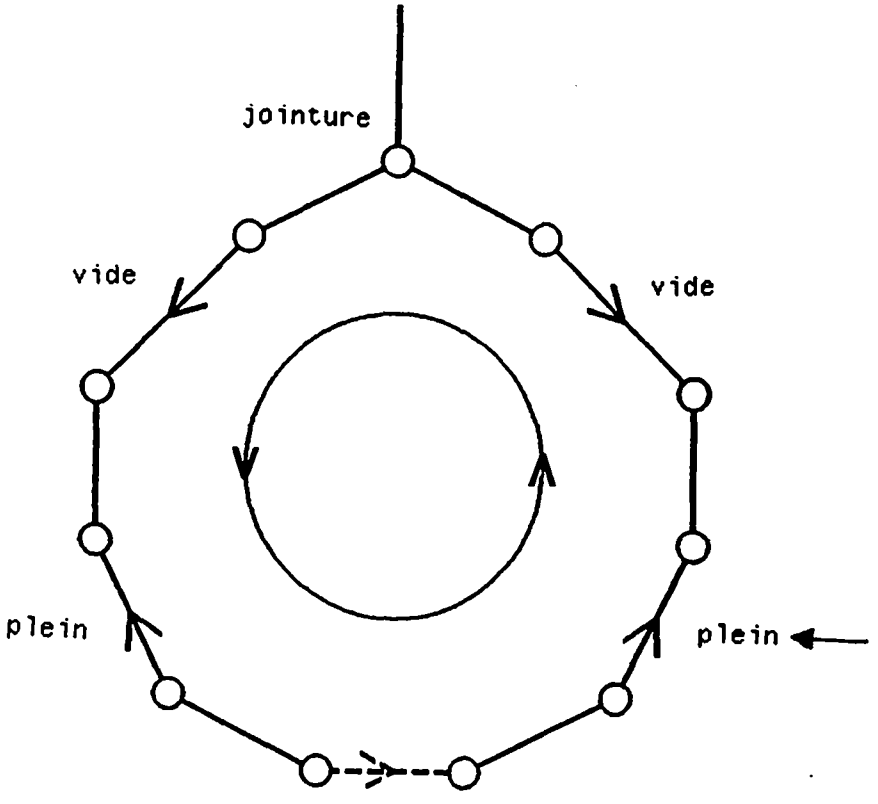
- si le pivot est dégénéré , T et X vérifiant la propriété , les arcs empêchant la modification du flot sont tous situés après l'arc (p,q) sur le cycle C ; ce sont des arcs vides orientés négativement sur C et dirigés loin de la racine ou des arcs saturés orientés positivement et dirigés vers la racine . En éliminant le premier on garde la propriété pour les autres (voir figure 1.12) .

Q.E.D.



cas non dégénéré

figure 1.11



cas dégénéré

figure 1.12

Ces deux propositions nous permettent de construire un algorithme qui ne cycle pas .

Théorème 1.6.

Un algorithme du simplexe où
- **la base et le flot initial respectent la propriété de Cunningham**
- **pour chaque pivot le choix de l'arc sortant se fait par la procédure proposée**

ne peut pas cycler .

Démonstration :

La proposition 1.5. nous assure que la propriété de Cunningham est vérifiée durant toute la durée de l'algorithme . Lors d'un pivot dégénéré , si l'arc entrant est vide il le restera et devra être dirigé loin de la racine dans la nouvelle base ; par contre s'il est saturé , il devra être dirigé vers la racine . Ceci n'est qu'une autre formulation pour l'hypothèse de la proposition 1.4. et ainsi on peut conclure que l'algorithme ne cycle pas .

Q.E.D.

Corollaire 1.7.

Un tel algorithme s'achève en un nombre fini de pivots .

Démonstration :

Le nombre d'arbres d'un réseau étant fini , ainsi que le nombre de flots de base par arbre , un algorithme ne cyclant pas est fini .

Q.E.D.

Bien que fini , le nombre de pivots peut être , en théorie , très grand ; Zadeh [14] a construit une suite de problèmes tels que le k-ième , qui possède 2k + 2 sommets , nécessite 2**k + 2**(k-2) - 2 pivots . De tels problèmes doivent être rares ; en effet la pratique montre que la durée de l'algorithme est proportionnelle à la taille du problème .

1.2.5. Stratégies pour le choix de l'arc entrant

Avant de passer à la programmation de l'algorithme , il est nécessaire de donner une règle pour le choix de l'arc entrant . Cette règle ou ***stratégie*** influence fortement les performances de l'algorithme ; c'est le coeur de celui-ci . L'élaboration d'une telle stratégie est un art basé sur l'intuition , l'expérience et beaucoup de tests .
On définit le ***coût réduit*** de l'arc (i,j) comme suit :

$$Cr_{(i,j)} = \begin{cases} C_{(i,j)} + W_j - W_i & \text{si (i,j) est vide} \\ - C_{(i,j)} - W_j + W_i & \text{si (i,j) est plein} \end{cases}$$

Les arcs ***candidats*** pour entrer dans la base sont ceux dont le coût réduit est négatif . Ce coût réduit d'un arc représente la diminution du coût total provoquée par l'avance d'une unité de flot le long du cycle associé .

La stratégie la plus simple consiste à parcourir la liste des arcs et à choisir le premier arc à coût réduit négatif . En procédant ainsi , la diminution du coût total peut être très petite et par conséquent la convergence lente . Pour accélérer on peut choisir l'arc de coût réduit minimum ; mais la recherche d'un tel minimum ralentit considérablement l'algorithme . En résumé la première stratégie a des pivots de courte durée , tandis que la seconde a un faible nombre de pivots . Entre les deux extrêmes il faut trouver un compromis . Les trois stratégies que nous allons présenter cherchent le meilleur candidat parmi un sous-ensemble des arcs .

Stratégie de Glover , Karney et Klingman ([8])

Les structures de données utilisées pour mémoriser des réseaux permettent un accès aisé à l'information relative aux arcs possédant la même extrémité initiale . Il est donc naturel de chercher , pour un sommet donné , l'arc de coût réduit minimum parmi les arcs qui en sortent ; c'est l'***examen*** du noeud . La stratégie consiste simplement à examiner cycliquement la liste des noeuds et à entrer dans la base le premier arc de coût réduit négatif ainsi obtenu .

Stratégie de Mulvey ([13])

Le principe de base est la création et l'entretien d'une liste de S arcs candidats pour entrer dans la base . On commence par établir cette liste en procédant à l'examen successif des sommets jusqu'à l'obtention de S candidats . Puis on pivote en entrant l'arc de coût réduit négatif minimum parmi ceux de la liste ; ceci jusqu'à épuisement de la liste ou accomplissement de s pivots . La liste est alors vidée , puis remplie en reprenant l'examen des sommets là où on l'avait interrompu . Le processus est contrôlé par un couple de paramètres (s , S) . Pour (s = 1 , S = 1) on retrouve la stratégie précédente et pour (s = 1 , S = n) on obtient la stratégie du coût réduit minimum . Les valeurs conseillées pour les paramètres sont (s = 10 , S = 35) .

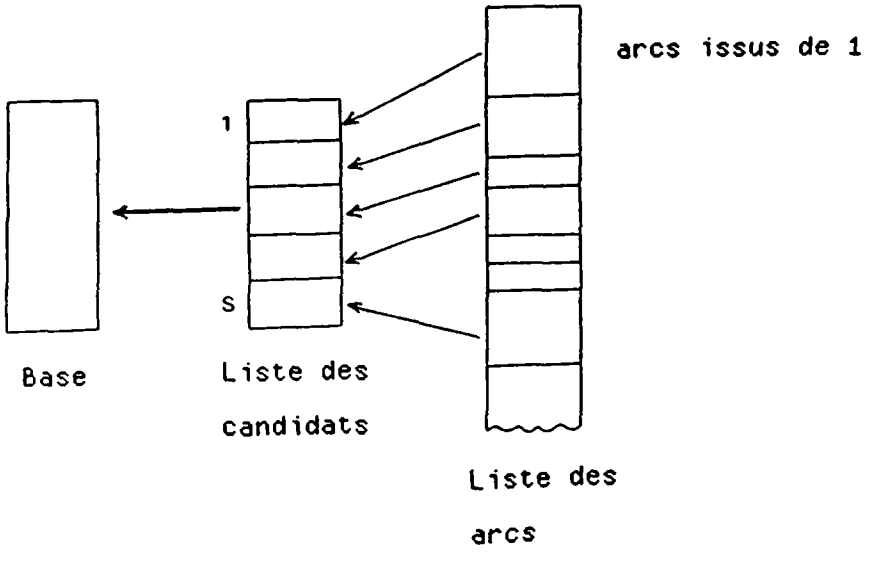


figure 1.13

Stratégie de Bradley , Brown et Graves ([2])

Le déroulement de l'algorithme du simplexe lorsqu'il emploie la méthode " big - M " se scinde en deux phases :

- premièrement il tente de trouver un flot n'utilisant pas les arcs fictifs ; c'est-à-dire qu'il construit des chaînes allant des producteurs aux consommateurs .

- secondement il optimise le flot .

Cette stratégie utilise une queue dont les éléments sont des sommets ou des arcs et qui est gérée de la façon suivante :

- pour choisir un arc entrant on extrait le bloc formé des LB premiers éléments de la queue . Parmi les arcs du bloc et ceux obtenus par examen des sommets du bloc , on retient celui de coût réduit le plus négatif et on réintroduit dans la queue ceux qui ont un coût réduit négatif .

- pour remplir la queue , on examine LP sommets de la liste des sommets ; cette liste est ainsi parcourue cycliquement par "page" de taille LP .

Pour suivre le déroulement du simplexe , la stratégie est différente suivant les phases . Au départ la queue est remplie avec les sommets producteurs . La première phase est composée de NI itérations . Durant chacune de ces itérations , on choisit un arc par le traitement d'un bloc , on pivote et on introduit dans la queue les extrémités de l'arc entré dans la base ; ceci afin de stimuler la création de chaînes dans la base liant les producteurs aux consommateurs . Le remplissage de la queue par l'examen d'une page ne se fait que lorsqu'elle est vide . Durant la seconde phase , plus aucun sommet n'est introduit dans la queue ; on ne fait que la traiter par blocs successifs . Le remplissage s'effectue chaque fois que tous les éléments ont été parcourus . Les valeurs des paramètres conseillées sont les suivantes :

LB = 32 ; LP = n / 10 + 1 ; NI = 3n / 4

Une étude comparative de ces 3 stratégies sera effectuée dans le paragraphe 1.4 consacré aux tests .

1.3. IMPLANTATION

Pour les problèmes de grande taille , l'algorithme du simplexe se compose de milliers ou de dizaines de milliers de pivots ; cette opération , facilement descriptible en termes graphiques , doit donc être programmée avec le plus grand soin . Son efficacité dépend fortement de la structure de données utilisée pour la représentation de la base en mémoire centrale . Ali , Helgason , Kennington et Lall ([1]) ont fait un survol des différentes implantations possibles du pivotage . Celle de Bradley , Brown et Graves ([2]) , considérée comme l'une des meilleures , va être présentée dans ce paragraphe .

1.3.1. Mémorisation du réseau

Pour stocker le réseau R = (V,E) , on utilise deux tableaux TAIL(.) de dimension (n + 1) et HEAD(.) de dimension m . Pour chaque sommet i , la liste de ses voisins est contenue dans le tableau HEAD à partir de la case TAIL(i) . Un arc est caractérisé par son extrémité initiale et son numéro , qui correspond à l'indice de la case de HEAD où est stockée son extrémité terminale . Les capacités U et les coûts C sont mémorisés dans deux tableaux COST(.) et CAP(.) de dimension m + 1 ; les caractéristiques d'un arc sont placées dans les cases indicées par son numéro . Les arcs issus d'un sommet i portent les numéros consécutifs TAIL(i) ,... , TAIL(i + 1) - 1 . Le réseau de la figure 1.2 est représenté par les quatre tableaux présentés dans la figure 1.14 .

Lorsqu'on utilise la méthode " big - M " , le sommet racine fictif porte le numéro (n + 1) ; de plus il n'est pas nécessaire de mémoriser les n arcs fictifs puisqu'ils possèdent les mêmes caractéristiques qui peuvent être stockées dans les cases (m + 1) des tableaux CAP et COST . On prend :

$$\mathrm{COST}(m+1) = n^{*}\left(\max_{(i,j)} |C((i,j))| \right) + 1$$

$$\mathrm{CAP}(m+1) = \sum_{i} |B(i)|$$

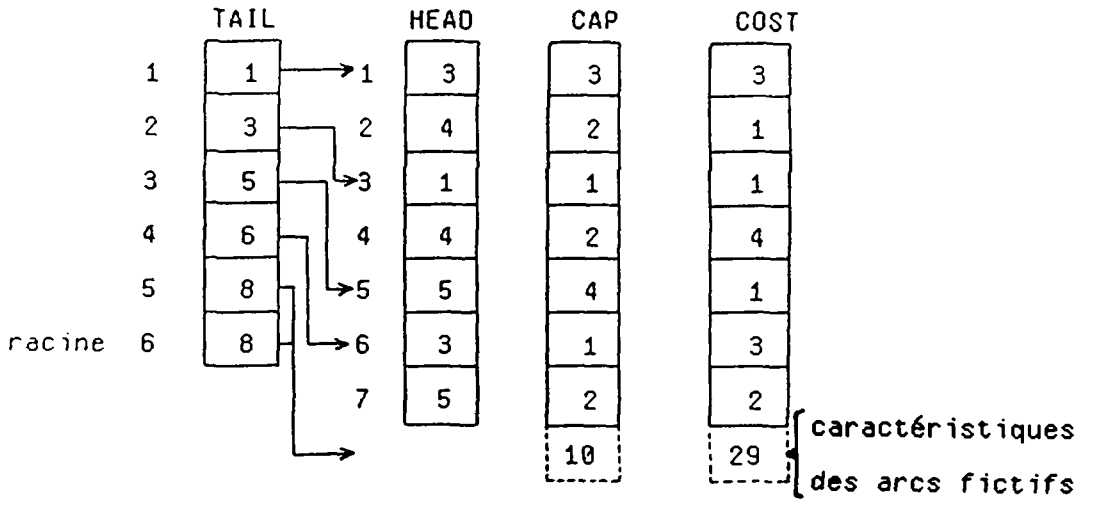


figure 1.14

1.3.2. Mémorisation des bases

Pour une base T , les fonctions prédécesseur et profondeur ainsi qu'un ordre transversal sont mémorisés . Pour ce faire on utilise quatre tableaux de dimension n :

- PRED(.) : on mémorise dans la case PRED(i) le prédécesseur p(i) du sommet i dans T .

- ARC(.) : on stocke dans la case ARC(i) le numéro k de l'arc d'extrémités i et p(i) . On adopte la convention suivante :
  - si l'arc est (i,p(i)) on pose ARC(i) = - k
  - si l'arc est (p(i),i) on pose ARC(i) = + k

- DEPTH(.) : DEPTH(i) contient la profondeur d(i) du sommet i dans T .

- TRAV(.) : ce tableau implante l'ordre transversal en stockant dans TRAV(i) le sommet qui suit i dans l'ordre .

On utilise un cinquième tableau MULT(.) de dimension n pour conserver les

variables duales ; la valeur de la variable W_i est mise dans la case MULT(i) .

Le flot de base X peut être mémorisé uniquement sur les arcs de la base T si on sait reconnaître les arcs hors base qui sont vidés ou saturés . On utilise un tableau VAR(.) de dimension n et une convention sur le signe des valeurs de CAP :

- VAR(.) : on mémorise dans la case VAR(i) le flot qui circule de p(i) à i à travers l'arc de numéro |ARC(i)| avec la convention suivante :
  - si ARC(i) > 0 on a VAR(i) = $X_{(p(i),i)}$
  - si ARC(i) < 0 on a VAR(i) = $U_{(i,p(i))} - X_{(i,p(i))}$

On repère les arcs hors base saturés en stockant dans CAP non pas leur capacité mais l'opposé de leur capacité .

La base T de la figure 1.3 et le flot de base de la figure 1.8 sont mémorisés dans les six vecteurs représentés dans la figure 1.15.

	PRED	ARC	DEPTH	TRAV	MULT	VAR
1	-	-	0	3	0	-
2	4	- 4	3	5	4	0
3	1	+ 1	1	4	- 3	3
4	3	- 6	2	2	0	0
5	4	+ 7	3	1	- 2	1

figure 1.15

De plus l'arc (3,5) étant saturé , CAP(5) = - 4

Lorsqu'on utilise la méthode " big - M " , les arcs fictifs forment la base initiale ; au cours de l'algorithme , ils sortent de la base pour ne plus y entrer . Les bilans B sont mémorisés dans le tableau VAR comme flot de base initial .

La figure 1.16 donne la base initiale obtenue par la méthode " big - M ".

```
capacité des arcs fictifs : 10
coût des arcs fictifs     : 29
```

	PRED	ARC	DEPTH	TRAV	MULT	VAR
1	6	- 8	1	2	29	7
2	6	- 8	1	3	29	8
3	6	8	1	4	- 29	0
4	6	8	1	5	- 29	0
5	6	8	1	6	- 29	5
6	-	-	0	1	0	-

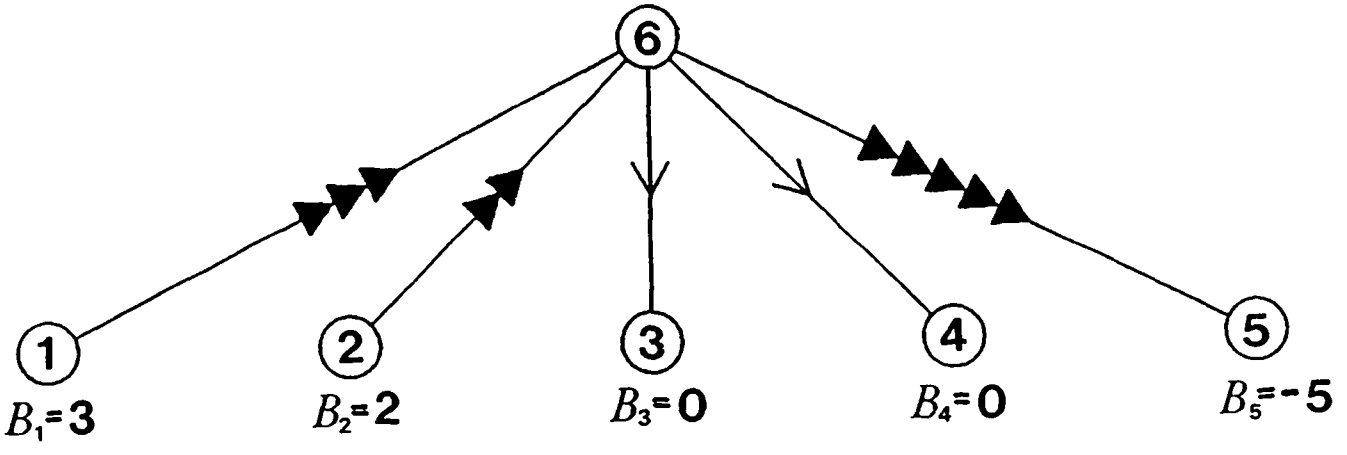


figure 1.16

1.3.3. Calcul de la modification

Sur la base de ces informations , il est possible de programmer une procédure qui , pour un arc entrant (p,q) donné , calcule la modification de flot en déterminant :

- la chaîne de q à p dans la base ainsi que la jointure

- la variation maximum de flot sur le cycle créé par (p,q)

- l'arc sortant (k,l)

Les tableaux PRED et ARC permettent , en passant de prédécesseur en prédécesseur , de connaître la chaîne de la base allant d'un sommet à la racine .

Avec l'information contenue dans DEPTH , il est possible de construire , en parallèle , les chaînes partant de p et de q jusqu'à la jointure et de s'arrêter . On commence par parcourir la chaîne issue du noeud le plus profond (DEPTH maximum) jusqu'à ce que l'on soit parvenu à la profondeur de l'autre sommet . Ensuite on avance de manière synchrone sur les deux chaînes ; lorsqu'elles se rencontrent on a trouvé la jointure .

Procédure 1.1. RECHERCHE DE LA CHAINE DE p A q.

```
Procedure chaîne ( input p,q (* arc entrant *) output jointure )
begin
   qq := q ; pp := p ;
   diffdepth = DEPTH(q) - DEPTH(p) ;
   if ( diffdepth > 0 ) then  (* q est le plus profond *)
      for iter := 1 to diffdepth do qq := PRED(qq)
   else (* p est le plus profond *)
      for iter := 1 to - diffdepth do pp := PRED(pp)
   end if ;
   (* pp et qq ont la même profondeur *)
```

```
   while ( pp ≠ qq ) do
      begin
       pp := PRED(pp) ; qq := PRED(qq)
       end ;
   (* on a atteint la jointure *)
   jointure := pp ;
end .
```

La détermination de la variation du flot Δ et de l'arc sortant peut être incorporée à cette procédure . La convention adoptée pour le tableau VAR facilite le calcul de Δ ; par exemple si l'arc (p,q) est vide on a :

$$\Delta = \min \begin{cases} CAP(.) & \text{pour l'arc (p,q)} \\ VAR(.) & \text{pour les arcs de la chaîne de q à la jointure} \\ CAP(.) - VAR(.) & \text{pour les arcs de la chaîne de p à la jointure} \end{cases}$$

1.3.4. Mise à jour du flot et de la base

Le choix de l'arc entrant (p,q) , de l'arc sortant (k,l) et le calcul de la modification du flot sont suivis de l'opération de pivotage , qui nécessite une mise à jour :

- du flot de base (tableau VAR) .

- de la base (tableaux TRAV , PRED , ARC , DEPTH) .

- des variables duales (tableau MULT) .

Le flot n'est modifié que sur l'arc entrant (p,q) et la chaîne de l'arbre allant de q à p . La mise à jour de VAR s'effectue sur les chaînes liant p et q à la

jointure ; ceci en passant de prédécesseur en prédécesseur .

Graphiquement , la transformation de la base T en la base T + (p,q) - (k,l) peut se diviser en deux phases : premièrement en enlevant l'arc (k,l) on déconnecte les sommets de $T_{(k,l)}$ de la racine , puis on les reconnecte par l'ajout de l'arc (p,q) . On définit ***l'axe du pivot*** comme étant la chaîne de T ayant tous les sommets dans $T_{(k,l)}$ qui lie une extrémité de l'arc entrant à une de l'arc sortant .

L'ordre transversal pour la base T doit être transformé en un ordre pour la base T + (p,q) - (k,l) . La procédure modifiant le tableau TRAV visite une et une seule fois les sommets de $T_{(k,l)}$. Elle traite séquentiellement les sommets de l'axe du pivot en partant de l'extrémité de l'arc entrant pour finir à celle de l'arc sortant . Lors du traitement d'un sommet i de l'axe , les descendants des sommets déjà traités ont été visités . Les descendants de i qui restent à visiter se divisent donc en deux groupes :

- les descendants à gauche de l'axe , qui sont visités selon l'ordre transversal (en itérant TRAV) depuis i jusqu'au dernier sommet de l'axe traité .

- les descendants à droite de l'axe , qui sont les descendants de i non visités restants . Ils sont visités selon l'ordre transversal jusqu'à l'obtention d'un sommet moins profond que i .

La description détaillée de cette procédure est la suivante :

Procédure 1.2. MISE A JOUR DE L'ORDRE TRANSVERSAL

```
procedure mise-à-jour ( input p,q (* arc entrant *) output k,l (* arc sortant *))
(* supposons que : DEPTH(l) = DEPTH(k) + 1 et q ∈ T(k,l) *)
begin
   (* visite des descendants de q *)
   d := q
   while ( DEPTH(TRAV(d)) > DEPTH(q) ) do d := TRAV(d)
```

```
    dernier := d (* dernier descendant de q *)
    pendant := TRAV(d)
    (* initialisation *)
    i := q
    niter := DEPTH(q) - DEPTH(l)
    (* l'axe du pivot est la chaîne de q à l longue de niter sommets *)
    for iter := 1 to niter do
       begin
       (* traitement du noeud i de l'axe *)
       j := i
       i := PRED(i)
       TRAV(dernier) := i
       (* visite des descendants de i à gauche de l'axe *)
       d := i
       while ( TRAV(d) ≠ j ) do d := TRAV(d)
       dernier := d
       if ( DEPTH(pendant) > DEPTH(i) ) then
          (* visite des descendants de i à droite de l'axe *)
          begin
          TRAV(d) := pendant
          while ( DEPTH(TRAV(d)) > DEPTH(i) ) do d := TRAV(d)
          dernier := d
          pendant := TRAV(d)
          end
       end
    (* visite des descendants de k à gauche de l'axe *)
    d := k
    while ( TRAV(d) ≠ l ) do d := TRAV(d)
    (* les sommets de T(k,l) sont placés juste après p dans l'ordre *)
    TRAV(d) := pendant
    TRAV(dernier) := TRAV(p)
    TRAV(p) := q
end ;
```

L'effet de cette procédure est illustré par les figures 1.17 et 1.18 .

ordre transversal : 1,2,3 ,...,17

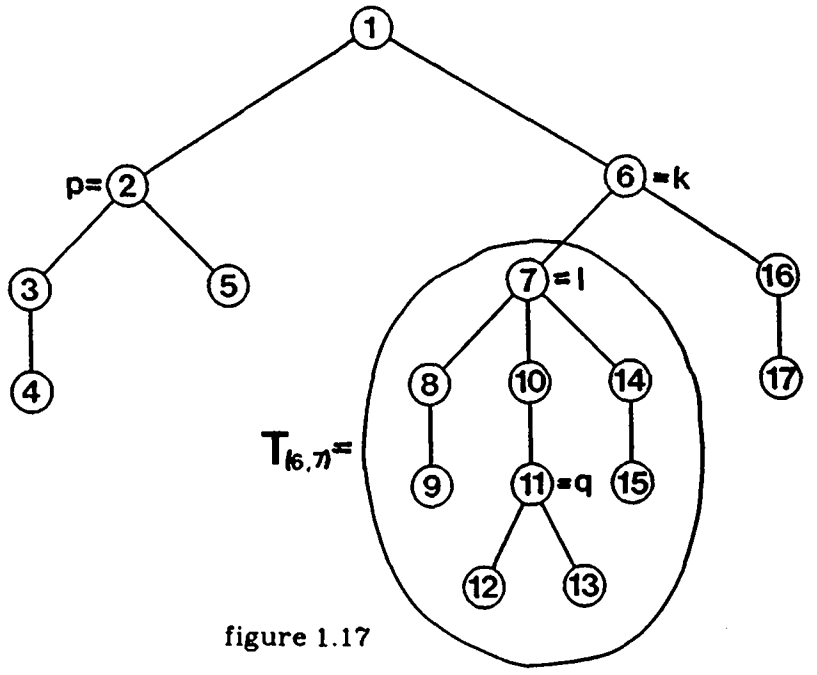


figure 1.17

T + (2,11) - (6,7)

axe : 7,10,11

ordre tranversal : 1,2,11,12,13,10,7,8,9,14,15,3,4,5,6,16,17

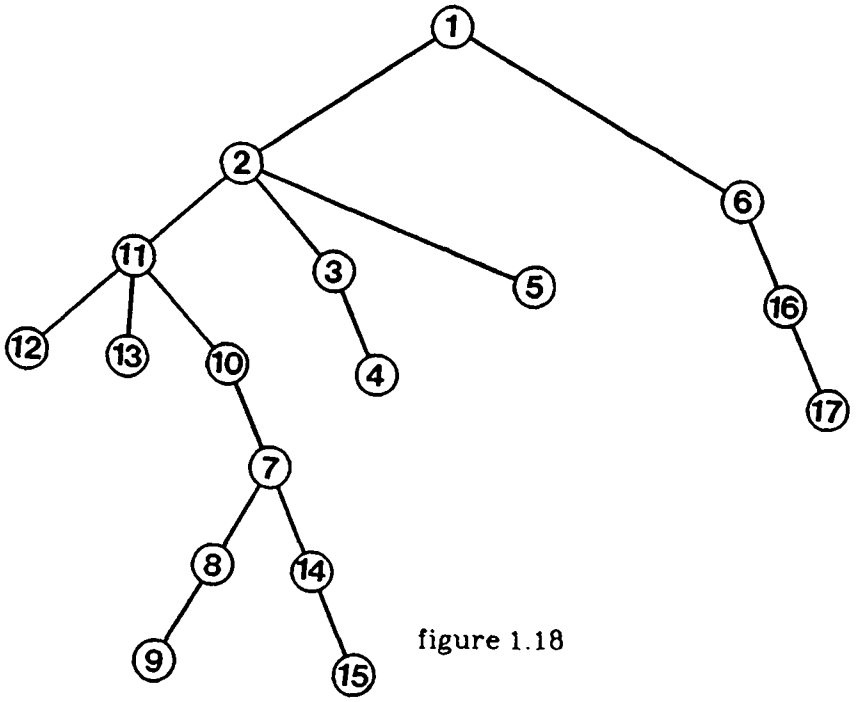


figure 1.18

Le pivotage a pour effet d'inverser la fonction de précédence pour les sommets de l'axe . La modification des tableaux PRED , ARC et VAR qui en résulte peut être introduite dans la procédure 1.2.

Seules les profondeurs des sommets de $T_{(k,l)}$ étant modifiées , la mise à jour du tableau *DEPTH* peut être incorporée à la procédure 1.2 . Lors du traitement d'un sommet i de l'axe , on calcule sa nouvelle profondeur ; la variation de profondeur des descendants de i à gauche et à droite de l'axe est la même que celle observée en i .

L'algorithme 1.5 pour la mise à jour des variables duales est facilement programmable . Durant la procédure 1.2. il suffit d'ajouter Δw à MULT(i) lorsqu'on visite le sommet i .

L'efficacité de cette procédure 1.8. de mise à jour réside dans le fait qu'elle visite uniquement les sommets pour lesquels les composantes des tableaux sont modifiées . Un énoncé détaillé peut être trouvé dans l'article de Ali , Helgason , Kennington et Lall [1] .

Pour mémoriser le réseau , la base et le flot on utilise sept tableaux de dimension n et trois tableaux de dimension m . Cette implantation est un bon compromis entre la rapidité et la place mémoire utilisée .

1.4. TESTS

Le but de ce chapitre est de comparer l'efficacité des trois stratégies de choix de l'arc entrant présentées en 1.2.5. . Selon les " règles de l'art " définies dans le paragraphe 1.3 , trois programmes ont été élaborés qui diffèrent uniquement par la routine de choix de l'arc entrant .

nom du programme	Stratégie utilisée	place mémoire requise
PNET	Glover , Karney et Klingmann	3m + 7n
LPNET	Mulvey	3m + 7n + 70
GNET	Bradley , Brown et Graves	3m + 9n

Ces trois programmes ont été écrits en Fortran 5 respectant la norme ANSI .

1.4.1. Les problèmes

De tels tests nécessitent la connaissance d'une batterie de problèmes . Trente problèmes ont été générés aléatoirement à l'aide d'un programme semblable à celui proposé par Klingmann , Napier et Stutz ([11]) .

Les douze paramètres d'entrée du programme permettent de contrôler la structure , la dimension et certaines caractéristiques du problème à générer .

1. *nombre de sommets*

2. *nombre d'arcs*

3. ***nombre de sources*** : une source est un sommet producteur où n'aboutit aucun arc

4. ***nombre de puits*** : un puits est un sommet consommateur d'où ne sort aucun arc

5. ***nombre de sommets de transit producteurs*** : un sommet de transit producteur est un sommet producteur où aboutit au moins un arc

6. ***nombre de sommets de transit consommateurs*** : un sommet de transit consommateur est un sommet consommateur d'où sort au moins un arc

7. ***production totale***

8. ***proportion de coût maximum*** : proportion des arcs qui ont le coût le plus élevé . Les autres arcs ont un coût réparti aléatoirement entre 1 et le coût maximum .

9. ***coût maximum***

10. ***proportion de capacité*** : proportion des arcs qui ont une capacité finie

11,12. ***les capacités minimales et maximales*** : les capacités finies seront générées uniformément entre ces deux bornes .

Les problèmes T1 , ... , T10 sont des problèmes de transport ; tout sommet est soit source soit puits et les arcs vont d'une source à un puits . Les problèmes F1 , ... , F20 sont des problèmes de flot généraux de dimensions croissantes . Pour que l'étude soit représentative , les paramètres de contrôle ont été choisis de manière à obtenir des problèmes les plus différents possibles . Les paramètres des 30 problèmes sont donnés dans le tableau 1.1 .

numéro des paramètres

nom	1	2	3	4	5	6	7	8	9	10	11	12
T1	300	1000	10	290	-	-	10000	0.0	100	1.0	40	60
T2	300	1000	10	290	-	-	20000	0.0	100	1.0	40	60
T3	300	1000	40	260	-	-	400000	0.8	100	0.5	200	500
T4	400	5000	40	360	-	-	500000	0.0	100	0.0	-	-
T5	400	5000	200	200	-	-	700000	0.0	100	0.5	0	200
T6	100	250	5	95	-	-	15000	0.8	100	1.0	50	200
T7	200	1000	10	190	-	-	300000	0.5	100	0.5	0	300
T8	200	2000	20	180	-	-	200000	0.8	100	0.0	-	-
T9	1000	5000	100	900	-	-	1000000	0.0	100	0.5	0	500
T10	2000	9000	500	1500	-	-	5000000	0.0	100	0.0	-	-
F1	400	1500	8	60	0	0	100000	0.3	100	0.2	1000	10000
F2	400	1500	8	60	0	0	500000	0.3	100	0.2	1000	10000
F3	400	3000	8	60	5	50	300000	0.3	100	0.5	1000	5000
F4	400	2000	4	12	0	0	100000	0.3	100	1.0	1000	10000
F5	400	2000	4	12	10	10	1000000	0.3	100	1.0	20000	120000
F6	1000	5000	50	50	0	0	1000000	0.3	100	0.0	-	-
F7	1000	5000	50	50	0	0	3000000	0.3	100	1.0	20000	120000
F8	1500	6000	50	200	50	50	3000000	0.3	100	1.0	50000	100000
F9	2000	6000	5	400	100	100	1000000	0.3	100	0.5	1000	100000
F10	2000	6000	200	200	0	0	3000000	0.3	100	0.0	-	-
F11	2000	10000	5	400	100	100	5000000	0.3	100	1.0	10000	50000
F12	3000	10000	50	200	0	0	500000	0.3	100	0.5	1000	10000
F13	3000	10000	5	200	45	0	500000	0.3	100	0.5	2000	10000
F14	3000	15000	100	900	0	0	1000000	0.3	100	1.0	2000	10000
F15	3000	15000	50	200	50	700	5000000	0.3	100	0.0	-	-
F16	4000	10000	10	1000	0	0	1000000	0.3	100	0.0	-	-
F17	4000	12000	10	500	10	100	2000000	0.0	100	0.5	50000	100000
F18	4000	14900	100	100	100	100	1000000	0.3	100	1.0	0	50000
F19	4000	14900	100	100	100	100	1000000	0.3	100	1.0	0	100000
F20	4500	13000	100	400	0	0	5000000	0.3	100	0.0	-	-

Tableau 1.1.

1.4.2. Analyse comparative

Les tableaux 1.2 , 1.3 et 1.4 contiennent les temps de calcul sur le Cyber 855 de l'EPFL . Ces temps sont les temps de résolution des problèmes ; les durées de lecture des données et d'écriture des résultats ne sont pas comprises .

Ces 3 tableaux montrent que les problèmes de transport T1 . . . T10 sont faiblement dégénérés ; tandis que la résolution des problèmes F1 . . . F20 produit entre 50 et 80% des pivots dégénérés .

Les temps de résolution semblent fortement influencés par la taille du problème (n et m) . Le programme PNET semble le plus rapide pour les problèmes de petite dimension (T1 . . . T8 et F1 . . . F7) , par contre le nombre de pivots est toujours supérieur à celui des deux autres stratégies . Sur ces 30 problèmes , LPNET et GNET ont des comportements très semblables ; tant du point de vue du temps de calcul que de celui du nombre de pivots . Pour les grands problèmes , ces deux programmes sont nettement plus rapides que PNET .

L'analyse de l'efficacité de la méthode " big - M " pour l'obtention d'un flot peut être effectuée en annulant les coûts des arcs du réseau . Les tableaux 1.5 , 1.6 et 1.7 contiennent les temps de calcul d'un flot dans les réseaux T5 , T10 , F5 , F14 et F19 . On remarque que la stratégie n'influence pas le nombre de pivots ; mais PNET est nettement plus rapide . Ceci est dû au fait que tous les coûts réduits valent - 2M , 0 ou 2M et qu'il est , par conséquent , inutile de perdre du temps à choisir l'arc entrant .

En conclusion on peut dire que pour de petits problèmes , une stratégie simple telle celle employée dans PNET fournit de bons résultats . Pour de grands problèmes , il faut utiliser une stratégie plus sophistiquée .

nom du problème	durée du calcul (s)	nombre de pivots total	non-dégénérés	dégénérés
T1	0.2	621	574	47
T2	0.2	821	760	61
T3	0.4	1006	904	102
T4	0.9	1290	1290	0
T5	2.3	3425	3359	66
T6	0.03	143	127	16
T7	0.2	548	503	45
T8	0.3	535	534	1
T9	4.2	5053	4740	313
T10	20.1	10005	9935	70
F1	0.4	900	217	683
F2	0.4	885	273	612
F3	0.9	2023	1056	967
F4	0.5	1481	240	1241
F5	0.5	1349	243	1106
F6	2.4	4392	717	3665
F7	3.1	5115	1184	3931
F8	6.0	6210	2288	3922
F9	12.8	8938	4052	4286
F10	8.5	7284	2302	4982
F11	17.8	12855	6602	6253
F12	16.6	12320	2433	9887
F13	17.2	12906	2555	10351
F14	36.4	17968	9268	8700
F15	26.4	15036	6511	8525
F16	19.4	7780	2699	5081
F17	23.5	12100	2855	9245
F18	36.7	22987	4882	18105
F19	33.9	21542	4164	17378
F20	36.4	18228	3743	14485

Tableau 1.2. Programme PNET.

nom du problème	durée du calcul	nombre de pivots total	non-dégénérés	dégénérés
T1	0.4	430	397	33
T2	0.4	526	493	33
T3	0.5	595	548	47
T4	1.2	607	607	0
T5	3.1	1708	1683	25
T6	0.1	131	116	15
T7	0.4	426	396	30
T8	0.4	248	246	2
T9	4.3	3022	2821	201
T10	10.7	4387	4376	11
F1	0.6	482	92	390
F2	0.7	560	134	426
F3	1.0	761	339	422
F4	0.8	691	55	636
F5	0.7	649	77	572
F6	2.9	1989	224	1765
F7	2.9	1985	375	1610
F8	4.9	2828	818	2010
F9	8.1	3899	1562	2337
F10	7.9	3947	1030	2917
F11	11.2	5435	2388	3047
F12	14.1	5761	918	4843
F13	14.0	5643	788	4855
F14	19.9	7118	3126	3992
F15	18.2	6669	2571	4098
F16	17.8	5126	1470	3656
F17	21.5	5902	1102	4800
F18	26.1	9238	1361	7877
F19	25.6	9089	1304	7785
F20	27.2	8264	1269	6995

Tableau 1.3. Programme LPNET.

nom du problème	durée du calcul	nombre de pivots total	non-dégénérés	dégénérés
T1	0.4	418	392	26
T2	0.5	550	514	36
T3	0.6	558	520	38
T4	1.0	538	538	0
T5	3.1	1641	1617	24
T6	0.1	125	114	11
T7	0.4	405	378	27
T8	0.4	268	267	1
T9	3.9	2433	2298	135
T10	10.9	3889	3884	5
F1	0.5	548	125	423
F2	0.6	606	149	457
F3	1.3	1009	461	548
F4	0.8	864	94	770
F5	0.8	771	110	661
F6	3.2	2156	271	1885
F7	3.6	2386	451	1935
F8	5.2	2953	855	2098
F9	7.9	3622	1436	2186
F10	7.5	3689	942	2747
F11	12.1	4935	2108	2827
F12	14.8	5740	948	4792
F13	13.8	5421	777	4644
F14	20.6	6653	2854	3799
F15	20.5	6674	2542	4132
F16	18.3	5268	1508	3760
F17	21.2	6484	1257	5227
F18	28.1	9315	1454	7861
F19	26.4	8989	1355	7634
F20	27.6	8080	1219	6861

Tableau 1.4. Programme GNET.

Obtention d'une solution admissible

nom du problème	durée du calcul	nombre de pivots total	non-dégénérés	dégénérés
T5	0.8	1019	884	195
T10	3.2	241	2395	15
F5	0.2	452	56	396
F14	9.8	4320	1897	2423
F19	14.2	5475	871	4604

Tableau 1.5. Programme PNET.

nom du problème	durée du calcul	nombre de pivots total	non-dégénérés	dégénérés
T5	2.2	952	837	115
T10	6.0	3560	2542	18
F5	0.6	465	05	400
F14	14.1	4481	1986	2495
F19	19.9	5211	775	4436

Tableau 1.6. Programme LPNET.

nom du problème	durée du calcul	nombre de pivots total	non-dégénérés	dégénérés
T5	2.2	908	799	109
T10	6.2	2338	2323	15
F5	0.4	430	53	337
F14	13.7	4034	1676	2358
F19	18.3	5048	690	4358

Tableau 1.7. Programme GNET

1.5. POSTOPTIMISATION

Lorsque l'on doit résoudre un problème de flot à coût minimum , il arrive que certaines des données soient modifiées . Il s'agit alors de calculer l'optimum du nouveau problème ; en général , on aura avantage à utiliser la solution optimale du problème initial comme solution de départ ; on appelle ***postoptimisation*** l'ensemble des techniques permettant d'obtenir l'optimum d'un problème lorsque certaines données ont subi de légères modifications . Dans ce paragraphe seront envisagées des variations de coût C , de bilans B , de capacité U et du réseau. Nous montrerons comment modifier la base T et le flot X optimal pour le problème initial en une base et flot de départ pour la résolution du nouveau problème par le simplexe .

1.5.1. Modification des coûts

Considérons tout d'abord la modification du coût d'un seul arc (i,j) :

- si cet arc est dans la base T , on doit recalculer les variables duales W des sommets de $T_{(i,j)}$. Les coûts réduits des arcs ayant une extrémité dans $T_{(i,j)}$ sont changés ; si l'un devient négatif , il faut appliquer le simplexe , sinon la solution reste optimale .

- si cet arc est hors base , seul son coût réduit change ; s'il devient négatif , on applique le simplexe sinon la solution reste optimale .

Lorsqu'un grand nombre de coûts sont modifiés , on a avantage à recalculer toutes les variables duales et tous les coûts réduits avant d'appliquer le simplexe .

1.5.2. Modification des bilans

Considérons une modification du vecteur des bilans B qui le transforme en $B + \Delta B$. Pour que le bilan total reste nul, il faut que les variations satisfassent

$$\sum_{i \in V} \Delta B_i = 0$$

Par un algorithme similaire à celui utilisé pour résoudre un système primal (algorithme 1.3.), on essaie de faire circuler le surplus de production à travers les arcs de la base T . Lorsqu'un arc de T est vidé ou saturé, on utilise des arcs fictifs semblables à ceux employés dans la méthode " big - M " (par. 1.2.3).

Algorithme 1.9. POSTOPTIMISATION POUR UNE MODIFICATION DE B

0. triangulation

Renuméroter les sommets de V selon l'ordre transversal

i := n

1. Traitement de la variation du bilan en i

Considérer l'arc e ∈ T d'extrémités i et p(i)

si (e = (p(i),i)) alors

$X_e := X_e - \Delta B_i$

si ($X_e < 0$) alors sortir e de T et entrer l'arc fictif (i,1)

$X_{(i,1)} := - X_e$; $X_e := 0$; $\Delta B_i := \Delta B_i - X_{(i,1)}$

si ($X_e > C_e$) alors sortir e de T et entrer l'arc fictif (1,i)

$X_{(1,i)} := X_e - C_e$; $X_e := C_e$; $\Delta B_i = \Delta B_i + X_{(1,i)}$

sinon (* e = (i,p(i)) *)

$X_e := X_e + \Delta B_i$

si ($X_e < 0$) alors sortir e de T et entrer l'arc fictif (1,i)

$X_{(i,1)} := X_e - C_e$; $X_e := C_e$; $\Delta B_i := \Delta B_i - X_{(i,1)}$

si ($X_e > C_e$) alors sortir e de T et entrer l'arc fictif (i,1)

$X_{(i,1)} := X_e - C_e$; $X_e := C_e$; $\Delta B_i := \Delta B_i - X_{(i,1)}$

Répercuter cette variation en p(i) : $\Delta B_{p(i)} := \Delta B_{p(i)} + \Delta B_i$

2. si (i > 2) alors i = i - 1

retour en **1**

3. Recalcul des variables duales

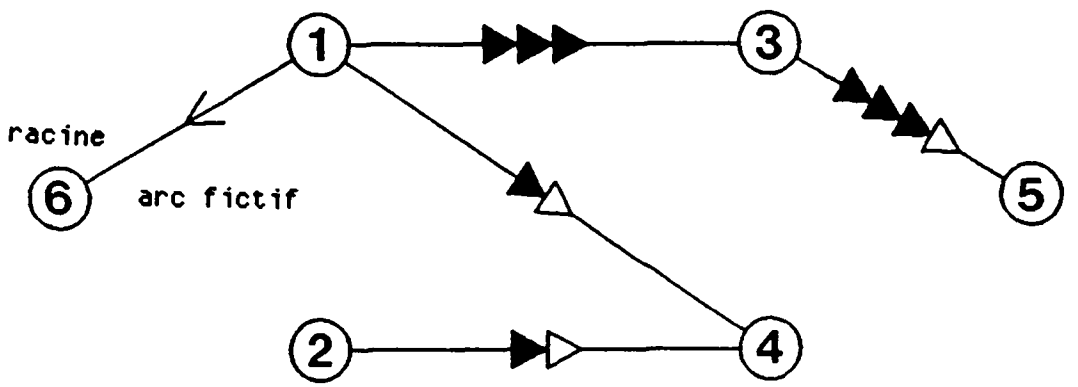
Utiliser l'algorithme 1.2 de résolution dual .

Remarques :

1) Il n'est pas nécessaire de renuméroter les sommets ; il suffit de les traiter selon l'inverse de l'ordre transversal ; le sommet racine (1) n'étant pas traité .

2) La programmation de l'opération qui consiste à sortir l'arc e de T et à entrer un arc fictif (i,1) ou (1,i) est semblable à celle du pivotage . On place les sommets de T_e à la fin de l'ordre transversal et on recalcule leurs profondeurs et leurs variables duales .

La résolution du problème exemple à l'aide de la méthode " big - M " fournit la base et le flot de base suivant donnés dans la figure 1.19.



ordre transversal : 6 , 1 , 4 , 2 , 3 , 5

figure 1.19

Considérons la variation $\Delta B_1 = -1$ $\Delta B_2 = 1$ $\Delta B_3 = 1$ $\Delta B_4 = -3$ $\Delta B_5 = 2$. Par l'algorithme , on obtient la base et le flot de base initiaux donnés dans la figure 1.20.

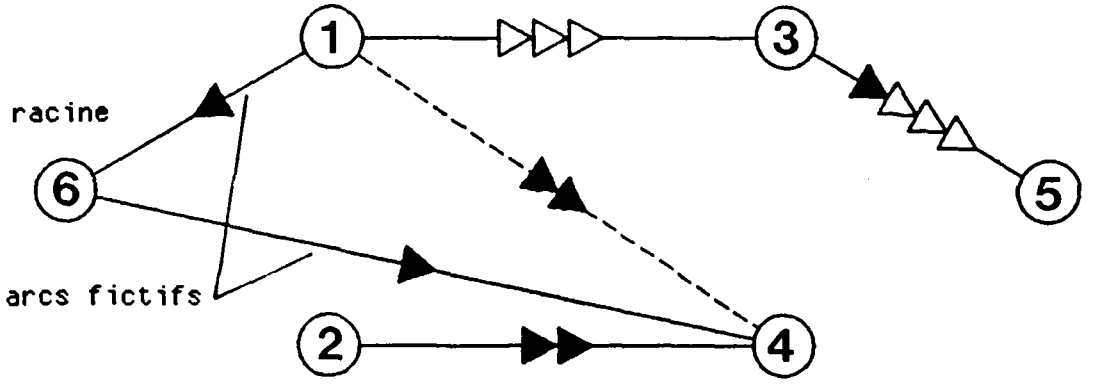


figure 1.20

1.5.3. Modification des capacités

Considérons une modification du vecteur des capacités U qui le transforme en $U + \Delta U$. En changeant les coûts et les flots on obtient aisément un flot de base optimal pour un problème où on a modifié les bilans ; on peut alors utiliser l'algorithme du paragraphe précédent .

Algorithme 1.10. POSTOPTIMISATION POUR UNE MODIFICATION DE U

1. Traitement des arcs où la capacité augmente

Pour tout e = (i,j) avec $\Delta U_e > 0$ faire

$C_e := C_e + \Delta U_e$

si (e est hors base et $X_e > 0$) alors

$\Delta B_i := -\Delta U_e$; $\Delta B_j := \Delta U_e$; $X_e := C_e$

2. Traitement des arcs où la capacité diminue

Pour tout e = (i,j) avec $\Delta U_e > 0$ faire

$C_e := C_e + \Delta U_e$

si ($X_e > C_e$) alors $\Delta B_i = C_e - X_e$; $\Delta B_j := X_e - C_e$; $X_e := C_e$

3. Appliquer l'algorithme de postoptimisation pour la variation ΔB des bilans

Cette façon de procéder rend aisée la combinaison de modifications de capacités et de bilans .

1.5.4. Modification du réseau

Il peut être intéressant pour la postoptimisation de modifier le réseau par la suppression ou l'ajout d'un ou plusieurs arcs .

Pour enlever un arc , on lui donne un coût très grand pour la postoptimisation .

Pour chaque arc supplémentaire , on calcule le coût réduit ; s'ils sont tous positifs le flot optimum est inchangé sinon il faut appliquer le simplexe . La structure de données utilisée pour mémoriser le réseau (par. 1.3.1) est mal adaptée à l'ajout d'un arc ; en effet pour stocker l'information d'un nouvel arc issu du sommet i , il faut décaler celles concernant les arcs issus de i+1 ... n . Ceci peut être évité en modélisant l'arc à ajouter par 2 nouveaux arcs issus d'un nouveau sommet ; ce sommet pouvant être placé en fin de liste , le transfert de l'information est supprimé . La création d'un arc (i,j) de capacité $U_{(i,j)}$ et coût $C_{(i,j)}$ peut être effectuée par l'algorithme 1.11.

Algorithme 1.11. AJOUT D'UN ARC

1. Création du sommet v

$W_v := W_i$

2. Création de l'arc (v,i)

$U_{(v,i)} := U_{(i,j)}$; $C_{(v,i)} := 0$; $X_{(v,i)} := U_{(i,j)}$

cet arc est mis dans la base T := T + (v,i)

3. Création de l'arc (v,j)

$U_{(v,j)} := U_{(i,j)}$; $C_{(v,j)} := C_{(i,j)}$; $X_{(v,j)} := 0$

calculer le coût réduit de cet arc

Les deux réseaux de la figure 1.21 sont équivalents ; en effet dans les deux cas , on peut faire transiter au plus 4 unités de flot de i vers j et cela avec un coût unitaire de 3 .

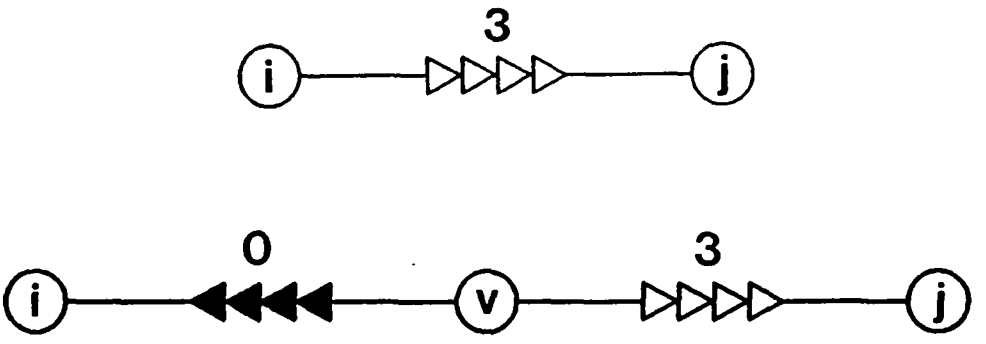


figure 1.21

On a démontré que l'algorithme du simplexe permet une postoptimisation pour n'importe quelles modifications du réseau . L'efficacité de telles procédures dépend fortement de l'importance des modifications ; lorsque le problème initial et le nouveau problème sont très différents , il est préférable de les traiter indépendamment .

BIBLIOGRAPHIE

[1] A.I. ALI , R.V. HELGASON , J.L. KENNINGTON , H.S. LALL
Primal simplex network codes : state-of-the-art implementation technology
Networks 8 (1978) p 315-339

[2] G.H. BRADLEY , G.G. BROWN , G.W. GRAVES
Design and implementation of large scale primal transhipment algorithms
Management science 24 (1977) p 1-34

[3] V. CHVATAL
Linear programming
Freeman and Co. (New-York , 1983)

[4] W.H. CUNNINGHAM
A network simplex method
Mathematical programming 11 (1976) p 105-116

[5] W.H. CUNNINGHAM
Theorical properties of the network simplex method
Mathematics of operations research 4 (1979) p 196-208

[6] D. DE WERRA , C. PASCHE , A. PETTER
Timetabling problems : Should they be canonical ?
INFOR 24 , No 4 (1986) p 304-308

[7] L.R. FORD , D.R. FULKERSON
Flots dans les graphes
Gauthier-Villars (Paris , 1967)

[8] F. GLOVER , D. KARNEY , D. KLINGMAN
Implementation and computational comparisons of primal , dual and primal - dual computer codes for minimum cost network flow problems
Networks 4 (1974) p 191 - 212

[9] P.A. JENSEN , J.W. BARNES
Network flow programming
J. Wiley (New-York , 1980)

[10] J.L. KENNINGTON , R.V. HELGASON
Algorithms for network programming
J. Wiley (New-York , 1980)

[11] D. KLINGMAN , A. NAPIER , J. STUTZ
NETGEN : a program for generating large scale capacitated assignment , transportation and minimum cost flow network problems
Management science 20 (1974) p 814-821

[12] D.G. LUENBERGER
Introduction to linear and nonlinear programming
Addison-Wesley (Reading , 1973)

[13] J.M. MULVEY
Testing of large scale network optimization program
Mathematical programming 15 (1978) p 291-314

[14] ZADEH
A bad network problem for the simplex method and other minimum cost flow algorithms
Mathematical programming 5 (1973) p 255-266

[15] S. ZIONTS
Linear and integer programming
Prentice Hall (Englenood Cliffs , 1974)

Chapitre 2

FLOT A COUT CONVEXE LINEAIRE PAR MORCEAUX

2.0. INTRODUCTION

Ce chapitre traite du problème de flot à coût minimum , où le coût de transit du flot sur chaque arc est une fonction convexe linéaire par morceaux . Ce problème sera appelé problème de flot à coût convexe linéaire (PFCCL) . Une publication a été extraite du présent chapitre ([10]) .

De nombreux problèmes pratiques peuvent se formuler comme un PFCCL ; c'est le cas par exemple :

- du problème d'augmentation à moindre coût de la capacité d'un réseau ([1]) ,
- de certains problèmes liés aux réseaux téléphoniques ([3],[9]) ,
- de certains problèmes de transport urbain ([5]) ,
- du problème de préservation d'un réseau de communication ([11]) ,
- du problème d'ordonnancement de tâches possédant des durées variables qui est , en fait , le dual de PFCCL ([1],[4]) ,
- du problème de flot où il faut minimiser une fonction convexe des transits , qui peut être approché par le PFCCL (voir le chapitre 3) .

En théorie le PFCCL est résolu en le transformant en un problème de flot à coût linéaire par décomposition des arcs en plusieurs arcs parallèles . Pour être efficace , un algorithme doit traiter ces variables supplémentaires de manière implicite . Monma et Segal ([9]) proposent une structure de données adaptées au PFCCL . Ahaja , Batra et Gupta ([1]) ont développé un algorithme dual spécialisé .

Pour utiliser au mieux la structure particulière du PFCCL , une nouvelle stratégie de pivotage a été introduite dans l'algorithme primal du simplexe . Plusieurs pivots successifs peuvent être combinés ; c'est-à-dire que plusieurs modifications de flot peuvent être effectuées pour une seule mise-à-jour de la base et des variables duales .

2.1. PRESENTATION DU PROBLEME

Considérons un réseau R = (V,E) formé d'un ensemble V de n sommets et d'un ensemble E de m arcs . Le problème se formule de la façon suivante :

$$\underset{X_{(i,j)}}{\text{Minimiser}} \quad \sum_{(i,j)\in E} C_{(i,j)}(X_{(i,j)}) \qquad (2.1)$$

s.c.

$$\sum_{(i,j)\in E} X_{(i,j)} - \sum_{(j,i)\in E} X_{(j,i)} = B_i \qquad \forall\, i \in V \quad (2.2)$$

$$0 \leqslant X_{(i,j)} \leqslant U_{(i,j)} \qquad \forall\, (i,j) \in E \quad (2.3)$$

où

- $X_{(i,j)}$ représente le flot transitant sur l'arc (i,j) de i vers j .

- $U_{(i,j)}$ est la capacité de l'arc (i,j) .

- B_i est le bilan au sommet i ; pour que le problème possède une solution on impose :

$$\sum_{i\in V} B_i = 0$$

- $\mathbf{C}_{(i,j)}$: [0 , $U_{(i,j)}$] → **R** , fonction convexe linéaire par morceaux , représente le coût de transfert du flot sur l'arc (i,j) . Elle est composée de **K**(i,j) fonctions linéaires données par :
  - les points de cassures de $\mathbf{C}_{(i,j)}$ sont donnés par $U_{(i,j)} \in \mathbf{R}^{(K(i,j)+1)}$:
    $0 = U_{(i,j),0} < U_{(i,j),1} < \ldots < U_{(i,j),K(i,j)} = U_{(i,j)}$
  - les coûts unitaires de $\mathbf{C}_{(i,j)}$ sont fournis par $C_{(i,j)} \in \mathbf{R}^{(K(i,j))}$:
    $C_{(i,j),1} < \ldots < C_{(i,j),K(i,j)}$.

  Sur le segment numéro k , correspondant à l'intervalle [$U_{(i,j),k-1}$, $U_{(i,j),k}$] , le coût est linéaire de pente $C_{(i,j),k}$. La figure 2.1 fournit un exemple de coût .

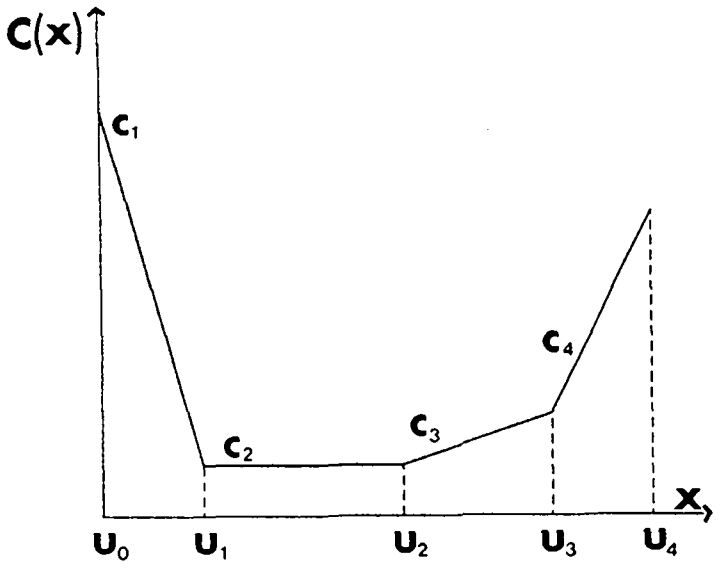


figure 2.1

Ce problème peut être transformé en un problème linéaire en introduisant pour chaque arc (i,j) ∈ E , **K**(i,j) variables $Y_{(i,j),1}$,..., $Y_{(i,j),K(i,j)}$. La variable $Y_{(i,j),k}$ représentant la part du flot $X_{(i,j)}$ transportée au coût unitaire $C_{(i,j),k}$. On a

$$X_{(i,j)} = \sum_{k=1}^{\mathbf{K}(i,j)} Y_{(i,j),k} \qquad (2.4)$$

Le problème se reformule alors de la manière suivante:

$$\text{Minimiser} \sum_{(i,j)\in E} \sum_{k=1}^{\mathbf{K}(i,j)} C_{(i,j),k}\, Y_{(i,j),k} \qquad (2.5)$$

$$\sum_{(i,j)\in E} \sum_{k=1}^{\mathbf{K}(i,j)} Y_{(i,j),k} - \sum_{(j,i)\in E} \sum_{k=1}^{\mathbf{K}(j,i)} Y_{(j,i),k} = B_i \qquad \forall\, i \in V \quad (2.6)$$

$$0 \leqslant Y_{(i,j),k} \leqslant U_{(i,j),k} - U_{(i,j),k-1} \qquad \forall\, (i,j) \in E \;\; \forall\, k = 1 ,..., \mathbf{K}(i,j) \quad (2.7)$$

$$Y_{(i,j),k} > 0 \;\Rightarrow\; Y_{(i,j),k-1} = U_{(i,j),k-1} - U_{(i,j),k-2} \qquad \forall\, (i,j) \in \mathbf{E} \;\; \forall\, k = 2 ,..., \mathbf{K}(i,j) \quad (2.8)$$

Les contraintes (2.8) sont superflues ; en effet les coûts unitaires $C_{(i,j),k}$ étant croissants pour k = 1 ,..., **K**(i,j) , elles sont forcément vérifiées dans une solution optimum . Les contraintes restantes définissent un problème de flot à coût linéaire minimum , où chaque arc (i,j) a été remplacé par **K**(i,j) arcs parallèles . La figure 2.2 linéarise le coût donné par la figure 2.1.

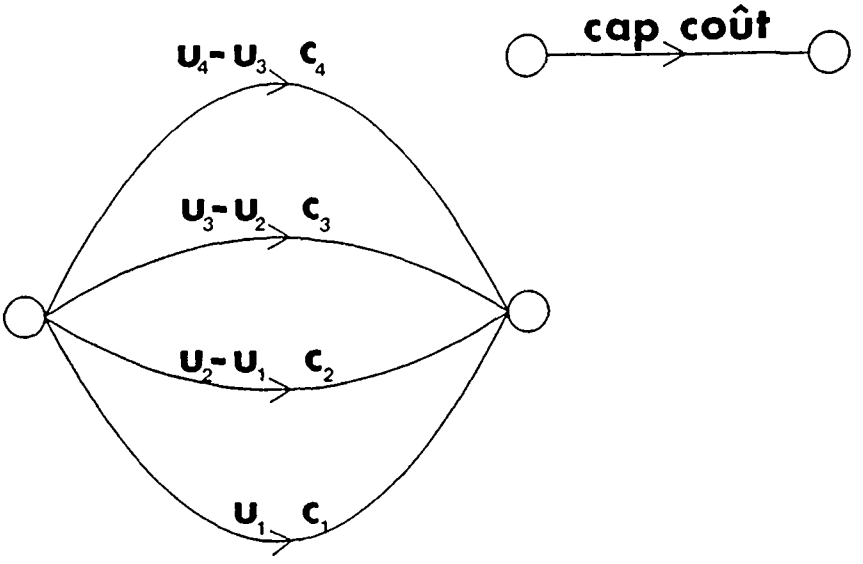


figure 2.2

Après cette transformation , le problème peut être résolu par tout algorithme classique de flot à coût minimum . L'efficacité d'un tel algorithme peut cependant être augmentée s' il est particularisé de manière à tenir compte des K(i,j)-uples d'arcs parallèles . Au lieu de modifier le réseau par éclatement des arcs , on introduit la notion d'état **e**(i,j) pour les arcs (i,j) ∈ E . **e**(i,j) est déterminé de sorte que :

$$\mathbf{e}(i,j) = k \quad \Rightarrow \quad X_{(i,j)} \in [\ U_{(i,j),k-1}\ ,\ U_{(i,j),k}\ ] \qquad (2.9)$$

L'implication (2.9) ne définit pas l'état de manière univoque . En effet pour $X_{(i,j)} = U_{(i,j),k}$ ($k \neq \mathbf{K}(i,j)$) l'état de (i,j) peut valoir k ou k+1 ; ceci dépend de la modification du flot $X_{(i,j)}$ envisagée :

- si c'est une augmentation alors **e**(i,j) = k+1
- si c'est une diminution alors **e**(i,j) = k .

Dans la suite , nous allons montrer comment intégrer la notion d'état à l'algorithme du simplexe pour les réseaux (cf [6],[7],[9]) .

2.2. L'ALGORITHME DU SIMPLEXE

Avant de décrire l'algorithme , il est nécessaire de définir les notions de base , de flot de base et de coût réduit pour le problème particulier.

Une base est formée d'un ensemble $T \subset E$ de n-1 arcs d'un arbre , d'un sommet racine $r \in V$ et des états $\mathbf{e}(i,j)$ pour les arcs $(i,j) \in E$.

Un flot de base $\{ X_{(i,j)} ; (i,j) \in E\}$ est un flot (contraintes 2.12 et 2.13) tel que :

- les variables de base $X_{(i,j)}$, $(i,j) \in T$, sont dans le segment numéro $\mathbf{e}(i,j)$; c'est-à-dire :

$$X_{(i,j)} \in [\ U_{(i,j),e(i,j)-1}\ ,\ U_{(i,j),e(i,j)}\ ] \qquad \forall\ (i,j) \in T \quad (2.10)$$

- les variables hors base $X_{(i,j)}$, $(i,j) \in E - T$, sont égales à un point de rupture ; c'est-à-dire :

$$X_{(i,j)} = U_{(i,j),e(i,j)-1} \text{ ou } U_{(i,j),e(i,j)} \qquad \forall\ (i,j) \in E - T \quad (2.11)$$

A chaque sommet $i \in V$, on attribue une variable duale W_i donnant la longueur de l'unique chaîne de T allant de i vers la racine r . Cette longueur est calculée en utilisant les coûts des arcs correspondant à leur état ; les arcs parcourus dans le bon sens ayant une contribution positive , ceux parcourus en sens inverse ayant une contribution négative .

Ces variables duales permettent de définir pour tout arc hors base (i,j) deux coûts réduits : $\mathbf{ra}(i,j)$ un coût réduit d'augmentation et $\mathbf{rd}(i,j)$ un coût réduit de diminution . Pour $(i,j) \in E - T$ avec $X_{(i,j)} = U_{(i,j),k}$ on pose :

$$\mathbf{ra}(i,j) = \begin{cases} \infty & \text{si } k = K(i,j) \\ W_j - W_i + C_{(i,j),k+1} & \text{sinon} \end{cases} \quad (2.12)$$

$$\mathbf{rd}(i,j) = \begin{cases} \infty & \text{si } k = 0 \\ W_i - W_j - C_{(i,j),k} & \text{sinon} \end{cases} \quad (2.13)$$

L'algorihme du simplexe va procéder par élimination successive de tous les cycles de longueur négative le long des quels le flot peut être modifié . L'algorithme 2.1 formule le simplexe pour le PFCCL .

Algorithme 2.1. ALGORITHME MODIFIE DU SIMPLEXE

0. Initialisation.

- Trouver une base T et un flot de base $\{ X_{(i,j)} ; (i,j) \in E \}$.

1. Choix de l'arc entrant : (p,q) .

- Choisir un arc $(p,q) \in E - T$ de coût réduit négatif .
- S'il n'en existe pas alors le flot est optimum .
- Changer l'état de l'arc entrant (p,q) (supposons que $X_{(p,q)} = U_{(p,q),k}$)
  si **ra**(p,q) < 0 alors **e**(p,q) := k+1 et on va augmenter le flot sur (p,q)
  si **rd**(p,q) < 0 alors **e**(p,q) := k et on va diminuer le flot sur (p,q) .

2. Calcul de l'amplitude de la modification .

- Chercher le cycle C formé par l'arc (p,q) et l'arbre T de la base .
- Orienter C selon (p,q) pour augmenter le flot et à l'inverse de (p,q) pour le diminuer .
- Calculer la quantité maximale Δ de flot qui peut avancer sur **C** sans qu'aucun arc ne doive changer d'état .
- Choisir comme arc sortant (u,v) l'un des arcs qui détermine Δ .

3. Modification du flot .

- Modifier $X_{(i,j)}$ pour $(i,j) \in C$ en faisant avancer les Δ unités de flot.

4. Mise à jour de la base .

- T := T + (p,q) - (u,v) .
- Mettre à jour les variables duales W_i , $i \in V$.

- Retourner à **1.**.

Remarques :

- La base et le flot de base initiaux peuvent être obtenu en utilisant la méthode "big-M" (paragraphe 1.2.3) .
- La régle de Cunningham (paragraphe 1.2.4) pour le choix de l'arc sortant produit un algorithme fini .
- Il est facile de modifier cet algorithme de façon à traiter des problèmes où les contraintes (2.3) sont remplacées par :

$$L_{(i,j)} \leqslant X_{(i,j)} \leqslant U_{(i,j)} \qquad \forall\ (i,j) \in E \quad (2.14)$$

si $L_{(i,j)}$ est négatif , l'arc est dit biorienté et $X_{(i,j)}$ négatif signifie que - $X_{(i,j)}$ unités transitent de j vers i .

2.3. COMBINAISON DE PIVOTS

Le choix de l'arc entrant dans la base est crucial pour l'efficacité de l'algorithme du simplexe et plusieurs stratégies de choix ont été proposées pour le cas d'une fonction coût linéaire . Pour exploiter la structure particulière du problème PFCCL (qui possède conceptuellement un grand nombre d'arcs parallèles) , il faut utiliser au maximum les cycles de coût négatif . Une méthode simple consiste à essayer de faire entrer la variable sortante . Après chaque pivot , on calcule le nouveau coût réduit de l'arc sortant ; s'il est négatif , on le choisit comme arc entrant pour le pivot suivant , sinon on utilise une des stratégies classiques.

La figure 2.3 illustre le traitement d'un cycle C par cette méthode :

- le pivot 1 fait entrer (5,7) et sortir (1,6) ; le coût réduit de (1,6) est négatif ;
- le pivot 2 fait entrer (1,6) et sortir (4,5) ; le coût réduit de (4,5) est négatif ;
- le pivot 3 fait entrer (4,5) et sortir (6,7) ; le coût réduit de (6,7) est positif .

Pour modifier le flot le long de C , les arbres T_2 et T_3 sont inutiles ; on peut passer directement de T_1 à T_4 en posant $T_4 = T_1 + (5,7) - (6,7)$. En règle générale quel que soit le nombre de modifications successives effectuées le long d'un cycle , un seul changement de base est nécessaire ; c'est le principe des pivots combinés .

Pour incorporer cette nouvelle manière de pivoter à l'algorithme 2.1 , il faut remplacer les points **2.** et **3.** par les points **2'.** et **3'.** suivants :

Algorithme 2.2. PIVOTS COMBINES

2'. Recherche du cycle C.

- Chercher le cycle C formé par l'arc (p,q) et l'arbre T de la base .
- Orienter C selon (p,q) pour augmenter le flot et à l'inverse de (p,q) pour le diminuer .

3'. Modification du flot le long de C.

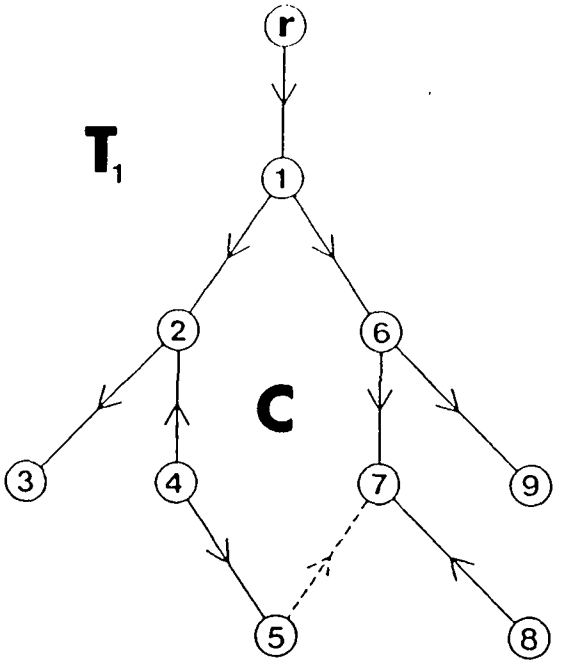
- Calculer la quantité maximale Δ de flot qui peut circuler sur C sans qu'aucun arc ne doive changer d'état .
- Choisir un arc (u,v) déterminant Δ .
- Modifier $X_{(i,j)}$ pour (i,j) $\in$ C en faisant avancer les Δ unités de flot et changer l'état de (u,v) .
- Si C possède encore un coût réduit négatif , alors aller en **3'**. .

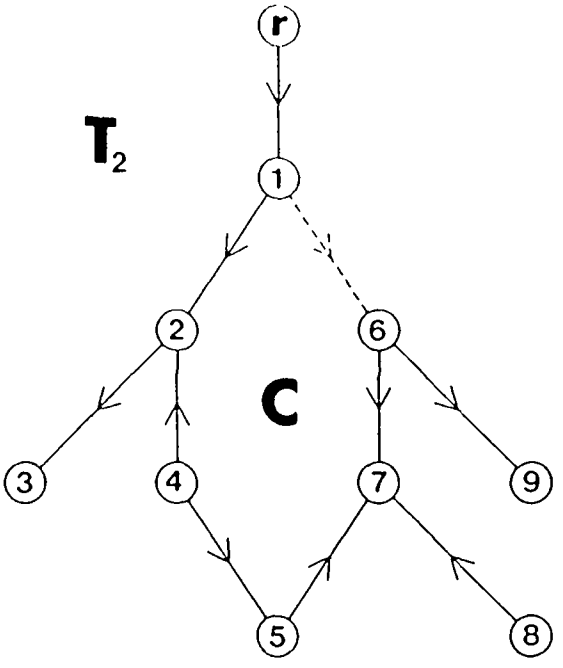
Dans l'algorithme classique , la mise-à-jour des variables duales s'effectue en additionnant une constante aux variables duales associées aux sommets déconnectés de la racine r par la suppression de l'arc sortant . Pour la nouvelle méthode de pivotage , cette opération est plus complexe :

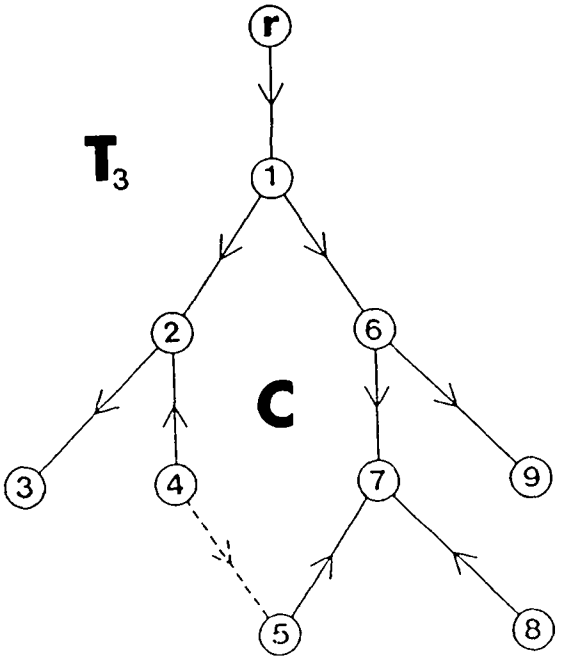
- les variables à modifier correspondent aux sommets déconnectés de la racine par le retrait des arcs ayant changé d'état au cours du pivot . Pour l'exemple les sommets 5,6,7,8 et 9.

- la modification des variables duales n'est plus uniforme ; il faut par conséquent utiliser l'algorithme 1.2 pour recalculer les variables de ces sommets . Dans l'exemple , on part du sommet 4 pour mettre à jour W_5 , W_7 et W_8 , puis à partir de 1 on recalcule W_6 et W_9 .

La règle de Cunningham ([2]) , évitant le cyclage , peut être adaptée au nouveau pivotage . Dans l'étape **3'**. on choisit comme arc (u,v) changeant d'état , le premier arc déterminant Δ rencontré le long de C en partant de la jointure (la jointure étant le sommet de C le moins profond dans l'arbre) .


T1
r
1
2
6
C
3
4
7
9
5
8


T2
r
1
2
6
C
3
4
7
9
5
8


T3
r
1
2
6
C
3
4
7
9
5
8

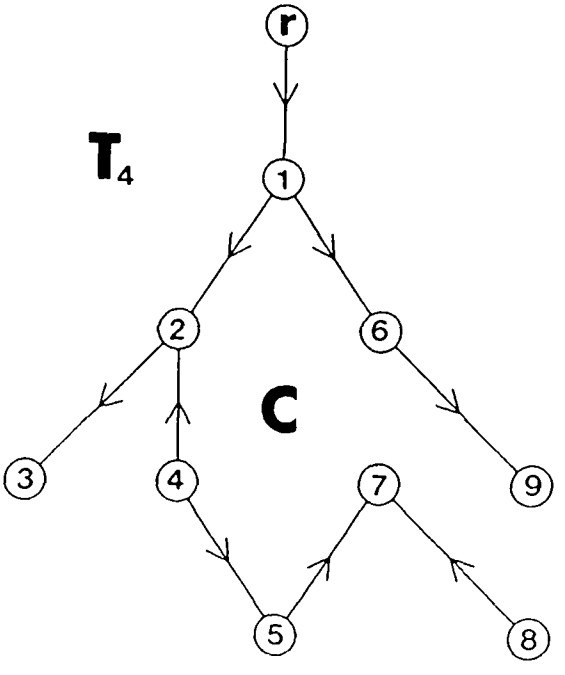
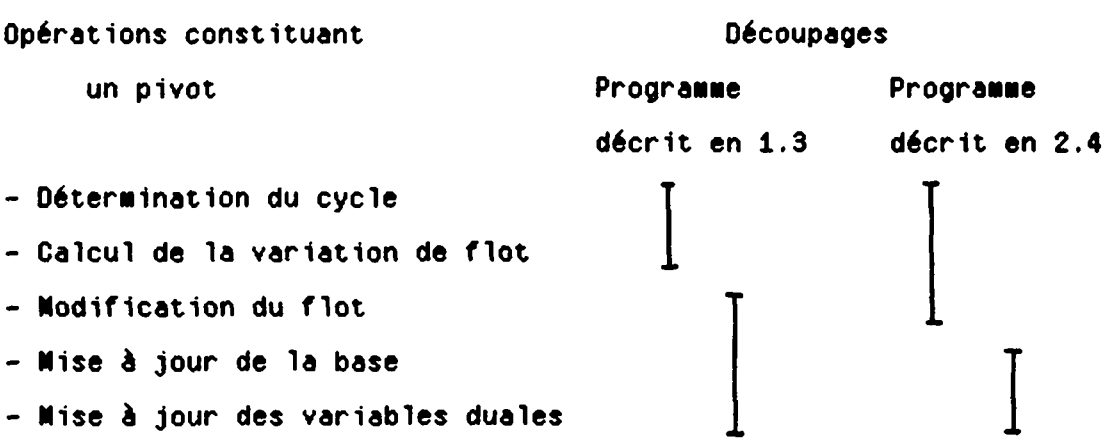

T4
r
1
2
6
C
3
4
7
9
5
8

figure 2.3

2.4. IMPLANTATION

Pour développer un programme efficace de résolution du PFCCL , il est nécessaire de particulariser non seulement l'algorithme mais également son implantation . Ce paragraphe présente les changements majeurs entre l'implantation de la méthode des pivots combinés et celle de l'algorithme classique décrite dans le paragraphe 1.3 . Les différences concernent la structure de données utilisée pour la représentation du réseau en mémoire et le découpage de l'algorithme en procédure :



2.4.1 Mémorisation du réseau et des bases

Le réseau R = (V,E) est mémorisé dans les deux tableaux TAIL(.) et HEAD(.) décrits dans le paragraphe 1.3.1 . La fonction coût $C_{(i,j)}(x)$ peut être entièrement caractérisée par les valeurs suivantes :

- la borne inférieure de flot transitant sur (i,j) : $L_{(i,j)} = U_{(i,j),0}$
- le coût de transit en cette borne inférieure : $C_{(i,j)}(L_{(i,j)})$
- les capacités des K(i,j) segments : $U_{(i,j),1} - U_{(i,j),0}$, ... , $U_{(i,j),K(i,j)} - U_{(i,j),K(i,j)-1}$
- les coûts unitaires sur les segments : $C_{(i,j),1}$, ... , $C_{(i,j),K(i,j)}$

La place utilisée en mémoire centrale est minimisée en mémorisant les caractéristiques des segments séquentiellement dans des tableaux . Pour ce faire les segments sont numérotés conformément à la numérotation des arcs . L'information sur les coûts de transit est stockée dans les tableaux suivants :

- MNCAP(.) : la case MNCAP(k) mémorise la borne inférieure L_k de flot sur l'arc numéro k .

- MNCOST(.) : le coût $C_k(L_k)$ est stocké en MNCOST(k)

- NSEG(.) : les segments composant l'arc numéro k portent les numéros NSEG(k) ,...., NSEG(k+1)-1

- SGCAP(.) : la case SGCAP(p) contient la capacité du segment numéro p

- SGCOST(.) : la case SGCOST(p) mémorise le coût unitaire sur le segment numéro p

Les tableaux MNCAP(.) , MNCOST(.) et NSEG(.) sont de dimension m (nombre d'arcs) , tandis que SGCAP(.) et SGCOST sont de dimension s où

$$s = \sum_{(i,j)\in E} K(i,j)$$

(s est le nombre total de segments)

Dans le paragraphe 2.1. la notion d''état a été introduite pour repérer les segments traités à un instant donné de l'algorithme . Les numéros et les caractéristiques des segments correspondants aux états sont mémorisés dans les tableaux suivants :

- STATE(.) : le numéro du segment correspondant à l'état de l'arc numéro k est stocké en STATE(k) en adoptant la convention suivante :
  - si l'arc est hors base , on mémorise l'état
  - si l'arc est dans la base , on mémorise l'opposé de l'état .

- CAP(.) : la case CAP(k) contient la capacité du segment correspondant à l'état de l'arc k . Comme en 1.3.2 on repère les arcs hors base ayant un segment saturé en stockant l'opposé de la capacité.

- COST(.) : le coût unitaire sur le segment correspondant à l'état de l'arc k est mémorisé dans COST(k) .

Ces trois tableaux sont de dimensions m . Les informations contenues dans les deux derniers sont redondantes ; en effet

CAP(k) = SGCAP(|STATE(k)|)
COST(k) = SGCOST(|STATE(k)|)

Cependant il s'est avéré que cet accès indirect aux caractéristiques des segments en cours d'utilisation ralentit le programme .

Les bases sont à nouveau mémorisées par les six tableaux PRED(.) , ARC(.) , DEPTH(.) , TRAV(.) , MULT(.) et VAR(.) . Le tableau VAR(.) mémorisant le flot sur le segment correspondant à l'état des arcs de la base .

Au total l'implantation de l'algorithme requiert sept tableaux de dimension n, sept de dimension m et deux de dimension s .

2.4.2. Implantation des pivots combinés

La procédure implantant la modification de flot lors d'une combinaison de pivots procède de la manière suivante :

- dans une première étape , on détermine le cycle , la jointure et la variation maximale de flot (sans changement d'état) de manière analogue à celle programmée dans la procédure 1.1 .

- dans les étapes suivantes , on effectue successivement une modification de flot sur le cycle , le changement d'état d'un arc et le calcul d'une nouvelle variation maximale de flot . Cette suite de pivots combinés s'achève lorsque le coût du cycle cesse d'être négatif .

Outre la jointure cette procédure doit également fournir les premiers arcs devant changer d'état rencontrés sur les chemins liant la jointure aux deux extrémités de l'arc entrant .

Le coût du flot ne pouvant plus être amélioré en utilisant ce cycle , il faut sortir de la base le dernier arc ayant changé d'état . La mise à jour de la base (tableaux PRED(.) , ARC(.) , DEPTH(.) et TRAV(.)) s'effectue à l'aide de la procédure 1.2 .

Finalement certains arcs ayant changé d'état , il est nécessaire de recalculer les variables duales (tableau MULT(.)) . La procédure 1.2 de mise à jour de l'ordre transversal ne visitant pas tous les sommets dont la variable duale est modifiée , les deux opérations de mise à jour doivent être séparées .

2.5. TESTS

Le but de ce chapitre est d'analyser l'efficacité des pivots combinés . Pour ce faire trois programmes ont été élaborés :

- FLOW qui implante l'algorithme du simplexe pour les flots à coût linéaire comme décrit dans le paragraphe 1.3 . (le temps nécessaire à la transformation d'un PFCCL en un problème à coût linéaire n'est pas pris en compte)

- SGFLOW qui implante l' algorithme 2.1 pour le PFCCL . On utilise la structure de donnée décrite en 2.4 et le découpage utilisé en 1.3 .

- SGFLOW1 qui est la variante de SGFLOW dans laquelle les pivots sont combinés (algorithme 3.2) . L'implantation se fait selon 2.4 .

Ces trois programmes ont été écrits en Fortran 5 respectant la norme ANSI . La stratégie de choix de l'arc entrant de Bradley , Brown et Graves est utilisée dans ces trois programmes .

Pour effectuer ces tests , nous avons développé deux générateurs de PFCCL selon les principes de NETGEN ([8]) :

- le premier engendre des problèmes où les caractéristiques des arcs (nombre de segments , capacités des segments et coûts des segments) sont uniformément distribuées . Il fournit les problèmes P1,...,P20 présentés dans le tableau 2.1 .

- le second produit des problèmes où les coûts des arcs sont quadratiques . Ces problèmes sont transformés en approchant le coût quadratique par une fonction linéaire par morceaux possédant un nombre fixé de segments égaux et en faisant une interpolation entre les points de cassure . Le tableau 2.2 donne les caractéristiques des problèmes Q1,...,Q10 ainsi obtenus .

nom	# sommets	# arcs	# max. de segments	
			négatifs	positifs
P1	100	500	0	4
P2	100	500	0	4
P3	100	500	5	5
P4	300	1500	0	4
P5	300	1500	0	8
P6	500	2000	4	5
P7	500	2000	0	5
P8	500	2500	0	7
P9	700	3000	5	5
P10	700	4000	0	8
P11	750	4000	2	4
P12	750	4000	2	4
P13	1000	5000	3	3
P14	1000	6000	0	4
P15	1000	6000	0	5
P16	1500	5000	0	4
P17	1500	6000	2	2
P18	1500	6000	0	4
P19	2000	6000	0	5
P20	2000	6000	0	3

Tableau 2.1

note : Les arcs sont biorientés ; les colonnes 3 et 4 du tableau donnent le nombre maximum de segments pour le flot positif et le flot négatif.

nom	# sommets	# arcs	# segments
Q1	100	500	30
Q2	100	500	20
Q3	100	1000	18
Q4	200	750	16
Q5	200	1000	14
Q6	200	2000	8
Q7	300	1000	14
Q8	300	1500	12
Q9	300	1500	10
Q10	500	2000	8

Tableau 2.2

note :Les arcs sont biorientés ; les segments sont égaux et disposés symétriquement par rapport à 0 .

L'analyse des temps de calcul donnés dans les tableaux 2.3 et 2.4 inspire les remarques suivantes :

- il est avantageux de développer une version du simplexe particularisée aux PFCCL . Pour résoudre les 30 problèmes , SGFLOW et SGFLOW1 utilisent respectivement 55 % et 35 % du temps de calcul de FLOW .

- l'efficacité de la méthode des pivots combinés dépend du nombre de morceaux composant les coûts de transit et de la charge du réseau . L'expérience montre que le nombre de parties remplies à l'optimum est un des facteurs déterminants . Le problème P11 possède le même rèseau que P12 , muni de capacités plus grandes ; ce réseau est peu chargé ce qui explique la mauvaise performance de SGFLOW1 .

nom	FLOW	SGFLOW	SGFLOW1
P1	1.4	1.1	0.5
P2	1.2	1.0	0.5
P3	2.8	1.6	1.2
P4	7.1	5.9	3.2
P5	17.8	10.0	3.8
P6	24.8	15.5	10.1
P7	15.8	12.6	6.7
P8	27.2	17.0	7.5
P9	46.1	28.3	18.6
P10	42.4	23.6	12.4
P11	35.6	26.9	27.7
P12	31.2	22.6	20.0
P13	50.2	36.9	26.5
P14	34.8	27.9	15.6
P15	41.3	31.2	22.0
P16	24.5	21.0	15.7
P17	54.0	48.9	62.0
P18	35.5	29.8	29.0
P19	45.3	36.4	21.7
P20	27.5	25.9	17.9
TOTAL:	566.5	424.1	322.6

Tableau 2.3

note : Les temps de calcul sont donnés en m.s. sur un CYBER 170/855 .

nom	FLOW	SGFLOW	SGFLOW1
Q1	60.0	9.7	2.9
Q2	23.6	4.8	2.1
Q3	118.3	18.5	5.0
Q4	41.5	9.9	4.8
Q5	35.5	9.6	5.8
Q6	77.3	24.3	11.6
Q7	41.1	14.8	5.6
Q8	62.9	20.3	10.9
Q9	38.9	14.9	9.9
Q10	55.9	26.6	17.5
TOTAL:	555.0	195.4	76.1

Tableau 2.4

note :Les temps de calcul sont donnés en m.s. sur un CYBER 170/855 .

En conclusion , la résolution de PFCCL en un temps minimum nécessite le développement d'un code spécialisé . La méthode des pivots combinés peut être facilement introduite dans tout code implantant un algorithme primal pour le PFCCL . Cette idée simple permet , selon les caractéristiques des réseaux , de réduire jusqu'à 50% du temps de calcul .

BIBLIOGRAPHIE

[1] R.K. AHUJA , J.L. BATRA , S.K. GUPTA
A parametric algorithm for convex cost network flow and related problems,
EJOR 16 (1984) p 222-235 .

[2] W.H. CUNNINGHAM
A network simplex method
Mathematical Programming 11 (1976) p 105-110 .

[3] T.R. ELKEN
The application of mathematical programming to loop feeder allocation
B.S.T.J. 59 no 4 (1980) p 479-500.

[4] L.R. FORD , D.R. FULKERSON
Flow in networks
Princeton (New Jersey, 1962).

[5] P.L. HAMMER , M. SEGAL
Area transfers in local area
unpublished work .

[6] P.A. JENSEN , J.W. BARNES
Network flow programming
Wiley (New-York, 1980).

[7] J.L. KENNINGTON , R.V. HELGASON
Algorithms for network programming
Wiley (New-York, 1980).

[8] D. KLINGMAN , A. NAPIER , J. STUTZ
NETGEN: a program for generating large scale capacited assignment, transportation and minimum cost flow network problems
Management Sciences 20 (1974) p 814-821.

[9] C.L. MONMA M. SEGAL

A primal algorithm for finding minimum cost flow in capacited networks with applications

Bell System Technical Journal 61 (1982) p 949-968.

[10] C. PASCHE

Flot à coût convexe linéaire par morceaux

RAIRO (1987) à paraître .

[11] J.M WOLLMER

Algorithm for targeting strikes in a lines of communication network .

Operations Research 18 (1970) p 497-515 .

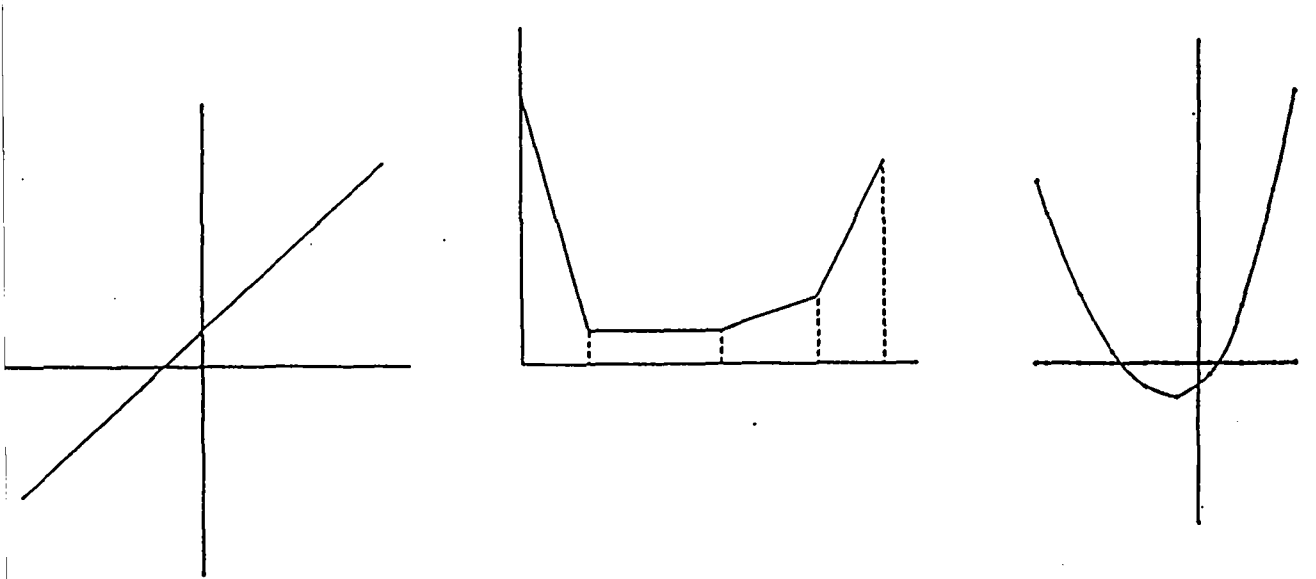
Partie 3

FLOT A COUT CONVEXE

3.0. INTRODUCTION .

Le problème de flot avec coûts convexes de transit (PFCC) est une extension naturelle du modèle de flot classique possédant un vaste champ d'application . Le paragraphe 3.6 traite de la modélisation de certains problèmes d'ingénieurs et classe les modèles équivalent au PFCC . En particulier , un nouveau modèle d'ordonnancement , caractérisé par des tâches de durée aléatoire , est présenté ainsi que la méthode de résolution par les flots .

Il est connu que le simplexe spécialisé pour traiter les contraintes de bilan résout de façon très performante le problème de flot avec des coûts linéaires . De manière identique , tous les algorithmes d'optimisation convexe sous contraintes linéaires peuvent être traduits en termes graphiques . Kennington et Helgason ([10]) présentent des algorithmes inspirés de la méthode de Frank-Wolfe , du gradient réduit et du convexe simplexe . Dembo ([2],[4]) a développé un algorithme utilisant l'information apportée par la seconde dérivée pour construire une direction de descente ; ce code sera notre base de référence pour les tests .



La figure ci-dessus inspire les constatations suivantes : une fonction convexe peut être considérée comme linéaire avec un nombre infini de morceaux et un coût linéaire par morceaux peut se ramener au cas linéaire en éclatant les arcs

du réseau . Le moyen le plus simple pour résoudre un PFCC consiste à approcher les coûts convexes par des fonctions linéaires par morceaux , puis à minimiser cette approximation par le simplexe . Cette méthode est efficace pour obtenir une solution grossière , mais inapplicable lorsqu'une grande précision est requise ([1]) .

Cette résolution par approximation linéaire a également donné lieu à des études théoriques . En particulier , Minoux ([12]) l'a utilisée pour prouver l'existence d'un algorithme polynomial dans le cas de coûts quadratiques . Meyer ([11]) a développé le concept de programmation en deux segments , où il considère une approximation locale formée de deux morceaux .

Cette partie présente un algorithme du simplexe résolvant de manière spécifique les problèmes approchés . Il est capable de traiter des approximations formées de plusieurs dizaines de milliers de morceaux par arcs et d'obtenir ainsi une solution très précise . L'étude théorique et les tests effectués montrent sa compétitivité tant du point de vue de la précision que de celui du temps calcul .

3.1. PRESENTATION DU PROBLEME

Considérons un réseau R=(V,E) formé d'un ensemble V de n sommets et d'un ensemble E de m arcs. Le PFCC s'énonce ainsi :

$$\text{Minimiser}_{X_{(i,j)}} \sum_{(i,j)\in E} \mathbf{F}_{(i,j)}(X_{(i,j)}) \qquad (3.1)$$

s.c.

$$\sum_{(i,j)\in E} X_{(i,j)} - \sum_{(j,i)\in E} X_{(j,i)} = B_i \qquad \forall\, i \in V \quad (3.2)$$

$$L_{(i,j)} \leqslant X_{(i,j)} \leqslant U_{(i,j)} \qquad (3.3)$$

où

- $X_{(i,j)}$ représente le flot transitant sur l'arc (i,j)

- $L_{(i,j)}$ est la borne inférieure de flot sur l'arc (i,j)

- $U_{(i,j)}$ est la borne supérieure de flot sur l'arc (i,j)

- B_i est le bilan au sommet i ; les bilans satisfont $\sum_{i\in V} B_i = 0$

- $\mathbf{F}_{(i,j)}$ est une fonction définie de $[\, L_{(i,j)} , U_{(i,j)} \,]$ dans $\mathbf{R}$ qui représente le coût de transfert de l'arc (i,j).

Deux hypothèses seront faites sur la forme de ces coûts :

- la convexité qui est nécessaire pour assurer que les algorithmes convergent vers une solution optimale.

- la dérivabilité des coûts qui est utilisée pour borner les erreurs d'approximation . La première dérivée de $\mathbf{F}_{(i,j)}$ est notée $\mathbf{G}_{(i,j)}$, la seconde $\mathbf{H}_{(i,j)}$.

Au problème d'optimisation PFCC , on associe un système appelé ***système de Kuhn et Tucker*** (SKT) . Il s'agit de déterminer un flot $X \in \mathbf{R}(E)$ (équations 3.2 et 3.3) et des variables duales $W \in \mathbf{R}(V)$ satisfaisant :

$$G_{(i,j)}(X_{(i,j)}) + W_j - W_i \geq 0 \qquad \forall\ (i,j) \in E \text{ avec } X_{(i,j)} < U_{(i,j)} \quad (3.4)$$

$$W_i - W_j - G_{(i,j)}(X_{(i,j)}) \geq 0 \qquad \forall\ (i,j) \in E \text{ avec } X_{(i,j)} > L_{(i,j)} \quad (3.5)$$

Une équivalence entre PFCC et SKT est définie par la propriété 3.1 .

Propriété 3.1.

Un flot $X \in \mathbf{R}(E)$ est solution du PFCC si et seulement si il existe $W \in \mathbf{R}(V)$ tel que (X,W) soit une solution de SKT .

Démonstration :

Dans le cadre général de l'optimisation d'une fonction convexe sous contraintes linéaires , le théorème de Kuhn et Tucker permet de caractériser les solutions optimales ([14]) . Le problème peut se formuler ainsi :

$$\begin{aligned} &\text{Minimiser } \mathbf{F}(X) \\ &\quad X \\ &\text{s.c. } AX = B \\ &\qquad MX \leq U \\ &\qquad PX \geq L \end{aligned}$$

où - $X \in \mathbf{R}(m)$ est la variable

- $A \in \mathbf{R}(n_1 \times m)$, $M \in \mathbf{R}(n_2 \times m)$ et $P \in \mathbf{R}(n_3 \times m)$ sont les matrices des contraintes .
- $B \in \mathbf{R}(n_1)$, $U \in \mathbf{R}(n_2)$ et $L \in \mathbf{R}(n_3)$ sont les seconds membres des contraintes .

Pour cette formulation du problème , l'énoncé du théorème de Kuhn et Tucker est le suivant :

$X \in \{ X \in \mathbf{R}(m) \mid AX = 0 , MX \leqslant U$ et $PX \geqslant L \}$ est une solution optimale du problème si et seulement si il existe des multiplicateurs où potentiels $W \in \mathbf{R}(n_1)$, $Y \in \mathbf{R}(n_2)$ et $Z \in \mathbf{R}(n_3)$ tels que :

$Y \geqslant 0 \quad Z \geqslant 0$
$Y'(U - MX) = 0$
$Z'(PX - L) = 0$
$\mathbf{GRAD(F)}(X) - W'A + Y'M - Z'P = 0$

(**GRAD(F)** note le vecteur gradient de **F**)

Dans le cas particulier du PFCC , ce théorème se simplifie en remarquant que :

- $A \in \mathbf{R}(V \times E)$ est la matrice d'incidence du réseau $R = (V,E)$;
- P et M sont des matrices identité ;
- $\mathbf{GRAD(F)}_{(i,j)} = \mathbf{G}_{(i,j)} \qquad \forall (i,j) \in E$

Les conditions nécessaires et suffisantes pour qu'un flot $X \in \mathbf{R}(E)$ soit optimal se reformulent alors de la manière suivante :

$\exists W \in \mathbf{R}(V)$ et $\exists Y , Z \in \mathbf{R}(E)$ tels que
$Y_{(i,j)} \geqslant 0 \quad Z_{(i,j)} \geqslant 0 \qquad \forall (i,j) \in E$
$Y_{(i,j)}(U_{(i,j)} - X_{(i,j)}) = 0 \qquad \forall (i,j) \in E$
$Z_{(i,j)}(X_{(i,j)} - L_{(i,j)}) = 0 \qquad \forall (i,j) \in E$
$\mathbf{G}_{(i,j)}(X_{(i,j)}) + W_j - W_i + Y_{(i,j)} - Z_{(i,j)} = 0 \qquad \forall (i,j) \in E$

Les variables Y et Z peuvent être éliminées pour obtenir SKT.

Q.E.D

Dans le paragraphe 3.6. , cette équivalence est utilisée pour modéliser des problèmes de réseau en termes de PFCC .

3.2. ALGORITHME

Afin de pouvoir utiliser les résultats des parties précédentes , on approche les fonctions de coût de transit par des fonctions linéaires par morceaux . Pour obtenir une solution très précise il est nécessaire de diviser le coût en un grand nombre de segments (plusieurs dizaines de milliers par arc) . Dès lors deux difficultés apparaissent :

- premièrement , il est impossible de mémoriser les données caractérisant tous les segments ;

- secondement , les évaluations de fonctions étant coûteuses en temps calcul , il est inconcevable de construire autant de segments .

Le premier obstacle est surmonté en stockant uniquement l'information sur les segments utilisés à un instant donné de l'algorithme . Le calcul d'une solution de départ grossière et son affinement progressif permettent d'augmenter grandement la vitesse de résolution .

3.2.1. Approximation linéaire

Soit un flot $O \in \mathbf{R}(E)$, un ensemble $\{ \mathbf{C}_{(i,j)} , (i,j) \in E \}$ de fonctions linéaires par morceaux est une ***interpolation d'ordre*** Δ ***centrée en*** O des coûts de transit $\{ \mathbf{F}_{(i,j)} , (i,j) \in E \}$ lorsque :

$$\mathbf{C}_{(i,j)}(O_{(i,j)}+k\Delta) = \mathbf{F}_{(i,j)}(O_{(i,j)}+k\Delta) \qquad \forall\, k \text{ t.q. } O_{(i,j)}+k\Delta \in [\, L_{(i,j)} , U_{(i,j)} \,] \quad \forall\, (i,j) \in E \quad (3.6)$$

Il est évident que le centre O et l'ordre Δ déterminent de façon univoque l'interpolation . Par contre un grand nombre de centres donne la même interpolation d'ordre Δ . La figure 3.1 illustre l'approximation d'une fonction convexe par des segments linéaires autour d' un centre O . On remarque que dans ce cas le domaine de variation du flot est diminué .

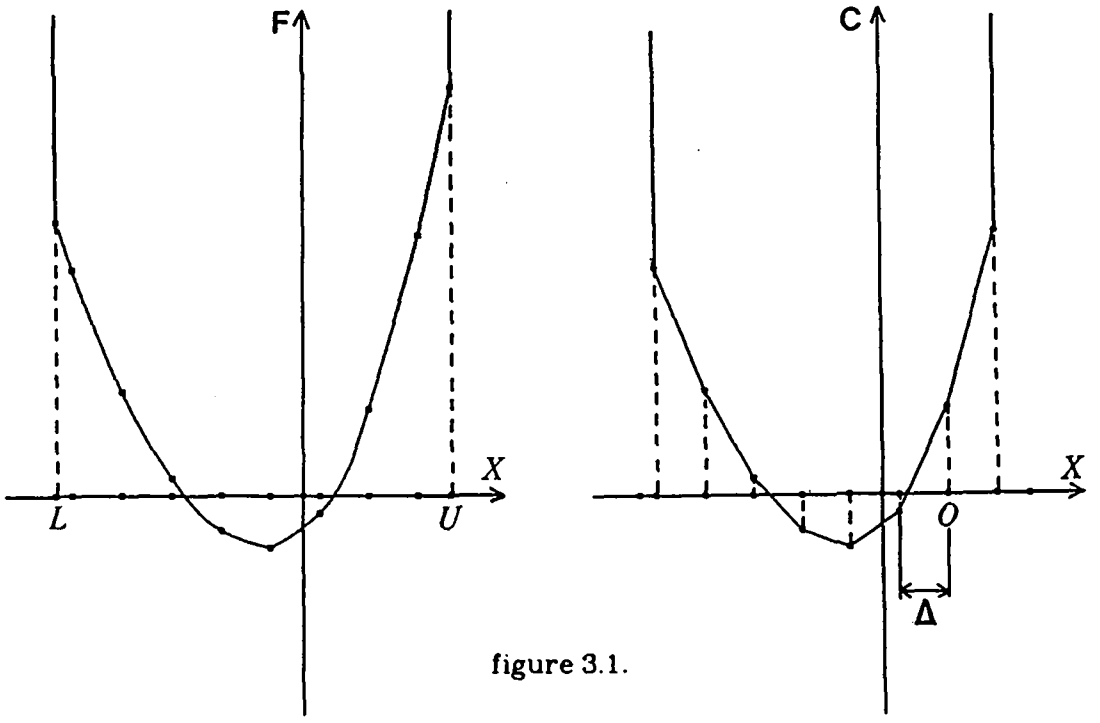


figure 3.1.

Les fonctions $\mathbf{F}_{(i,j)}$ étant convexes , les interpolations le seront également et l'algorithme développé dans la partie 2 peut être utilisé pour résoudre le problème de flot associé à une interpolation d'ordre Δ autour du flot initial . Cependant on peut améliorer cet algorithme en calculant les pentes des fonctions d'interpolation non pas à priori mais lorsqu'elles sont utilisées . Pour ce faire on associe deux coûts à tout arc (i,j) où circule un flot $X_{(i,j)}$:

- ***le coût d'augmentation*** :

$$\mathbf{ca}(i,j) = \begin{cases} (\mathbf{F}_{(i,j)}(X_{(i,j)}+\Delta) - \mathbf{F}_{(i,j)}(X_{(i,j)}))/\Delta & \text{si } X_{(i,j)}+\Delta \leq U_{(i,j)} \\ \infty & \text{sinon} \end{cases} \tag{3.8}$$

- ***le coût de diminution*** :

$$\mathbf{cd}(i,j) = \begin{cases} (\mathbf{F}_{(i,j)}(X_{(i,j)}) - \mathbf{F}_{(i,j)}(X_{(i,j)}-\Delta))/\Delta & \text{si } X_{(i,j)}-\Delta \geq L_{(i,j)} \\ \infty & \text{sinon} \end{cases} \tag{3.9}$$

Une étape de l'algorithme est caractérisée par :

- une base $T \subset E$ qui est un arbre de racine $r \in V$;

- un flot de base $X \in \mathbf{R}(E)$ satisfaisant :

$$(X_{(i,j)} - O_{(i,j)})/\Delta \in \mathbb{Z} \qquad \forall (i,j) \in E$$

- les états e(i,j) associés aux arcs (i,j) ∈ E . Cette fonction binaire e(i,j) vaut a si on veut augmenter le flot sur (i,j) et d si on veut le diminuer ;

- les variables duales W_i associées aux sommets i ∈ E . La longueur W_i de l'unique chaîne de T allant de i vers la racine r est calculée en utilisant les coûts des arcs correspondant à leur état .

- les deux coûts réduits associés aux arcs hors base (i,j) ∈ E-T :
  - *le coût réduit d'augmentation* : **ra**(i,j) = W_j - W_i + **ca**(i,j) (3.9)
  - *le coût réduit de diminution* : **rd**(i,j) = W_i - W_j - **cd**(i,j) (3.10)

Ces définitions permettent de formuler un algorithme du type simplexe pour l'optimisation dans un réseau avec coûts convexes .

Algorithme 3.1. ALGORITHME DU SIMPLEXE D'ORDRE Δ

0. Initialisation

- Trouver un flot initial { $X_{(i,j)} = O_{(i,j)}$; (i,j) ∈ E }.
- Choisir une base T et des états e(i,j) , ∀ (i,j) ∈ T .
- Calculer les variables duales W_i ,∀ i ∈ V .

1. Choix de l'arc entrant

- Choisir un arc (p,q) ∈ E-T de coût réduit négatif .
- S'il n'en existe pas alors aller à **5**.
- Fixer l'état de l'arc entrant (p,q) ainsi :
  si **rd**(p,q) < 0 alors e(p,q) := d
  si **ra**(p,q) < 0 alors e(p,q) := a

2. Recherche du cycle de descente

- Chercher le cycle C formé par l'arc entrant (p,q) et l'arbre T .
- Orienter C selon (p,q) si e(p,q)=a et à l'inverse de (p,q) si e(p.q)=d

3. Modification du flot le long du cycle

3.1. Modification des états le long du cycle

- Si le coût réduit de C est positif alors aller en **4**.
- Choisir un arc (u,v) de C tel que :

  soit e(u,v)=a et (u,v) est orienté négativement sur C

  soit e(u,v)=d et (u,v) est orienté positivement sur C
- Si (u,v) existe alors inverser e(u,v) et aller en **3.1**.

3.2. Avance de flot le long du cycle

- Modifier $X_{(i,j)}$ pour les arcs (i,j) $\in$ C en faisant avancer Δ unités de flot
- Inverser les états e(i,j) des arcs (i,j) $\in$ C .
- Retour en **3.1**.

4. Mise à jour de la base

- T := T + (p,q) - (u,v)
- Mettre à jour les variables duales W_i ,$\forall$ i $\in$ V
- Retour à **1**.

5. Fin .

Remarque : Le test de 3.1 est toujours vrai lors du premier passage ; en effet pour entrer dans le point 3 le coût réduit de C doit être négatif .

La propriété 3.1 justifie que pour Δ petit l'algorithme 3.1 fournit une solution du PFCC acceptable .

Propriété 3.1.

L'algorithme 3.1 obtient un flot $X \in \mathbf{R}(E)$ minimisant l'interpolation d'ordre Δ centrée autour du flot initial O.

Démonstration :

Le lecteur vérifiera aisément que l'algorithme 3.1 correspond à l'algorithme 2.2 où l'on a tenu compte du fait que les modifications de flot sont soit nulles soit d'amplitude égale à l'ordre Δ . Q.E.D

Corollaire 3.2.

L'algorithme 3.1 obtient un flot $X \in \mathbf{R}(E)$ minimisant l'interpolation d'ordre Δ centrée en X.

Démonstration :

Tous les flots rencontrés par l'algorithme peuvent être pris comme centre de l'interpolation initiale .

Q.E.D

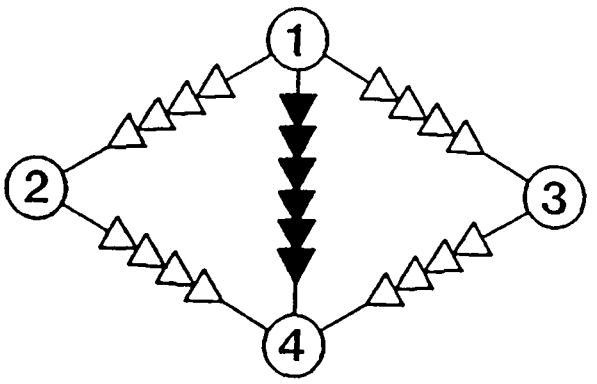
Dans la figure 3.2 , l'algorithme 3.1 est utilisé pour résoudre un problème de flot à coût quadratique pour une interpolation donnée par les entiers ($\Delta = 1$).

Caractéristiques des arcs

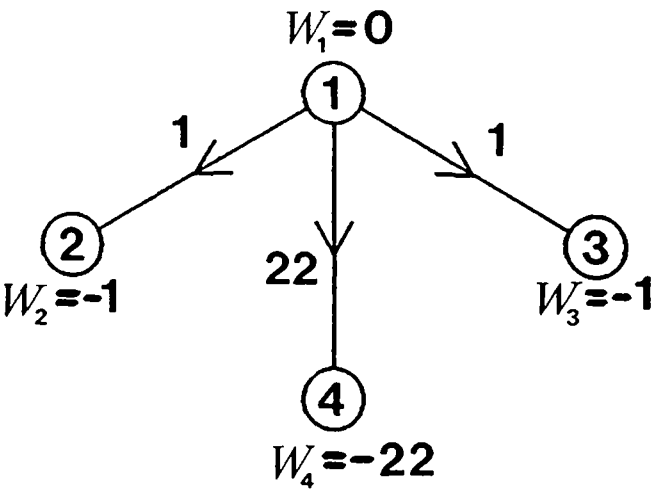
arcs	(1,2)	(1,3)	(1,4)	(2,4)	(3,4)
F(x)	x**2	x**2	2x**2	x**2	x**2
ca	2x+1	2x+1	4x+2	2x+1	2x+1
cd	2x-1	2x-1	4x-2	2x-1	2x-1
e	a	a	d	a	a
0	0	0	6	0	0
L	0	0	0	0	0
U	4	4	6	4	4

figure 3.2.

Réseau :

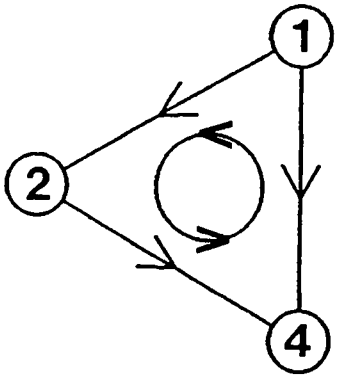


Base :



L'arc entrant est (2,4) : ra(2,4)= -20

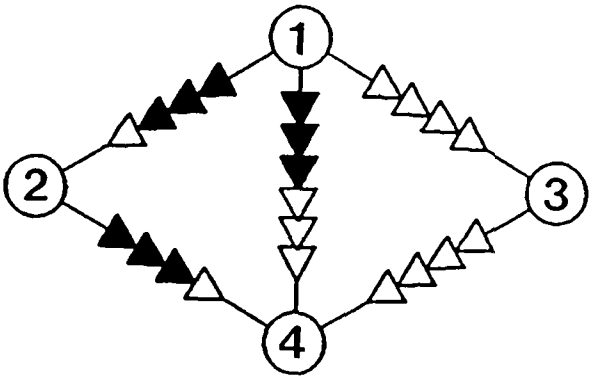
Les points 3.1 et 3.2 donnent les itérations suivantes :

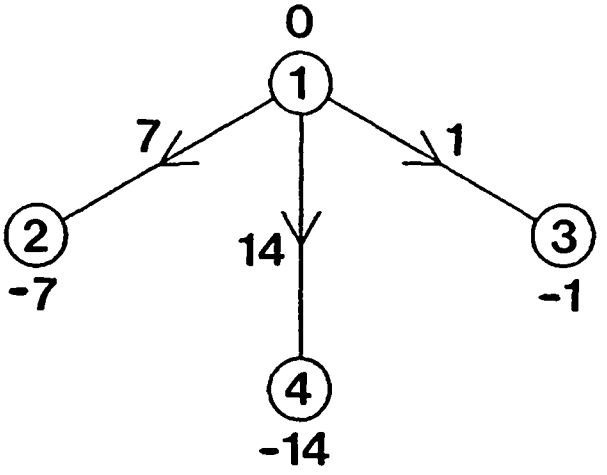


(1,2)	(2,4)	(1,4)	coût
x e	x e	x e	de C
0 a	0 a	6 d	-20
1 d	1 d	5 a	-20
1 a	1 d	5 a	-18
1 a	1 a	5 a	-16
1 a	1 a	5 d	-12
2 d	2 d	4 a	-12
2 a	2 d	4 a	-10
2 a	2 a	4 a	-8
2 a	2 a	4 d	-4
3 d	3 d	3 a	-4
3 a	3 d	3 a	-2
3 a	3 a	3 a	0

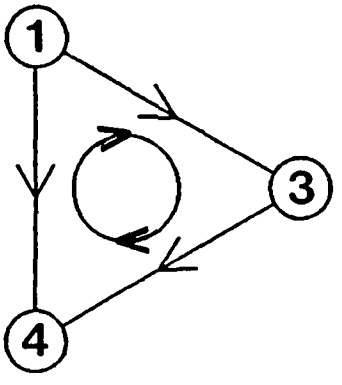
L'arc (2,4) sort de la base

figure 3.2. (suite)





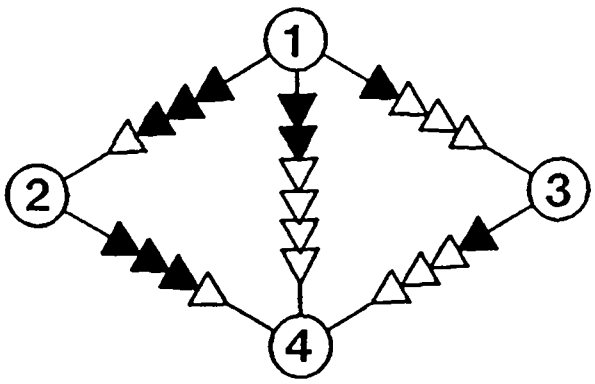
L'arc entrant est (3,4) : ra(3,4)= -12

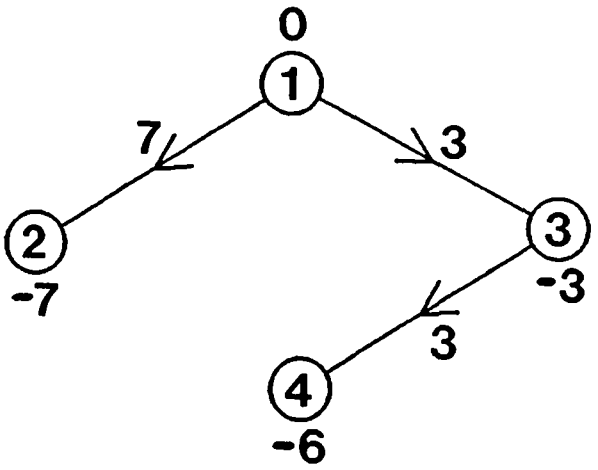


(1,3)	(3,4)	(1,4)	coût
x e	x e	x e	de C
0 a	0 a	3 a	-12
0 a	0 a	3 d	-8
1 d	1 d	2 a	-8
1 a	1 d	2 a	-6
1 a	1 a	2 a	-4
1 a	1 a	2 d	0

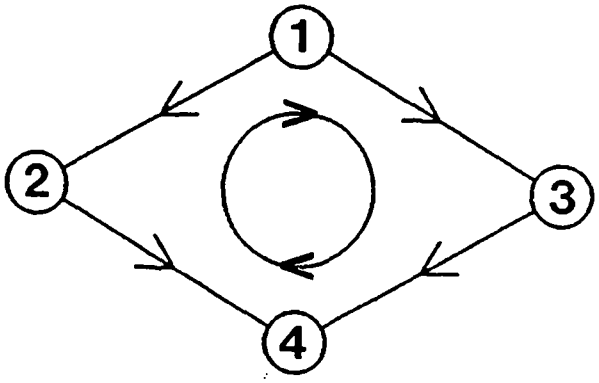
L'arc (1,4) sort de la base

figure 3.2. (suite)





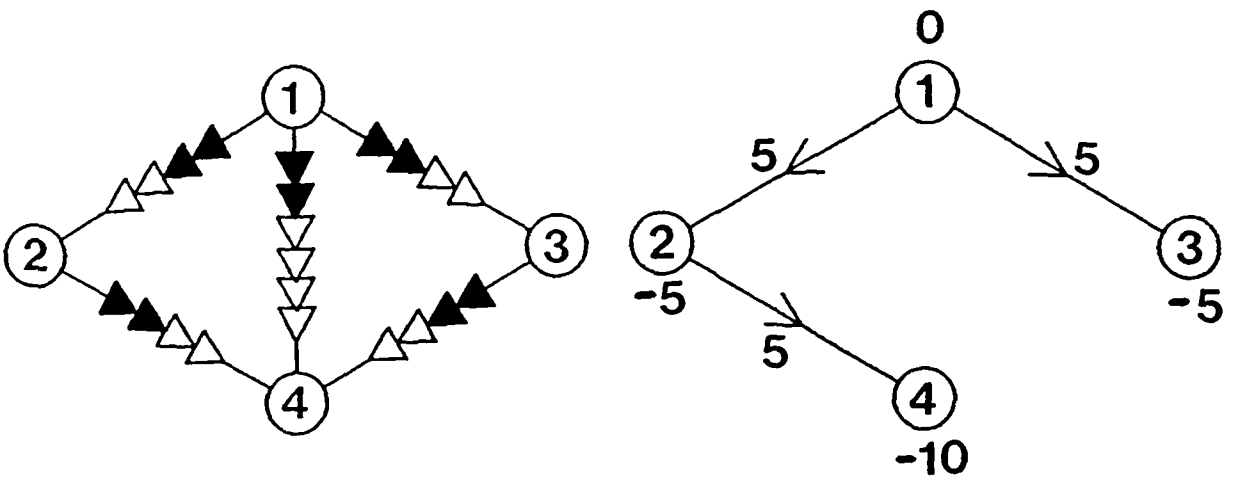
L'arc entrant est (2,4) : rd(2,4)= -6



(1,3)	(3,4)	(2,4)	(1,2)	coût
x e	x e	x e	x e	de C
1 a	1 a	3 d	3 a	-6
1 a	1 a	3 d	3 d	-4
2 d	2 d	2 a	2 a	-4
2 a	2 d	2 a	2 a	-2
2 a	2 a	2 a	2 a	0

L'arc (3,4) sort de la base

figure 3.2. (suite)



Les coûts réduits sont tous positifs ; l'algorithme s'arrête .

figure 3.2. (fin)

L'algorithme d'optimisation convexe dans les réseaux qui ressemble le plus à l'algorithme 3.1 est le convexe simplexe (les pages 192 et 193 de [10] donnent une description détaillée de cet algorithme). Bien que ces deux algorithmes utilisent des directions de descente engendrées par l'adjonction d'un arc à l'arbre de base , leurs trajectoires sont en général différentes . Appliqué à l'exemple de la figure 3.2 , le convexe simplexe conserve toujours la même base et produit la suite des flots :

arcs		(1,2)	(2,4)	(1,4)	(1,3)	(3,4)
itération	0	0.0	0.0	6.0	0.0	0.0
	1	3.0	3.0	3.0	0.0	0.0
	2	3.0	3.0	1.5	1.5	1.5
	3	2.25	2.25	2.25	1.5	1.5
	4	2.25	2.25	1.875	1.875	1.875
	...	...	...	...	...	...

Il est bien évident que l'algorithme 3.1 n'a pas toujours la chance d'obtenir une solution optimale pour le PFCC . La différence principale entre ces deux algorithmes est que l'algorithme 3.1 change plus souvent de base que le convexe simplexe ; ceci permet de considérer un plus grand nombre de directions de descente .

3.2.2. Approximation itérative

Etant donnée une séquence de k ordres d'interpolation $\Delta_1 \geq \Delta_2 \geq ... \geq \Delta_k$, il est facile d'utiliser l'algorithme 3.1 de manière itérative .

Algorithme 3.2. ALGORITHME DU SIMPLEXE POUR UNE SEQUENCE DE k ORDRES

0. Initialisation

- $\Delta := \Delta_1$
- i := 1
- Trouver un flot initial $\{ X_{(i,j)} = O_{(i,j)} ; (i,j) \in E \}$
- Choisir une base T et des états e(i,j) , $\forall$ (i,j) $\in$ E
- Calculer les variables duales W_i , $\forall$ i $\in$ V

1,2,3,4. de l'algorithme 3.1

5. Initialisation pour le traitement de l'ordre Δ_{i+1}

- si i=k alors fin de l'algorithme
- i := i+1
- $\Delta := \Delta_i$
- Calculer les variables duales W_i , $\forall$ i $\in$ V
- retour en **1.**

Corollaire 3.3

L'algorithme 3.2 obtient un flot $X \in \mathbb{R}(E)$ minimisant l'interpolation d'ordre Δ_k centrée en X .

Démonstration

Pour la dernière étape on applique l'algorithme 3.1 en partant du flot obtenu à l'avant-dernière étape ; le corollaire 3.2 permet de conclure . Q.E.D

La contrainte d'intégralité

Dans certains modèles , l'unité de flot est indivisible ; c'est en particulier le cas lorsqu'elle correspond à une personne ou à un véhicule . On doit alors résoudre le ***problème de flot entier à coût convexe*** qui est défini comme le PFCC (points (3.1),(3.2) et (3.3)) plus la contrainte d'intégralité du flot ; c'est-à-dire :

$$X_{(i,j)} \in \mathbb{Z} \qquad \forall\ (i,j) \in E \quad (3.11)$$

Pour garantir l'existence d'une telle solution on suppose de plus l'intégralité des bornes de flot ($U,L \in \mathbb{Z}(E)$) et celle des bilans ($B \in \mathbb{Z}(V)$) . L'algorithme 3.2 permet de résoudre ce problème de manière exacte .

Propriété 3.4.

En partant d'un flot initial entier $O \in \mathbb{Z}(E)$ **et pour une séquence d'ordres entiers** $\Delta_1 \geq \Delta_2 \geq \ldots \geq \Delta_k = 1$, $\Delta_i \in \mathbb{N}$ **l'algorithme 3.2 fournit un flot** $X \in \mathbb{Z}(E)$ **qui est une solution du problème du flot entier à coût minimum .**

Démonstration :

Le flot initial étant entier et les modifications de flot Δ_i l'étant également , tous les flots obtenus en cours d'algorithme sont entiers . Par le corollaire 3.3 et comme $\Delta_k = 1$, le flot final est optimal pour la fonction coût obtenue en interpolant le coût convexe entre les entiers . Il est évident que , sous la contrainte d'intégralité , la valeur du coût entre les entiers n'a aucune importance et que par conséquent le flot entier obtenu est optimal pour le coût convexe considéré .

Q.E.D

Une autre approximation .

L'interpolation linéaire n'est pas l'unique façon d'approcher une fonction convexe par une fonction linéaire par morceaux . Pour les tests une variante de l'algorithme à été considérée , où les coûts d'augmentation (3.7) et de diminution (3.8) sont remplacés par :

$$\mathbf{ca}(i,j) = \begin{cases} G_{(i,j)}(X_{(i,j)}+\Delta/2) & \text{si } X_{(i,j)}+\Delta \leqslant U_{(i,j)} \\ \infty & \text{sinon} \end{cases} \qquad (3.12)$$

$$\mathbf{cd}(i,j) = \begin{cases} G_{(i,j)}(X_{(i,j)}-\Delta/2) & \text{si } X_{(i,j)}-\Delta \geqslant L_{(i,j)} \\ \infty & \text{sinon} \end{cases} \qquad (3.13)$$

3.3. CONVERGENCE ET PRECISION

Ce paragraphe rassemble tous les résultats visant à évaluer la précision de l'algorithme 3.2 pour résoudre un PFCC . Les démonstrations sont basées sur des développements de Taylor d'ordre 2 et sont inspirées de celles données en [12] pour le cas quadratique .

Considérons les éléments suivants :

- $\{ \mathbf{F}_{(i,j)} , (i,j) \in E \}$ les coûts de transits du PFCC . La première dérivée de $\mathbf{F}_{(i,j)}$ est notée $\mathbf{G}_{(i,j)}$, la seconde $\mathbf{H}_{(i,j)}$.

- $Y \in \mathbf{R}(E)$ un flot optimal pour le PFCC ; c'est-à-dire que :
  $\mathbf{F}(Y) = \sum_{(i,j)\in E} \mathbf{F}_{(i,j)}(Y_{(i,j)})$ est minimum .

- $\{ \mathbf{C}_{(i,j)} , (i,j) \in E \}$ une interpolation d'ordre Δ centrée en X.

- $X \in \mathbf{R}(E)$ un flot obtenu par l'algorithme 3.2 ; c'est-à-dire que :
  $\mathbf{C}(X) = \sum_{(i,j)\in E} \mathbf{C}_{(i,j)}(X_{(i,j)})$ est minimum .

Erreur d'approximation .

Pour commencer , il est nécessaire de comparer les fonctions $\mathbf{F}_{(i,j)}$ et $\mathbf{C}_{(i,j)}$; la propriété 3.5 énonce un résultat d'analyse élémentaire qui permet de borner l'erreur commise en remplaçant les coûts par leurs interpolations .

Propriété 3.5.

$\forall\ x \in [\ L_{(i,j)} , U_{(i,j)}\ ] , \forall\ (i,j) \in E$ **il existe** $T_{(i,j)} \in [-\Delta , \Delta]$ **tel que**
$\mathbf{C}_{(i,j)}(x) - \mathbf{F}_{(i,j)}(x) \leq \Delta^2\, \mathbf{H}_{(i,j)}(x+T_{(i,j)})/2$

Démonstration :

Supposons que x soit dans le segment [d , d+Δ[

Erreur = $C_{(i,j)}(x) - F_{(i,j)}(x) = F_{(i,j)}(d) + (F_{(i,j)}(d+\Delta)-F_{(i,j)}(d))(x-d)/\Delta - F_{(i,j)}(x)$

Considérons les développements de Taylor

$F_{(i,j)}(x) = F_{(i,j)}(d) + G_{(i,j)}(d)(x-d) + H_{(i,j)}(d+t_1\Delta)(x-d)^2/2$ $\quad t_1 \in [0,1]$

$F_{(i,j)}(d+\Delta) = F_{(i,j)}(d) + G_{(i,j)}(d)\Delta + H_{(i,j)}(d+t_2\Delta)\Delta^2/2$ $\quad t_2 \in [0,1]$

Ainsi

Erreur = $H_{(i,j)}(d+t_2\Delta)(x-d)\Delta/2 - H_{(i,j)}(d+t_1\Delta)(x-d)^2/2$

Or $x-d < \Delta$ et par convexité $H_{(i,j)}(x+t_1\Delta) \geq 0$; par conséquent

Erreur $\leq H_{(i,j)}(d+t_2\Delta)\Delta^2/2 = H_{(i,j)}(x+T_{(i,j)})\Delta^2/2$

en posant $T_{(i,j)} = t_2\Delta+d-x \in [-\Delta,\Delta]$

Q.E.D

Erreur sur le coût total .

Le but de l'algorithme étant de minimiser le coût total de transfert , la distance séparant la solution approchée X de la solution réelle Y peut être mesurée par $F(X)-F(Y)$.

Propriété 3.6.

$$F(X) - F(Y) \leq \Delta^2 \sum_{(i,j)\in E} H_{(i,j)}(Y_{(i,j)}+T_{(i,j)})/2$$

avec $T_{(i,j)} \in [-\Delta,\Delta]\ \forall (i,j) \in E$

Démonstration :

On tire des définitions la suite d'inégalités suivante :

$F(Y) \leq F(X) \leq C(X) \leq C(Y)$

Ainsi $F(X) - F(Y) \leq C(Y) - F(Y)$

Par la propriété 3.4 , on obtient le résultat désiré .

Q.E.D

Erreur sur les flots .

Dans la pratique , on s'intéresse généralement au flot et non à la valeur minimum du coût ; il est donc important d'évaluer l'erreur commise sur les flots . La propriété 3.7 transpose la propriété 3.6 en termes de flot .

Propriété 3.7.

$$\sum_{(i,j)\in E} H_{(i,j)}(Y_{(i,j)}+t(X_{(i,j)}-Y_{(i,j)}))(X_{(i,j)}-Y_{(i,j)})^2 \leq \Delta^2 \sum_{(i,j)\in E} H_{(i,j)}(Y_{(i,j)}+T_{(i,j)})$$

avec $T_{(i,j)} \in [-\Delta,\Delta]$ $\forall$ $(i,j) \in E$ **et** $t \in [0,1]$

Démonstration :

Considérons le développement de Taylor de $\mathbf{F}$ autour de Y:

$$\mathbf{F}(X) - \mathbf{F}(Y) = \sum_{(i,j)\in E} G_{(i,j)}(Y_{(i,j)})(X_{(i,j)}-Y_{(i,j)}) + \sum_{(i,j)\in E} H_{(i,j)}(Y_{(i,j)}+t(X_{(i,j)}-Y_{(i,j)}))(X_{(i,j)}-Y_{(i,j)})^2 /2$$

La première somme est positive car Y est optimum et $X-Y$ est une direction admissible . La propriété 3.6 permet d'obtenir la borne désirée .

Q.E.D

Erreur sur le SKT .

Dans le paragraphe 3.6 , l'algorithme 3.2 est utilisé pour résoudre des systèmes du même type que SKT . Soit $W \in \mathbf{R}(V)$ le vecteur des variables duales obtenues par l'algorithme , la propriété 3.8 montre que X et W sont solutions d'un système très proche de SKT .

Propriété 3.8.

$\forall$ $(i,j) \in E$, **il existe** $T_{(i,j)} \in [0,\Delta]$ **et** $S_{(i,j)} \in [-\Delta,0]$ **tels que** :

$G_{(i,j)}(X_{(i,j)}) + W_j - W_i \geq -H_{(i,j)}(X_{(i,j)}+T_{(i,j)})\Delta/2$ **si** $X_{(i,j)} \leq U_{(i,j)}-\Delta$

$W_i - W_j - G_{(i,j)}(X_{(i,j)}) \geq -H_{(i,j)}(X_{(i,j)}+S_{(i,j)})\Delta/2$ **si** $X_{(i,j)} \geq L_{(i,j)}+\Delta$

Démonstration :

Par définition de X, $\forall$ (i,j) $\in$ E avec $X_{(i,j)} \leq U_{(i,j)}-\Delta$ alors

ra(i,j) = $W_j - W_i$ + **ca**(i,j) $\geq$ 0

Par développement de Taylor

ca(i,j) = ($\mathbf{F}_{(i,j)}(X_{(i,j)}+\Delta) - \mathbf{F}_{(i,j)}(X_{(i,j)})$)/Δ = $\mathbf{G}_{(i,j)}(X_{(i,j)}) + \mathbf{H}_{(i,j)}(X_{(i,j)}+T_{(i,j)})\Delta/2$

où $T_{(i,j)} \in [0,\Delta]$.

En introduisant cette expression dans **ra**(i,j) on obtient la première inégalité .

La seconde se démontre de manière similaire en utilisant **rd**(i,j) et **cd**(i,j) .

Q.E.D

Il est intéressant de souligner que cette erreur varie en fonction de Δ alors que les erreurs précédentes dépendent de Δ^2 .

Cas quadratique .

Les résultats précédents peuvent être affinés et simplifiés pour des coûts quadratiques de transit ; c'est-à-dire :

$\mathbf{F}_{(i,j)}(x) = Q_{(i,j)}x^2/2 + P_{(i,j)}x$ $\qquad$ $Q_{(i,j)} \geq 0$ $\forall$ (i,j) $\in$ E

Propriété 3.9.

- $\mathbf{C}_{(i,j)}(x) - \mathbf{F}_{(i,j)}(x) \leq Q_{(i,j)}\Delta^2/8$ $\qquad$ $\forall$ x , $\forall$ (i,j) $\in$ E

- $\mathbf{F}(X) - \mathbf{F}(Y) \leq \Delta^2 \sum\limits_{(i,j)\in E} Q_{(i,j)}/8$

- $\sum\limits_{(i,j)\in E} Q_{(i,j)}(X_{(i,j)}-Y_{(i,j)})^2 \leq \Delta^2 \sum\limits_{(i,j)\in E} Q_{(i,j)}/4$

- $\forall$ (i,j) $\in$ E :

  $Q_{(i,j)}X_{(i,j)} + P_{(i,j)} + W_j - W_i \geq -Q_{(i,j)}\Delta/2$ $\qquad$ **si** $X_{(i,j)} \leq U_{(i,j)}-\Delta$

  $W_i - W_j - Q_{(i,j)}X_{(i,j)} - P_{(i,j)} \geq -Q_{(i,j)}\Delta/2$ $\qquad$ **si** $X_{(i,j)} \geq L_{(i,j)}+\Delta$

Démonstration :

De la démonstration de la propriété 3.5 , nous tirons :

Erreur = $C_{(i,j)}(x) - F_{(i,j)}(x) = Q_{(i,j)}(x-d)\Delta/2 - Q_{(i,j)}(x-d)^2/2$

Le maximum de l'erreur pour $x \in [d,d+\Delta]$ est atteint en $x = d+\Delta/2$; ainsi

Erreur $\leqslant Q_{(i,j)}\Delta^2/4 - Q_{(i,j)}\Delta^2/8 = Q_{(i,j)}\Delta^2/8$

Les autres points ont la même démonstration que dans le cas général.

Q.E.D

3.4. IMPLANTATION

Ce paragraphe présente les différences majeures entre l'implantation de l'algorithme 3.2 et celles décrites en 1.3 et 2.4 . La difficulté principale est de mémoriser les coûts de transit et leurs interpolations linéaires ; les procédures de pivotage restant dans les grandes lignes inchangées .

Il faut remarquer que , pour pouvoir utiliser les mêmes procédures de pivotage dans le calcul du flot initial et dans les itérations suivantes , le cas d'un flot non situé à un point de cassure doit être pris en compte . En effet la première itération trouve , par la méthode big-M , un flot admissible optimisant l'interpolation initiale . Or les productions et les demandes de flot n'étant ,en général , pas des multiples de l'ordre initial Δ_1 , il est impossible de trouver un flot admissible uniquement par des modifications de Δ_1 unités de flot . Pour utiliser au mieux cette possibilité de manipuler des flots quelconques , l'interpolation définie en 3.2.1 est complétée par un segment à gauche et un segment à droite pour atteindre les bornes réelles de flot . Ainsi dans le programme les segments et les modifications de flot peuvent être différents de l'ordre .

3.4.1. Mémorisation des fonctions coûts

Comme il est impossible de lire sur un fichier de données les fonctions coûts , un certain nombre de fonctions convexes sont disponibles dans le programme de façon standard ; il est facile d'en définir d'autres . Ces fonctions prédéfinies sont numérotées et dépendent de trois paramètres . Cette implantation est identique à celle décrite dans [3] . Ainsi les coûts de transit peuvent être mémorisés par les quatre tableaux de dimension m suivant :

NFONC(.) : la case NFONC(k) mémorise le numéro de la fonction prédéfinie correspondant au coût de transit

PARAM(1..3,.): la case PARAM(i,k) contient le i-ème paramètre de la fonction correspondant à l'arc numéro k.

Les domaines de définition des fonctions sont donnés par les tableaux LB(.) et UB(.) mémorisant les bornes inférieures et supérieures de flot sur les arcs.

Pour calculer les coûts de diminution ou d'augmentation , il faut connaitre soit la pente de l'interpolation entre deux points , soit la fonction dérivée . Ces deux fonctions sont également prédéfinies .

3.4.2. Mémorisation des interpolations

L'information utilisée par l'algorithme est différente selon que l'arc appartient à la base ou est hors base . L'approximation sur les arcs hors base ne sert qu'à calculer les coûts réduits . Pour les arcs de la base , elle est utilisée pour calculer les variables duales et pour déterminer l'amplitude des modifications de flot .

Pour un arc hors base on mémorise le flot et les deux coûts associés à l'arc . On utilise trois tableaux :

FLOW(.) : la composante FLOW(k) donne le flot sur l'arc numéro k .

LCOST(.) : le coût de diminution de l'arc k est stocké dans LCOST(k) .

RCOST(.) : le coût d'augmentation de l'arc k est stocké dans RCOST(k) .

Pour un arc de la base les caractéristiques du segment où est situé le flot sont mémorisées dans trois tableaux :

LFLOW(.) : la composante LFLOW(k) donne le point de cassure à gauche du segment courant de l'arc k .

CAP(.) : la capacité du segment courant de l'arc k est stockée en CAP(k)

COST(.) : le coût sur le segment courant de l'arc k est stocké en COST(k).

La fraction de flot circulant dans les segments courants est mémorisée dans le tableau VAR(.) selon la convention décrite en 1.3.2 . Un arc étant soit dans la base soit hors base , trois tableaux de dimensions m suffisent à mémoriser l'interpolation :

tableau 1 : FLOW(.) ≡ LFLOW(.)
tableau 2 : LCOST(.) ≡ CAP(.)
tableau 3 : RCOST(.) ≡ COST(.)

En résumé la place mémoire totale occupée par ce code est de dix tableaux de dimension m et de sept tableaux de dimension n .

3.5. TESTS

L'algorithme 3.2 est paramétrisé par la séquence d'ordres $\Delta_1 \geqslant \Delta_2 \geqslant \ldots \geqslant \Delta_k$. L'influence de cette séquence est double :

- la précision de la solution dépend de l'ordre final Dk , comme montré au paragraphe 3.3.

- Δ_k étant fixé , la rapidité de l'algorithme varie en fonction de la séquence d'approximations successives utilisées pour obtenir cet ordre final .

Le premier but de ces tests est d'étudier la précision et la rapidité de l'algorithme relativement à la séquence employée . Cette analyse permet de formuler des recommandations pour le choix de cette séquence .

Le second est de comparer le code développé avec NLPNET ([3]) fournit par R. Dembo . Les trois programmes testés sont les suivants :

- CFLOW0 : implantation de l'algorithme 3.2 décrite dans le paragraphe 3.4. Les coûts linéaires sont évalués à l'aide des fonctions de coût (équations 3.7 et 3.8) .

- CFLOW1 : variante où les coûts linéaires sont évalués à l'aide de la dérivée des fonctions de coût (équations 3.12 et 3.13) .

- NLPNET : code de R. Dembo . L'algorithme utilisé génère des directions de descente admissibles par une variante de l'algorithme de Newton (la description de cet algorithme se trouve dans [2] et [4]). La connaissance des deux premières dérivées est requise .

Ces trois programmes sont écrits en FORTRAN 77 .

3.5.1. Les problèmes

Les tests seront effectués sur cinq problèmes issus de la pratique . Pour chacun de ces problèmes , les trois programmes convergent vers le même flot . Par la suite , nous noterons Y le flot "optimal" obtenu avec un ordre final très petit (plus de vingt fois inférieur à ceux indiqués dans les tableaux) . Dans les tableaux de résultats , X désignera à la solution obtenue par l'algorithme étudié .

Nom	Nb. de sommets	Nb. d'arcs	F(Y)	||Y||	Moyenne de |Y|	[min ,max]
PB1	150	196	-482E2	144.	6.	[-40, 40]
PB2	667	906	-206E3	809.	11.	[-126,228]
PB3	136	191	588E1	178.	9.	[-50, 50]
PB4	466	668	675.	6.	1E-1	[-2, 2]
PB5	64	117	-713E4	2E6	1E5	[0 ,6E5]

tableau 3.1

notes :
- ||Y|| est la norme usuelle du vecteur Y.
- |Y| est le vecteur formé des valeurs absolues des composantes de Y.
- [min,max] sont les valeurs extrêmes des composantes de Y.

Le tableau 3.1 présente les caractéristiques numériques des réseaux . Les problèmes PB1 , PB2 et PB5 ont été fournis par R. Dembo et sont utilisés dans [2] . Il est également intéressant de connaître le modèle correspondant à chacun de ces problèmes :

- PB1 et PB2 sont des données tirées du réseaux de distribution d'eau de la ville de Dallas . Pour modéliser les pompes et les variations de pression , trois types de fonctions coût de transit sont utilisées :

```
F(X) = A*X + B
F(X) = C*abs(X)**2.85
F(X) = -0.5*(X*sqrt(B*(A-X*X)) + D*arcsin(X/C)
```

- PB3 et PB4 sont des réseaux électriques . Il faut minimiser la perte totale d'énergie .Pour chaque arc la fonction perte est du type :

```
F(X) = A*X*X
```

- Pb5 est tiré d'un modèle d'estimation de matrice input-output pour la Thaîlande . Les fonctions d'entropie utilisées sont :

```
F(X) = A*X + B
F(X) = X*(log(X/A) - 1)
```

3.5.2. Analyse des résultats

Analyse du processus itératif

Le but de ce paragraphe est de déterminer la séquence d'ordre à utiliser pour obtenir un ordre final donné . Le tableau 3.2 présente , dans le cas du programme CFLOW0 appliqué à PB2 , les temps de calcul nécessaires pour obtenir un ordre final de 0.0005 par différentes séquences . L'analyse de ces temps inspire les remarques suivantes :

- Pour les tests a , b et c , l'algorithme passe d'un ordre initial de 30.0 à un ordre final de 0.0005 en divisant , à chaque itération , l'ordre par 256 , 16 , 4 respectivement . En comparant b et c , on remarque que le petit nombre d'itérations de b compense partiellement la grande variation d'ordre d'une itération à l'autre .

- Les tests c , g et h correspondent à des divisions successives de l'ordre par 4 , 5 et 3 respectivement . L'algorithme semble très robuste relativement à ce facteur .

- Dans les tests c , d , e et f , plusieurs ordres initiaux ont été comparés. La variation du temps de calcul est , là encore , très faible .

Séquence d'ordres							
a	b	c	d	e	f	g	h
30.	30.	30.	120.	8.	2.	30.	30.
.12	2.	8.	30.	2.	.5	6.	10.
.0005	.12	2.	8.	.5	.12	1.2	3.3
	.008	.5	2.	.12	.03	.24	1.1
	.0005	.12	.5	.03	.008	.05	.4
		.03	.12	.008	.002	.01	.13
		.008	.03	.002	.0005	.002	.04
		.002	.008	.0005		.0005	.013
		.0005	.002				.004
			.0005				.0013
							.0005
46.7	13.3	12.8	13.4	12.8	14.1	11.9	13.3

Temps de calcul

tableau 3.2

note : les temps de calcul sont donnés en secondes sur un CYBER 170/855

Sur la base de tests équivalents pour les autres problèmes , la séquence d'ordre optimale doit possèder les caractéristiques suivantes :

- L'ordre initial est une estimation de l'amplitude moyenne du flot .

- A chaque itération l'ordre est divisé par 4 où 5 .

Analyse de la précision

Le dernier paramètre à déterminer est l'ordre final ; il dépend bien évidemment de la précision désirée . Le tableau 3.3 présente la convergence de l'algorithme pour le test c du tableau précédent . Pour ce test l'ordre est divisé par 4 à chaque itération . Les observations sont les suivantes :

- La durée d'une itération est constante (sauf pour l'itération initiale) .

- L'erreur sur le coût est divisée par 16 . C'est l'illustration de la propriété 3.6.

- La norme du vecteur des erreurs est divisée par 4 ; ce phénomène est à rapprocher de la propriété 3.7.

- L'erreur de flot maximale commise sur un arc peut être estimée par l'ordre final .

Ordre final	Temps calcul	F(X)-F(Y)	||X-Y||	Maximum de |X-Y|
.5	5.6	18.6671	5.214	.554
.12	7.0	1.0974	1.367	.166
.03	8.5	.0704	.334	.037
.008	9.9	.0048	.083	.010
.002	11.3	.0003	.022	.002
.0005	12.8	.0000	.005	.001

tableau 3.3

note : les temps de calcul sont donnés en secondes sur un CYBER 170/855.

Dans tous les tests effectués sur les cinq problèmes , l'erreur de flot maximale n'a jamais dépassé le double de l'ordre final . Cette observation permet de formuler une règle heuristique d'arrêt .

Comparaison des trois codes

Le tableau 3.4 présentent les durées de résolution des cinq problèmes par CFLOW0 , CFLOW1 et NLPNET . Les ordres finaux ont été choisis de sorte que les solutions obtenues par les trois algorithmes soient de précisions comparables . L'analyse de ces résultats montre que CFLOW0 et CFLOW1 sont compétitifs ; le petit nombre de problèmes ne permet pas d'être plus précis .

L'avantage majeur de CFLOW0 et CFLOW1 est une grande facilité d'utilisation . Seuls deux paramètres , l'ordre initial et l'ordre final , sont à déterminer et leurs interprétations sont simples puisqu'ils correspondent à l'amplitude moyenne de flot et à la précision désirée . De plus l'algorithme semble robuste relativement à la séquence d'ordre choisie . Les paramètres de NLPNET sont plus nombreux et plus difficiles à ajuster .

L'utilisation de NLPNET nécessite l'existence de la seconde dérivée des fonctions coûts . L'information que tire NLPNET de cette hypothèse ne semble pas produire un gain significatif de précision ou de rapidité par rapport à CFLOW0 qui n'utilise que les fonctions coûts .

CFLOW0 et CFLOW1 n'ont pas été comparés directement à une implantation du convexe simplexe . Mais les tests effectués dans [2] montrent que NLPNET est beaucoup plus rapide que l'algorithme du convexe simplexe spécialisé pour les réseaux proposé en [10] .

Prob	Code	Ordre	Temps calcul	F(X)-F(Y)	||X+Y||	Maximum de |X-Y|
PB1	CFLOW0	.002	1.5	6E-5	7E-3	1E-3
PB1	CFLOW1	.002	1.3	6E-5	9E-3	2E-3
PB1	NLPNET	-	0.8	10E-5	22E-3	7E-3
PB2	CFLOW0	.0001	14.4	0.0	1E-3	1E-4
PB2	CFLOW1	.0001	13.0	0.0	1E-3	1E-4
PB2	NLPNET	-	17.0	0.0	3E-3	12E-4
PB3	CFLOW0	.008	1.0	30E-5	4E-2	7E-3
PB3	CFLOW1	.008	1.0	30E-5	4E-2	7E-3
PB3	NLPNET	-	0.7	5E-5	2E-2	11E-3
PB4	CFLOW0	.0005	4.7	1E-5	4E-3	4E-4
PB4	CFLOW1	.0005	4.7	1E-5	4E-3	4E-4
PB4	NLPNET	-	7.3	2E-5	12E-3	62E-4
PB5	CFLOW0	1.5	1.0	4E-2	6.6	1.6
PB5	CFLOW1	1.5	0.9	4E-2	6.2	1.9
PB5	NLPNET	-	1.1	6E-2	60.8	18.0

tableau 3.4.

note : les temps de calcul sont donnés en secondes sur un CYBER 180/855.

3.6. PROBLEMES EQUIVALENTS

Le champ d'application de l'algorithme peut être étendu en utilisant la propriété 3.1 qui établit une équivalence entre un problème d'optimisation (PFCC) et la résolution d'un système (SKT) . Il est possible de définir trois types d'utilisation :

- ***l'utilisation primale*** où on calcule un flot qui minimise un coût ; c'est le PFCC , qui est à la base de l'algorithme .

- ***l'utilisation primale-duale*** où le problème est posé sous la forme d'un système du type SKT qu'il faut résoudre pour connaître le flot et les variables duales . Le paragraphe 3.6.1 décrit un ensemble de problèmes où l'on cherche un flot satisfaisant certaines conditions d'équilibre .

- ***l'utilisation duale*** où on doit résoudre un problème d'optimisation dans un réseau , les variables étant attachées aux sommets . L'ordonnancement à coût convexe décrit dans 3.6.2 en est un exemple .

3.6.1. Les problèmes d'équilibres

De nombreux problémes d'ingénierie possèdent comme structure de base un réseau et , après modélisation , la solution mathématique est un flot respectant certaines conditions d'équilibre . La littérature foisonne de tels modèles , en particulier :

- des modèles électriques [8]

- des modèles de réseau de distribution d'eau [1]

- des modèles d'étude de trafic urbain [13]

- des modèles d'estimation de matrice input-output en économie [2] , [5] et [12]

Le problème d'équilibre le plus simple dans un réseau R = (V,E) se formule mathématiquement de la façon suivante :

Déterminer $X \in \mathbf{R}(E)$ et $W \in \mathbf{R}(V)$ satisfaisant :

$$\mathbf{G}_{(i,j)}(X_{(i,j)}) + W_j - W_i = 0 \qquad \forall\ (i,j) \in E \quad (3.14)$$

$$\sum_{(i,j)\in E} X_{(i,j)} - \sum_{(j,i)\in E} X_{(j,i)} = B_i \qquad \forall\ (i,j) \in E \quad (3.15)$$

où

- B_i est le bilan au sommet i.

- $\mathbf{G}_{(i,j)} : \mathbf{R} \to \mathbf{R}$ est la ***fonction de potentiel*** qui est supposée croissante.

Par la propriété 3.1 , on montre que ce problème est équivalent au PFCC suivant :

$$\text{Minimiser} \sum_{(i,j)\in E} \mathbf{F}_{(i,j)}(X_{(i,j)}) \qquad (3.16)$$

s.c.

$$\sum_{(i,j)\in E} X_{(i,j)} - \sum_{(j,i)\in E} X_{(j,i)} = B_i \qquad \forall\ (i,j) \in E \quad (3.17)$$

où

- $\mathbf{F}_{(i,j)} : \mathbf{R} \to \mathbf{R}$ est la ***fonction d'énergie*** définie comme la primitive du potentiel $\mathbf{G}_{(i,j)}$.

Ainsi pour résoudre le problème d'équilibre par l'algorithme 3.2 , on l'applique au PFCC associé et on prend comme solution le flot et les variables duales obtenues à la fin de l'algorithme . Il est à remarquer que si l'on utilise la variante de l'algorithme où les coûts sont définis par (3.12) et (3.13) (page 113) , le calcul des fonctions $F_{(i,j)}$ n'est pas nécessaire . Ce type de problèmes va être illustré par un problème d'estimation (cette présentation est inspirée de [2] et [5]).

Un problème d'estimation de matrice

Etant donné une matrice de $\mathbf{R}_+(I \times J)$ qui varie au cours du temps , les données du problème sont :

- la matrice $M \in \mathbf{R}_+(I \times J)$ de l'année précédente
- la prévision $S \in \mathbf{R}_+(I)$ des sommes par ligne pour l'année en cours (Si est la valeur estimée de la somme pour la ligne i)
- la prévision $T \in \mathbf{R}_+(J)$ des sommes par colonne pour l'année en cours .

Il s'agit d'estimer sur la base de ces informations la matrice $X \in \mathbf{R}_+(I \times J)$ de l'année courante .

Les contraintes induites par les sommes sont les suivantes :

$$\sum_{j \in J} X_{i,j} = S_i \qquad \forall\, i \in I \quad (3.18)$$

$$\sum_{i \in I} X_{i,j} = T_j \qquad \forall\, j \in J \quad (3.19)$$

$$X_{i,j} \geqslant 0 \qquad \forall\, i \in I \ , \ \forall\, j \in J \quad (3.20)$$

Un des modèles pour le passage de M à X suppose l'existence de facteurs multiplicateurs cachés $P \in \mathbf{R}_+(I)$ et $Q \in \mathbf{R}_+(J)$ tels que :

$$X_{(i,j)} = P_i M_{(i,j)} Q_j \qquad \forall\, i \in I \ , \ \forall\, j \in J \quad (3.21)$$

On remarque que $X_{(i,j)}$ ne peut être nul que si $M_{(i,j)}$ est nul ou si on supprime la ligne i ($P_i = 0$ et $S_i = 0$) ou encore en éliminant la colonne j ($Q_j = 0$ et $T_j = 0$) . Le réseau $R = (V,E)$ sous-jacent est donc le suivant :

- l'ensemble des sommets est $V = \{ i \in I \mid S_i \neq 0 \} \cup \{ j \in J \mid T_j \neq 0 \}$
- l'ensemble des arcs est $E = \{ (i,j) \in V \times V \mid i \in I , j \in J \text{ et } M_{i,j} \neq 0 \}$

Lorsque les sommets de I sont des producteurs et ceux de J des consommateurs , les contraintes (3.18) et (3.19) sont les équations de bilan dans ce réseau . Les bornes inférieures de flot sont nulles et les bornes supérieures infinies . En prenant le logarithme de (3.21) on obtient :

$$\ln(X_{i,j}) = \ln(M_{i,j}) + \ln(P_i) + \ln(Q_j) \qquad \forall\ (i,j) \in E \quad (3.22)$$

Posons

$$W_k = \begin{cases} \ln(P_k) & \forall\ k \in I \cap V \\ -\ln(Q_k) & \forall\ k \in J \cap V \end{cases}$$

Les équations obtenues par cette substitution sont des équations d'équilibres comme définies par (3.14) :

$$\ln(X_{i,j}/M_{i,j}) + W_j - W_i = 0 \qquad \forall\ (i,j) \in E \quad (3.23)$$

Ce problème d'estimation se reformule en un PFCC dans le réseau biparti R = (V,E) :

$$\text{Minimiser} \sum_{(i,j)\in E} X_{(i,j)}\{ \ln(X_{(i,j)}/M_{i,j} - 1 \} \qquad (3.24)$$

s.c.

$$\sum_{(i,j)\in E} X_{(i,j)} = S_i \qquad i \in I \cap V \quad (3.25)$$

$$\sum_{(i,j)\in E} X_{(i,j)} = T_j \qquad j \in J \cap V \quad (3.26)$$

$$X_{(i,j)} \geq 0 \qquad (i,j) \in E \quad (3.27)$$

Un flot optimum $X \in \mathbb{R}(E)$ pour ce problème peut être obtenue par l'algorithme 3.2 ; il fournit l'estimation de la matrice suivante :

$$X_{i,j} = \begin{cases} X_{(i,j)} & \text{si } (i,j) \in E \\ 0 & \text{sinon} \end{cases} \qquad (3.28)$$

3.6.2. L'ordonnancement à coût convexe

Un problème important en pratique est la planification d'un projet ; un projet étant constitué d'un ensemble partiellement ordonné de tâches élémentaires . Cet ordre partiel provient de contraintes technologiques qui forcent certaines tâches à être achevées avant que d'autres puissent débuter . Lorsque les durées d'exécution des tâches sont disponibles , le problème peut être résolu efficacement par la méthode PERT ([7]) . En pratique , les durées d'exécution sont souvent des variables aléatoires ; le modèle d'ordonnancement proposé essaie de traiter cette indétermination .

De manière similaire à la méthode PERT , le projet est décrit par un réseau G = (V,E) , où chaque arc (i,j) ∈ E correspond à une tâche et chaque sommet à une étape . Dans ce modèle , la tâche (i,j) est caractérisée par :

- une durée minimum d'exécution $D_{(i,j)}$.
- un retard maximum $R_{(i,j)}$.
- la fonction de répartition **PROB**(retard de (i,j) ⩽ t) . Le retard de la tâche (i,j) est une variable aléatoire à valeurs dans [0 , $R_{(i,j)}$] . Les retards sont supposés être indépendants .

Le problème est de planifier le projet sur une période de D unités de temps ; ce qui implique de fixer , par avance , la date à laquelle chaque tâche devrait commencer . Cette date de début d'une tâche doit être choisie ni trop tard pour limiter la probabilité d'occasionner un retard pour le projet dans son ensemble , ni trop tôt pour éviter les coûts induits par un retard sur le début de cette tâche . Cet ordonnancement est basé sur des prévisions $D_{(i,j)} + T_{(i,j)}$ des temps d'exécution ; les retards prévus $T_{(i,j)}$ sont calculés pour maximiser la probabilité de n'avoir aucun retard :

$$\text{Maximiser} \quad \prod_{(i,j)\in E} \textbf{PROB}(\text{ retard de } (i,j) \leqslant T_{(i,j)})$$

Lorsque cette probabilité est non nulle , cet objectif peut se réécrire comme suit :

$$\text{Minimiser} \sum_{(i,j)\in E} \ln(\mathbf{PROB}(\text{ retard de } (i,j) \leq T_{(i,j)}))$$

Plus généralement , la fonction objectif sera de la forme

$$\text{Minimiser} \sum_{(i,j)\in E} \mathbf{F}_{(i,j)}(T_{(i,j)}) \qquad (3.29)$$

Les coûts $\mathbf{F}_{(i,j)}(T_{(i,j)})$ peuvent être par exemple :

- $C_{(i,j)}\mathbf{PROB}(\text{ retard de } (i,j) > T_{(i,j)})$, où $C_{(i,j)}$ est un poids positif .

- l'espérance du coût occasionné par un retard supérieur à la prévision $T_{(i,j)}$

- le prix à payer pour s'assurer que la durée de la tâche (i,j) est inférieur à la prévision $D_{(i,j)} + T_{(i,j)}$.

Cette troisième interprétation de $\mathbf{F}_{(i,j)}$ n'est plus probabiliste ; elle s'avère correspondre au problème d'ordonnancement avec coût de compression défini par Ford et Fulkerson ([6]) . Ces auteurs ont étudié le cas de fonctions linéaires et ont montré comment la solution peut être calculée en utilisant les flots . Dans notre modèle les coûts sont des fonctions convexes en général non linéaires .

Le développement de la méthode de résolution va s'effectuer en trois phases :

- premièrement le problème d'ordonnancement est formulé et des conditions d'optimalité sont données pour ce problème primal ,

- deuxièmement un problème de flot à coût convexe est défini ; c'est le problème dual ,

- finalement un théorème liant ces deux problèmes fournit une méthode pour résoudre le problème d'ordonnancement .

Le problème primal .

Les hypothèses suivantes sont faites sur les fonctions coûts $\mathbf{F}_{(i,j)} : [0 , R_{(i,j)}] \mapsto \mathbf{R}$ associées aux tâches:

- $\mathbf{F}_{(i,j)}$ est positive , décroissante , convexe et deux fois différentiable .

- la dérivée de $\mathbf{F}_{(i,j)}$, notée $\mathbf{DF}_{(i,j)}$, est inversible sur l'intervalle $[\mathbf{DF}_{(i,j)}(0) , \mathbf{DF}_{(i,j)}(R_{(i,j)})]$.

Des exemples de telles fonctions seront donnés ultérieurement . Une tâche (i,j) ayant une durée fixe est caractérisée par $R_{(i,j)} = 0$.

Le sommet représentant le début du projet est noté d et celui qui correspond à la fin est noté f . Le réseau peut être complété par un arc (f,d) ($E'' = E + (f,d)$).

Les variables du problème sont les suivantes :

- W_i représente la date correspondant à l'etape $i \in V$

- $T_{(i,j)}$ est le retard prévu pour la tâche $(i,j) \in E$; la durée prévue pour (i,j) est $D_{(i,j)} + T_{(i,j)}$.

Le problème d'ordonnancement se formule alors comme suit :

$$\text{Minimiser}_{T_{(i,j)}} \quad \sum_{(i,j)\in E} \mathbf{F}_{(i,j)}(T_{(i,j)}) \qquad (3.30)$$

s.c.

$$W_f - W_d \leq D \qquad (3.31)$$

$$W_j - W_i - D_{(i,j)} - T_{(i,j)} \geq 0 \qquad \forall (i,j) \in E \qquad (3.32)$$

$$0 \leq T_{(i,j)} \leq R_{(i,j)} \qquad \forall (i,j) \in E \qquad (3.33)$$

De par la forme des coûts , à l'optimum on aura :

$$T_{(i,j)} = \text{Minimum} (W_j - W_i - D_{(i,j)} , R_{(i,j)})$$

Pour ce problème , les conditions de Kuhn et Tucker sont nécessaires et suffisantes . A l'optimum , il existe des variables duales Y , $Z \in \mathbb{R}(E)$ et $X \in \mathbb{R}(E'')$ telles que :

$$X_{(t,d)} \geq 0$$

$$X_{(i,j)} \geq 0 \quad , \quad Y_{(i,j)} \geq 0 \quad , \quad Z_{(i,j)} \geq 0 \qquad \forall\ (i,j) \in E$$

$$\sum_{(i,j)\in E''} X_{(i,j)} - \sum_{(j,i)\in E''} X_{(j,i)} = 0 \qquad \forall\ i \in V$$

$$\mathbf{DF}_{(i,j)}(T_{(i,j)}) = - X_{(i,j)} + Y_{(i,j)} - Z_{(i,j)} \qquad \forall\ (i,j) \in E$$

$$X_{(t,d)}(W_t - W_d - D) = 0$$

$$X_{(i,j)}(W_j - W_i - D_{(i,j)} - T_{(i,j)}) = 0 \qquad \forall\ (i,j) \in E$$

$$T_{(i,j)} Y_{(i,j)} = 0 \qquad \forall\ (i,j) \in E$$

$$(R_{(i,j)} - T_{(i,j)}) Z_{(i,j)} = 0 \qquad \forall\ (i,j) \in E$$

Les variables Y et $Z \in \mathbb{R}(E)$ peuvent être supprimées pour donner les conditions équivalentes suivantes :

$$X_{(i,j)} \geq 0 \qquad \forall\ (i,j) \in E'' \qquad (3.34)$$

$$\sum_{(i,j)\in E''} X_{(i,j)} - \sum_{(j,i)\in E''} X_{(j,i)} = 0 \qquad \forall\ i \in V \qquad (3.35)$$

$$X_{(t,d)}(W_t - W_d - D) = 0 \qquad (3.36)$$

$$X_{(i,j)}(W_j - W_i - D_{(i,j)} - T_{(i,j)}) = 0 \qquad \forall\ (i,j) \in E \qquad (3.37)$$

$$T_{(i,j)} > 0 \ \Rightarrow\ \mathbf{DF}_{(i,j)}(T_{(i,j)}) \leq -X_{(i,j)} \qquad \forall\ (i,j) \in E \qquad (3.38)$$

$$T_{(i,j)} < R_{(i,j)} \ \Rightarrow\ \mathbf{DF}_{(i,j)}(T_{(i,j)}) \geq -X_{(i,j)} \qquad \forall\ (i,j) \in E \qquad (3.39)$$

Tout triplet de vecteurs X , T et W vérifiant (3.31),...,(3.39) correspond à une solution optimale du problème primal .

Le problème dual .

A chaque arc (i,j) ∈ E , on associe la fonction duale $G_{(i,j)}$ ayant comme dérivée la fonction suivante :

$$DG_{(i,j)}(x) = \begin{cases} R_{(i,j)} & x \in [0 \ , \ -DF_{(i,j)}(R_{(i,j)})] \\ (DF_{(i,j)})^{-1}(-x) & x \in [-DF_{(i,j)}(R_{(i,j)}) \ , \ -DF_{(i,j)}(0)] \\ 0 & x \in [-DF_{(i,j)}(0) \ , \ \infty[\end{cases} \quad (3.40)$$

Par intégration , on obtient la fonction non-décroissante , concave et différentiable suivante :

$$G_{(i,j)}(x) = \begin{cases} R_{(i,j)}x + F_{(i,j)}(R_{(i,j)}) - F_{(i,j)}(0) \\ x(DF_{(i,j)})^{-1}(-x) + F_{(i,j)}((DF_{(i,j)})^{-1}(-x)) - F_{(i,j)}(0) \\ 0 \end{cases} \quad (3.41)$$

On peut alors définir le problème de flot suivant :

$$\underset{X_{(i,j)}}{\text{Minimiser}} \quad DX_{(t,d)} + \sum_{(i,j)\in E} -(G_{(i,j)}(X_{(i,j)}) + D_{(i,j)}X_{(i,j)}) \quad (3.42)$$

s.c.

$$\sum_{(i,j)\in E''} X_{(i,j)} - \sum_{(j,i)\in E''} X_{(j,i)} = 0 \qquad \forall\ i \in E \quad (3.43)$$

$$X_{(i,j)} \geq 0 \qquad \forall\ (i,j) \in E'' \quad (3.44)$$

Pour ce problème , les conditions de Kuhn et Tucker sont également nécessaires et suffisantes . A l'optimum , il existe des variables duales $W \in \mathbf{R}(V)$ et $V \in \mathbf{R}(E'')$ satisfaisant :

$$V_{(i,j)} \geq 0 \qquad \forall\ (i,j) \in E''$$

$$D = W_t - W_d + V_{(t,d)}$$

$$V_{(t,d)}X_{(t,d)} = 0$$

$$-D_{(i,j)} - DG_{(i,j)}(X_{(i,j)}) = -W_j + W_i + V_{(i,j)} \qquad \forall\ (i,j) \in E$$

$$V_{(i,j)}X_{(i,j)} = 0 \qquad \forall\ (i,j) \in E$$

Les variables $V \in \mathbb{R}(E)$ peuvent être éliminées pour obtenir les conditions équivalentes suivantes :

$$W_t - W_d - D \geqslant 0 \tag{3.45}$$

$$X_{(t,d)}(W_t - W_d - D) = 0 \tag{3.46}$$

$$W_j - W_i - D_{(i,j)} - DG_{(i,j)}(X_{(i,j)}) \geqslant 0 \qquad \forall\ (i,j) \in E \tag{3.47}$$

$$X_{(i,j)}(W_j - W_i - D_{(i,j)} - DG_{(i,j)}(X_{(i,j)})) = 0 \qquad \forall\ (i,j) \in E \tag{3.48}$$

L'ordonnancement optimal .

L'algorithme pour trouver l'ordonnancement est le suivant :

Algorithme 3.3. ALGORITHME D"ORDONNANCEMENT A COUT CONVEXE

1. **Résolution du primal .**

   Par l'algorithme 3.2 résoudre le problème de flot . La solution optimale est donnée par un flot optimal $X \in \mathbb{R}(E'')$ et des variables duales associées $W \in \mathbb{R}(V)$.

2. **Construction de l'ordonnancement .**

   - Les dates de début sont données par les variables duales W_i , $\forall\ i \in V$
   - Les retards prévus sont

     $T_{(i,j)}$ = Minimum ($R_{(i,j)}$, $W_j - W_i - D_{(i,j)}$) $\qquad \forall\ (i,j) \in E$

Il est à remarquer qu' en utilisant les coûts réduits définis en (3.12) et (3.13) (page 113) seule DG(i,j) est utilisée ; ce qui évite d'avoir à intégrer .

Propriété 3.10.

A toute solution optimale $X \in \mathbb{R}(E'')$ et $W \in \mathbb{R}(V)$ du problème dual , le point 2 de l'algorithme associe une solution optimale du problème primal .

Plus précisément , $X \in \mathbb{R}(E'')$ et $W \in \mathbb{R}(V)$ satisfont aux contraintes (3.43) à (3.48) et cette propriété équivaut à montrer que $W \in \mathbb{R}(V)$ et $T \in \mathbb{R}(E)$ satisfont aux contraintes (3.31) à (3.39) .

Lemme $\quad T_{(i,j)} = \mathbf{DG}_{(i,j)}(X_{(i,j)}) \qquad \forall\ (i,j) \in E$

Démonstration

- cas $X_{(i,j)} = 0$

  Par (3.47) $\quad W_j - W_i - D_{(i,j)} - \mathbf{DG}_{(i,j)}(0) = W_j - W_i - D_{(i,j)} - R_{(i,j)}$

  ainsi , au point 2 , $\quad T_{(i,j)} = R_{(i,j)}$

- cas $X_{(i,j)} > 0$

  Par (3.48) $\quad W_j - W_i - D_{(i,j)} = \mathbf{DG}_{(i,j)}(X_{(i,j)}) \geqslant 0$

  ainsi , au point 2 , $\quad T_{(i,j)} = W_j - W_i - D_{(i,j)} = \mathbf{DG}_{(i,j)}(X_{(i,j)})$

Q.E.D.

Démonstration de la propriété .

- (3.31) est équivalent à (3.45) .
- (3.32) est équivalent à (3.47) par le lemme .
- (3.33) est vérifié trivialement par la définition de $\mathbf{DG}_{(i,j)}$.
- (3.34),(3.35),(3.36) sont exactement semblables à (3.43),(3.44),(3.46) .
- (3.37) est équivalent à (3.48) par le lemme .
- (3.38) si $0 < T_{(i,j)} = \mathbf{DG}_{(i,j)}(X_{(i,j)}) < R_{(i,j)}$ alors par définition de $\mathbf{DG}_{(i,j)}$

  $\mathbf{DF}_{(i,j)}(T_{(i,j)}) = \mathbf{DF}_{(i,j)}(\mathbf{DG}_{(i,j)}(X_{(i,j)})) = -X_{(i,j)}$

  si $T_{(i,j)} = \mathbf{DG}_{(i,j)}(X_{(i,j)}) = R_{(i,j)}$ alors par définition de $\mathbf{DG}_{(i,j)}$

  $\mathbf{DF}_{(i,j)}(R_{(i,j)}) \leqslant -X_{(i,j)}$
- (3.39) si $T_{(i,j)} = \mathbf{DG}_{(i,j)}(X_{(i,j)}) = 0$ alors par définition de $\mathbf{DG}_{(i,j)}$

  $\mathbf{DF}_{(i,j)}(0) \geqslant -X_{(i,j)}$

  si $0 < T_{(i,j)} = \mathbf{DG}_{(i,j)}(X_{(i,j)}) < R_{(i,j)}$ alors par définition de $\mathbf{DG}_{(i,j)}$

  $\mathbf{DF}_{(i,j)}(T_{(i,j)}) = \mathbf{DF}_{(i,j)}(\mathbf{DG}_{(i,j)}(X_{(i,j)})) = -X_{(i,j)}$

Q.E.D.

Il est possible d'étendre cette approche à des fonctions coûts qui ne statisfont aux hypothèses que par morceaux . Pour conclure voici quelques exemples de fonctions coûts duales.

Exemple 1 : Coût linéaire $\mathbf{F}_{(i,j)}(x) = C_{(i,j)}(R_{(i,j)}-x)$ avec $C_{(i,j)} > 0$

$$\mathbf{DG}_{(i,j)} = \begin{cases} R_{(i,j)} & x \in [0\ ,\ C_{(i,j)}] \\ 0 & x \in [C_{(i,j)}\ ,\ \infty[\end{cases}$$

$$\mathbf{G}_{(i,j)} = \begin{cases} R_{(i,j)}(X_{(i,j)} - C_{(i,j)}) & x \in [0\ ,\ C_{(i,j)}] \\ 0 & x \in [C_{(i,j)}\ ,\ \infty[\end{cases}$$

Exemple 2 : Coût quadratique $\mathbf{F}_{(i,j)}(x) = C_{(i,j)}(x - R_{(i,j)})^2$ avec $C_{(i,j)} > 0$

$$\mathbf{DG}_{(i,j)} = \begin{cases} R_{(i,j)} - x/2C_{(i,j)} & x \in [0,2C_{(i,j)}R_{(i,j)}] \\ 0 & x \in [2C_{(i,j)}R_{(i,j)}\ ,\ \infty[\end{cases}$$

$$\mathbf{G}_{(i,j)} = \begin{cases} R_{(i,j)}x - x^2/4C_{(i,j)} - C_{(i,j)}R_{(i,j)}^2 & x \in [0,2C_{(i,j)}R_{(i,j)}] \\ 0 & x \in [2C_{(i,j)}R_{(i,j)}\ ,\ \infty[\end{cases}$$

Exemple 3 : Coût exponentiel $\mathbf{F}_{(i,j)}(x) = \mathbf{exp}(-C_{(i,j)}x)$ avec $C_{(i,j)} > 0$

$$\mathbf{DG}_{(i,j)} = \begin{cases} R_{(i,j)} & x \in [0\ ,\ P_{(i,j)}] \\ (\mathbf{ln}(C_{(i,j)}) - \mathbf{ln}(x))/C_{(i,j)} & x \in [P_{(i,j)}\ ,\ C_{(i,j)}] \\ 0 & x \in [C_{(i,j)}\ ,\ \infty[\end{cases}$$

$$\mathbf{G}_{(i,j)} = \begin{cases} R_{(i,j)}x + \mathbf{exp}(-C_{(i,j)}R_{(i,j)}) - 1 & x \in [0\ ,\ P_{(i,j)}] \\ (x\mathbf{ln}(C_{(i,j)}) - x\mathbf{ln}(x) + x - C_{(i,j)})/C_{(i,j)} & x \in [P_{(i,j)}\ ,\ C_{(i,j)}] \\ 0 & x \in [C_{(i,j)}\ ,\ \infty[\end{cases}$$

où $P_{(i,j)} = C_{(i,j)}\mathbf{exp}(-C_{(i,j)}R_{(i,j)})$

BIBLIOGRAPHIE

[1] M. COLLINS , L. COOPER , R. HELGASON , J. KENNINGTON , L. LE BLANC
Solving the pipe network analysis problem using optimization techniques
Management Science 7 (1978) p 147 - 760

[2] R.S. DEMBO
A primal truncated Newton algorithm with application to large-scale nonlinear network optimization
Yale School of Organization and Management (1983)

[3] R.S. DEMBO
NLPNET - User's guide and system documentation
Yale School of Organization and Management (1983)

[4] R.S. DEMBO , J.G. KLINCEWICZ
A scaled reduced gradient algorithm for network flow problems with convex separable costs
Mathematical Programming Study 15 (1981) p 125 - 147

[5] J. ERIKSON
A note on the solution of large sparse maximum entropy problems with linear equality constraints
Mathematical Programming 18 (1980) p 146 - 154

[6] L.R. FORD , D.R. FULKERSON
Flots dans les graphes
Gauthier-Villars (Paris , 1967)

[7] M. GONDRAN , M. MINOUX
Graghes et algorithmes
Eyrolles (Paris , 1979)

[8] E. HOBSON , D.L. FLETCHER , W.O. STADLIN
Network flow linear programming techniques and their application to fluel scheduling and contingency analysis
IEEE PAS 103 (1984) p 1684 - 1691

[9] C.Y. KAO , R.R. MEYER
Secant approximation methods for convex optimization
Mathematical Programming Study 14 (1981) p 143 - 162

[10] J.L. KENNINGTON , R.V. HELGASON
Algorithms for network programming
J. Wiley (New-York , 1980)

[11] R.R. MEYER
Two-segment separable programming
Management Science 4 (1979) p 385 - 395

[12] M. MINOUX
A polynomial algorithm for minimum quadratic cost flow problems
EJOR 18 (1984) P 377 - 387

[13] S.NGUYEN
Une approche unifiée des méthodes d'équilibre pour l'affectation du trafic
Thèse de doctorat (Université de Montréal , 1974)

[14] W.I. ZANGWILL
Nonlinear programming : a unified approach
Prentice Hall (Englewood Cliffs , 1969)

Chapitre 4

CALCUL DE L'EQUILIBRE DANS UN RESEAU DE TRANSPORT

4.0. INTRODUCTION

Toute étude d'un système de transport comprend une phase d'analyse et de prévision de la demande ; le lecteur intéressé trouvera une description détaillée du processus de planification d'un système de transport dans [3] . Pour un réseau routier , l'affectation du trafic correspond à la détermination des itinéraires empruntés par les automobilistes pour se rendre de leur origine à leur destination et de la charge induite sur les tronçons de route . Pour calculer cette affectation , il est nécessaire de faire des hypothèses sur le comportement de l'usager . Le lecteur trouvera en [1] un survol des modèles d'affectation et une bibliographie détaillée .

Pour prédire un trafic moyen ou un trafic à l'heure de pointe , le modèle d'équilibre du trafic est généralement utilisé . Ce dernier découle de l'hypothèse de Wardrop , à savoir que l'automobiliste choisit un itinéraire minimisant une certaine fonction d'utilité (en général son temps de transport) . Un inventaire des méthodes de calcul de ce trafic d'équilibre est présenté dans [4] . Le but principal de cette partie est de développer un algorithme d'affectation basé sur les algorithmes d'optimisation convexe décrits précédemment .

Bien évidemment , la prévision du trafic est précédée d'une phase d'acquisition de données pour déterminer d'une part les caractéristiques du réseau (capacité et longueur des routes , sens uniques) et d'autre part la demande . Pour évaluer la demande en transport , la région étudiée est découpée en zones ; le caractère homogène de la zone permettra de la traiter ultérieurement comme un seul point générant et absorbant du trafic . Le volume du déplacement interzonal est obtenu par des comptages ou par des enquêtes .

4.1. LE MODELE

Le but de ce paragraphe est d'introduire certaines notations et de définir mathématiquement les équations d'équilibre modélisant l'affectation du trafic . Pour pouvoir mettre en oeuvre un algorithme d'optimisation , le problème sera reformulé en termes de flot avec des coûts convexes .

4.1.1. Le réseau

Le réseau de transport est schématisé de manière naturelle par un réseau $G = (V,E)$. Les sommets représentent :

- une zone origine ou destination ; l'ensemble des sommets générateurs de trafic est noté $O \subset V$.

- une intersection de routes .

- un point de changement des caractéristiques d'une route .

Cette classification n'est pas exclusive ; une zone peut être à la fois génératrice de trafic , destination d'un trafic provenant d'une autre zone et lieu de transit . Les arcs correspondent à des tronçons unidirectionnels de route ; pour modéliser une route où la circulation se fait dans les deux sens , on introduit deux arcs de sens opposés .

Le trafic interzonal est donné par une matrice $B \in \mathbb{R}(O \times V)$. La ligne $B_{l.}$, indicée par l'origine $l \in O$, est un vecteur de bilan . L'interprétation des termes de cette ***matrice de déplacement interzonal*** est la suivante :

- $B_{l,l} > 0$ correspond à une génération de trafic en $l \in O$,

- $(-B_{l,i}) > 0$ est le volume du trafic se déplaçant de la zone $l \in O$ à la zone $i \in V$,

- $B_{l,i} = 0$ indique que le sommet $i \in V$ n'est qu'un point de transit pour le trafic issu de la zone $l \in O$.

Le but de l'étude est de déterminer le volume de trafic sur les tronçons constituant le réseau . Dans ce modèle , on différencie les flots de véhicules en fonction de leur origine . Formellement , on veut calculer une ***matrice de trafic*** $M \in \mathbf{R}(O \times E)$ où $M_{l,(i,j)}$ est le trafic issu de la zone $l \in O$ circulant sur le tronçon $(i,j) \in E$. Les équations de bilan sont les suivantes :

$$\sum_{(i,j)\in E} M_{l,(i,j)} - \sum_{(j,i)\in E} M_{l,(j,i)} = B_{l,i} \qquad \forall\, i \in V\,,\ \forall\, l \in O \quad (4.1)$$

De plus le trafic doit respecter le sens de circulation :

$$M_{l,(i,j)} \geq 0 \qquad \forall\, i \in V\,,\ \forall\, l \in O \quad (4.2)$$

Les équations (4.1) et (4.2) montrent que , pour toute origine $l \in O$, la ligne $M_{l.}$ est un flot selon la définition donnée dans la partie 1 . On a donc plusieurs flots non miscibles qui circulent dans le même réseau . Une telle matrice M est dit être un ***multiflot*** .

Le flot total $S \in \mathbf{R}(E)$ circulant sur les arcs est donné par

$$S_{(i,j)} = \sum_{l\in O} M_{l,(i,j)} \qquad \forall\, (i,j) \in E \quad (4.3)$$

Généralement , les flots composant le multiflot sont liés par :

- soit des contraintes de capacité sur S,
- soit des équations d'équilibre sur S,
- soit une fonction de coût dépendant non-linéairement de S.

Ce sont ces contraintes qui différencient un problème de multiflot pur de plusieurs problèmes de flot juxtaposés .

4.1.2. Les équations d'équilibre

L'hypothèse fondamentale du modèle présenté est que l'automobiliste choisit l'itinéraire de façon à minimiser son temps de parcours . L'état d'équilibre du trafic est donc atteint lorsqu'aucun automobiliste ne peut diminuer son temps de transport en changeant d'itinéraire . Une conséquence de cette régle de comportement de l'usager est que tous les itinéraires utilisés entre une origine et une destination sont équivalents et que les temps de parcours sur les itinéraires restants ne sont pas plus courts .

Le temps de parcours d'un tronçon de route croît avec l'augmentation du trafic sur celui-ci . La fonction $G_{(i,j)} : \mathbf{R}_+ \rightarrow \mathbf{R}_+$ exprime cette relation . Lorsque $S_{(i,j)}$ véhicules circulent sur (i,j) , ils mettent $G_{(i,j)}(S_{(i,j)})$ unités de temps pour se rendre de i à j .

A tout multiflot $M \in \mathbf{R}(O \times E)$, on associe une matrice des plus courts itinéraires $W \in \mathbf{R}(O \times V)$. La composante $W_{l,i}$ donne la durée minimum pour aller de la zone origine $l \in O$ au sommet $i \in V$; ces temps étant calculés en utilisant les fonctions $G_{(i,j)}$.

Mathématiquement , le multiflot M est un ***état d'équilibre*** si et seulement si :

$$G_{(i,j)}(S_{(i,j)}) + W_{l,i} - W_{l,j} = 0 \qquad \text{si } M_{l,(i,j)} > 0 \quad \forall\, l \in O ,\ \forall\, (i,j) \in E \qquad (4.4)$$

$$G_{(i,j)}(S_{(i,j)}) + W_{l,i} - W_{l,j} \geqslant 0 \qquad \text{si } M_{l,(i,j)} = 0 \quad \forall\, l \in O ,\ \forall\, (i,j) \in E \qquad (4.5)$$

Ces deux équations expriment que des voitures issues de l circulent sur (i,j) , uniquement si ce tronçon est situé sur un plus court chemin partant de l (les distances étant données par les $G_{(i,j)}(S_{(i,j)})$) .

4.1.3. Le problème de multiflot optimum

Comme précédemment , les équations d'équilibre peuvent être remplacées par une fonction objective . Le problème de multiflot à coût convexe minimum est le suivant :

$$\text{Minimiser}_{M_{l,(i,j)}} \quad \sum_{(i,j)\in E} F_{(i,j)}\Big(\sum_{l \in O} M_{l,(i,j)} \Big) \qquad (4.6)$$

s.c.

$$\sum_{(i,j)\in E} M_{l,(i,j)} - \sum_{(j,i)\in E} M_{l,(j,i)} = B_{l,i} \qquad \forall\, i \in V,\ \forall\, l \in O \qquad (4.7)$$

$$M_{l,(i,j)} \geq 0 \qquad \forall\, i \in V,\ \forall\, l \in O \qquad (4.8)$$

où

$F_{(i,j)} : \mathbf{R}_+ \rightarrow \mathbf{R}$ est une fonction convexe dont la dérivée est $G_{(i,j)}$.

Le théorème de Kuhn et Tucker permet de montrer l'équivalence entre le calcul d'un multiflot à coût minimum et la recherche d'un état d'équilibre .

4.2. L'ALGORITHME

Pour résoudre ce problème de multiflot , l'algorithme d'optimisation de flot décrit précédemment est utilisé itérativement pour diminuer la fonction objective en modifiant le trafic issu d'une origine fixée .

4.2.1. La modification optimale pour une origine

A un multiflot $M \in \mathbf{R}(O \times E)$ et une origine $l \in O$, on associe le problème de flot suivant :

$$\text{Minimiser} \quad \sum_{(i,j)\in E} Z_{(i,j)}(X_{(i,j)}) \qquad (4.9)$$
$$X_{(i,j)}$$
s.c.

$$\sum_{(i,j)\in E} X_{(i,j)} - \sum_{(j,i)\in E} X_{(j,i)} = 0 \qquad \forall\, i \in V \quad (4.10)$$

$$X_{(i,j)} \geq -M_{l,(i,j)} \qquad \forall\, (i,j) \in E \quad (4.11)$$

où

- $X_{(i,j)}$ est la modification de trafic sur le tronçon $(i,j) \in E$

- $Z_{(i,j)} : \mathbf{R} \to \mathbf{R}_+$ est une fonction convexe de la forme :

$$Z_{(i,j)}(X_{(i,j)}) = F_{(i,j)}\Big(\sum_{l\in O} M_{l,(i,j)} + X_{(i,j)}\Big) \qquad (4.12)$$

Le flot $X \in \mathbf{R}(E)$ représente la variation optimale du multiflot lorsqu'on se limite au trafic provenant de l'origine $l \in O$. Ce problème de flot à coût convexe sera appellé ***calcul de la modification optimale*** .

4.2.2. L'algorithme

Etant donnée une séquence de k ordres d'interpolation $\Delta_1 \geq ... \geq \Delta_k$, on peut utiliser l'algorithme 3.1. pour déterminer des modifications du multiflot . Comme précédemment , ces modifications seront calculées avec une précision croissante .

Algorithme 4.1. ALGORITHME D'OPTIMISATION D'UN MULTIFLOT

0. Initialisation

- Trouver un multiflot initial $M \in \mathbf{R}(O \times E)$
- Trouver un arbre initial $T \subset E$
- $p := 1$

1. Modifications du multiflot pour l'ordre Δ_p

- $\Delta := \Delta_p$
- Pour toutes les origines $l \in O$, effectuer les points **1.0** à **1.5**

1.0. Initialisation de la modification

- $X_{(i,j)} := 0 \qquad \forall\ (i,j) \in E$
- $e(i,j) := a \qquad \forall\ (i,j) \in T$
- Calculer les variables duales $W_i\ \forall\ i \in V$

1.1. à 1.4. Calcul de la modification $X \in \mathbf{R}(E)$ pour l'origine l

Application des pas **1.** à **4.** de l'algorithme 3.1.

1.5. Modification du multiflot

- $M_{l,(i,j)} := M_{l,(i,j)} + X_{(i,j)} \quad \forall\ (i,j) \in E$

2. si (p = k) alors fin de l'algorithme

sinon p := p+1 retour en **1.**

Une variante de cet algorithme consiste à considérer un arbre par origine . Bien que certainement plus rapide , cette variante a été écartée pour économiser la place mémoire .

Le problème initial étant de trouver un état d'équilibre , il est important d'étudier les équations (4.4) et (4.5) dans le cas du multiflot obtenu par l'algorithme . La propriété 4.1 propose une condition d'arrêt et montre que le multiflot ainsi calculé est proche de l'équilibre .

Propriété 4.1.

Soit une séquence d'ordres $\Delta_1 \geq \ldots \geq \Delta_{k-1} = \Delta_k = \Delta$ pour laquelle l'étape k ne fait pas décroitre la fonction objectif , alors l'algorithme 4.1 fournit un multiflot $M \in \mathbf{R}(O \times E)$ et des variables duales $W \in \mathbf{R}(O \times V)$ pour lesquels :

$\forall\, l \in O$, $\forall\, (i,j) \in E$, **il existe** $T_{l,(i,j)} \in [0,\Delta]$ **et** $U_{l,(i,j)} \in [-\Delta,0]$ **tels que :**

$$G_{(i,j)}(S_{(i,j)}) + W_{l,j} - W_{l,i} \geq -H_{(i,j)}(S_{(i,j)} + T_{l,(i,j)})\Delta/2$$

$$G_{(i,j)}(S_{(i,j)}) + W_{l,j} - W_{l,i} \leq H_{(i,j)}(S_{(i,j)} + U_{l,(i,j)})\Delta/2 \qquad \text{si } M_{l,(i,j)} \geq \Delta$$

où $H_{(i,j)}$ **est la dérivée de** $G(i,j)$

$$S_{(i,j)} = \sum_{l \in O} M_{l,(i,j)}$$

Démonstration :

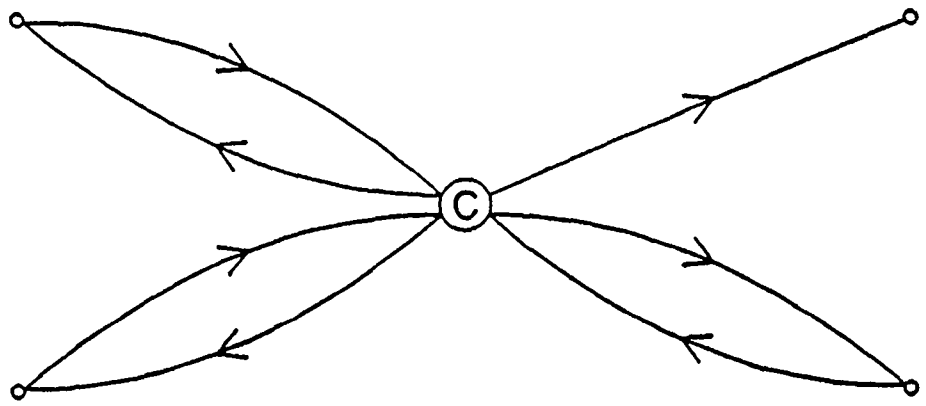
Le multiflot n'étant pas modifié durant l'étape k , l'algorithme a calculé des variations de flot nulles pour toutes les origines . Ainsi appliqué au problème de flot associé à une origine quelconque $l \in O$ (paragraphe 4.2.1) , l'algorithme du simplexe d'ordre Δ (algorithme 3.1) a trouvé un flot nul . La propriété 3.8 (page 116) permet alors d'établir les inégalités liées à l'origine l . Q.E.D.

De par la nature du problème , les bilans sont entiers ($B \in \mathbb{Z}(O \times V)$) et l'on recherche une solution entière . Un multiflot entier peut être obtenu par l'algorithme 4.1 en partant d'une solution initiale entière et en utilisant une séquence d'ordre entière $\Delta_1 \geq \ldots \geq \Delta_k = 1$. Si de plus on applique la condition d'arrêt de la propriété 4.1, le multiflot entier obtenu ne peut pas être amélioré en modifiant un seul des flots le composant . Ainsi , pour faire décroître la fonction objectif , il faudrait combiner des variations entières du trafic issu de plusieurs origines .

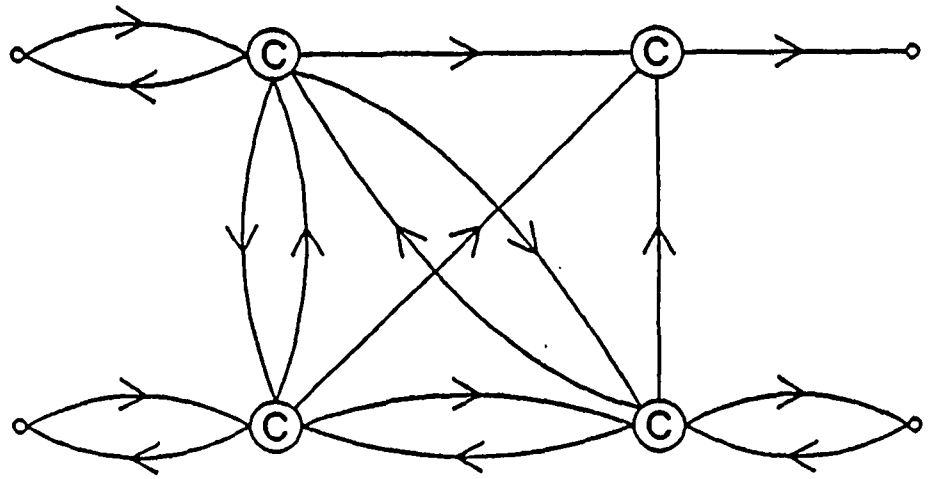
4.3. TESTS

4.3.1. Présentation du problème

Cette méthode de calcul de l'affectation du trafic a été appliquée à un cas réel . Le but principal de l'étude était l'analyse de la circulation dans les carrefours principaux de la région de Bellinzone . Pour pouvoir distinguer les différents flux de véhicules traversant un carrefour , il a été nécessaire d'introduire des arcs et des noeuds supplémentaires . La figure 4.1 illustre le procédé d'éclatement des carrefours .



carrefour C



carrefour C après
éclatement

figure 4.1.

Le temps de parcours d'un arc (i,j) ∈ E en fonction du trafic y est donné par :

$$G_{(i,j)}(y) = T_{(i,j)}(1 + a(y/C_{(i,j)})^b)$$

où

- $T_{(i,j)}$ temps de parcours de (i,j) à vide
- $C_{(i,j)}$ trafic maximum sur le tronçon (i,j).

Les paramètres a et b sont différents si (i,j) correspond à un tronçon réel ou à un flux traversant un carrefour . Ce type de fonction a été proposé par le Bureau of Public Road [5].

Le tableau 4.1 donne les caractéristiques principales du réseau modélisant la région de Bellinzone . La description détaillée de cette étude se trouve en [2]

nb. de noeuds	122
nb. d'origines	47
nb. de carrefours	11
nb. d'arcs	298
nb. de tronçons	100

tableau 4.1.

4.3.2. Analyse des résultats

Ce paragraphe consiste en une analyse comparative de deux programmes de calcul d'affectation :

- AFFLOW : code implantant l'algorithme 4.1. . Ce n'est qu'une adaptation de l'algorithme CFLOW1 présenté dans la partie 3.

- AFROUTE : code développé à l'ITEP ([2]) pour l'étude de la région de Bellinzone . L'algorithme utilisé est une adaptation de la méthode de Frank et Wolfe pour les réseaux . Une analyse détaillée de cet algorithme se trouve aux pages 54 à 66 de [4] .

Avant de comparer les temps de calcul , il est important d'analyser la qualité du résultat obtenu . La première différence entre les deux codes est le traitement de la contrainte d'intégralité des flots . Pour AFROUTE , tout le calcul se fait avec des flots réels ; l'intégralité est introduite artificiellement en arrondissant la solution obtenue . Cette façon de procéder a un grand inconvénient : la conservation du flot n'est plus garantie . Pour AFFLOW , l'intégralité du multiflot est inhérente à la nature de l'algorithme et par conséquent la conservation du flot est parfaitement vérifiée .

La seconde différence est la nature de la solution . AFFLOW résout un problème de multiflot ; pour chaque tronçon , il calcule non seulement le volume total de trafic mais également la répartition en fonction de la provenance des voitures . AFROUTE ne fournit que le trafic total . Il est alors évident que AFFLOW utilise plus de place mémoire que AFROUTE . Cependant , il faut souligner que :

- AFFLOW stocke le multiflot $M \in \mathbf{R}(O \times E)$, mais n'utilise la matrice des bilans que ligne par ligne pour calculer le multiflot initial .

- AFROUTE ne mémorise que le flot total , mais il doit conserver en mémoire la matrice des bilans $B \in \mathbf{R}(O \times V)$ durant tout le calcul .

Le tableau 4.2 présente les performances des deux codes pour le réseau de Bellinzone ; deux valeurs des paramètres ont été considérées :

-P1 a = 0.5 , b = 4.0 pour les tronçons et dans les carrefours . Ce sont les paramètres préconisés par le Bureau of Public Road ([5]) .

-P2 a = b = 0 pour les tronçons et a = 3.7 , b = 1.3 dans les carrefours . Ces paramètres , utilisés pour l'étude de Bellinzone , permettent d'analyser la congestion dans les carrefours .

Problème	Code	Temps calcul	Fonction objectif
P1*	AFFLOW	27.2	4373372.5
P1	AFFLOW	23.4	4373372.5
P1	AFROUTE	41.7	4373437.0
P2*	AFFLOW	35.1	4482851.5
P2	AFFLOW	21.3	4482860.5
P2	AFROUTE	36.6	4482862.0

tableau 4.2.

notes: - les temps de calcul sont donnés en secondes sur un VAX
- * désigne le multiflot entier obtenu par la condition d'arrêt de la propriété 4.1.

Ces deux codes peuvent être utilisés , moyennant de légères modifications , pour résoudre le modèle d'équilibre du trafic avec demande élastique ; c'est-à-dire lorsque le nombre de déplacements entre deux zones dépend du niveau de service du réseau . Plus précisément

$B_{l,i}$ = **fonction**($W_{l,i}$ - $W_{l,l}$) $\quad \forall\, l \in O\,,\ \forall\, i \in V$

BIBLIOGRAPHIE

[1] M.FLORIAN

Nonlinear cost network models in transportation analysis .

Mathematical Programming Study 26 (1986) p 167-196.

[2] R.FRIDEL

Algorithme d'affectation des circulations sur un réseau de transport individuel .

Travail de semestre ITEP-EPFL (1985) .

[3] D.GENTON

Système de transport .

Cours de l'ITEP-EPFL (1980) .

[4] S.NGUYEN

Une approche unifiée des méthodes d'équilibre pour l'affectation du trafic .

Thèse de doctorat (Université de Montréal , 1974) .

[5] U.S. DEPARTMENT OF COMMERCE , BUREAU OF PUBLIC ROADS.

Traffic assignment manuel .

Washington , D.C. (1964) .

Chapitre 5

PROBLEME DE SECURITE DES RESEAUX D'ENERGIE ELECTRIQUE

5.0. INTRODUCTION

Tous les pays industrialisés sont extrêmement dépendants de l'énergie électrique et toute interruption générale ou partielle du courant peut avoir des conséquences économiques très importantes . Les entreprises de production , de transport et de distribution d'électricité ont donc un besoin vital de méthodes aidant à la prise de décision en cas de défaillances de certains éléments de réseau . Les concepts fondamentaux du problème de sécurité dans les réseaux sont décrits en détail dans [11] . Toute analyse de sécurité comporte deux phases : la surveillance du bon fonctionnement du réseau et le choix des actions à entreprendre pour pallier certaines perturbations .

Une des méthodes d'évaluation du degré de sécurité d'un réseau est l'analyse de contingences ; connaissant l'état du réseau , on étudie systématiquement les conséquences du déclenchement de L lignes (en général on se limite à L = 1 ou 2) . Le niveau de sécurité peut alors se définir comme le nombre maximum de lignes défaillantes que peut supporter le réseau .

Ce chapitre présente quelques nouvelles méthodes pour l'analyse d'un réseau basées sur le concept de flot ; l'ensemble de ce travail a été effectué en collaboration avec le Laboratoire des Réseaux Electriques dans le cadre d'un projet du fonds national suisse de la recherche scientifique ([3]) .

Avant de traiter le problème de sécurité , il a été nécessaire de modéliser par les flots la répartition des puissances dans un réseau . Cette étude a été publiée dans la revue IEEE/PES sous le titre "Network simplex method applied to AC Load-flow calculation" ([1]) . La modélisation des divers éléments constituant un réseau électrique a été développée par J.Barras et S.Alec et elle est présentée dans l'article . Les paragraphes 5.1 et 5.2 ne traitent que le cas le plus simple et mettent l'accent sur le problème de biflot sous-jacent et sa résolution par les algorithmes du chapitre 3 . De cette étude découle un nouveau concept de calcul local qui consiste en une méthode d'analyse située à mi-chemin entre l'approche topologique pure et un calcul complet des puissances .

L'approche topologique consiste à étudier la structure du réseau et à utiliser les résultats comme informations complémentaires pour l'analyse de sécurité . Le problème de base est de savoir reconnaître les ensembles de lignes dont le déclenchement simultané met en péril la connexité du réseau . Le paragraphe 5.3 présente un algorithme pour l'énumération des ensembles de une ou deux lignes dangereuses . N.Chahal a adapté et utilisé itérativement cet algorithme pour déterminer également les ensembles de trois lignes . L'implantation efficace de cet algorithme utilise essentiellement les procédures de manipulation d'arbres décrites dans le paragraphe 1.3 .

Les concepts de calcul local et d'ensemble de lignes déconnectant ne suffisent pas à résoudre le problème vaste et complexe de l'analyse de sécurité . Mais ce pourront être des outils utiles car il sont basés essentiellement sur la structure de réseau .

5.1. LE CALCUL DU COURANT DANS UN RESEAU RESISTIF

Ce paragraphe décrit la modélisation de la répartition du courant par un problème de flot à coûts quadratiques . Cette méthode de résolution est esquissée dans [8] , mais l'interprétation des variables duales en terme de tension n'y est pas mentionnée ni utilisée pour traiter les sources de tension . Le propos n'est pas de fournir une nouvelle méthode de calcul mais de présenter le cas simple du courant continu avant de traiter le cas du courant alternatif .

5.1.1. La formulation électrique

Le réseau résistif est caractérisé par sa topologie décrite par un ensemble de noeuds reliés par des branches et les résistances des branches .

L'injection et le retrait de courant en certains noeuds va induire un courant sur l'ensemble des branches du réseau . La répartition du courant obéit aux deux lois de Kirchoff :

- la somme algébrique des courants circulant dans l'ensemble des branches incidentes au même noeud est nulle ,

- la somme algébrique des tensions aux bornes des branches constituant une maille (terminologie électrique pour désigner un cycle) est nulle .

Pour une branche de résistance R , la tension ΔU aux bornes et le courant I la traversant sont liés par la loi d'Ohm :

$$\Delta U = RI$$

Il est aisé de généraliser le modèle aux cas de résistances non-linéaires ; où la tension est une fonction continue et strictement croissante du courant .

5.1.2. Le modèle de flot

Le réseau électrique est transformé en un réseau R = (V,E) en orientant de manière arbitraire les branches . Cette orientation correspond à une convention de signe pour le courant ; la branche reliant les noeuds p et q orientée en un arc (p,q) ∈ E équivaut à la définition suivante du courant $I_{(p,q)}$:

- $I_{(p,q)} > 0$ signifie que $I_{(p,q)}$ unités de courant transitent sur la branche de p vers q .
- $I_{(p,q)} < 0$ signifie que $-I_{(p,q)}$ unités de courant transitent sur la branche de q vers p .

En chaque noeud , l'injection ou le retrait de courant est fixé ; ces données sont fournies sous la forme d'un vecteur de bilan $B \in \mathbb{R}(V)$. Ce type de noeud est appelé *source de courant* .

La première loi de Kirchoff est modélisée par les équations de bilan :

$$\sum_{(p,q)\in E} I_{(p,q)} - \sum_{(q,p)\in E} I_{(q,p)} = B_p \qquad \forall\ p \in V \quad (5.1)$$

La seconde loi de Kirchoff équivaut à déterminer des tensions nodales $U \in \mathbb{R}(V)$ telles que la tension $\Delta U_{(p,q)}$ aux bornes d'une branche (p,q) soit :

$$\Delta U_{(p,q)} = U_p - U_q \qquad \forall\ (p,q) \in E$$

Pour des résistances $R \in \mathbb{R}(E)$, ces équations se récrivent en utilisant la loi d'Ohm :

$$U_p - U_q = R_{(p,q)} I_{(p,q)} \qquad \forall\ (p,q) \in E \quad (5.2)$$

Ce sont des équations d'équilibre selon la définition du paragraphe 3.6.1. Le calcul de la répartition du courant se reformule en un problème d'optimisation quadratique .

$$\text{Minimiser}_{I_{(p,q)}} \quad \sum_{(p,q)\in E} R_{(p,q)} I_{(p,q)}^2 \qquad (5.3)$$

sc

$$\sum_{(p,q)\in E} I_{(p,q)} - \sum_{(q,p)\in E} I_{(q,p)} = B_p \qquad \forall\ p \in V \quad (5.4)$$

La fonction coût sur une branche s'interprète électriquement comme la perte par effet Joule . La seconde loi de Kirchhoff modélise la tendance de la nature à minimiser l'énergie des systèmes .

5.1.3. Les contraintes de tension

Le réseau peut également contenir des ***sources de tension*** ; ce sont des noeuds où la tension est fixée et le bilan du courant variable . De manière plus générale , il est possible de modéliser des ***consignes de tension*** ; la tension en certains noeuds doit être comprise entre deux bornes . Ces consignes sont respectées en modifiant les injections et les retraits de courant .

Le sommet r étant pris comme sommet de référence ($U_r = 0$) , une consigne de tension au noeud p correspond à la contraintes (5.5) sur la tension (variable duale) U_p .

$$\alpha \leqslant U_p - U_r \leqslant \beta \qquad (5.5)$$

Cette contrainte est introduite sous la forme d'un arc (p,r) avec une fonction coût $\mathbf{F}_{(p,r)}$ définie comme suit :

$$\mathbf{F}_{(p,r)}(I_{(p,r)}) = \begin{cases} -\alpha I_{(p,r)} & \text{si } I_{(p,r)} \leqslant 0 \\ \beta I_{(p,r)} & \text{si } I_{(p,r)} \geqslant 0 \end{cases} \qquad (5.6)$$

La fonction $\mathbf{F}_{(p,r)}$ est linéaire par morceaux ; les conditions d'optimalité présentée dans le paragraphe 2.2 montrent qu'à l'optimum les variables duales vérifient la contrainte (5.5) . En effet les inégatilés (2.12) et (2.13) s'écrivent dans ce cas :

$$\mathbf{ra}(p,r) = U_r - U_p + \beta \geqslant 0$$
$$\mathbf{rd}(p,r) = U_p - U_r + (-\alpha) \geqslant 0$$

De plus , le courant $I_{(p,r)}$ donne les deux modifications d'injection ou de retrait à effectuer :

- si $I_{(p,r)} < 0$, on doit retirer $-I_{(p,r)}$ unités en p et les ajouter en r ; dans ce cas , la tension est maximale $U_p - U_r = \beta$

- si $I_{(p,r)} > 0$, on doit ajouter $I_{(p,r)}$ unités en r ; dans ce cas , la tension est minimale $U_p - U_r = \alpha$.

5.1.4. Le calcul local

Le but de ce paragraphe est de montrer comment l'algorithme 3.1 peut être utilisé pour étudier le déclenchement d'une branche . Le calcul local consiste à déterminer les modifications les plus importantes de courant provoquées par le retrait d'une branche .

Dans le cas du courant continu , la différence entre la répartition de courant $I \in \mathbf{R}(E)$ dans le réseau R = (V,E) et la répartition $I' \in \mathbf{R}(E-(s,t))$ après le déclenchement de l'arc (s,t) est encore une répartition . Pour le vérifier considérons le ***problème de déclenchement*** de (s,t) défini comme un problème d'équilibre sur l'ensemble des arcs E' = E - (s,t) . Ce problème se formule comme suit :

Déterminer $X \in \mathbb{R}(E')$ et $W \in \mathbb{R}(V)$ satisfaisant :

$$\sum_{(p,q)\in E'} X_{(p,q)} - \sum_{(q,p)\in E'} X_{(q,p)} = \begin{cases} 1 & p = s \\ -1 & p = t \\ 0 & \forall\ p \in V - s - t \end{cases} \qquad (5.7)$$

$$W_p - W_q = R_{(p,q)} X_{(p,q)} \qquad \forall\ (p,q) \in E' \qquad (5.8)$$

Par combinaison linéaire des équations (5.1) , (5.2) , (5.7) et (5.8) , on vérifie aisément que la répartition après déclenchement s'exprime comme :

$$I'_{(p,q)} = I_{(p,q)} + I_{(s,t)} X_{(p,q)} \qquad \forall\ (p,q) \in E' \quad (5.9)$$

Cette propriété est connue sous le nom de principe de superposition .

L'algorithme 3.1 permet de calculer une solution au problème de déclenchement aprés transformation en un problème d'optimisation quadratique . Pour le calcul local le flot initial est nul , l'ordre est $\Delta = 0.25$; l'algorithme trouve un flot $X \in \mathbb{R}(E')$ tel que

$$X_{(p,q)} \in \{ -1 , -0.75 , -0.5 , -0.25 , 0 , 0.25 , 0.5 , 0.75 , 1 \} \qquad \forall\ (p,q) \in E'$$

Ce flot fournit certaines informations utiles pour l'analyse de sécurité , comme par exemple :

- $\{ (p,q) \in E' \mid X_{(p,q)} \neq 0 \}$ détermine le sous-réseau le plus perturbé par le déclenchement de (s,t) . Les modifications de courant y seront estimées supérieures à 25% du courant $I_{(s,t)}$.

- $\{ (p,q) \in E' \mid X_{(p,q)} = 1 \}$ est l'ensemble des branches qui , déclenchées simultanément à (s,t) , risquent :
  - soit de couper le réseau en deux ,
  - soit de provoquer d'importantes modifications de courant sur des chemins de grande résistivité .

5.2.LE CALCUL DE LA REPARTITION DES PUISSANCES

La répartition des puissances actives et réactives dans un réseau de haute tension est un problème qui peut s'énoncer ainsi : ([3])

> Déterminer les tensions aux noeuds d'un réseau électrique ainsi que les transits de puissance active et réactive dans les branches (lignes et transformateurs) , connaissant la topologie du réseau , les caractéristiques des branches (impédances longitudinales et admittances transversales) de même que les productions ou demandes actives et réactives à chaque jeu de barres .

Ce paragraphe présente deux modèles classiques de la répartition des puissances et propose des algorithmes de résolution utilisant les flots .

5.2.1. Le modèle actif "DC-flow"

Le modèle du "DC-flow" ([1],[3]) ne se préoccupe que de la répartition des puissances actives ; il est basé sur des hypothèses simplificatrices permettant d'obtenir un système d'équations d'équilibre du type de ceux décrits dans le paragraphe 3.6 . Le problème est de calculer :

- $P \in \mathbb{R}(E)$: le vecteur des puissances actives ; $P_{(p,q)}$ représentant la puissance transitant de p vers q ,

- $O \in \mathbb{R}(V)$: le vecteur des phases .

L'équation du transit de puissance active s'écrit :

$$P_{(p,q)} = (O_p - O_q) / X_{(p,q)} \qquad \forall\ (p,q) \in E \quad (5.10)$$

où $X \in \mathbb{R}(E)$ est le vecteur des réactances longitudinales des branches .

Les hypothèses impliquant la conservation de la puissance active en chaque noeud , le calcul du "DC-flow" est tout à fait similaire à celui de la répartition de courant présentée dans la paragraphe 5.1.

5.2.2. Le modèle actif-réactif "AC load-flow"

S'il est possible de linéariser au prix de quelques hypothèses réalistes les équations de transit de la puissance active , le transit de la puissance réactive ne peut pas être approché par une fonction linéaire ([3]) . De plus la puissance réactive est , en général , différente aux deux extrémités d'une ligne et par conséquent elle ne peut pas être traitée comme un flot . Ce sont les raisons pour lesquelles , il faut utiliser un modèle de courants réels et imaginaires induits par les puissances actives et réactives produites ou consommées aux noeuds .

Dans la suite , un nombre complexe est la donnée d'un vecteur à deux composantes $Z \in \mathbb{R}(\{Re,Im\})$ et il peut s'écrire également $Z_{Re} + jZ_{Im}$ où j est la racine de -1 ($j^2 = -1$) .

Les courants complexes $I \in \mathbb{R}(E\times\{Re,Im\})$ et les tensions complexes $U \in \mathbb{R}(V\times\{Re,Im\})$ vérifient les deux lois de Kirchhoff :

- la conservation du courant réel et imaginaire :

$$\sum_{(p,q)\in E} I_{(p,q),Re} - \sum_{(q,p)\in E} I_{(q,p),Re} = B_{p,Re} \qquad \forall\, p \in V \quad (5.11)$$

$$\sum_{(p,q)\in E} I_{(p,q),Im} - \sum_{(q,p)\in E} I_{(q,p),Im} = B_{p,Im} \qquad \forall\, p \in V \quad (5.12)$$

où $B \in \mathbb{R}(V\times\{Re,Im\})$ est le vecteur des injections ou retraits de courant complexe .

- par hypothèse simplificatrice , seule la partie active $X_{(p,q)}$ de l'impédance longitudinale d'une ligne (p,q) est prise en compte ; la différence de tension aux bornes de cette ligne vérifie :

$$U_{p,Im} - U_{q,Im} = X_{(p,q)} I_{(p,q),Re} \qquad \forall\ (p,q) \in E \quad (5.13)$$
$$U_{p,Re} - U_{q,Re} = X_{(p,q)} I_{(p,q),Im} \qquad \forall\ (p,q) \in E \quad (5.14)$$

La forme des équations (5.11) à (5.12) permet de modéliser le courant complexe comme un multiflot qui est formé d'une composante réelle et d'une composante imaginaire et qui satisfait les équations d'équilibre (5.13) et (5.14) .

Dans un problème de calcul de répartition des puissances , seules les demandes et productions de puissance active $P \in \mathbb{R}(V)$ et réactive $Q \in \mathbb{R}(V)$ sont connues . Les équations (5.15) et (5.16) expriment les bilans de courant réel et imaginaire en fonction des bilans de puissance active et réactive .

$$B_{p,Re} = (P_p U_{p,Re} + Q_p U_{p,Im}) / ((U_{p,Re})^2 + (U_{p,Im})^2) \qquad \forall\ p \in V \quad (5.15)$$
$$B_{p,Im} = (P_p U_{p,Im} + Q_p U_{p,Re}) / ((U_{p,Re})^2 + (U_{p,Im})^2) \qquad \forall\ p \in V \quad (5.16)$$

5.2.3. L'algorithme

Pour des bilans de courant réel et imaginaire fixés , le problème de multiflot se scinde en deux problème de flot indépendant :

- le ***problème réel*** défini par les équations (5.11) et (5.13) . L'algorithme 3.2 peut être utilisé pour calculer les courants réels $I_{(p,q),Re}$ sur les branches $(p,q) \in E$ et les tensions imaginaires $U_{p,Im}$ aux noeuds $p \in V$,

- le ***problème imaginaire*** défini par les équations (5.12) et (5.14) . L'algorithme 3.2 peut être utilisé pour calculer les courants imaginaires $I_{(p,q),Im}$ sur les branches $(p,q) \in E$ et les tensions réelles $U_{p,Re}$ aux noeuds $p \in V$.

Le principe du découplage actif-réactif suggéré dans [5] a été adapté par J.Barras pour prendre en compte l'interdépendance des problèmes réel et imaginaire ([1]) . Le calcul découplé du courant réel et imaginaire est utilisé itérativement pour déterminer des injections et retraits de courant satisfaisant aux équations (5.15) et (5.16) ; ceci constitue l'algorithme 5.1 .

Algorithme 5.1. ALGORITHME DE CALCUL DE LOAD-FLOW

0. Initialisation

$U_{p,Re}$ = tension de référence

$U_{p,Im} = 0$

1. Résolution du problème réel

1.1. Calcul des bilans réels

$$B_{p,Re} = (P_p U_{p,Re} + Q_p U_{p,Im}) / ((U_{p,Re})^2 + (U_{p,Im})^2) \qquad \forall\, p \in V$$

1.2. Calcul des courants réels et des tensions imaginaires

$I_{(p,q),Re}$, $\forall\, (p,q) \in E$ et $U_{p,Im}$, $\forall\, p \in V$ sont obtenus par application de l'algorithme 3.2.

2. Résolution du problème imaginaire

2.1. Calcul des bilans imaginaires

$$B_{p,Im} = (P_p U_{p,Im} + Q_p U_{p,Re}) / ((U_{p,Re})^2 + (U_{p,Im})^2) \qquad \forall\, p \in V$$

2.2. Calcul des courants imaginaires et des tensions réelles

$I_{(p,q),Im}$ $\forall\, (p,q) \in E$ et $U_{p,Re}$ $\forall\, p \in V$ sont obtenus par application de l'algorithme 3.2.

3. Si la solution converge alors STOP

sinon retour en **1.**

Remarques :

- D'une itération à la suivante , les variations des bilans de courant réel et imaginaire sont petites en valeur relative . Il est donc avantageux de calculer une solution de départ par l'algorithme 1.9 de postoptimisation décrit en 1.5.2 .

- Pour l'application de l'algorithme 3.2 , l'ordre final et l'ordre initial vont être modifiés d'une itération à l'autre . L'ordre initial est choisi égal au quart de la variation maximale du bilan et l'ordre final est calculé comme l'ordre initial divisé par 64 .

Des tests systématiques de cet algorithme sont présentés et analysés dans [1] et [3] . Les conclusions de cette étude sont essentiellement les suivantes :

- l'algorithme converge rapidement ; en général, le nombre d'itérations n'excède pas 10.

- la précision des tensions et des puissances obtenues est satisfaisante

- l'algorithme n'est pas compétitif du point de vue de la vitesse de calcul . Les temps de calcul de cette méthode sont supérieurs de plusieurs ordres de grandeur à ceux de la méthode de Newton-Raphson ([13]) .

Bien que non compétitif pour le calcul du cas de base d'un load-flow , ce modèle a le grand avantage de conserver la structure de réseau et pourra , de ce fait , plus facilement être utilisé en combinaison avec des méthodes de théories des graphes pour aborder le problème de sécurité des réseaux .

5.3. L'ENUMERATION DES 2-COUPES

Ce chapitre traite des réseaux de distribution d'énergie ; mais il existe un grand nombre de réseaux dont les liaisons sont soumises à des défaillances et dont le niveau de fonctionnement dépend de la connexité . Les exemples les plus courants sont les réseaux de télécommunication , de transport et de distribution d'eau .

De par son vaste champ d'application , ce problème a suscité de nombreuses études . Certains auteurs utilisent une approche probabiliste pour calculer le rendement moyen à partir des probabilités de rupture des arcs ([3],[5]) ; d'autres déterminent l'arc le plus vital , c'est-à-dire celui dont le déclenchement provoque la plus forte augmentation des coûts ou la plus forte diminution du flot maximum entre producteurs et consommateurs ([6],[8],[14],[15]) .

Le problème traité dans ce paragraphe est la détermination des arcs les plus importants du point de vue de la connexité . L'algorithme présenté énumère tous les ensembles de un ou deux arcs qui déconnectent le réseau ; il est généralisable aux ensembles de trois arcs . Ces ensembles permettent de mieux connaitre la structure du réseau et ses points faibles . Dans [11] , une étude analogue est présentée ; mais elle traite du problème de flot maximum et ne prend en compte que les coupes séparant une source et un puits donnés . La limitation à deux ou trois arcs n'est pas restrictive ; en effet il est rare , en pratique , d'avoir un déclenchement simultané de quatre arcs ou plus .

5.3.1. Le problème

Le problème est d'énumérer toutes les 1-coupes et les 2-coupes d'un réseau connexe R = (V,E) ; ces notions sont définies de la manière suivante :

- une *1-coupe* (ou *isthme*) est un arc $e \in E$ tel que le retrait de cet arc déconnecte ce réseau ; c'est-à-dire que (V,E - e) n'est pas connexe ,

- une *2-coupe* est une paire d'arcs $e,f \in E$ tels que ni e ni f ne sont des 1-coupes et que $(V,E - e - f)$ n'est pas connexe .

Un cycle ***simple*** est un cycle dont aucun arc n'est parcouru plusieurs fois . La propriété 5.1 , qui caractérise les 2-coupes en utilisant de tels cycles , est à la base de l'algorithme .

Propriété 5.1.

La paire d'arcs e,f **forment une 2-coupe si et seulement si tout cycle simple contenant** e **contient aussi** f .

Démonstration :
$\Rightarrow$: Supposons , par l'absurde , qu'il existe un cycle simple C contenant e et pas f . Par hypothèse , $(V,E - f)$ est connexe ; comme tout chemin passant par e peut être reconnecté par le chemin $C - e$, alors $(V,E - f - e)$ est également connexe . Ceci contredit le fait que e et f forment une 2-coupe.

$\Leftarrow$: Supposons , par l'absurde , que $(V,E - f - e)$ est connexe . Alors il existe un chemin simple $C \subset E - f - e$ liant les deux extrêmités de l'arc e . Le cycle simple $C + e$ ne contient pas f ce qui contredit l'hypothèse.

Q.E.D.

Propriété 5.2.

Si d'une part e **et** f **forment une 2-coupe et que d'autre part** f **et** g **forment une 2-coupe alors** e **et** g **forment aussi une 2-coupe** .

Démonstration :
Par la propriété 5.1 , tout cycle simple contenant e contient f et tout cycle contenant f contient g . Ainsi tout cycle contenant e contient g et par conséquent e et g forment une 2-coupe.

Q.E.D

Cette propriété de transitivité , mise en évidence par N.Chahal , permet de définir des classes de 2-coupes ; une *classe de 2-coupes* est un ensemble d'arcs qui , pris deux à deux , forment des 2-coupes .

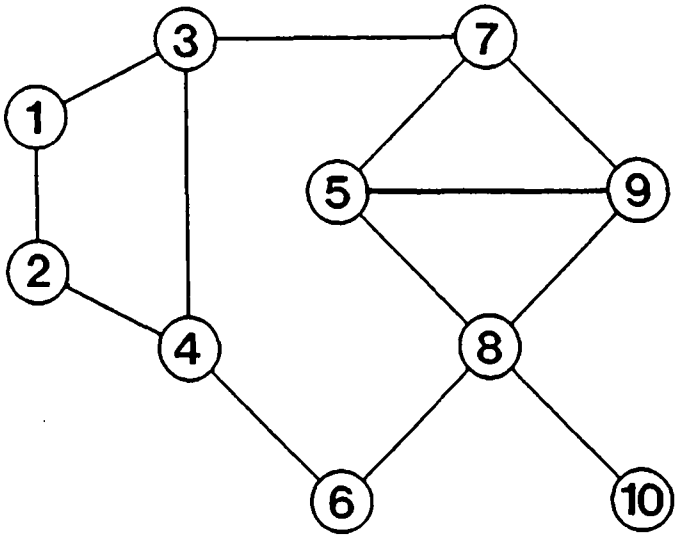


Figure 5.1.

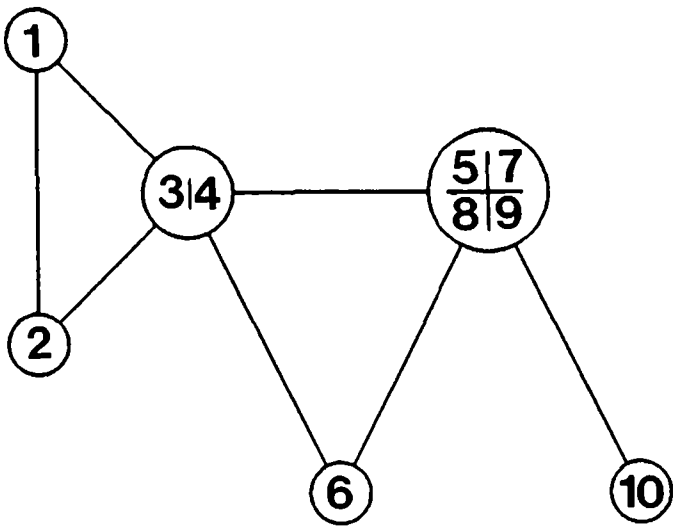


Figure 5.2.

La connaissance de ces classes permet d'analyser la structure du réseau . Une manière simple de représenter l'information données par les 2-coupes est de contracter les arcs n'appartenant ni à une 1-coupe ni à une 2-coupe . La contraction d'un arc est l'opération qui consiste à enlever l'arc et à identifier ses deux extrêmités . La figure 5.2 illustre cette opération sur le réseau de la figure 5.1.

5.3.2. L'algorithme

L'algorithme 5.2 détermine les 1-coupes et les classes de 2-coupes d'un réseau $R = (V,E)$. Tout comme le simplexe pour les réseaux , cet algorithme emploie la notion d'arbre et traite les cycles associés aux arcs hors de l'arbre . L'ensemble M désigne les arcs qu'il reste à analyser .

Algorithme 5.2. ALGORITHME D'ENUMERATION DES 1 et 2-COUPES

0. Initialisation

$M := E$

1. Calcul de l'arbre T

construire un arbre T qui minimise $|M \cap T|$

2. Calcul des coupes associées à T

Calculer $W(e) = \{ f \notin T \mid$ le cycle unique de $T+f$ contient $e \}$ $\forall$ $e \in T$

Pour tout arc $e \in T \cap M$ faire

si $W(e) = \emptyset$ alors e est une 1-coupe , $M := M - e$

si $W(e) = \{f\}$ alors e est dans la classe des 2-coupes de f , $M := M - e$

3. Test d'arrêt

$M := M \cap T$

si $M = \emptyset$ alors le réseau est analysé

sinon retour à **1.**

Le meilleur nouvel arbre est celui qui minimise $|M \cap T|$; mais tout arbre tel que $M \not\subset T$ serait également possible . Un tel arbre peut être obtenu par l'algorithme glouton .

Propriété 5.3.

L'algorithme 5.2 énumère toutes les 1-coupes et toutes les 2-coupes.

Démonstration :

Lors du calcul des coupes associées à l'arbre initial , toutes les 1-coupes sont trouvées . En effet , un arc e formant une 1-coupe ne peut pas appartenir à un cycle simple et , par conséquent , il est élément de l'arbre et l'ensemble associé $W(e)$ est vide .

Pour tout $f \notin T$, le calcul des 2-coupes associées à T trouve la classe des 2-coupes de f si elle existe . La propriété 5.1 implique que seuls les arcs du cycle unique de $T + f$ peuvent former une 2-coupe avec f . Soit e un arc de ce cycle qui n'est pas une 1-coupe et $W(e)$ l'ensemble associé , on a deux cas :

- soit il existe $f' \in W(e)$ $f' \neq f$ et alors e et f ne forment pas une 2-coupe car e fait partie d'un cycle ne contenant pas f ; c'est le cycle unique de $T + f'$,
- soit f est le seul arc de $W(e)$ et alors e et f forment une 2-coupe car f est le seul arc qui peut remplacer e dans T .

Au début de l'algorithme , M est égal à E . En cours d'algorithme , un arc e n'est retiré de M que dans un des trois cas suivant :

- si e est une 1-coupe ,
- si $e \in T$ et la classe des 2-coupes contenant e a été trouvée
- si $e \notin T$ et il a été démontré que la classe des 2-coupes associée à e est connue si elle existe .

A la fin de l'algorithme , M est vide et toutes les 1 et 2-coupes ont donc été trouvées .

Q.E.D.

5.3.3. L'implantation

Tout comme le simplexe dans les réseaux , l'algorithme 5.2 manipule des arbres . L'arbre T utilisé dans cet algorithme est mémorisé à l'aide de la structure de donnée décrite en 1.3 et ce paragraphe montre comment utiliser cette représentation pour implanter les deux opérations de base que sont :

- le calcul des ensembles W(e) associés aux arcs de l'arbre ,
- le calcul de l'arbre de l'étape suivante .

Calcul de W(e)

En fait , la connaissance entière de W(e) n'est pas requise ; il suffit de pouvoir tester si W(e) = 0 , si W(e) = {f} ou si |W(e)| > 1 . Il est possible de représenter les ensembles W(e) , ∀ e ∈ E , par un tableau CYCLE(.) de dimension m en adoptant la convention suivante :

$$\text{CYCLE}(e) = \begin{cases} 0 & \text{si } W(e) = 0 \\ f & \text{si } W(e) = \{f\} \\ -1 & \text{si } |W(e)| > 1 \end{cases}$$

L'algorithme 5.3 permet de calculer le tableau CYCLE(.) . La détermination du cycle associé à un arc hors de l'arbre T peut être implantée par la procédure 1.1 .

Algorithme 5.3 : CALCUL DES ENSEMBLES W(e)

0. Initialisation

CYCLE(e) := 0 ∀ e ∈ E

1. Parcourt des cycles

Pour tous les arcs f ∉ T faire

pour tous les arcs e du cycle unique de T + f faire

si CYCLE(e) > 0 alors CYCLE(e) := -1

si CYCLE(e) = 0 alors CYCLE(e) := f

Calcul d'un arbre optimal

A chaque étape , l'arbre T dont l'analyse apporte le plus d'information est celui qui minimise $|T \cap M|$. Ceci est un cas très particulier du problème d'arbre maximum de poids minimum ([9]) . L'algorithme 5.4 présente une implantation spécifique de l'algorithme glouton par pivotage .

Algorithme 5.4 : CACUL DE L'ARBRE OPTIMAL

0. Initialisation

Trouver un arbre initial T

1. Choix de l'arc entrant f et de l'arc sortant e

Trouver $f \notin T$ et $e \in T$ tel que :

- $f \notin M$
- $e \in C \cap M$ où C est le cycle unique de $T + f$

Si de tels arcs n'existent pas alors l'arbre est optimal

2. Mise à jour de l'arbre (pivotage)

$T := T + f - e$

retour à **1.**

La procédure 1.1 peut être adaptée pour trouver , s'il existe , un arc e dans M sur le cycle associé à l'arc f hors de l'arbre . Pour la mise à jour de l'arbre , la procédure 1.2 peut être utilisée telle quelle .

Dans une première phase, l' algorithme 5.4 est utilisé pour calculer un arbre initial . Comme décrit dans 1.2.3 un arbre trivial est obtenu par l'introduction d'arcs fictifs , puis l'algorithme est appliqué en prenant M égal à l'ensemble des arcs fictifs . Dans la seconde phase , cet algorithme construit , en partant d'un arbre pour lequel les 2-coupes ont déjà été calculées , l'arbre devant être analysé à l'étape suivante .

Complexité de l'algorithme

L'étude de complexité essaie d'évaluer théoriquement le comportement du temps calcul en fonction des dimensions du problème qui sont le nombre n de noeuds et le nombre m d'arcs .

Pour un arbre T , le calcul des ensembles W(e) nécessite le parcours de m - n + 1 cycles simples comprenant au plus n arcs . Dans le pire des cas n arbres sont analysés et le calcul des W(e) se comporte comme mn^2 .

Pour le calcul de l'arbre initial , les n - 1 arcs *fictifs* doivent être remplacés par des arcs du réseau . Or le choix d'un couple d'arc entrant et sortant peut nécessiter au plus m parcours de cycle ; ainsi la construction de l'arbre initial se comporte comme mn^2 .

Après analyse de l'arbre initial , tous les arcs hors de cet arbre sont enlevés de M et ainsi |M| = n - 1 . Les itérations suivantes ont pour but de sortir ces n - 1 arcs de l'arbre . Pour vider M l'algorithme doit donc choisir , généralement en plusieurs itérations , n - 1 couples d'arcs entrant et sortant . Cette partie de l'algorithme se comporte comme mn^2 .

Les trois remarques précédentes montrent que la complexité de l'algorithme 5.2 avec l'implantation proposée est mn^2 .

5.3.4. Les tests

Cet algorithme a été programmé puis testé d'une part sur les réseaux électriques suisse et français et d'autre part sur les réseaux aléatoires F1...F20 du paragraphe 1.4 .

L'analyse des résultats présentés dans le tableaux 5.2 suggère les remarques suivantes :

- le nombre d'arbres pour lesquels les 2-coupes sont calculées est petit ,

- la comparaison des temps de calcul et des nombres de pivots donnés dans les tableaux 5.2 et 1.3 montre que le problème d'énumération des 1 et 2-coupes est du même ordre de difficulté que celui du calcul d'un flot à coût linéaire minimal ,

- les deux réseaux réels étant moins denses que les réseaux aléatoires , le nombre des 1 et 2-coupes est plus grand.

Nom	Nombre de 1-coupes	Nombre de classes	Temps calcul	Nombre d'arbres	Nombre de pivots
CH	34	31	0.04	3	193
EDF	126	122	0.3	4	624
F1	20	5	0.5	3	773
F3	4	6	0.8	3	788
F4	3	1	0.5	2	794
F5	2	3	0.6	3	793
F6	10	12	2.6	3	1976
F7	10	9	2.6	3	1979
F8	35	26	4.7	3	2936
F9	127	100	7.2	3	3771
F10	89	110	6.5	3	3792
F11	61	30	8.7	3	3907
F12	70	95	15.7	3	5830
F13	65	85	15.2	3	5849
F14	145	59	18.3	3	5794
F15	26	25	17.4	3	5947
F16	564	318	25.8	3	7087
F17	159	163	26.1	3	7673
F18	26	70	27.1	3	7901
F20	140	228	32.2	3	8624

Tableau 5.2.

BIBLIOGRAPHIE

[1] J.BARRAS , S.ALEC , C.PASCHE , P-A.CHAMOREL , A.GERMOND , D.DE WERRA
Network simplex method applied to AC load-flow calculation
IEEE Transactions on Power Systems Vol PWRS-2 1 (1987) p 197 - 203

[2] J.BARRAS , C.PASCHE , A.GERMOND
Network simplex method applied to AC load-flow calculation and to security analysis
Papport interne LRE-EPFL (1986)

[3] J.BARRAS , S.ALEC
Application de méthodes de recherche opérationnelle aux problèmes de la sécurité des réseaux d'énergie électrique
Fonds national suisse de la recherche scientifique projet No 2.718-0.82

[4] M.CAREY , C.HENDRICKSON
Bounds on expected performance of networks with links subject to failure
Networks 14 (1984) p439-456

[5] P-A.CHAMOREL
Optimisation des puissances actives et réactives par la programmation linéaire dans les réseaux électriques à haute tension
Thèse No 496 , EPFL , 1983

[6] P.DOULLIEZ , E.JAMOULLE
Transportation networks with random arc capacities
RAIRO 3 (1972) p 45-60

[7] P.DOULLIEZ , M.R.RAO
Maximal flow in a multi terminal network with anyone arc subject to failure
Management science 18 (1971) p 40-58

[8] E.HOBSON , D.L.FLETCHER , W.O.STADLIN
Network flow linear programming techniques and their application to fuel scheduling and contengency analysis
IEEE/PAS-103 7 (1984) p 1684-1691

[9] E.L.LAWLER
Combinatorial optimization : networks and matroids
Holt , Rinehart and Winston (New-York 1976)

[10] S.H.LUBORE , H.D.RATLIFF , G.T.SICILIA
Determining lhe most vital link in a flow network
Naval research logistic quaterly 18 (1971) p 497-502

[11] J-C.PICARD , M.QUEYRANNE
On the structure of all minimum cuts in a network and applications
Mathematical programming study 13 (1980) p 8-16

[12] C.ROSSIER
Analyse de la sécurité d'exploitation d'un réseau électrique
Bulletin ASE/UCS 1 (1981) p 2-7

[13] G.W.STAGG , A.H.EL-ABIAD
Computer methods in power system analysis
Mc-Graw-Hill (New-York , 1968)

[14] R.D.WOLLMER
Removing arcs from a network
Operations research 12 (1964) p 934-940

[15] R.D.WOLLMER
Algorithms for targeting strikes in a lines-of-communication network
Operations research 18 (1970) p 497-515

CONCLUSION .

Ce présent rapport , synthèse du travail de quatre années dans le domaine des flots , voudrait être également un guide pour le praticien de la recherche opérationnelle confronté à un problème de flot et pour l'ingénieur étudiant un réseau .

Sur la base des applications qui illustrent ce travail , il est possible d'esquisser une méthodologie pour la modélisation et la résolution de problèmes de réseaux :

- dans une première phase , il faut représenter le système technique ou le problème sous la forme d'un réseau et identifier l'entité , satisfaisant des équations de conservation , qui jouera le rôle de flot ,

- dans la seconde phase , il faut formuler mathématiquement les lois ou les fonctions objectives qui régissent la répartition du flot dans le réseau ,

- le problème peut alors être résolu soit par un des algorithmes proposés dans les chapitres 1 , 2 ou 3 , soit par une version prenant en compte des contraintes supplémentaires comme l'illustrent les chapitres 4 et 5 .

Les algorithmes d'optimisation convexe permettent ainsi de résoudre des problèmes de nature très différente tels : l'ordonnancement , le trafic routier ou la répartition du courant . L'élégance et la puissance de modélisation de la théorie des flots résident donc dans la grande variété de ses applications .

CURRICULUM VITAE

Nom :	Pasche
Prénom :	Claude
Adresse :	ch. de la Verne 7d
	1073 Savigny
Date de naissance :	12 juin 1958
Lieu d'origine :	Servion
Etat civil :	marié, un enfant

Formation :

- Baccalauréat scientifique au gymnase de la Cité 1977
- Diplôme d'ingénieur mathématicien de l'EPFL 1982
  Prix Dommer
- VII^{ème} Cours postgrade d'informatique technique ..1986
  "l'intelligence artificielle et ses applications"

Position actuelle :

assistant en recherche opérationnelle à l'EPFL
(Prof. D. de Werra , Département de Mathématiques).