# Problèmes de multiflots : état de l'art et approche par décomposition décentralisée du biflot entier de coût minimum

Wafa Rezig

Thèse

présentée par

Wafa REZIG

pour obtenir le titre de

Docteur de l'Université Joseph Fourier - Grenoble 1

(arrêtés ministériels du 5 juillet 1984 et du 30 mars 1992)

Spécialité : Informatique

Formation Doctorale : Recherche Opérationnelle

---

# Problèmes de multiflots : état de l'art et approche par décomposition décentralisée du biflot entier de coût minimum

---

Soutenue le 23 novembre 1995 devant le jury suivant :

| | | |
|---|---|---|
| MM | Philippe Jorrand | Président |
| | Pierre Dejax | Rapporteur |
| | Philippe Mahey | Rapporteur |
| | Jean Mailfert | Examinateur |
| | Gerd Finke | Directeur |

Thèse préparée au sein du laboratoire : ARTEMIS - IMAG de Grenoble

*"Un seul être vous manque et tout est dépeuplé."*

Lamartine

*A la mémoire de mon inspiratrice de toujours, à ma Mammoka*
*qui ne cesse d'occuper tout mon cœur et tout mon esprit...*

# REMERCIEMENTS

*Je tiens tout d'abord à remercier M. Philippe Jorrand, Directeur de Recherche au CNRS, pour l'honneur qu'il me fait en présidant le jury de cette thèse.*

*Ma reconnaissance va ensuite à mon directeur de thèse, M. Gerd Finke, Professeur à l'Université Joseph Fourier, pour la confiance et la patience qu'il m'a accordées tout au long de ces années de thèse. Je peux lui dire, à présent, que ses qualités humaines et scientifiques exceptionnelles m'ont toujours apporté le conseil et la sérénité nécessaires à l'aboutissement d'un tel travail.*

*J'adresse également ma gratitude à M. Philippe Mahey, Professeur à l'Université Blaise Pascal, pour son ouverture d'esprit, son extrême rigueur et sa générosité. Je ne fais ici que lui renouveler mes sentiments d'amitié qui lui étaient déjà acquis depuis mon DEA.*

*Mes remerciements vont aussi à M Pierre Dejax, Professeur à l'École Centrale de Paris, pour avoir accepté d'être rapporteur de cette thèse, pour ses corrections constructives ainsi que pour son contact chaleureux et toujours sympathique.*

*Je remercie également M. Jean Mailfert, Maître de conférences à l'Université d'Auvergne, pour l'intérêt spontané qu'il a accordé à ce travail, et pour avoir accepté de faire partie du jury de cette thèse.*

*Je remercie ma petite soeur et mon père pour avoir toujours cru en moi, et pour leurs encouragements sans cesse renouvelés.*

*Je n'oublie pas tous mes amis de l'équipe ARTEMIS, et de l'extérieur, qui m'ont beaucoup soutenue et qui ont assez bien supporté mes dernières semaines de caprices.*

# TABLE DES MATIÈRES

# Chapitre II

# TECHNIQUES DE RÉSOLUTION

## Chapitre III

## CAS PARTICULIER : LE BIFLOT

## Chapitre IV

## BIFLOT ENTIER DE COÛT MINIMUM : MÉTHODE DE RÉSOLUTION PAR DÉCOMPOSITION DÉCENTRALISÉE

## Chapitre V

## VALIDATION DES HEURISTIQUES SUR UN MODÈLE DE BIFLOT POUR LE PROBLÈME DU VOYAGEUR DE COMMERCE

# INTRODUCTION

Les problèmes de multiflots constituent une classe de problèmes très représentative des problèmes d'optimisation combinatoire. Tout d'abord leurs applications sont multiples. Que ce soit en communication, en ordonnancement, en gestion de production, ou dans bien d'autres domaines encore, on rencontre très souvent des problèmes dans lesquels il s'agit de tenir compte de la juxtaposition de plusieurs produits sur un réseau quelconque, à capacité finie. D'une façon générale, on retrouve des modèles de multiflots, comme sous-structures, dans de nombreux problèmes combinatoires complexes. D'autre part, les problèmes de multiflots ont permis, grâce à une structure sous-jacente, particulièrement attractive, de flot simple, d'engendrer des approches de résolution pour certaines classes de problèmes combinatoires.

Les modèles de multiflots se divisent en deux grandes classes : les modèles linéaires et les modèles non linéaires. Nous nous intéresserons, ici, aux modèles linéaires qui offrent déjà, un très vaste champ d'application, et une panoplie de méthodes de résolution non moins diversifiée.

C'est ainsi que nous consacrerons le chapitre I de cette thèse à la présentation des problèmes de multiflots, à travers l'introduction de leurs principales formulations et la description de leurs nombreuses applications. Ce chapitre permettra, en outre, d'illustrer la différence, et par là même, la difficulté relative des problèmes de multiflots par rapport aux problèmes de flot simple.

Le chapitre II est l'occasion d'un état de l'art sur la résolution des problèmes de multiflots. Il faut savoir que, contrairement à leurs homologues en flot simple, les problèmes de multiflots, présentés sous forme de programmes linéaires, sont bien plus difficiles à résoudre. D'une part, les opérations de pivotage du simplexe, qu'on peut effectuer directement sur des réseaux de flot simple, ne peuvent plus être reproduites sur des réseaux de multiflots. D'autre part, le théorème des valeurs entières pour le flot simple ne se généralise pas au cas des multiflots. C'est pourquoi nous détaillerons, dans un

premier temps, les techniques de résolution concernant les problèmes de multiflots dans le cas continu. Si l'on rajoute une contrainte d'intégralité, quasiment implicite pour ce genre de problèmes, ils deviennent NP-difficiles. Nous présenterons donc, dans un deuxième temps, l'existant des méthodes de résolution pour le cas entier. Ce tour d'horizon permettra de mettre l'accent sur le déséquilibre énorme entre l'effort de recherche consacré aux multiflots continus, et celui consacré aux multiflots entiers.

Cette première partie nous permettra de dégager une problématique intéressante, à savoir la grande popularité des multiflots entiers, leur extrême difficulté, et incontestablement le manque flagrant de méthodes résolution efficaces. Partant de là, nous avons voulu contribuer à l'élargissement du panorama des méthodes de résolution existant pour le cas entier. Cette motivation s'est concrétisée dans le cadre d'un cas particulier, le plus simple pour les problèmes de multiflots : le problème du biflot. Les raisons de ce choix peuvent aisément se déduire de tout ce qui vient d'être mentionné, concernant la complexité des problèmes de multiflots. Mais nous devons également rajouter que le problème du biflot présente une spécificité de structure qui donne lieu à des résultats qu'on ne peut pas toujours étendre aux autres problèmes de multiflots, et qu'il permet, en outre, de modéliser un certain nombre de problèmes combinatoires.

Nous nous intéresserons tout d'abord, dans le chapitre III, à la description du problème de biflot dans sa version orientée et non orientée. Cette description se fera en termes de formulations mathématiques, d'analyse de complexité, d'approches algorithmiques et de résultats théoriques existant pour ce genre de problème.

Le chapitre IV constitue l'essentiel de notre contribution en matière de multiflots entiers. Plus précisément, nous présenterons une méthode heuristique pour résoudre le problème du biflot entier à coût minimum. Cette méthode est basée essentiellement sur l'approche de décomposition mixte, proposée par Mahey, dans le cadre de la programmation linéaire. Nous nous attacherons particulièrement à souligner les principales difficultés suscitées par une telle approche. De fait, cette analyse nous permettra de déboucher sur des heuristiques innovantes, qui prouveront toute leur efficacité sur des problèmes de biflot purs générés aléatoirement. Dans ce contexte d'expérimentation, ce chapitre est aussi une occasion de présenter le logiciel que nous avons développé pour l'élaboration de problèmes de biflots aléatoires, et la mise en œuvre de toutes les heuristiques.

Encouragés par les résultats, très prometteurs, dégagés du chapitre IV, nous avons décidé de tester nos heuristiques sur des problèmes de biflot plus structurés. Cette seconde

validation de nos heuristiques sur un modèle de biflot, proposé par Finke et al. pour le problème du voyageur de commerce, fera l'objet du Chapitre V. Nous présentons une relaxation du modèle de biflot en question, qui sera résolue dans le cadre d'une approche par les flots, mettant en œuvre un principe de décomposition mixte. Nous montrerons, enfin, que le graphe de support du flot présente des caractéristiques intéressantes pouvant donner lieu à l'application d'une méthode exacte pour la recherche, dans ce graphe, d'un circuit hamiltonien. Malheureusement, on se rend compte que l'occurrence de tels circuits dans le graphe de support du flot est très rare. C'est pourquoi nous proposons une méthode heuristique basée sur des techniques d'insertion classiques, et permettant d'exploiter le graphe de support du flot pour construire un circuit hamiltonien. Il est clair que notre but, dans ce cadre, n'est pas de rivaliser avec les techniques récentes de résolution du problème de voyageur de commerce, mais plutôt de montrer qu'il est plus avantageux d'utiliser l'information, contenue dans le graphe de support du flot, comme préalable à une quelconque technique d'insertion. Cette stratégie globale d'optimisation sera évaluée sur des problèmes test aléatoires, à la fois pour le cas asymétrique et le cas symétrique, et ce par rapport aux techniques constructives classiques pour le problème du voyageur de commerce.

# Chapitre I

# MODÈLES DE MULTIFLOTS ET APPLICATIONS

## I.0. INTRODUCTION

Les problèmes de multiflots apparaissent lorsque plusieurs produits circulent sur un réseau quelconque à capacité finie. Ces problèmes qui permettent de tenir compte de la juxtaposition de plusieurs produits (non miscibles) sur un même graphe, constituent par conséquent un modèle naturel des réseaux de communication. On peut citer, par exemple, les réseaux téléphoniques ou d'ordinateurs, ou encore les réseaux routiers, ferroviaires ou aériens. Tous ces domaines d'application confèrent au problème de multiflots une grande importance pratique. Sur un plan théorique, les modèles de multiflots se divisent en deux grandes classes: les modèles linéaires et les modèles non linéaires. Nous aborderons dans ce cadre, les principaux modèles linéaires existant dans la littérature pour ce type de problème, ainsi que leurs différentes extensions. Nous détaillerons, dans un deuxième temps, quelques exemples d'applications pour illustrer ces modèles dans des domaines aussi variés que la planification de la production, la distribution et la gestion.

## I.1. DÉFINITIONS ET FORMULATIONS

Étant donné un réseau formé de deux ensembles finis $V$, $E$, respectivement de nœuds et d'arcs $(|V| = N,\ |E| = M)$, sur lequel circulent $K$ marchandises de type différent (cas orienté), on désigne par $x_{ij}^k \geq 0$, le flot de marchandise $k$ circulant sur l'arc $(i,j)$, $(i,j) \in E,\ k = 1,\ K$. Cette inégalité exprime que les arcs sont parcourus dans le sens direct. Pour $k = 1,\ K$, le vecteur $x^k = (x_{ij}^k)$ est un flot simple entre une source $s_k$ et une destination $t_k$ donnés appartenant à $V$, de valeur $d^k$. Autrement dit $Ax^k = r^k$ où $A$ désigne la matrice d'incidence sommets-arcs du graphe $G$ associé au réseau et où $r^k$ est un $N$-

vecteur à composantes toutes nulles sauf $s_k$ et $t_k$ qui valent respectivement $+d^k$ et $-d^k$. Considérons le vecteur $F=(F_{ij})$, $(i,j)\in E$, défini par $F_{ij}=\sum_k x_{ij}^k$. On dit que $F=(F_{ij})$ est un multiflot sur $G$, de valeur $d^1, d^2, \ldots, d^K$. La définition précédente peut être étendue de différentes manières:

**a.** Tout d'abord on peut associer à chaque flot $k$ $(k=1, K)$ un ensemble de sources $S_k \subset V$ et un ensemble de puits $T_k \subset V$ $(S_k \cap T_k = \varnothing)$. Pour se ramener à la définition précédente, il suffit alors d'ajouter pour chaque flot $k$, un sommet $s_k$ (super-source) relié a chacun des sommets de $S_k$ et un sommet $t_k$ (super-puits) auquel sont reliés tous les sommets de $T_k$. On imposera à chacun de ces nouveaux flots de n'écouler que du flot de type $k$.

**b.** On peut imposer à chaque flot $k=1, K$ d'avoir pour support un sous-graphe partiel $G^k$ de $G$. Évidemment $G^k$ doit contenir les sommets $s_k$ et $t_k$ et comporter au moins un chemin entre $s_k$ et $t_k$. Si $A^k$ est la matrice d'incidence sommets-arcs de $G^k$ alors le multiflot $F$ sera défini par le système d'équations:

$$\begin{cases} A^k x^k = r^k & \forall\, k=1, K, \\ x_{ij} = \sum_k x_{ij}^k & \forall\, (i,j)\in E, \\ x^k \geq 0 & \forall\, k=1, K. \end{cases}$$

Remarquons que ceci revient aussi à prendre une borne supérieure égale à zéro sur les arcs appropriés.

**c.** On peut très bien définir un multiflot $F$ sans imposer a chacun des $x^k$ $(k=1, K)$ d'être non négatif. Dans ce cas, la composante $F_{ij}$ du multiflot est définie comme la somme des valeurs absolues des composantes $F_{ij}^k$ des différents flots. Ceci nous amène à la définition d'un multiflot sur un graphe $H=(S, W)$ non orienté. En remplaçant chaque arête $w=(i, j)$ appartenant a $W$ par deux arcs $w^+=(i, j)$ et $w^-=(j, i)$, on définit le graphe orienté $G=(S, W)$ avec $W=(W^+ \cup W^-)$, où: $W^+=\{w^+/w \in W\}$ et $W^-=\{w^-/w \in W\}$.
Soient alors $x^1, x^2, \ldots, x^K$, $K$ flots à composantes positives sur $G$, c'est à dire vérifiant:

$$\begin{cases} A^k x^k = r^k & \forall\, k=1, K, \\ x^k \geq 0 & \forall\, k=1, K. \end{cases}$$

Où $A$ est la matrice d'incidence sommets-arcs de $G$. Le vecteur $L=(L_w)$ $w\in W$ défini par $L_w=\sum_k L_{w^+}(k)+\sum_k L_{w^-}(k)$ est appelé multiflot sur le graphe $H$.

Il serait intéressant à présent, d'illustrer sur un exemple (Gondran et Minoux [1984]), la différence fondamentale existant entre un multiflot et un flot simple. Considérons le graphe de la Figure I.1 parcouru par deux flots: f1 $(s_1 = 1,\ t_1 = 5)$ et f2 $(s_2 = 2,\ t_2 = 6)$. Tous les arcs ont une capacité égale à l'unité. Si les deux flots f1 de valeur $z^1$ et f2 de valeur $z^2$ sont de même nature (courant électrique par exemple) ils peuvent s'ajouter algébriquement sur chaque arc pour former un flot résultant de valeur $z^1 + z^2$. Par exemple si f1 a pour valeur 1 et pour support le chemin {1, 3, 4, 5} et si f2 a pour valeur 1 et pour support le chemin {2, 4, 3, 6}, la somme algébrique $z = z^1 + z^2$ est un flot de valeur 2 compatible avec les capacités et ayant pour support les chemins {1, 3, 6} et {2, 4, 5}. Au total les flots étant de même nature, tout se passe comme si en chacun des sommets 3 et 4, il s'opérait un échange entre une unité de flot provenant de 1 et une unité de flot provenant de 2. Il est clair que lorsque les flots ne sont pas de même nature, cette opération n'a plus de sens. Si le flot f1 symbolise le transport d'une unité d'un certain produit A entre 1 et 5 et le flot f2 le transport d'une unité d'un autre produit B entre 2 et 6, la somme algébrique ne permet pas de traduire le bilan global de l'opération. En effet le client en 5 qui a réclamé une unité de produit A recevrait une unité de produit B (venant de 2) et l'autre client en 6 qui a demandé une unité de produit B recevrait une unité de produit A (venant de 1). En réalité pour traduire le bilan de l'opération, on voit bien qu'il faut considérer sur chaque arc la somme des valeurs absolues des différents flots. Ainsi la quantité totale transportée sur l'arc (3, 4) est de 2 unités (une dans un sens et une dans l'autre) et le multiflot n'est pas compatible avec les capacités.

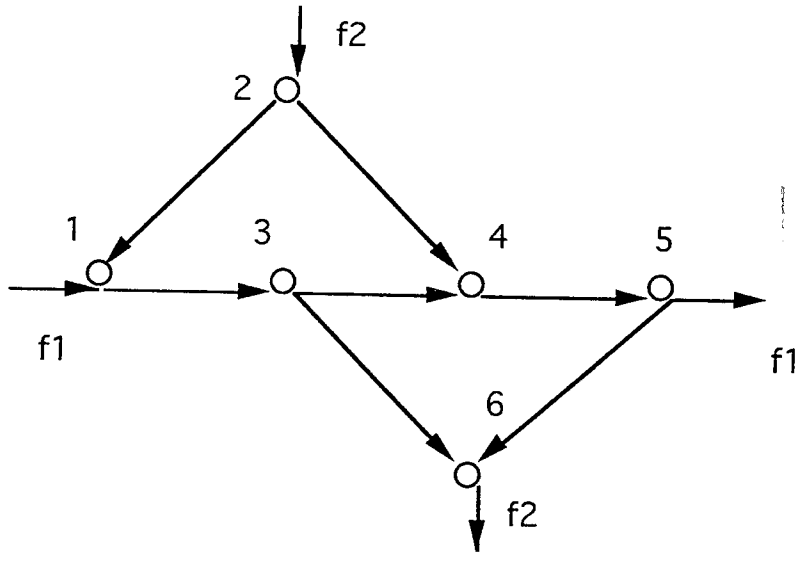



**Figure I.1** Exemple de multiflot.

Après ces notions préliminaires, nous pouvons donner la formulation mathématique d'un problème de multiflot. On distingue deux principales formulations: la première fait intervenir la matrice d'incidence sommets-arcs du graphe considéré, nous dirons qu'il s'agit de la formulation sommets-arcs; la deuxième met en jeu la matrice d'incidence arcs-chemins, elle sera dite formulation arcs-chemins. Il existe bien une troisième formulation basée sur la matrice incidence arcs-circuits, mais nous ne l'utiliserons pas dans ce contexte. Quelques détails de base sur cette formulation peuvent être trouvés dans Assad [1978].

### I.1.1. FORMULATION SOMMETS-ARCS

Le problème le plus populaire dans l'optimisation des multiflots consiste à trouver un multiflot de coût minimum. Ce problème qui a retenu l'attention de bon nombre de chercheurs sera considéré ici, comme le modèle de base pour les multiflots, en ce sens qu'il sera au centre de tous les développements ultérieurs. D'autres problèmes comme le non moins populaire problème de multiflot maximal ou le problème plus simple de multiflot compatible suscitent un intérêt moins accentué, dans la mesure où ils peuvent être considérés comme des cas particuliers du multiflot de coût minimum

#### Multiflot de coût minimum

Dans ce problème les arcs sont assortis de coûts et de capacités individuels. Il s'agit alors, connaissant les source $s_k$ et puits $t_k$ pour chaque produit $k$, ainsi que le montant $d^k$ disponible au niveau de la source et demandé au niveau de la destination ou du puits, de trouver un multiflot qui satisfasse cette demande de trafic sans violer les capacités disponibles et ce, au moindre coût. Il vient alors:

$$\text{Minimiser} \sum_k \sum_{(i,j)\in E} c_{ij}^k x_{ij}^k \qquad \text{(I.1a)}$$

sous les contraintes

$$Ax^k = r^k \qquad \forall\, k = 1,\, K, \qquad \text{(I.1b)}$$

$$\sum_k x_{ij}^k \le b_{ij} \qquad \forall\, (i,j) \in E, \qquad \text{(I.1c)}$$

$$0 \le x_{ij}^k \le u_{ij}^k \quad \forall\, (i,j) \in E,\ \forall\, k = 1,\, K. \qquad \text{(I.1d)}$$

Avec les conventions de notation suivantes:

$n$ : indice correspondant aux nœuds, $n = 1, N$.

$k$ : indice correspondant aux différents produits en circulation, $k = 1, K$.

$b_{ij}$ : capacité associée à l'arc $(i, j)$, pour $(i, j) \in E$.

$r^k$ : $N$-vecteur lié aux ressources; $r^k$ est égal à $+d^k$ si $n = s_k$, à $-d^k$ si $n = t_k$ et à 0 sinon.

$c_{ij}^k$ : coût unitaire de circulation du produit $k$ sur l'arc $(i, j)$, pour $(i, j) \in E$ et $k = 1, K$.

$x_{ij}^k$ : variable de décision désignant le flot de marchandise $k$ circulant sur l'arc $(i, j)$, pour $(i, j) \in E$ et $k = 1, K$.

$u_{ij}^k$ : borne supérieure imposée sur le flot de produit $k$ circulant sur l'arc $(i, j)$, pour $(i, j) \in E$ et $k = 1, K$.

Cette formulation met en jeu $K$ contraintes individuelles de conservation de flot classiques (I.1b) pour chaque produit $k = 1, K$, et des contraintes de couplage entre les différents produits en circulation représentées par les $M$ contraintes de capacité (I.1c) (on impose ainsi à la somme de tous les flots de produits passant sur chaque arc d'être au plus égale à sa capacité). Enfin les contraintes (I.1d) expriment que chaque arc est parcouru dans le sens direct (contraintes de positivité) et qu'en plus les flots individuels ne doivent pas dépasser une borne supérieure $u_{ij}^k$. Par la suite, il nous arrivera d'utiliser les contraintes de capacité (I.1c) sous forme d'égalités, plutôt que d'inégalités. Introduisons, dans ce cas, des variables d'écart $s_{ij}$ non négatives (mesurant la capacité inutilisée de l'arc $(i, j)$). Nous pouvons alors réécrire les contraintes de couplage (I.1c) sous la forme:

$$\sum_k x_{ij}^k + s_{ij} = b_{ij} \quad \forall (i, j) \in E. \tag{I.1c'}$$

**Multiflot maximal**

Ici les $d^k$ constituent des variables de décision, le but étant de maximiser la somme de ces montants, il vient:

$$\text{Maximiser} \sum_k d^k$$

sous les contraintes

(I.1b), (I.1c) et (I.1d).

Le multiflot maximal peut être interprété comme un cas particulier du multiflot de coût minimum. En effet, si on rajoute un arc retour (artificiel) de $t_k$ vers $s_k$ représenté par $-g^k$ et qu'on désigne par $d^k$ le flot sur cet arc, on peut reformuler le problème de multiflot maximal comme suit:

$$\text{Minimiser} \sum_k -d^k$$

sous les contraintes

(1.1c), (1.1d) et

$$\begin{array}{l} Ax^k - g^k d^k = 0 \\ d^k \geq 0. \end{array} \quad \forall\, k = 1,\, K$$

Il est clair que le vecteur coût $c^k = (0,\ -1)$ associé au vecteur flot $(x^k,\ d^k)$ définit, avec les contraintes ci-dessus, un multiflot de coût minimum.

**Multiflot compatible**

C'est le problème de décision associé au problème de multiflot de coût minimum. Ici les montants $d^k$ sont connus, on cherche alors un multiflot qui satisfait la demande sans violer les capacités. Autrement dit, on recherche une solution réalisable (si elle existe) du polyèdre défini par les contraintes (I.1b), (I.1c) et (I.1d). Il est évident que le multiflot compatible est le cas particulier du multiflot de coût minimum où l'on identifie le vecteur coût avec le vecteur nul.

## I.1.2. FORMULATION ARCS-CHEMINS

On s'attachera ici uniquement à la formulation du problème de multiflot de coût minimum. Pour simplifier la présentation, supposons que les coûts soient non négatifs. On pourra alors pour modéliser le problème, se baser sur l'ensemble $\mathbf{P}^k$ (pour tout $k$) de tous les chemins existant dans le graphe $G$ entre $s_k$ et $t_k$ (en effet, on pourra dans ce cas, éliminer les variables flot sur les cycles puisque pour toute solution optimale le flot sur n'importe quel cycle sera nul). Désignons par $f(P)$ la variable de décision correspondant au flot sur un chemin quelconque $P$ de $\mathbf{P}^k$ et ce, pour tout $k$. Soit $\delta_{ij}(P)$ un indicateur arc-chemin égal à 1 si l'arc $(i, j)$ est contenu dans le chemin $P$, et à 0 sinon. Les coûts sont associés ici à des variables chemins. Soit $c^k(P) = \sum_{(i,j)\in E} c_{ij}^k \delta_{ij}(P)$, le coût unitaire de circulation sur le chemin $P$ de $\mathbf{P}^k$ en rapport avec le produit $k$. La formulation arcs-chemins prend alors la forme suivante:

$$\text{Minimiser} \sum_{k} \sum_{P \in \mathbf{P}^k} c^k(P) f(P) \tag{I.2a}$$

sous les contraintes

$$\sum_{P \in \mathbf{P}^k} f(P) = d^k \qquad \forall\, k = 1,\, K, \tag{1.2b}$$

$$\sum_{k} \sum_{P \in \mathbf{P}^k} \delta_{ij}(P) f(P) \le b_{ij} \qquad \forall\, (i,j) \in E, \tag{I.2c}$$

$$\sum_{P \in \mathbf{P}^k} \delta_{ij}(P) f(P) \le u_{ij}^k \quad \forall\, (i,j) \in E,\ \forall\, k = 1,\, K, \tag{I.2d}$$

$$f(P) \ge 0 \qquad \forall\, P \in \mathbf{P}^k,\ \forall\, k = 1,\, K. \tag{I.2e}$$

On retrouve le même type de contraintes que dans la formulation précédente, à savoir des contraintes de conservation de flot pour chaque produit (I.2b), des contraintes de capacité (I.2c), en plus des contraintes de borne supérieure et de non négativité des flots (I.2d) et (I.2e) respectivement.

Nous avons donc présenté les principales formulations du problème de base des multiflots (multiflot de coût minimum), en faisant intervenir les mêmes types de contraintes mais sous des formes différentes. Ainsi, la formulation arcs-chemins fait intervenir une seule contrainte de conservation du flot pour chaque produit $k$, stipulant que la somme des flots sur tous les chemins connectant $s_k$ et $t_k$ est égale à $d^k$. Pour un réseau à $N$ nœuds, $M$ arcs et $K$ produits, la formulation arcs-chemins exhibe $M + K$ contraintes (en plus des contraintes de non négativité et des contraintes de borne supérieure). En comparaison, la formulation sommets-arcs met en jeu $M + NK$ contraintes, du fait qu'il existe une contrainte de conservation de flot en chaque nœud et pour chaque produit. Pour une meilleure appréciation, prenons un exemple avec $N$ = 1000 nœuds, $M$ = 5000 arcs, et un produit entre chaque paire de nœuds, soit approximativement $K \approx N^2 = 1.000.000$ produits. Par conséquent la formulation arcs-chemins contient 1.005.000 contraintes, contre quelque 1.000.005.000 contraintes pour la formulation sommets-arcs, d'où une différence appréciable en faveur de la première formulation. Néanmoins, la formulation arcs-chemins présente une variable pour chaque chemin entre un nœud source et un nœud puits et ce pour chaque produit. Il s'en suit que le nombre de variables croît exponentiellement en fonction de la taille du réseau, ce qui peut constituer un inconvénient pour cette formulation. Heureusement il se trouve qu'en pratique, toute solution optimale contient malgré un nombre énorme de chemins mis en jeu, un nombre relativement petit de chemins transportant un flot positif. Nous verrons un peu plus loin la prise en compte de

toutes ses caractéristiques en abordant les techniques de résolution des problèmes de multiflots, où nous mettrons également en valeur l'influence de la formulation sur certaines méthodes de résolution.

## I.2. EXTENSIONS DU MODÈLE DE BASE

Les principales variantes des modèles précédents consistent généralement à inclure des contraintes additionnelles ou à prendre des critères d'optimisation quelque peu modifiés. Dans certaines applications par exemple, les contraintes supplémentaires sont imposées par une limitation de ressources (autres que la capacité) devant être partagées par deux ou plusieurs arcs du réseau. Pour modéliser cela supposons que chaque unité de flot de produit $k$ consomme le montant $\rho_{ij}^k$ d'une quelconque ressource, en passant à travers l'arc $(i,j)$. Si $\alpha_{ij}$ est le montant de la ressource disponible, alors nous pouvons rajouter au modèle précédent la contrainte de ressource $\sum_k \rho_{ij}^k x_{ij}^k \leq \alpha_{ij}$, pour tout arc $(i,j) \in E$. Il est clair que la prise en compte à la fois de contraintes de capacité et de ressource revêt une importance primordiale dans les réseaux de transport. Dans les régions peu développés, ce sont les limitations en routes, lignes de train etc. ... qui contraignent le plus le mouvement des hommes. Tandis que dans les régions fortement urbanisées, les limitations effectives de mouvement proviennent plutôt des restrictions en véhicules de transport ou autres ressources, le développement du système de transport donnant lieu, dans ce cas, à une capacité excédant largement celle requise. Cremeans, Smith et Tyndall [1970] ont développé cette extension en utilisant une formulation arcs-chemins. De nombreuses autres applications faisant intervenir le facteur ressource figurent dans Wollmer [ 1972].

D'autres modèles de multiflots sont également proposés dans le cadre des réseaux de transports urbains. Dans ce type de réseau, les nœuds sont représentés par les différents carrefours, les arcs, par les rues, les voies express ou autres. Les demandes et les disponibilités sont données par le nombre de véhicules qui voyagent sur le réseau pendant une période de temps donnée (par exemple le matin à l'heure de pointe). Les produits en circulation peuvent être interprétés soit par les flots entre paires individuelles origine-destination, soit par les flots entre chaque origine et toutes ses destinations. Chaque arc possède une capacité fixe $b_{ij}$. En supposant que cette dernière puisse être augmentée moyennant des travaux routiers à un coût unitaire $h_{ij}$, on peut élaborer un autre modèle en introduisant une nouvelle variable de décision $y_{ij}$ représentant l'accroissement de la capacité. Si $B$ dénote le budget total disponible pour toutes les améliorations, alors la contrainte de capacité (I.1c) du modèle classique est remplacée par:

$$\begin{cases} \sum_k x_{ij}^k \le b_{ij} + y_{ij}, \quad y_{ij} \ge 0 \quad \forall (i,j) \in E, \\ \sum_{(i,j)\in E} h_{ij} y_{ij} \le B. \end{cases}$$

Par ailleurs, il est bien connu que le temps total de circulation sur un arc est une fonction convexe non linéaire (voir Wardrop [1952], Carter et Stowers [1963], Jorgenson [1963]). Dans ce domaine, une multitude de travaux (Prager [1954], Charnes et Cooper [1961], Jorgenson [1963], Bradley [1965], Tomlin [1966b, 1967], Pots et Olivier [1972]) proposent des approximations sous forme de fonctions linéaires ou linéaires par morceaux.

Les modèles de multiflots interviennent également dans les systèmes de communication informatique. Dans cet environnement, un réseau de communication comporte des arcs représentant des lignes de transmission et des nœuds représentant les terminaux informatiques ou d'autres systèmes plus importants. Les demandes et les disponibilités sont représentées par les niveaux de transmission entre les nœuds. Comme dans les réseaux de transport, les produits peuvent être soit des messages entre paires de nœuds, soit des messages entre une source et ses multiples destinations. Par ailleurs, chaque ligne de transmission peut accroître sa capacité $b_{ij}$, à un coût unitaire $h_{ij}$. Si $y_{ij}$ représente la variable de décision associée à l'augmentation de la capacité, la fonction objective de ce modèle s'écrit alors:

$$\text{Minimiser} \sum_k \sum_{(i,j)\in E} c_{ij}^k x_{ij}^k + \sum_{(i,j)\in E} h_{ij} y_{ij}.$$

Soit $k_{ij}$ l'augmentation maximale autorisée pour la capacité de l'arc $(i,j)$, les contraintes de capacité s'écrivent alors:

$$\sum_k x_{ij}^k \le y_{ij} + b_{ij}, \quad 0 \le y_{ij} \le k_{ij} \quad \forall (i,j) \in E.$$

Il existe deux applications essentielles pour ce modèle. Dans la première, l'objectif est de déterminer la capacité du réseau qui réalise la demande au moindre coût. Dans ce cas les coûts $c_{ij}^k$ ainsi que les capacités $b_{ij}$ sont nuls, alors que les $k_{ij}$ prennent de très grandes valeurs. Dans la seconde application, le réseau existe déjà avec des capacités fixes. Dans ce cas les $k_{ij}$ sont égaux à zéro et on cherche une répartition des flots réalisant un coût minimum.

Précisons que dans les réseaux de communication aussi bien que dans les réseaux de transport, les produits peuvent interagir de manière plus compliquée, en ce sens que si le flot d'un produit quelconque croît sur un arc, il induit un coût croissant et non linéaire sur cet arc. Donnons, à titre d'exemple simplement, le cas d'un réseau de transport où la fonction objective consiste à déterminer une répartition des flots de produits permettant de réaliser un délai global minimal pour le système. Ce genre de modèle ne contient pas de contraintes de capacité et met en jeu une fonction objective non linéaire de la forme:

$$\text{Minimiser} \sum_{(i,j)\in E} \frac{x_{ij}}{l_{ij} - x_{ij}}.$$

Cette forme de la fonction objective est dérivée des résultats de base de la théorie des files d'attente. Dans ce modèle, $l_{ij}$ représente la capacité nominale de l'arc $(i, j)$. Ainsi si le flot total $(x_{ij} = \sum_k x_{ij}^k)$ sur un arc se rapproche de sa capacité nominale, le délai tend vers $+\infty$. Fratta, Gerla et Kleïn-rock [1973] ont formulé et développé des algorithmes pour différents modèles de communication qui rentrent dans cette classe de modèles non linéaires et non contraints. D'autres développements similaires peuvent être trouvés dans Kalaba et Juncosa [1957], Hakimi [1962], Chien, Gomory et Hu [1964], White [1972a] et McCallum [1977].

Enfin, d'autres généralisations non moins importantes ont été proposées par Demmy [1969] et Swoveland[1971]. Demmy considère le cas des multiflots généralisés où le flot croît ou décroît à son passage sur un arc, ce modèle de flots avec gains est également repris par Panagiotakopoulos [1976]. Swoveland suppose, quant à lui, qu'une unité de flot de produit $k$ ne consomme plus une seule unité de capacité (comme auparavant) mais un montant $q_{ij}^k$ de cette capacité. La contrainte de capacité se généralise alors en $\sum_k q_{ij}^k x_{ij}^k \leq b_{ij}$.

## I.3. EXEMPLES D'APPLICATIONS

Les notions de modèles et d'applications des multiflots se recoupent irrémédiablement. Ainsi nous avons évoqué précédemment des modèles de multiflots en mettant l'accent sur les applications correspondantes, notamment dans les domaines de la communication et du transport. Il est clair pourtant que ces secteurs sont loin de représenter complètement le vaste champ d'application de ces problèmes. En effet, les modèles de multiflot interviennent dans de nombreux autres domaines tels que la planification de la production, la distribution, l'import-export et le management. Pour saisir l'ampleur de cette

diversification, précisons que dans un cadre général, les applications des problèmes de multiflots peuvent être de deux types. D'une part les multiflots qui mettent en jeu, comme on pourrait le penser intuitivement, des marchandises de type différent partageant un même réseau. D'autre part ceux qui impliquent un seul type de produit (en tant qu'entité physique) mais avec des origines et des destinations multiples induisant des produits "différents" par leur sources et destinations spécifiques. Ce deuxième type d'application se rencontre surtout dans les réseaux de communication et de transport que nous avons exposés précédemment. Par conséquent, pour compléter notre analyse, nous présenterons ici quatre applications adaptées sous forme d'exemples personnalisés (Ahuja, Magnanti et Orlin [1993]) pour illustrer le premier type d'application de ces problèmes, en même temps que leur grande popularité.

### I.3.1. STOCKAGE DE PRODUITS SAISONNIERS (Jewell [1957].)

Une entreprise fabrique des produits saisonniers. Leur demande est variable sur une semaine, un mois ou un trimestre. Pour utiliser au mieux ses forces de travail, l'entreprise décide de stocker ses produits en période creuse pour mieux pourvoir aux fortes demandes de la période de pointe. L'entreprise dispose d'un entrepôt de capacité $R$ utilisé pour le stockage de toutes les marchandises fabriquées. Le problème qui se pose à l'entreprise consiste à identifier, pour chaque période (semaine, mois, trimestre) les niveaux de production de tous les produits qui lui permettrons de satisfaire la demande, tout en induisant le coût le plus faible en production et en stockage. Nous pouvons modéliser ce problème d'optimisation comme un problème de multiflots défini sur un réseau approprié. Pour plus de simplicité, considérons que l'entreprise fabrique seulement deux produits et désire un ordonnancement de sa production sur chacun des quatre prochains trimestres. Soient $d_j^1$ et $d_j^2$ les demandes respectives des produits 1 et 2 au cours du trimestre $j$. Supposons que les capacités de production pour ce trimestre soient $u_j^1$ et $u_j^2$ respectivement pour les produits 1 et 2, de même pour les coûts unitaires de fabrication $c_j^1$ et $c_j^2$. Notons $h_j^1$, $h_j^2$ les coûts de stockage (manutention) des produits 1 et 2, du trimestre $j$ au trimestre $j + 1$. La Figure I.2 présente le réseau associé à ce problème de stockage. Ce réseau contient un nœud pour chaque période de temps (ici un trimestre) et deux paires de sommets source-destination pour chacun des deux produits. Les montants ressource (à la source) et demande (à la destination) sont égaux au total de la demande de chaque produit sur les quatre trimestres. Chaque sommet source $s_k$ possède quatre arcs sortants correspondant aux quatre trimestres. Un seul des deux produits peut circuler sur chacun de ces arcs. A chaque arc $(s_k, j)$ est associé un coût $c_j^k$ et une capacité $u_j^k$. De façon analogue,

chaque puits $t_k$ possède quatre arcs rentrants tels que chaque arc $(j, t_k)$ est assorti d'un coût nul et d'une capacité $d_j^k$. Le reste des arcs est de la forme $(j, j + 1)$ pour $j = 1, 2, 3$, le flot sur ces arcs représente les quantités stockées de la période $j$ à $j + 1$. Chacun de ces arcs est pourvu d'une capacité $R$ et d'un coût unitaire de circulation $h_j^1$ pour le produit 1 et $h_j^2$ pour le produit 2, c'est sur ces arcs que les flots de produits se partagent la capacité.

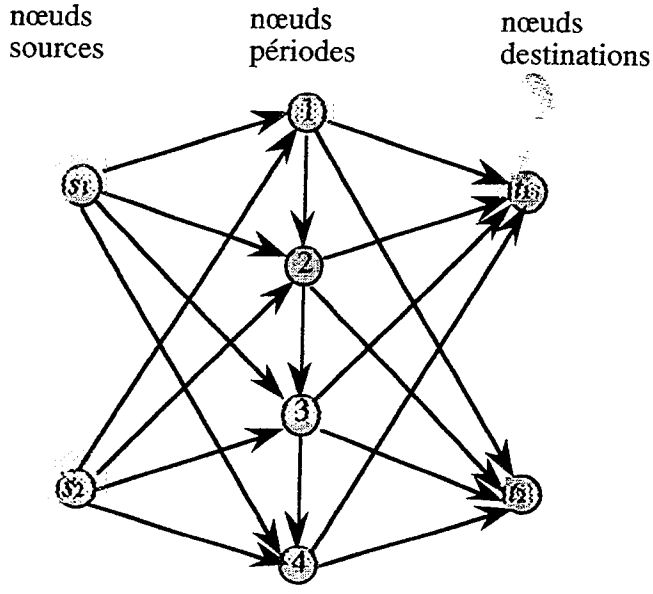



**Figure I.2** Stockage optimal de produits saisonniers.

Il est clair qu'en optimisant ce problème de multiflot, l'entreprise résoudra son problème en trouvant le plan de production et de stockage optimal.

### I.3.2. ORDONNANCEMENT D'UNE FLOTTE DE NAVIRES-CITERNES

(Bellmore, Bennigton et Lubore [1971].)

Supposons que l'on veuille déterminer un plan d'acheminement optimal pour des navires-citernes devant satisfaire un plan de livraison donné. Chaque livraison est une expédition d'un point ravitaillement à un point demande, effectuant le transport d'un produit quelconque ayant une date de livraison fixée. Considérons dans notre cas une flotte non homogène de navires-citernes différant par leur vitesse, leur capacité de transport et leur coût d'exploitation. Nous pouvons formuler ce problème sous forme d'un problème de multiflots en identifiant les différents flots de produits aux différents types de navires. Le réseau associé est représenté en Figure I.3. Pour simplifier, nous avons considéré le cas de deux livraisons s'effectuant au moyen de deux types de navires. Chaque type de navire

démarre d'une origine unique $s_k$. Ce réseau possède quatre variétés d'arcs: les arcs de mise en service, les arcs livraison, les arcs retour et les arcs de mise hors service. Un arc de mise en service correspond, comme son nom l'indique, à la mise en service d'un navire de type donné. Le coût associé à cet arc correspond au coût de déploiement du navire au point origine de la livraison, de même pour les arcs de mise hors service, mais cette fois en rapport avec le retrait du navire. Un arc livraison $(i, j)$ correspond à un parcours entre une origine $i$ et une destination $j$, le coût relatif à cet arc correspond au coût d'exploitation associé au transport du chargement par un navire de type $k$ donné. Un arc retour existe entre deux livraisons $(i, j)$ et $(k, l)$ si on peut effectuer la livraison $(k, l)$ après la livraison $(i, j)$ (conformément aux dates de livraison des deux chargements et des temps mis pour effectuer les parcours correspondants à vide et en mode chargement); ces arcs sont assortis de coûts appropriés. Chaque arc du réseau a une capacité unitaire, ce qui permet aux arcs livraison de ne se faire emprunter que par un seul type de navire. De plus pour assurer que toutes les livraisons soient exécutées dans l'ordonnancement choisi, on impose une borne inférieure égale à 1 sur le flot des arcs livraison. Dans ce réseau, chaque chemin entre une source $s_k$ et la destination $t$ correspond à un ordonnancement réalisable pour un seul navire de type $k$. Par ailleurs on peut imposer, selon le besoin, des bornes supérieures sur les flots. Par exemple si le navire de type 2 est incapable de faire la première livraison, on posera $u_{ij}^2 = 0$ ou alors $u_{jk}^1 = 0$ si le navire de type 1 est incapable d'utiliser l'arc retour $(j, k)$ (suite à une trop faible vitesse). Notons que cette fois nous devons rechercher une solution entière à ce problème de multiflot, de façon à pouvoir lui associer un ordonnancement réalisable pour la flotte. Ce qu'il faut remarquer en effet, c'est que la contrainte d'intégrité est une contrainte importante et souvent implicite dans les problèmes de multiflots.

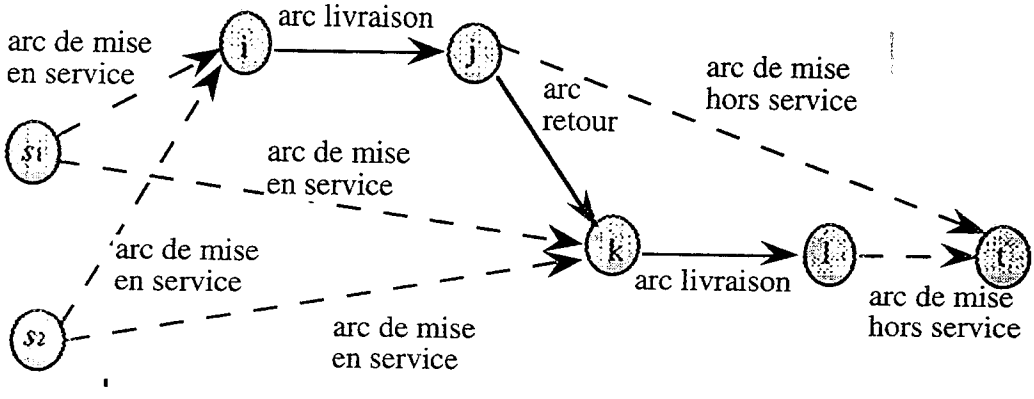



**Figure I.3** Ordonnancement d'une flotte de navires-citernes.

### I.3.3. MODÈLE DE PLANIFICATION PRODUCTION-STOCKAGE (Evans [1977].)

Considérons un processus de production composé par une séquence d'opérations que l'on peut réaliser sur différentes machines et à des périodes différentes. Supposons que ces machines disposent de capacités de production variables d'une période à une autre et qu'elles fabriquent $K$ produits différents. Dans ce cas nous pouvons traiter chaque opération comme une étape indépendante en imposant à chaque produit de passer par toutes les étapes avant d'achever sa fabrication. Nous utiliserons les notations suivantes :

$d_j^k$ : demande du produit $k$ à la période $j$.
$x_{rj}^k$ : quantité de produit $k$ fabriquée à l'étape $r$ en période $j$.
$I_{rj}^k$ : stock du produit $k$ utilisé de la période $j$ à $j+1$, à l'étape $r$.
$P_{rj}^k$ : capacité de production du produit $k$ à l'étape $r$ en période $j$.
$u_{rj}^k$ : borne supérieure sur le niveau des stocks du produit $k$ entre les périodes $j, j+1$ à l'étape $r$.
$c_{rj}^k$ : coût de production unitaire du produit $k$ à l'étape $r$, en période $j$.
$h_{rj}^k$ : coût de manutention du stock de produit $k$ de la période $j$ à $j+1$, à l'étape $r$.

La Figure I.4 montre le réseau de multiflot associé à ce problème de planification de la production. Ce réseau possède $RT + 1$ nœuds: pour chaque $r = 1, R$, et chaque $j = 1, T$, le nœud $j^r$ représente la *rème* étape de la période $j$. Le nœud 0 constitue le nœud source pour tous les produits, on lui attribue un montant égal à la somme de toutes les demandes de tous les produits sur toutes les périodes. Les variables de décision représentent les différents flots et sont indiquées en regard des arcs pour un seul flot, en omettant l'indice du produit (pour ne pas surcharger la figure) . Les variables $x$ symbolisent la production, les variables $I$ les stocks et ce, pour des étapes et des périodes données. Comme le montre la construction, il existe une correspondance directe entre un multiflot entier réalisable sur ce réseau et un plan de production et de stockage. Par conséquent, un plan de production optimal pourra s'obtenir en recherchant un multiflot entier de coût minimal sur le réseau en question.

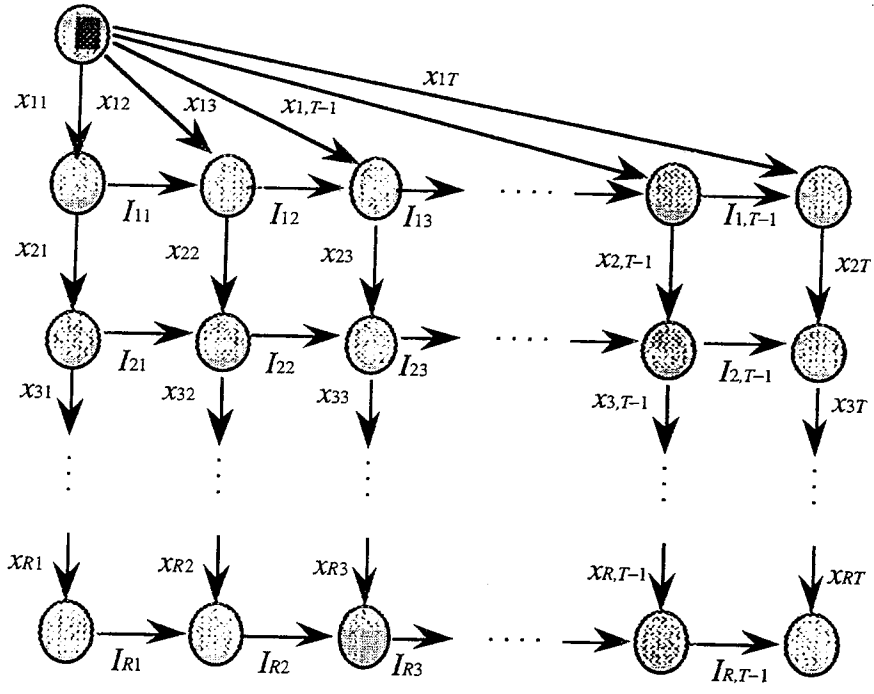



**Figure I.4** Modèle de planification production-stockage.

### I.3.4. AFFECTATION DE GROUPES ETHNIQUES À UN ENSEMBLE D'ÉCOLES (Clarke et Surkis[1968].)

En 1968, la cour suprême des États-Unis statuait que toutes les institutions d'enseignement devaient mettre en place des techniques efficaces pour promouvoir des écoles moins homogènes à travers tout le territoire national, de façon à ce que les étudiants soient accueillis au sein des écoles sans aucune discrimination. Mais la cour suprême ne précisant aucune spécification sur ce que devait être un "bon équilibre racial", chaque école se mit à définir son propre équilibre et ses propres critères pour réaliser son plan de "déconcentration". Cette application décrit un modèle de multiflot qui permet de déterminer une affectation optimale des étudiants aux écoles, en termes de minimisation de la distance parcourue par les étudiants et de satisfaction d'un équilibre racial pour chaque école. Supposons qu'un département donné dispose de $S$ écoles, et que chaque école $j$ possède une capacité d'accueil $u_j$. Pour les besoins de la formulation, nous allons diviser ce département en $L$ zones de population. Ces zones peuvent être des arrêts de bus, ou des blocs d'immeubles par exemple. La seule restriction est que ces zones soient en nombre fini et que l'on puisse par une seule mesure de distance, approximer raisonnablement la distance que doit parcourir un étudiant du centre $i$, s'il est affecté à l'école $j$. Notons $S_{ik}$ le nombre d'étudiants du *kème* groupe ethnique présent dans la zone de population $i$.

L'objectif est d'affecter les étudiants aux écoles en respectant la composition ethnique prévue par l'école et en minimisant la distance totale parcourue par les étudiants. Chaque école concrétise son critère d'équilibre en définissant deux bornes $l_{jk}$ et $u_{jk}$ représentant le nombre minimal et maximal d'étudiants du groupe ethnique $k$ à faire admettre. Nous pouvons modéliser ce problème comme un problème de multiflot défini sur un réseau adéquat. La Figure I.5 donne une représentation de ce réseau pour une application à trois zones de population et trois écoles. Ce réseau possède trois nœuds $b$ représentant les zones, trois nœuds $c$ représentant les écoles et une paire de nœuds source-destination $(s,t)$ pour chaque groupe ethnique. Les étudiants du *kème* groupe ethnique circulent de la source $s_k$ au puits $t_k$ via les sommets "zones" et "écoles". Associons à l'arc $(s_k, b_i)$ connectant la source du *kème* groupe à la zone de population $b_i$, une borne supérieure égale à $S_{ik}$, et à l'arc $(b_i,c_j)$ connectant la zone $i$ à l'école $j$, un coût $f_{ij}$ égal à la distance entre les deux points. Nous pouvons noter ici un autre type d'extension: la capacité sur les nœuds, chaque école possédant une capacité d'accueil $u_j$. Toutefois cette situation peut-être ramenée au modèle de base, en dédoublant les sommets "écoles" comme indiqué sur le réseau de la Figure I.5 et en attribuant la capacité $u_j$ à l'arc $(c_j,d_j)$. Ainsi nous pouvons garantir que le nombre total d'étudiants affecté à cette école, reste plus petit ou égal à sa capacité d'accueil. Finalement nous pouvons introduire les contraintes de composition ethnique en imposant des bornes inférieure et supérieure sur les flot de l'arc $(d_j, t_k)$. Il est alors clair qu'un multiflot de coût minimum sur ce réseau donnera lieu à une affectation optimale des étudiants aux écoles.

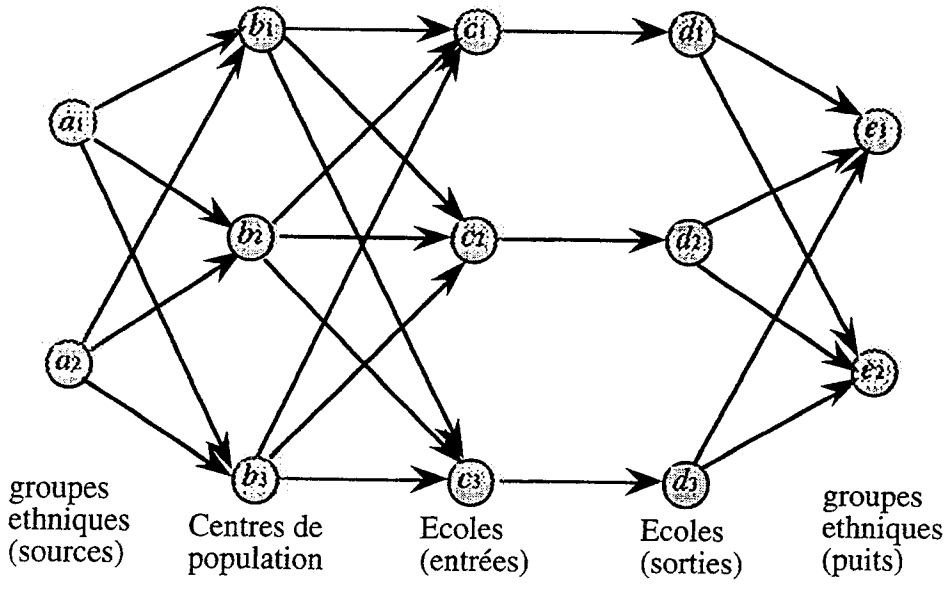



**Figure I.5** Modèle de multiflots pour le problème d'équilibre racial dans une école.

Pour compléter cette quadruple illustration des applications de multiflots, nous allons mentionner d'autres applications non moins intéressantes. Citons, par exemple, le modèle d'import-export de Golden [1975], ou alors le modèle de Korte [1988] dans le domaine de la conception de circuits VLSI. Dans un cadre plus général, nous pouvons évoquer les modélisations de Rao et Zionts [1968] concernant les problèmes d'affectation dans les réseaux de transport, où ils prennent en compte la dimension espace et la dimension temps. Cette approche a été utilisée par White et Bomberault [1969] pour modéliser l'affectation de wagons vides dans un réseau ferroviaire. Il apparaît d'ailleurs que dans le secteur ferroviaire, les problèmes de multiflots trouvent un domaine d'application privilégié, à en juger par le nombre de modèles traités : White [1972b], Bodin, Golden Schuster et Rowing [1980], Assad [1980a], et Crainic Ferland et Rousseau [1984]. Dans ce même contexte de réseaux de transport dynamiques, citons également l'application de Powell et Sheffi [1983, 1989] dans laquelle ils appréhendent un problème de transport de moteurs grâce à un système de remorquage, par agrégation de plusieurs chargements, pour diminuer les coûts de transport.

Trois autres modèles typiques des multiflots continuent à être utilisés dans de nombreux cas réels pour résoudre des problèmes de distribution. Il s'agit du modèle développé par Barnett, Binkley et McCarl [1982] pour analyser l'effet de l'altération de la capacité de certains ports aux U.S.A. sur les exportations de blé, de mais et de soja. Le deuxième modèle a été mis au point par Helgason, Kennington et Wong [1981] ainsi que Kirby, Hager et Wong [1982]. Ce modèle est intervenu comme aide à la décision pour l'aménagement des forêts en Californie. Enfin le troisième modèle développé par Ali, Helgason et Kennington [1982] a été utilisé pour assister le personnel des forces aériennes américaines à établir un bon schéma logistique pour distribuer des pièces de rechange pour différents avions, missiles et radars.

Un autre domaine de prédilection pour les multiflots est celui de la gestion, à ce propos on peut mentionner les modèles respectifs de production et de stockage par Zangwill [1969b] et Rao [1969]; une description plus complète de ces modèles est donnée par Elmaghraby [1970].

Pour finir, disons que d'autres critères d'optimisation peuvent intervenir dans les modèles de multiflots. C'est le cas de l'application de Kaplan [1973] où il considère que des ressources vitales (vivres et médicaments par exemple) peuvent être mises en jeu, d'où une minimisation conjointe des coûts de transport et des coûts induits par les demandes

insatisfaites de ces ressources. Un autre schéma de fonction objective est mis en évidence par le modèle de multiflot avec conditions de sécurité de Chen et Chin [1992]. Dans leur réseau de multiflot, ils attribuent à un arc trois valeurs dont deux nous sont maintenant familières (coût et capacité), la troisième reflétant le degré de sécurité comme quantification des conditions de circulation sur cet arc. Pour le transport de produits fragiles ou précieux, par exemple, la minimisation conjointe du coût et de la sécurité du transport prend évidemment tout son sens. Ces auteurs montrent que malgré cette généralisation du modèle conventionnel de multiflot, les deux problèmes sont en fait polynomiallement équivalents. Enfin dans les secteur de la défense, on peut modéliser la neutralisation d'un système de protection par un modèle de multiflots. Bellemore, Greenberg et Jarvis [1970], Bellmore et Ratliff [1971], décrivent ce problème concernant la recherche d'une coupe de capacité minimale (pour une offensive d'attaque en l'occurrence) dans un réseau de multiflots, où la capacité d'un arc équivaut à son coût de destruction par les forces ennemies.

## I.4. CONCLUSION

Ce chapitre nous a permis de mettre l'accent sur les principaux modèles de multiflots et leurs multiples applications. Il est clair que, sans vouloir être exhaustifs, nous avons tout de même débouché sur une large panoplie d'applications où rivalisent les thèmes de production, distribution, gestion, transport et communication. A travers cette description mettant en valeur la richesse de ces modèles et l'étendue de leur champ d'application, nous avons voulu souligner l'importance des problèmes de multiflots à la fois sur un plan pratique et sur un plan théorique.

# Chapitre II

# TECHNIQUES DE RÉSOLUTION

## II.0. INTRODUCTION

Tel qu'il a été formulé, le problème de multiflot de coût minimum se présente comme un programme linéaire dont la résolution pourrait se faire par application de la méthode du simplexe. Cependant, la dimension souvent trop grande et la structure en blocs de ce genre de problème, rend prohibitive l'application directe d'une telle méthode. Par ailleurs, comme nous l'avons vu dans certaines applications, il arrive que les composantes des différents flots soient astreintes à ne prendre que des valeurs entières. Il en résulte des programmes linéaires en nombres entiers (généralement de grandes dimensions) pour lesquels on ne connaît pas actuellement de méthodes de résolution exactes. Ce chapitre va nous permettre d'envisager les différentes techniques de résolution existant dans la littérature pour ce genre de problème et ce, dans le cas continu et dans le cas entier. Dans le cadre des multiflots fractionnaires, nous présenterons un panorama complet des méthodes classiquement proposées, puis nous donnerons un aperçu sur les nouvelles approches récemment développées pour la résolution de ces problèmes. Dans le cas des multiflots entiers, la démarche sera quelque peu différente, en raison de la complexité du problème. Les techniques proposées sont moins nombreuses, plus personnalisées et consistent en des méthodes approchées. Nous avons retenu deux approches fondamentalement différentes permettant d'embrasser la résolution des trois types de problèmes de multiflots; la première est une heuristique procédant par relaxations successives pour la résolution des problèmes de multiflot compatible et de coût minimum, la deuxième est une approche d'approximation déterministe pour le problème de multiflot maximal. L'approche proposée

pour la résolution du multiflot maximal représente le meilleur algorithme d'approximation actuellement disponible et permet d'illustrer la grande difficulté des problèmes de multiflots par rapport au cas particulier de flot simple. Ceci, entre autres, a motivé l'introduction, dans ce contexte, d'une autre forme de méthode pour la résolution des problèmes de multiflots. Cette "méthode" consiste à transformer (quand cela est possible) un réseau de multiflot en un réseau de flot simple approprié, et à le résoudre (beaucoup plus efficacement) en tant que tel.

## II.1. CAS CONTINU

Cette première partie sera consacrée à l'optimisation du modèle de base des multiflots, à savoir le problème de multiflot à coût minimum. Les deux formulations de ce problème seront mises en jeu à travers, les conditions d'optimalité pour ce type de problème, dans un premier temps, et les approches de base classiquement proposées pour le résoudre, dans un deuxième temps. Nous présenterons, dans cet ordre, les méthodes de décomposition, puis les méthodes de partitionnement. Rappelons, au passage, que toutes ces techniques de résolution ont déjà fait l'objet de nombreux travaux de synthèse, notamment ceux de Kennington [1978] et Assad [1978], et plus récemment encore, celui de Schneur [1991]. Pour finir, nous établirons une comparaison entre les différentes approches proposées, en termes d'efficacité et de tailles de problèmes résolus. En même temps, nous donnerons un bref aperçu sur les nouvelles approches récemment mises à jour pour résoudre des problèmes de multiflots de plus en plus énormes...

### II.1.2. CONDITIONS D'OPTIMALITÉ

Les conditions d'optimalité d'un problème permettent de décider si oui ou non une solution réalisable quelconque est optimale. Par ailleurs, ces conditions peuvent donner une interprétation à certains algorithmes comme étant des méthodes particulières visant à satisfaire ces mêmes conditions. Plus encore, ces conditions d'optimalité peuvent inspirer certaines approches de résolution, comme nous le verrons un peu plus loin. Nous allons, dans ce qui suit, énoncer les conditions d'optimalité pour le problème de multiflot de coût minimum que nous avons défini dans le chapitre précédent. Pour ce faire, nous utiliserons dans un premier temps la formulation sommets-arcs de ce problème puis dans un deuxième temps, la formulation arcs-chemins.

### II.1.1.1. Conditions d'optimalité en formulation sommets-arcs

reprenons la formulation (I.1) en supposant, pour simplifier la présentation, qu'il n'y ait pas de bornes supérieures sur les flots individuels, autrement dit $u_{ij}^k = +\infty$. Nous utiliserons ici, la forme (I.1 c') de la contrainte de capacité. Comme le problème de multiflot est un problème linéaire, nous pouvons utiliser les conditions d'optimalité issues de la programmation linéaire pour caractériser les solutions optimales. Soient $w_{ij}$ les variables associées aux contraintes de capacité sur chaque arc $(i,j)$ et $\pi^k(i)$ les variables duales associées aux contraintes de conservation de flot pour toute combinaison entre les nœud $i$ et produit $k$. Nous pouvons alors définir le coût réduit de l'arc $(i,j)$ relatif au produit $k$ par:

$$c_{ij}^{\pi,k} = c_{ij}^k + w_{ij} - \pi^k(i) + \pi^k(j).$$

Les flots $y_{ij}^k$ sont optimaux pour le problème de multiflot (I.1) si et seulement si ils sont réalisables, et pour un certain choix des variables $w_{ij}$ (non négatives) et $\pi^k(i)$ (quelconques), les coûts réduits et les variables flot sur les arcs satisfont les conditions suivantes:

$$w_{ij}(\sum_k y_{ij}^k - b_{ij}) = 0 \quad \forall\, (i,\, j) \in E, \tag{II.1a}$$

$$c_{ij}^{\pi,k} \geq 0 \quad \forall\, (i,\, j) \in E,\ \forall\, k = 1,\, K, \tag{II.1b}$$

$$c_{ij}^{\pi,k} y_{ij}^k = 0 \quad \forall\, (i,\, j) \in E,\ \forall\, k = 1,\, K. \tag{II.1c}$$

Nous nous référerons aux variables $w_{ij}$ qui satisfont ces conditions comme les *prix optimaux sur les arcs*, pendant que les variables $\pi^k(i)$ seront désignées par *potentiels nœuds optimaux*. Le théorème suivant fait alors ressortir le lien existant entre multiflot et flot simple:

***Théorème II.1.*** *Soit $y_{ij}^k$ et $w_{ij}$ les flots et les prix optimaux sur les arcs respectivement pour le problème* (I.1). *Alors pour chaque produit k, les variables $y_{ij}^k$ sont également solution du problème de flot simple de coût minimum défini par:*

$$min\left\{ \sum_{(i,j)\in E} (c_{ij}^k + w_{ij}) x_{ij}^k \,\middle|\, Ax^k = r^k,\ x_{ij}^k \geq 0\ \forall (i,\, j) \in E \right\}. \tag{II.2}$$

*Preuve.* Comme $y_{ij}^k$ et $w_{ij}$ sont optimaux pour le problème (I.1), ils satisfont les conditions d'optimalité (II.1) et en particulier les conditions (II.1b) et (II.1c). Remarquons alors que ces conditions ne sont rien d'autre que les conditions d'optimalité d'un problème de flot

simple de coût minimum, sans contraintes de capacité et avec des coûts sur les arcs égaux à $(c_{ij}^k + w_{ij})$. (Voir Ahuja, Magnanti et Orlin [1993] chap. 9 par exemple.)

Cette propriété montre que l'on peut utiliser une approche séquentielle pour obtenir les prix optimaux et les potentiels nœuds optimaux: en déterminant d'abord les prix optimaux puis en déduisant les potentiels nœuds et les flots optimaux, par résolution du problème de flot simple à coût minimum défini en (II.2).

### II.1.1.2. Conditions d'optimalité en formulation arcs-chemins

De la même manière nous pouvons réécrire les mêmes conditions d'optimalité relativement à la formulation arcs-chemins du problème de multiflot de coût minimum. Désignons par $w_{ij}$, les mêmes variables prix sur les arcs, associées aux contraintes de capacité. Soient $\sigma^k$ les nouvelles variables rattachées aux contraintes demande, se rapportant à chaque produit $k$. Ces variables seront désignées par *les prix du produit k* (pour tout *k)*. Le coût réduit associé à chaque variable chemin est alors défini par:

$$c_P^{\sigma,w} = c^k(P) + \sum_{(i,j)\in P} w_{ij} - \sigma^k.$$

Les conditions des écarts complémentaires prennent alors la forme suivante: les flots *f(P)* de la formulation (I.2) sont optimaux pour ce problème, si et seulement si pour un certain choix des variables $w_{ij}$ et $\sigma^k$, les coûts réduits et les variables flot satisfont les conditions suivantes:

$$w_{ij}\left(\sum_k \sum_{P\in \mathbf{P}^k} \delta_{ij}(P) f(P) - b_{ij}\right) = 0 \;\forall\; (i,j)\in E, \qquad \text{(II.3a)}$$

$$c_P^{\sigma,w} \geq 0 \quad \forall\, P \in \mathbf{P}^k \text{ et } \forall\, k = 1, K, \qquad \text{(II.3b)}$$

$$c_P^{\sigma,w} f(P) = 0 \quad \forall\, P \in \mathbf{P}^k \text{ et } \forall\, k = 1, K. \qquad \text{(II.3c)}$$

La condition (II.3a) exprime que le prix de l'arc $(i,j)$ est égal à 0 si la solution optimale *f(P)* n'utilise pas toute la capacité de cette arc. Comme le coût réduit relatif au chemin $P$ est défini par $c^k(P) = \sum_{(i,j)\in P} c_{ij}^k$, on peut alors réécrire ce coût comme:

$$c_P^{\sigma,w} = \sum_{(i,j)\in P} (c_{ij}^k + w_{ij}) - \sigma^k.$$

La condition (II.3b) exprime alors que le coût modifié $\sum_{(i,j)\in P}(c_{ij}^k + w_{ij})$ de chaque chemin connectant une source $s_k$ et une destination $t_k$ doit être supérieur ou égal au prix du produit $k$, soit $\sigma^k$. La condition (II.3c) implique que chaque chemin transportant un flot non nul

dans la solution optimale, doit avoir un coût modifié exactement égal au prix du produit transporté $\sigma^k$. Conjointement ces deux conditions débouchent sur l'observation suivante: $\sigma^k$ est la plus courte distance entre $s_k$ et $t_k$, conformément aux coûts modifiés $(c_{ij}^k + w_{ij})$, et dans la solution optimale, chaque chemin entre $s_k$ et $t_k$ véhiculant un flot positif est un plus court chemin relativement à ces mêmes coûts modifiés.

## II.1.2. MÉTHODES DE DÉCOMPOSITION

Les méthodes de décomposition sont intéressantes à plus d'un titre. En effet, elles évitent le recours à la programmation linéaire et sont assez simples à mettre en œuvre. De plus, elles procèdent par décomposition, c'est à dire ramènent la résolution du problème d'ensemble à une séquence de problèmes de flots simples ou de plus courts chemins, pour lesquels des algorithmes efficaces peuvent être employés. Nous aborderons d'abord les approches classiques de décomposition, à savoir la décomposition par les prix et la décomposition par les ressources. Dans le cadre de la décomposition par les prix, nous présenterons une analyse récemment effectué par Jones et Lustig, Farvolden et Powell [1993] sur un aspect de l'optimisation des multiflots jusqu'ici occulté par les chercheurs. Il s'agit en fait de l'influence de la formulation sur l'efficacité d'une méthode de décomposition très populaire: la méthode de Dantzig-Wolfe. Enfin pour finir, nous avons jugé intéressant, pour être exhaustifs entre autres, d'introduire un algorithme de décomposition mixte mettant en jeu les deux types de décomposition précédents. C'est un algorithme qui a été proposé dans le cadre plus général des problèmes à structure bloc-angulaire, et que nous avons adapté au problème de multiflot.

### II.1.2.1. Décomposition par les prix

L'approche de décomposition par les prix consiste comme son nom l'indique, à mettre à jour des variables prix (variables duales associées aux contraintes de capacité), et à résoudre un ensemble de sous-problèmes de flot simple. Il lui faut alors trouver des prix appropriés de façon à ce que les solutions optimales de ces sous-problèmes permettent conjointement, de résoudre le problème de multiflot original. Nous allons décrire ici deux méthodes pour calculer les prix adéquats, la relaxation lagrangienne et la méthode de génération de colonnes. Dans le cadre de la technique de génération de colonnes, nous aborderons partiellement la méthode de décomposition de Dantzig-Wolfe pour donner une autre interprétation au calcul des variables prix, et pour introduire de récents développements concernant l'effet de la formulation sur cette même technique de décomposition par les prix.

II.1.2.1.1. Relaxation lagrangienne

Appliquons la théorie de la relaxation lagrangienne au problème (I.1) en associant des multiplicateurs de Lagrange $w_{ij}$, non négatifs, aux contraintes de capacité (I.1c). Écrivons alors le sous-problème de Lagrange correspondant:

$$L(w) = \min \sum_k c^k x^k + \sum_{(i,j)\in E} w_{ij} (\sum_k x_{ij}^k - b_{ij}) \qquad \text{(II.4a)}$$

ou alors de manière équivalente:

$$L(w) = \min \sum_k \sum_{(i,j)\in E} (c_{ij}^k + w_{ij}) x_{ij}^k - \sum_{(i,j)\in E} w_{ij} b_{ij} \qquad \text{(II.4b)}$$

sous les contraintes

$$Ax^k = r^k \quad \forall\, k = 1,\ K, \qquad \text{(II.4c)}$$

$$x_{ij}^k \geq 0 \qquad \forall\, k = 1,\ K,\ \forall\ (i,\ j) \in E. \qquad \text{(II.4d)}$$

Notons que pour un quelconque choix des multiplicateurs de Lagrange, le terme $-\sum_{(i,j)\in E} w_{ij} b_{ij}$ peut-être ignoré (sa valeur étant constante dans ce cas). Il en découle que le problème (II.4) peut être décomposé en plusieurs problèmes de flot simple de coût minimum, correspondant chacun à un produit $k$ donné. Par conséquent, nous pouvons mettre en œuvre une procédure de sous-gradient pour mettre à jour les multiplicateurs $w_{ij}$, tout en résolvant alternativement un ensemble de problèmes de flot simple de coût minimum. Si $y_{ij}^k$ représente la solution optimale aux sous-problèmes de flot simple pour des valeurs $w_{ij}^t$ des multiplicateurs de Lagrange à la *tème* itération, on peut écrire:

$$w_{ij}^{t+1} = [w_{ij}^t + \theta_t (\sum_k y_{ij}^k - b_{ij})]^+.$$

La notation $[a]^+$ représente la partie positive de $a$ et le scalaire $\theta_t$, le pas de déplacement. Ces pas sont calculés conformément aux techniques algorithmiques de relaxation lagrangienne (voir par exemple Ahuja, Magnanti et Orlin [1993] Chap. 16). Remarquons que le multiplicateur $w_{ij}^t$ de l'arc $(i, j)$ croît du montant pondéré $(\sum_k y_{ij}^k - b_{ij})$ si le flot optimal $y_{ij}^k$ utilise plus que la capacité $b_{ij}$ de cet arc et décroît du même montant dans le cas inverse. Par ailleurs le problème de multiflot étant un problème linéaire, la valeur optimale du problème dual $L^* = \max_{w\geq 0} L(w)$ est égale à la valeur optimale du programme (I.1).

L'utilisation dans ce contexte d'une technique de sous-gradient est attractive à plus d'un titre: premièrement elle permet d'utiliser la structure particulière des problèmes de

multiflots liée aux contraintes (non relaxées) de conservation du flot pour chaque produit. Deuxièmement les formules de mise à jour des multiplicateurs de Lagrange sont très faciles à implémenter dans un programme informatique. Cependant cette méthode compte certaines limitations dues essentiellement à de faibles propriétés de convergence (pour assurer la convergence, les pas de déplacement doivent être assez petits, ce qui rend la convergence assez lente). Par ailleurs, cette méthode étant une technique duale, la convergence vers des variables optimales $w_{ij}^t$ ne signifie pas toujours que les solutions optimales $y_{ij}^k$ convergent vers la solution optimale du problème de multiflot.

Avant de conclure cette partie, il serait utile de mentionner que l'on peut combiner relaxation lagrangienne et programmation linéaire pour résoudre les problèmes de multiflots. En effet, la relaxation lagrangienne peut fournir une bonne base initiale pour démarrer l'application de l'algorithme du simplexe au programme (I.1). Si nous résolvons les sous-problèmes de Lagrange, pour un choix quelconque des variables $w_{ij}^t$, par la méthode adaptée du simplexe dans les réseaux, nous obtenons une solution arbre optimale associée à chaque produit $k$ (voir Ahuja, Magnanti et Orlin [1993] chap. 11.) Cette solution arbre correspond en fait à une base $B^k$ du système $Ax^k = r^k$. nous pouvons alors étendre ces bases à une base globale $\boldsymbol{B}$ comme l'indique la Figure II.1, la matrice identité dans le coin supérieur gauche correspond aux variables d'écart $s_{ij}$. Notons que pour la solution associée à la base $\boldsymbol{B}$: (1) chaque variable flot $x^k$ est égale à la solution $y^k$ du sous-problème de Lagrange et (2) chaque variable $s_{ij}$ est donnée par $s_{ij} = b_{ij} - \sum_k y_{ij}^k$. Si toutes les variables $s_{ij}$ sont non négatives, la base est réalisable, sinon elle est irréalisable. Dans un cas comme dans l'autre, on peut appliquer les méthodes standard de la programmation linéaire pour résoudre le problème en question.

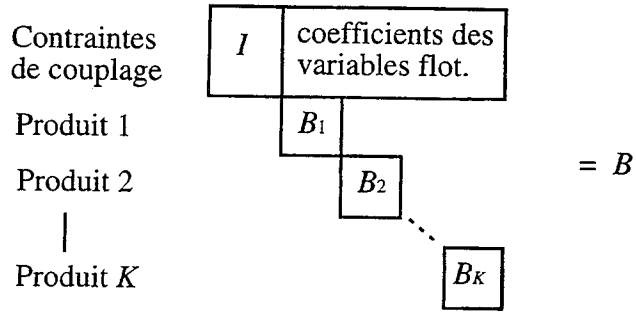



**Figure II.1** Base initiale formée des bases individuelles des sous-problèmes.

Bixby [1991] utilise une variante de cette approche pour résoudre des problèmes de multiflots de très grande taille. Sur un plan pratique, son expérience montre qu'en utilisant

une bonne base initiale, pouvant être en l'occurrence une bonne approximation de la solution optimale, la méthode du simplexe pouvait réaliser de très bonnes performances.

II.1.2.1.2. Technique de génération de colonnes

Cette technique sera illustrée à partir de la formulation arcs-chemins (I.2). Remarquons alors que le nombre de colonnes de ce programme linéaire, correspond au nombre total de chemins pour tous les produits, entre chaque paire de sommets source-destination. Ce nombre souvent énorme peut alors inhiber l'application d'une quelconque méthode de programmation linéaire. La technique de génération de colonnes permet de surmonter cet inconvénient en mettant en œuvre l'idée que toutes les colonnes ne doivent pas être complètement explicitées, mais générées au fur et à mesure de leur "utilité". La méthode révisée du simplexe apparaît alors comme la méthode la plus adaptée pour ce type de stratégie. Dans ce cadre, la méthode révisée du simplexe met à jour les variables duales $w_{ij}$ et $\sigma^k$, de façon à ce que le coût réduit de chaque variable en base soit égal à zéro. Par conséquent, si un chemin $P$, connectant pour le produit $k$ les sommets source-destination $s_k$ et $t_k$ est en base, alors $c_P^{\sigma,w} = 0$, ou de manière équivalente $\sum_{(i,j)\in P}(c_{ij}^k + w_{ij}) = \sigma^k$. Notons que chaque base contient $K + M$ variables et que par conséquent chaque base donnera lieu à $K + M$ équations de la forme:

$$\sum_{(i,j)\in P}(c_{ij}^k + w_{ij}) = \sigma^k, \quad \text{pour chaque chemin } P \text{ dans la base.}$$

Ces équations contiennent $K + M$ variables dont $M$ variables prix $w_{ij}$ et $K$ variables distances $\sigma^k$ correspondant à des plus courts chemins. La condition d'optimalité (II.4c) impose que $c_P^{\sigma,w} f(P) = 0$ pour tout chemin $P$ du réseau. Comme chaque chemin $P$ de la base vérifie cette condition, nous pouvons lui faire véhiculer n'importe quel montant de flot sans violer la condition (II.4c). Si la variable $s_{ij} = [\sum_k \sum_{P\in \mathbf{P}^k} \delta_{ij}(P) f(P) - b_{ij}]$ n'est pas en base, $s_{ij} = 0$ et la condition (II.4a) est satisfaite. D'un autre côté si la variable $s_{ij}$ est en base, son coût réduit $(0 - w_{ij})$ est nul et la solution satisfait (II.4a). Reste donc la condition (II.4b) à vérifier: le coût réduit de chaque variable est positif ou nul, autrement dit, pour chaque produit $k$:

$$c_P^{\sigma,w} = \sum_{(i,j)\in P}(c_{ij}^k + w_{ij}) - \sigma^k \geq 0 \quad \forall P \in \mathbf{P}^k,$$

ou de manière équivalente,

$$\min_{P \in \mathbf{P}^k} \sum_{(i,j) \in P} (c_{ij}^k + w_{ij}) \geq \sigma^k.$$

Pour vérifier cette condition, nous pouvons résoudre un problème de plus court chemin pour chaque produit $k$. Si pour tous les produits $k$, la longueur du plus court chemin est au moins égale à $\sigma^k$, nous sommes à l'optimum. Dans le cas contraire, il existe au moins un produit $k$ pour lequel $Q$ représente le plus court chemin relativement aux coûts modifiés $(c_{ij}^k + w_{ij})$, tel que la longueur de ce chemin soit inférieure à $\sigma^k$. Il vient alors:

$$c_P^{\sigma,w} = \sum_{(i,j) \in Q} (c_{ij}^k + w_{ij}) - \sigma^k < 0.$$

En termes de programmation linéaire, le chemin $Q$ a un coût réduit négatif, il peut donc remplacer un des chemins $P$ déjà en base. Le changement de base qui s'ensuit s'effectue alors de la même manière que dans l'application classique du simplexe. D'autres schémas d'implémentation sont également possibles, par exemple le pivotage sur toutes les colonnes favorables à chaque étape, ou encore la sauvegarde des colonnes devenant non basiques, au cas où leur coût réduit deviendrait négatif ultérieurement. Une illustration de ce schéma est décrite en détails par Bazaraa et Jarvis [1977].

Avant de finir, précisons que cette technique de génération de colonnes offre un avantage intéressant par rapport à la relaxation lagrangienne. En effet, soit $g^*$ la valeur optimale de la fonction objective du problème (I.2) et soit $g^{pl}$ la valeur de la fonction objective à une étape de résolution du même problème, par la méthode du simplexe. Comme $g^{pl}$ correspond à une solution réalisable de (I.2), $g^* \leq g^{pl}$. Par ailleurs comme nous l'avons mentionné précédemment, pour un choix quelconque des variables $w_{ij}$, la valeur optimale $L(w)$ du sous-problème de Lagrange, représente une borne inférieure pour $g^*$. Du moment que nous calculons à chaque étape les plus courts chemins relatifs aux coûts modifiés $(c_{ij}^k + w_{ij})$, nous mettons à jour pour chaque produit $k$, les plus petites distances $l^k(w)$. La valeur de $L$ est alors donnée par:

$$L(w) = \sum_k l^k(w) - \sum_{(i,j) \in E} w_{ij} b_{ij},$$

et par la théorie de la relaxation lagrangienne,

$$L(w) \leq g^* \leq g^{pl}.$$

Ainsi nous pouvons à chaque étape juger de la qualité de la solution courante, de façon à pouvoir interrompre les itérations pour une différence relative entre $g^{pl}$ et $L(w)$ jugée satisfaisante. Remarquons pour conclure que si la borne supérieure $g^{pl}$ ne peut pas croître strictement, la borne inférieure, quant à elle, tend à osciller de façon très variable. Il serait donc plus judicieux de prendre comme borne inférieure, la plus grande valeur de $L(w)$ pour avoir la meilleure borne à chaque étape.

Pour donner un caractère un peu plus général à cette présentation nous allons tout simplement reconsidérer les questions de bornes supérieures sur les flots individuels et l'unicité des sources et des destinations pour les différents produits. Nous nous apercevons alors que ces hypothèses sont très faciles à mettre en œuvre. En effet, elles interviennent uniquement au niveau des sous-problèmes qui ne sont plus en l'occurrence des problèmes de plus court chemin, mais des problèmes de flot simple de coût minimum. Rappelons alors que ces problèmes de flot simple peuvent être résolus de manière efficace, par de nombreuses techniques que l'on peut trouver dans Srinivasan et Thompson [1973], Glover, Karney, Klingman et Napier [1974], Langley, Kennington et Shetty [1974], Aashtiani et Magnanti [1976], Bradley, Brown et Graves [1977], Mulvey [1978], Kennington et Helgason [1980], ou plus récemment encore, Grigoriadis [1986], Bertsekas et Tseng [1988a], Chang et Chen [1989] et Bazaraa, Jarvis et Sherali [1990].

Pour la note historique, Ford et Fulkerson [1958] ont été les premiers à proposer cette technique de génération de colonnes pour résoudre le problème de multiflot maximal. Il a été établi par Jewell [1966] que leur travail inspira la non moins populaire méthode de décomposition de Dantzig-Wolfe [1960]. Cette méthode a été attractive à plus d'un titre, d'une part, pour sa mise en œuvre d'un formalisme particulier qui a été au cœur des économies planifiées et d'autre part, pour l'utilité de l'algorithme qui en a découlé et son applicabilité dans de nombreux contextes. Nous reviendrons plus en détails sur cette méthode dans le paragraphe suivant, pour analyser l'impact de la formulation sur l'efficacité de cette méthode. En attendant, précisons simplement que l'idée est la suivante: imaginons que pour résoudre notre problème de $K$ flots, nous faisons appel à $K$ décideurs et à un élément coordinateur. Le rôle du coordinateur consisterait à résoudre le problème de multiflot (I.2) en se restreignant à un sous-ensemble de colonnes disponibles; le problème ainsi tronqué sera désigné par problème maître. Le coordinateur transmettrait alors des informations sous forme de prix aux $K$ décideurs. Ces derniers vont à leur tour, soit fournir d'autres colonnes "profitables" (en résolvant des problèmes de flot simple), soit prouver l'optimalité de la solution courante. Comparée à la technique de relaxation

lagrangienne, cette méthode converge en moins d'itérations. Cependant il est beaucoup plus coûteux en temps de mettre à jour les variables prix en résolvant un programme linéaire, qu'en utilisant une méthode de sous-gradient, d'où une lenteur caractéristique. Néanmoins la méthode de décomposition de Dantzig-wolfe, à l'image de la technique de génération de colonnes, offre le double avantage de maintenir à chaque itération une solution réalisable ainsi qu'une garantie de qualité de la solution en termes de "distance" à l'optimum.

Plus tard, Tomlin [1966a] appliqua la technique de génération de colonnes de Ford et Fulkerson au cas du multiflot de coût minimum, en même temps qu'il établit une spécialisation de la méthode de Dantzig-Wolfe dans le cas d'une formulation arcs-chemins. Ce travail lui permettra de mettre en valeur l'équivalence entre ces deux formulations. On retrouvera des résultats similaires dans Bradley [1965] et Jarvis [1969].

Les généralisations du problème de multiflot de coût minimum ont fait l'objet de nombreuses procédures de décomposition par les prix, qu'on peut trouver dans Cremeans, Smith et Tyndall [1970], Swoveland [1971], Swoveland [1973], Weigel et Cremeans [1972], Wollmer [1972] et Chen et Dewald [1974]. Ces généralisations concernent notamment les contraintes de capacité sur les nœuds, ou la prise en compte de ressources quelconques.

Par ailleurs précisons que nous pouvons combiner les principes de relaxation lagrangienne et de génération de colonnes. Bazaraa (article non daté) présente à cet effet, une spécialisation de la méthode de Balas [1966] pour le multiflot maximal. Les sous-problèmes engendrés sont des problèmes de flot maximal et la mise à jour des variables duales se fait par l'intermédiaire d'un programme linéaire résolu grâce à une technique de génération de colonnes.

II.1.2.1.3. Effet de la formulation sur la décomposition par les prix de Dantzig-Wolfe

Nous cherchons ici à mettre en évidence l'impact de la formulation d'un problème de multiflot sur sa résolution par la méthode de décomposition de Dantzig-Wolfe. Comme nous l'avons vu précédemment, les problèmes de multiflots peuvent se formuler de trois manières différentes: avec des origines et destinations spécifiques, des destinations (origines) spécifiques et des produits spécifiques. En pratique, lorsqu'on est face à la prise de décision concernant la formulation la plus appropriée, on choisit celle qui met en œuvre le plus petit nombre de flots, minimisant ainsi la taille du programme linéaire

correspondant. Si on opte pour une résolution du problème par la méthode de Dantzig-Wolfe, il est clair qu'un tel choix de formulation débouche sur une réduction de la taille du problème maître ainsi que du nombre de sous-problèmes à résoudre. Jones et Lustig, Farvolden et Powell [1993] montrent qu'une telle formulation aussi compacte soit elle, débouche paradoxalement sur un problème plus complexe, en impliquant cette lenteur de convergence caractéristique de la méthode. Les motivations de cette hypothèse sont de deux types: premièrement il existe moins de points extrêmes dans la formulation en chemins que dans la formulation en arbres. Naturellement ceci dépend de la structure du réseau, mais quand ceci est le cas, la formulation chemin, surclasse la formulation arbre. Deuxièmement, la décomposition de la structure d'arbre en structure chemin permet à la méthode de tenir compte d'un plus grand nombre de solutions réalisables dans le domaine du problème maître; ainsi par exemple, dans un réseau quelconque, un chemin ayant un coût de transport faible et une grande capacité, doit vraisemblablement faire passer un flot positif. Cependant si ce chemin est groupé dans une structure d'arbre, il est possible que le flot sur ce chemin soit inhibé par quelque contrainte et le coût de l'arbre dans sa globalité. Cette hypothèse de meilleure convergence via une formulation origine-destination spécifique est également étayée par de nombreux tests numériques.

II.1.2.1.3.1. Formulations et décomposition

Nous allons envisager trois formulations en fonction de trois définitions uniques des produits. Dans la première formulation, un produit est défini comme étant un flot de marchandise circulant entre une origine spécifique et une destination spécifique, ce problème sera noté *problème origine-destination (POD)*. Dans cette formulation $\kappa$ représente le triplet *(k, s, t)*, tel que $k$ représente la marchandise $(1 \leq k \leq K)$, $s \in S$ est l'origine spécifique et $t \in T$ correspond à la destination spécifique. Le *(POD)* peut représenter un problème d'ordonnancement d'équipage où l'identité d'un membre $k$ de l'équipage doit être maintenue pendant que l'on cherche à satisfaire les contraintes d'origine et de destination. Dans la deuxième formulation, nous définissons un produit comme étant un flot de marchandise entre une origine spécifique et des destinations multiples (ou de façon symétrique, avec des origines multiples et une destination spécifique); le produit $\kappa$ sera représenté par la paire *(k, t)* identifiant la marchandise $k$ et sa destination spécifique $t$, et le problème associé sera appelé *problème destination spécifique (PDS)*. Le *(PDS)* se rencontre typiquement dans les problèmes d'affectation de trafic, où $K$ véhicules doivent être orientés à travers un réseau, à partir de différentes origines à une destination commune $t$. Dans la troisième formulation, un produit est défini comme une

marchandise circulant entre de multiples origines et destinations. Le produit κ sera représenté par le singleton $k$ désignant la marchandise et le problème correspondant sera noté *(PPS): problème produit spécifique*. Le *(PPS)* permet de modéliser des problèmes d'allocation pour une multiflotte où un équipement spécifique doit pourvoir à un ensemble de services de transport.

$$\mathrm{K} = \begin{cases} \left\{(k,s,t): d^{\kappa} = d^{k}_{st} \neq 0\right\} = \mathrm{K}(POD) & \text{pour } (POD), \\ \left\{(k,t): d^{\kappa} = d^{k}_{t} \neq 0\right\} = \mathrm{K}(PDS) & \text{pour } (PDS), \\ \left\{k: d^{\kappa} = d^{k} \neq 0\right\} = \mathrm{K}(PPS) & \text{pour } (PPS). \end{cases} \tag{II.5}$$

La différence entre les formulations touche au nombre de blocs |K| et à la densité du vecteur second membre $d^{\kappa}$; ces différences sont notées dans les formules (II.5) où l'on voit que *(POD)* possède potentiellement plus de blocs que *(PDS)* et *(PPS)*. Le vecteur second membre $d^{\kappa}$ varie de façon significative à travers les différentes formulations. Ceci peut se voir en examinant le nombre d'éléments différents de zéro dans chaque $d^{\kappa}$: il existe seulement deux éléments non nuls pour *(POD)* correspondant à l'origine et la destination spécifique du produit, contre plusieurs éléments non nuls pour les deux autres problèmes, correspondant aux multiples origines et à la destination spécifique pour *(PDS)*, et aux multiples origines et destinations pour *(PPS)*. En conclusion *(POD)* possède potentiellement plus de blocs que *(PDS)* et *(PPS)*, mais son vecteur second membre est nettement plus creux que celui des deux autres formulations. Cette constatation va jouer un rôle prépondérant dans la décomposition. De fait, nous en profitons ici pour donner une description générique du problème maître impliqué par le schéma de décomposition de Dantzig-Wolfe. Les trois types de problèmes de multiflot sont basés sur la formulation sommets-arcs donnée en (I.1) et donnent lieu, suite à l'analogie de la matrice des contraintes, au même problème maître. (pour plus de détails voir l'algorithme de Dantzig-Wolfe [1961].)

Soit $\mathbf{J}^{\kappa}$ le nombre total de colonnes dans le problème maître restreint associé à l'élément κ, à l'étape courante. Par conséquent le problème maître restreint peut s'écrire comme suit:

$$\text{Minimiser} \sum_{\kappa \in \mathrm{K}} \sum_{J \in \mathbf{J}^{\kappa}} ((c^{\kappa})^{T} v^{\kappa}_{J}) \lambda^{\kappa}_{J} \tag{II.6a}$$

sous les contraintes

$$\sum_{\kappa \in \mathrm{K}} \sum_{J \in \mathbf{J}^{\kappa}} (\hat{I}^{\kappa} v^{\kappa}_{J}) \lambda^{\kappa}_{J} \leq b, \tag{II.6b}$$

$$H\lambda = e, \tag{II.6c}$$

$$\lambda_J^\kappa \geq 0 \quad \forall J \in \mathbf{J}^\kappa,\ \kappa \in \mathrm{K}. \tag{II.6d}$$

Où $v_J^\kappa$ est le *Jème* point extrême, solution du sous-problème de flot simple d'indice κ. Notons que $\hat{I}^\kappa$ est une matrice diagonale définie comme suit:

$$\hat{I}_{ii}^\kappa = \begin{cases} 1 \text{ si l' arc est couplé} \\ 0 \text{ sinon.} \end{cases}$$

Le vecteur $b$ est le vecteur capacité associé aux arcs couplés, $e$ est un vecteur unitaire représentant les contraintes de convexité. La matrice $H$ est une matrice bloc-diagonale de $e$-vecteurs, telle que:

$$H = \begin{bmatrix} (e^1)^T & & & \\ & (e^2)^T & & \\ & & \ddots & \\ & & & (e^{|\mathrm{K}|})^T \end{bmatrix}$$

le nombre d'éléments de $e^\kappa$ est égal au nombre de colonnes $\mathbf{J}^\kappa$ associées au sous-problème κ. A chaque itération de l'algorithme de décomposition de Dantzig-Wolfe un sous-problème de flot simple est résolu pour chaque κ:

$$\text{Minimiser } (c^\kappa - \hat{I}^\kappa w)^T v^\kappa \tag{II.7a}$$

sous les contraintes

$$Av^\kappa = d^\kappa, \tag{II.7b}$$

$$v_{ij}^\kappa \geq 0 \quad \forall (i,j) \in E. \tag{II.7c}$$

Où les éléments $w$ correspondent aux variables duales associées aux contraintes de capacité (II.6b). La solution de ces sous-problèmes de flot simple est déterminée de façon unique relativement à $d^\kappa$. Ainsi pour *(POD)* la solution est *un plus court chemin* entre une origine $s$ et une destination $t$ pour chaque produit $k$. Pour *(PDS)* la solution est *un plus court chemin-arbre* de toutes les sources à la destination. Enfin pour *(PPS)* la solution consiste à trouver une solution arbre (entre les multiples sources et destinations) correspondant à *un problème de flot simple de coût minimum.* Par conséquent il paraît assez probable que le point extrême solution de *(POD)* sera plus creux que celui généré par les deux autres. Par ailleurs comme nous l'avons vu précédemment, l'ensemble K*(POD)* possède

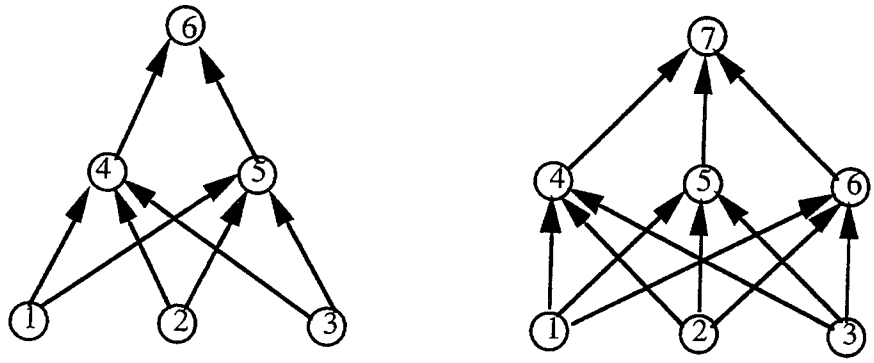
potentiellement plus d'éléments que K*(PDS)* ou K*(PPS)*. Il s'en suit que le nombre de sous-problèmes traités et donc de colonnes rajoutées au problème maître est sensiblement plus grand dans *(POD)* que dans les deux autres problèmes. Jones et Lustig, Farvolden et Powell [1993] soutiennent que pour un certain type de réseau, il y aura au total moins de points extrêmes rajoutés au problème maître de *(POD)* qu'à celui des deux autres, parcequ'il y aura moins d'itérations associées au problème maître. Pour montrer ceci, définissons $\mathbf{J}_t^{\kappa}$ comme le nombre de points extrêmes associés au produit $\kappa$ à l'étape $t$. Soit $\mathbf{J}_t$ le nombre maximum de colonnes associées à l'itération $t$, correspondant à un quelconque élément $\kappa$, i.e.,

$$\mathbf{J}_t = \max_{\kappa \in \mathrm{K}} \left\{ \mathbf{J}_t^{\kappa} \right\}.$$

Par conséquent, pour toutes les formulations, il existe potentiellement $|\mathrm{K}| \times \mathbf{J}_t$ colonnes dans le problème maître à l'itération $t$. L'hypothèse suivante est alors soutenue par Jones et Lutsig, Farvolden et Powell, pour les grands réseaux et spécialement les réseaux dynamiques:

$$|\mathrm{K}(POD)| \times \mathbf{J}_{T_0}(POD) << |\mathrm{K}(PDS)| \times \mathbf{J}_{T_d}(PDS).$$

$T_0$ et $T_d$ représentent le nombre d'itérations à l'optimalité pour *(POD)* et *(PDS)* respectivement. Ceci peut se voir en examinant le nombre total de points extrêmes dans chaque formulation. Le petit exemple de la Figure II.2a (Farvolden [1989]) montre un réseau dynamique avec un horizon à deux périodes. Ce réseau possède 6 chemins possibles contre 8 arbres possibles. Mais une petite extension de cet exemple, telle que présentée dans la Figure II.2b, montre que l'addition d'un seul nœud fait passer le nombre de chemins de 6 à 9, pendant que le nombre d'arbres passe de 8 à 27. Il parait clair que pour un problème impliquant un horizon plus long et un plus grand nombre de nœuds par période, la différence entre le nombre de chemins et d'arbres risque d'être extrêmement importante. On voit donc que malgré un plus grand nombre de sous-problèmes pour *(POD)*, il y aura moins de colonnes au total dans le problème maître correspondant, en comparaison avec *(PDS)*, suite à cet effet combinatoire de chemins et d'arbres.




a. b.

**Figure II.2** Énumération de points extrêmes pour les chemins et les arbres.

II.1.2.1.3.2. Comparaison des formulations

Pour établir et valider les différences entre la formulation chemin de *(POD)* et les formulations arbre de *(PDS)* et *(PPS)*, il faut que ces problèmes soient mathématiquement équivalents. Comme la solution globale est une combinaison convexe de points extrêmes obtenus à partir des sous-problèmes, les problèmes sont équivalents si les sous-problèmes le sont. Pour distinguer les différentes formulations notons $x^k$ le vecteur flot sur les arcs décrivant la solution arbre correspondant à *(PPS)*. $y_t^k$ représente la solution de plus court chemin arbre associée à *(PDS)* et $z_{st}^k$ le vecteur flot sur les arcs relatif à la solution de plus court chemin associée à *(POD)*. Les relations existant entre *(POD)* et *(PDS)* découlent du lemme suivant établi par Rockaffelar [1984]:

***Lemme II.2.*** *Si $y_t^k$ est réalisable pour (PDS) alors il existe des valeurs $z_{st}^k$ $\forall s \in S$, $t \in T$ et $1 \leq k \leq K$ telles que:*

$$y_t^k = \sum_{s \in S} z_{st}^k \qquad \text{(II.8)}.$$

Ce lemme exprime que les flots agrégés en structure d'arbre partant de toutes les sources à la destination, peut être décomposé en un sous-ensemble de flots chemins partant individuellement de ces sources à cette destination. Les corollaires qui suivent sont des conséquences directes de ce lemme.

***Corollaire II.3.*** *Si les coûts sur les arcs du réseau ne dépendent pas des contraintes origine-destination et que $z_{st}^k$ est optimal pour (POD) alors $y_t^k$ défini en* (II.8) *est optimal pour (PDS).*

***Corollaire II.4.*** *Si les coûts sur les arcs sont indépendantes des contraintes origine-destination et si* $y_r^k$ *est optimal pour (PDS) alors une solution optimale* $z_{si}^k$ *pour (POD) peut être construite à partir de* $y_r^k$ *en vérifiant l'équation* (II.8).

Grâce à ces corollaires , on peut considérablement réduire le nombre de sous-problèmes à résoudre pour *(POD)*. En effet, on peut résoudre seulement |K(*PDS*)| sous-problèmes et décomposer ensuite chaque solution optimale en solutions équivalentes pour *(POD)*. Notons cependant que ces résultats ne sont plus applicables entre *(PDS)* et *(PPS)*. Ceci peut être démontré par l'exemple de la Figure II.3 où l'on compare la solution optimale de *(POD)* et la solution *(PPS)*. Cet exemple utilise la matrice demande *(POD)* suivante:

| $D_1$ | 3 | 4 |
|---|---|---|
| 1 | 1 | 0 |
| 2 | 0 | 1 |

Ce problème est résolu en envoyant 1 unité sur chacun des arcs (1,3) et (2,4) avec un coût total de 20. Par contre la formulation *(PPS)* débouche sur une solution de coût 2, en utilisant les arcs (1,4) et (2,3).

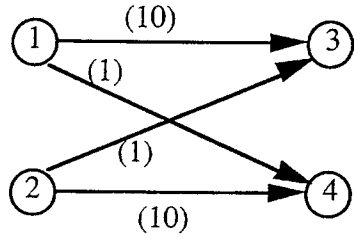



**Figure II.3** Exemple de réseau.

En récapitulation, on constate que la comparaison entre *(POD)* et *(PPS)* peut se faire directement (moyennant l'hypothèse de non spécificité des coûts par rapport aux diverses origines et destinations, ce qui est une hypothèse assez courante dans les problèmes de multiflots.) Par contre la comparaison entre *(POD)* et *(PPS)* est autrement plus complexe et nécessite une transformation de la formulation *(PPS)* en formulation chemins; ce qui fait l'objet du prochain paragraphe.

II.1.2.1.3.3. Extension du problème produit spécifique

Le graphe $G=(V, E)$ peut être augmenté de certains nœuds et arcs afin de créer une formulation en chemins équivalente à *(PPS)*, soit *(EPPS)* cette formulation. Pour ce faire, introduisons $K$ nœuds super-sources $\hat{S}=\{\hat{s}_1, ...,\hat{s}_K\}$ et $K$ nœuds super-demandes $\hat{T}=\{\hat{t}_1, ...,\hat{t}_K\}$ en imposant au flot $k$ de circuler entre une source spécifique $\hat{s}_k$ et une destination spécifique $\hat{t}_k$. Nous sommes alors en présence d'un nouveau graphe $\hat{G}=(\hat{V}, \hat{E})$ où $\hat{V}=V\cup\hat{S}\cup\hat{T}$ et $\hat{E}=E\cup\{(\hat{s},s)\mid \hat{s}\in\hat{S}, s\in S\}\cup\{(t,\hat{t})\mid t\in T, \hat{t}\in\hat{T}\}$. (Voir Figure II.4)

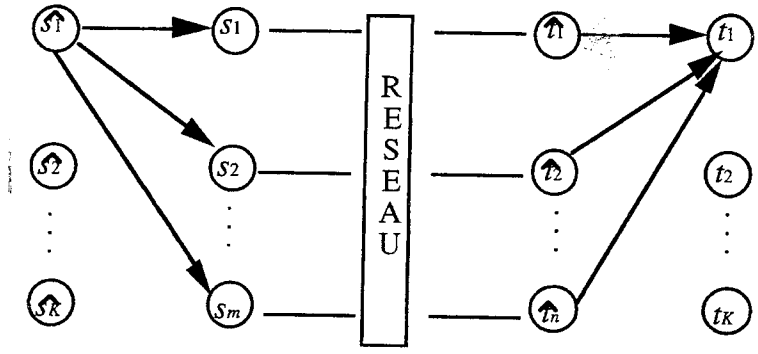



**Figure II.4** Exemple de réseau *(EPPS)*.

Ce nouveau problème est équivalent à *(PPS)* si le flot total $\bar{d}^k$ entre $\hat{s}_k$ et $\hat{t}_k$ est tel que:

$$\bar{d}^k=\sum_{s\in S} d_s^k=\sum_{t\in T} d_t^k,$$

et si le flot sur les nouveaux arcs satisfait les contraintes de ravitaillement et de demande. Notons $d_s^k$ et $d_t^k$ les quantités respectives de marchandise disponible au nœud source $s$ et demandée à la destination $t$. On peut alors envisager le programme linéaire suivant pour résoudre *(EPPS)* :

$$\text{Minimiser } \sum_k c^k x^k \qquad \text{(II.9a)}$$

$$\sum_k x_{ij}^k \le b_{ij} \quad \forall\,(i,j)\in E, \qquad \text{(II.9b)}$$

$$\bar{x}_{\hat{s}_k s}^k \le d_s^k \quad \forall\,\hat{s}\in\hat{S},\ s\in S,\ 1\le k\le K, \qquad \text{(II.9c)}$$

$$\bar{x}_{s\hat{t}_k}^k \le d_t^k \quad \forall\,t\in T,\ \hat{t}\in\hat{T},\ 1\le k\le K, \qquad \text{(II.9d)}$$

$$Ax^k - \bar{x}^k_{\hat{s}_k \cdot} + \bar{x}^k_{\cdot \hat{t}_k} = 0 \quad \forall\, 1 \le k \le K, \tag{II.9e}$$

$$\sum_{s \in S} \bar{x}^k_{\hat{s}_k s} = \bar{d}^k \quad \forall\, 1 \le k \le K,\ \hat{s}_k \in \hat{S}, \tag{II.9f}$$

$$\sum_{t \in T} \bar{x}^k_{t \hat{t}_k} = -\bar{d}^k \quad \forall\, 1 \le k \le K,\ \hat{t}_k \in \hat{T}, \tag{II.9g}$$

$$x^k_{ij} \ge 0 \qquad \forall\, (i,j) \in E,\ 1 \le k \le K, \tag{II.9h}$$

$$\bar{x}^k_{\hat{s}_k s} \ge 0 \qquad \forall\, \hat{s} \in \hat{S},\ s \in S,\ 1 \le k \le K, \tag{II.9i}$$

$$\bar{x}^k_{t \hat{t}_k} \ge 0 \qquad \forall\, t \in T,\ \hat{t} \in \hat{T},\ 1 \le k \le K. \tag{II.9j}$$

Dans cette formulation les notations $\bar{x}^k_{\hat{s}_k \cdot}$ et $\bar{x}^k_{\cdot \hat{t}_k}$ indiquent les flots entrant respectivement dans chaque nœud $s$ en provenance de chaque nœud $\hat{s}_k \in \hat{S}$, dans chaque nœud $\hat{t}_k \in \hat{T}$ en provenance de chaque nœud $t$. Les contraintes de conservation de flot prennent alors la forme (II.9e). Les contraintes de capacité sur les nouveaux arcs sont exprimées par (II.9c) et (II.9d). Il est clair qu'à l'optimum ces contraintes seront satisfaites à égalité. La matrice d'incidence sommets-arcs du graphe augmenté est représentée par les trois dernières égalités. Comme dans *(POD)*, le second membre comporte deux valeurs non nulles, la différence est due aux contraintes de bornes supérieures sur $\bar{x}^k_{\hat{s}_k s}$ et $\bar{x}^k_{t \hat{t}_k}$.

Lors de la décomposition de *(EPPS)* les contraintes additionnelles peuvent être incorporées soit dans le problème maître soit dans les sous-problèmes. Si elles sont incorporées dans les sous-problèmes, la formulation du problème maître est identique à (II.6) pendant que les sous-problèmes imposent des bornes supérieures et inférieures sur les arcs additionnels de façon à ce que les flots sur ces arcs soient *fixés*. Dans ce cas la solution des sous-problèmes *(EPPS)* est identique à celle des sous-problèmes *(PPS)* et par conséquent il n'y a aucun avantage à le reformuler en une formulation basée sur les chemins. D'un autre côté si les contraintes de flot sur les arcs $(\hat{s}_k, s)$ et $(t, \hat{t}_k)$ sont prises en compte dans le problème maître, la solution des sous-problèmes est identique à celle de *(POD)*, en ce sens que la solution consiste en un chemin unique entre une super-source et une super-destination spécifiques. Comme il n'existe pas de contraintes de capacité individuelles dans les sous-problèmes, il est important de noter que la solution est un chemin unique *irréalisable* pour le problème linéaire initial. De même, vu qu'il existe un flot extérieur $\bar{d}^k$ au nœud super-source $\hat{s}_k$ dans les sous-problèmes résolus, la solution doit nécessairement envoyer le flot total $\bar{d}^k$ de $\hat{s}_k$ à un nœud source unique s. Ceci est un flot irréalisable dans le problème

original *(PPS)* qui est rectifié dans le problème maître en forçant la réalisabilité, par les contraintes de capacité des arcs adjacents à tous les nœuds super-source et super-demande. Même si les contraintes sur ces arcs sont prises en compte sous forme d'inégalité dans le problème maître restreint de *(EPPS)*, la procédure de décomposition les satisfait à égalité (conditions sur le flot total au niveau des sous-problèmes). Par conséquent si une solution réalisable est obtenue pour ce problème, elle satisfait les contraintes de demande et de ravitaillement dans la formulation *(PPS)*. Il reste que la difficulté de cette formulation réside dans l'obtention d'une solution initiale réalisable. Pour qu'une solution soit réalisable pour chaque produit, il faut générer dans chaque sous-problème autant de chemins qu'il en faut pour "toucher" tous les nœuds ravitaillement-demande originaux. Considérons, par exemple, un biflot sur un réseau à 3 nœuds ravitaillement, 3 nœuds demande et 2 nœuds intermédiaires. La Figure II.5 donne la solution pour le premier flot indiquée en gras.

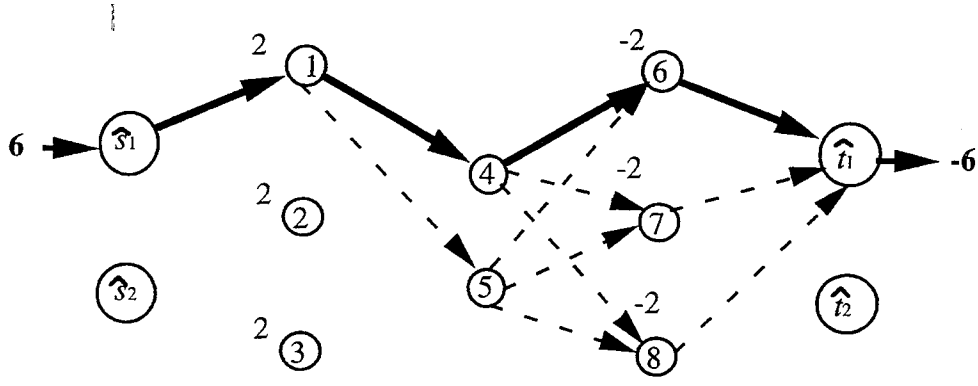



**Figure II.5** Exemple de sous-problème *(EPPS)*.

Cette solution est incluse dans le problème maître sous forme de colonne pour le chemin $\hat{s}_1 \rightarrow \hat{t}_1$. Cette colonne contiendra deux valeurs non nulles dans les contraintes de couplage du problème maître associées aux arcs $(\hat{s}_1, 1)$ et $(6, \hat{t}_1)$. Cependant, pour satisfaire toutes les contraintes ravitaillement-demande (pour le premier produit), le problème maître doit également inclure des colonnes avec des éléments non nuls associés aux arcs $(\hat{s}_1, 2)$, $(\hat{s}_1, 3)$, $(7, \hat{t}_1)$ et $(8, \hat{t}_1)$. Pour satisfaire ces contraintes, le sous-problème associé au produit 1, doit être résolu au moins 3 fois pour générer les chemins:

$$\hat{s}_1 \rightarrow 1 \rightarrow 4 \rightarrow 6 \rightarrow \hat{t}_1,$$

$$\hat{s}_1 \rightarrow 2 \rightarrow 4 \rightarrow 7 \rightarrow \hat{t}_1,$$

$$\hat{s}_1 \rightarrow 3 \rightarrow 5 \rightarrow 8 \rightarrow \hat{t}_1.$$

Mais il existe 18 chemins susceptibles d'être choisis par le sous-problème. Ceci veut dire que le sous-problème doit être résolu de 3 à 18 fois, et ce seulement pour le produit 1. Par conséquent, même pour un petit exemple, l'obtention d'une solution initiale réalisable peut prendre beaucoup de temps. La question qu'on se pose alors est la suivante: est ce que la rapidité de convergence caractéristique d'une formulation basée sur les chemins suffit à contrebalancer les difficultés dues à l'obtention d'une solution initiale réalisable La réponse à cette question ainsi que les résultats obtenus dans le cadre de cette analyse, font l'objet du paragraphe suivant.

II.1.2.1.3.4. Résultats obtenus

Des tests numériques ont été effectués par Jones et Lustig, Farvolden et Powell [1993] sur plusieurs ensembles de problèmes test issus de la littérature. Les résultats obtenus soutiennent l'hypothèse que la décomposition appliquée à des structures plus importantes mais plus creuses, associées à des formulations basées sur les chemins telles que *(POD)* et *(EPPS)*, donne lieu à une meilleure convergence par rapport aux formulations *(PDS)* et *(PPS)* respectivement. La reformulation de *(PDS)* en *(POD)* entraîne une légère augmentation du nombre de contraintes du problème maître, comparée à l'augmentation du nombre de sous-problèmes traités à chaque itération. Simplement il faut mettre l'accent sur le fait que même si un plus grand nombre de sous-problèmes sont résolus à chaque itération, il y a moins de points extrêmes énumérés dans cette formulation. De plus les sous-problèmes sont plus faciles à résoudre en tant que problèmes de plus court chemin. D'autre part, cet inconvénient peut être évité en résolvant les sous-problèmes correspondants en formulation arbre, puis en décomposant la solution en chemins.

La puissance des formulations en chemins est plus notable dans la formulation *(EPPS)*. Dans ce cas il y a un léger accroissement dans le nombre de lignes du problème maître avec une complication supplémentaire due aux contraintes de capacité, le nombre de sous-problèmes restant le même. Mais même si cette reformulation de *(PPS)* entraîne une augmentation du temps requis pour obtenir une solution réalisable, la performance globale de *(EPPS)* s'avère meilleure que pour *(PPS)*.

### II.1.2.2. Décomposition par les ressources

Robacker [1956] suggère une décomposition par les ressources pour résoudre les problèmes de multiflots, sans spécifier un quelconque schéma d'implémentation. En général cette approche consiste à déterminer une allocation de la capacité sur chaque arc,

pour tous les flots, de façon à ce que l'optimum du problème global soit atteint à travers la résolution de $K$ sous-problèmes de flot simple. La somme des capacités allouées à chaque flot sur un arc doit être inférieure ou égale à la capacité de cet arc. Il s'ensuit que les flots individuels de chaque sous-problème constituent un multiflot réalisable pour le problème initial. Appliquons cette approche au problème (I.1) en allouant au produit $k$, $f_{ij}^k \leq u_{ij}^k$ unités de la capacité $b_{ij}$ de l'arc $(i, j)$. Il en découle le problème suivant:

$$z = \min \sum_k c^k x^k \tag{II.10a}$$

sous les contraintes

$$\sum_k f_{ij}^k \leq b_{ij} \quad \forall\, (i, j) \in E, \tag{II.10b}$$

$$Ax^k = r^k \quad \forall\, k = 1,\ K, \tag{II.10c}$$

$$0 \leq x_{ij}^k \leq f_{ij}^k \quad \forall\, (i, j) \in E,\ \forall\, k = 1,\ K. \tag{II.10d}$$

Où $c^k$ et $x^k$ représentent respectivement les vecteurs coût et flot sur les arcs, pour le produit $k$. Notons $f = (f_{ij}^k)$ le vecteur allocation de la capacité et faisons quelques observations sur le problème ainsi défini.

***Propriété II.5.*** *Le problème* (II.10) *est équivalent au problème de multiflot original* (I.1) *en ce sens que: (1) si (x, f) est une solution réalisable pour* (II.10), *alors x est réalisable de* (I.1) *et produit la même valeur de la fonction objective; (2) si x est réalisable de* (I.1) *alors en prenant x = f, (x, f) est réalisable dans le problème* (II.10) *et produit la même valeur de la fonction objective.*

Imaginons maintenant une approche séquentielle pour résoudre le problème (II.10). En effet au lieu de résoudre le problème en choisissant $f$ et $x$ simultanément, on peut le faire séquentiellement. Pour cela, fixons en premier lieu une allocation de capacité $f_{ij}^k$ et déterminons ensuite les flots $x_{ij}^k$. Désignons par $z(f)$ la valeur optimale du problème (II.10) pour une valeur fixée du vecteur allocation $f$, et considérons le problème d'allocation dérivé:

$$\text{Minimiser } z(f) \tag{II.11a}$$

sous les contraintes

$$\sum_k f_{ij}^k \leq b_{ij} \quad \forall\, (i, j) \in E, \tag{II.11b}$$

$$0 \leq f_{ij}^k \leq u_{ij}^k \quad \forall\, (i, j) \in E,\ \forall\, k = 1,\ K. \tag{II.11c}$$

La fonction objective de ce problème est connue implicitement, comme solution d'un problème d'optimisation en variables $x_{ij}^k$. Remarquons alors que pour des valeurs quelconques des variables allocation $f_{ij}^k$, le problème (II.10) se décompose en $K$ sous-problèmes de flot simple de valeur $z^k(f^k)$. Dans ce cas $z(f)=\sum_k z^k(f^k)$, avec $z^k(f^k)$ défini comme suit:

$$z^k(f^k)=\min c^k x^k \tag{II.12a}$$

$$Ax^k=r^k \quad \forall k=1,\ K, \tag{II.12b}$$

$$0\le x_{ij}^k\le f_{ij}^k \ \ \forall\ (i,\ j)\in E,\ \forall\ k=1,\ K. \tag{II.12c}$$

***Propriété II.6.*** *Le problème* (II.10) *est équivalent au problème d'allocation* (II.11) *en ce sens que: (1) si (x, f) est réalisable de* (II.10) *alors f est réalisable dans le problème d'allocation* (II.11) *et* $z(f)\le cx$, *(2) si f est réalisable pour le problème d'allocation* (II.11) *alors il existe un vecteur x tel que (x, f) est réalisable du problème original et* $cx=z(f)$.

*Preuve.* Si $(x,\ f)=(x^1,\ x^2,\ ...,\ x^K,\ f^1,\ f^2,\ ...,\ f^K)$ est réalisable de (II.10) alors $f$ est réalisable pour (II.11). De plus si $x^k$ est seulement réalisable pour chacun des sous-problèmes correspondants (II.7), $z^k(f^k)\le c^k x^k$. Par conséquent $z(f)=\sum_k z^k(f^k)\le\sum_k c^k x^k=cx$. Réciproquement si $f$ est réalisable dans le problème (II.6), alors par définition $z^k(f^k)=c^k x^k$ pour le vecteur $x^k$, d'où $(x^1,\ x^2,\ ...,\ x^K)$ satisfait la condition $cx=z(f)$.

Les propriétés (II.5) et (II.6) impliquent qu'au lieu de résoudre directement le problème de multiflot, nous pouvons le ramener à un problème d'allocation dont la structure des contraintes est très simple, mais avec une fonction objective complexe. Nous verrons d'ailleurs que cette fonction peut être évaluée assez simplement et de plusieurs manières différentes, à travers la résolution de $K$ problèmes de flot simple. Reprenons d'abord l'écriture du problème (II.11) sous une forme matricielle plus compacte, en rajoutant des variables d'écart aux contraintes (II.11b), il vient:

$$\text{Minimiser } z(f)=\sum_k z^k(f^k) \tag{II.11'a}$$

$$\sum_k f^k+s=b, \tag{II.11'b}$$

$$0\le f^k\le u^k \quad \forall\ k=1,\ K, \tag{II.11'c}$$

$$s\ge 0. \tag{II.11'd}$$

Avec:

$$z^k(f^k) = \min\left\{c^k x^k \middle| Ax^k = r^k,\ 0 \le x^k \le f^k\right\}$$

et par dualité, pour $\mu^k$, $v^k$ variables duales associées respectivement aux contraintes de conservation de flot et de capacité:

$$z^k(f^k) = \max\left\{r^k\mu^k - f^k v^k \middle| \mu^k A - v^k \le c^k,\ v^k \ge 0\right\}.$$

Les techniques de décomposition par les ressources différent par la manière dont le problème (II.11) est résolu. Plusieurs approches ont été proposées dans la littérature, nous décrirons ici deux techniques essentielles représentées respectivement par la méthode d'approximation tangentielle et la méthode de sous-gradient.

II.1.2.2.1. Approximation Tangentielle

Soit $R_k = \left\{\mu^k,\ v^k \middle|\ \mu^k A - v^k \le c^k,\ v^k \ge 0 \text{ et } (\mu^k,\ v^k) \text{ point extrême}\right\}$, alors le problème (II.11') est équivalent au problème suivant:

$$\text{Minimiser} \sum_k \beta^k \qquad \text{(II.13a)}$$

sous les contraintes

$$\beta^k \ge r^k\mu^k - f^k v^k \qquad \forall\ (\mu^k, v^k) \in R_k,\ k = 1,\ K, \qquad \text{(II.13b)}$$

$$\sum_k f^k + s = b, \qquad \text{(II.13c)}$$

$$0 \le f^k \le u^k \quad \forall\ k = 1,\ K, \qquad \text{(II.13d)}$$

$$s \ge 0. \qquad \text{(II.13e)}$$

Supposons que $Q_k \subset R_k$; désignons par $z(R)$ la valeur optimale de la fonction objective du problème (II.13) et par $z(Q)$ la valeur optimale du même problème en remplaçant $R_k$ par $Q_k$ dans les contraintes (II.13b). Alors $z(Q) \le z(R)$ est une borne inférieure pour le problème (II.10). L'algorithme qui en découle prend la forme suivante:

*Algorithme de décomposition par les ressources utilisant l'approximation tangentielle*

*Pas 0 : Initialisations*

Posons $t = 0$ ($t$, compteur d'itérations.)

$\beta^k = -\infty$, $Q_k = \varnothing$.

Soit $f_0 = (f_0^1, \ldots, f_0^K)$, un élément de $\left\{(f^1, \ldots, f^K) \middle| \sum_k f^k \leq b,\ 0 \leq f^k \leq u^k\right\}$.

*Pas 1 : Résolution des sous-problèmes (détermination d'une borne supérieure)*

Résoudre pour chaque $k=1, K$, le sous-problème de flot simple:

$$\begin{aligned} z^k(f_t^k) &= \min c^k x_t^k \\ Ax_t^k &= r^k \quad (\mu_t^k) \\ x_t^k &\leq f_t^k \quad (v_t^k) \\ x_t^k &\geq 0. \end{aligned}$$

*Pas 2 : Test d'optimalité*

Calculer les bornes supérieure et inférieure *BS* et *BI* respectivement par:

$$BS = \sum_k z^k(f_t^k) \quad \text{et} \quad BI = \sum_k \beta_k.$$

Si $BS = BI$, arrêter on est à l'optimum. Sinon rajouter $(\mu_t^k, v_t^k)$ à $Q_k$ et aller au pas 3.

*Pas 3 : Résolution du problème maître*

Poser $t = t+ 1$ et résoudre le problème maître:

$$\text{Minimiser} \sum_k \beta_k$$

sous les contraintes

$$\begin{aligned} \beta^k \geq r^k \bar{\mu}^k - f_t^k \bar{v}^k \quad & \forall\, (\bar{\mu}^k, \bar{v}^k) \in Q_k,\ k = 1, K, \\ \sum_k f_t^k + s &= b, \\ 0 \leq f_t^k &\leq u^k, \\ s &\geq 0, \end{aligned}$$

et revenir au pas 1.

Les $K$ sous-problèmes impliqués au pas 1 sont des problèmes de flot simple et peuvent être résolus par de nombreuses techniques efficaces (voir Srinivasan et Thompson [1973], Glover, Karney, Klingman et Napier [1974], Langley, Kennington et Shetty [1974], Aashtiani et Magnanti [1976], Bradley, Brown et Graves [1977], Mulvey [1978], Kennington et Helgason [1980], ou plus récemment encore, Grigoriadis [1986], Bertsekas

et Tseng [1988a], Chang et Chen [1989] et Bazaraa, Jarvis et Sherali [1990]). De plus, les seuls changements dans les opérations successives de pivotage, touchent uniquement les bornes supérieures; il s'ensuit que ces programmes peuvent tirer avantage d'une technique duale simplexe lors de leur réoptimisation. Par ailleurs le problème maître possède une structure bloc-angulaire pouvant être exploitée grâce à une technique spécialisée du simplexe, connue sous le nom de *Generalized upper bounding*. Soweveland [1971] implémenta cette procédure sans tenir compte de cette structure particulière du problème maître. Son code de décomposition par les ressources s'avéra alors moins performant que son code de décomposition par les prix. De fait, une approche similaire tenant compte de cet avantage, appliquée par Géoffrion et Graves [1974] à un problème de multiflot sur la localisation de dépôts, enregistra un plus grand succès.

II.1.2.2.2. Méthode de sous-gradient

Une autre manière d'interpréter la fonction $z(f)$ serait de la considérer comme le coût du programme linéaire (II.10), fonction des paramètres du second membre $f$. Un résultat connu de la programmation linéaire montre alors que la fonction $z(f)$ est une fonction convexe linéaire par morceaux (cf. par exemple Lasdon [1970] chap. 9). Une approche naturelle pour résoudre le problème d'allocation serait alors d'utiliser une méthode de sous-gradient. Pour ce faire, il faut trouver une direction de déplacement correspondant à un sous-gradient, et un pas de déplacement permettant de maintenir la faisabilité et d'assurer la convergence vers une allocation optimale de la capacité. L'approche adoptée procède en deux étapes: (1) trouver un sous-gradient $\gamma$ de la fonction $z(f)$ et un pas de déplacement assurant la convergence, (2) transformer toute nouvelle allocation $\overline{f} = f + t_n \gamma$ irréalisable (i.e. $\sum_k f_{ij}^k > b_{ij}$ ) en une allocation réalisable. Pour ce qui est de l'étape 1, rappelons qu'un sous-gradient $\gamma$ de $z(f)$ au point $f = \overline{f}$ est un vecteur satisfaisant la condition:

$$z(f) \geq z(\overline{f}) + \gamma(f - \overline{f}) \text{ pour tout } f = (f^1, f^2, \ldots, f^K), \text{ avec } f^k \in F^k.$$

Dans cette expression, $F^k$ est l'ensemble de toutes les allocations possibles associées au produit $k$, pour lesquelles le problème (II.11) est réalisable. La propriété suivante montre que trouver un sous-gradient de la fonction $z(f)$ revient à trouver indépendamment les sous-gradients des fonctions constituantes $z^k(f^k)$.

***Propriété II.7.*** *Soit $\gamma^k$ un sous-gradient de $z^k(f^k)$ au point $\overline{f}^k$ pour $k = 1, K$. Alors $\gamma = (\gamma^1, \gamma^2, \ldots, \gamma^K)$ est un sous-gradient de $z(f)$ au point $\overline{f} = (\overline{f}^1, \overline{f}^2, \ldots, \overline{f}^K)$.*

*Preuve.* Comme $\gamma^k$ est un sous-gradient de $z^k(f^k)$ on a:

$$z^k(f^k) \geq z^k(\overline{f}^k) + \gamma^k(f^k - \overline{f}^k) \quad \forall f^k \in F^k.$$

En sommant ces expressions sur $k$ et en utilisant le fait que $z(f) = \sum_k z^k(f^k)$ et $\gamma(f - \overline{f}) = \sum_k \gamma^k(f^k - \overline{f}^k)$, on démontre le résultat énoncé.

Ce résultat montre que trouver un sous-gradient de la fonction $z$ en un point $f$, requiert des informations sur les problèmes de flot simple définis en (II.12). Fort heureusement ces informations sont le produit de toute procédure de résolution de ces problèmes: pour $k = 1, K$ notons $v^k = (v_{ij}^k)_{(i,j)\in E}$ les variables duales associées aux contraintes $x^k \leq f^k$ (tensions) à l'optimum de (II.12), alors le vecteur $v = (v^k)_{k=1, K}$ est un sous-gradient de $z$ en $f$. (Voir Lasdon [1970] ou Géoffrion [1970].)

Une fois la direction de déplacement choisie, on s'intéresse au pas de déplacement. Le pas de déplacement classiquement utilisé dans les techniques de sous-gradient est:

$$t_n = \rho_n [z(f) - z^*] / \gamma\gamma.$$

Où $\rho_n$ est une série décroissante (avec $0 \leq \rho_n \leq 2$), $z^*$ est une borne inférieure du problème (II.11) et $\gamma$ un sous-gradient de $z(f)$ en $f$. La convergence de cette procédure est assurée lorsque $z^*$ est la valeur optimale de la fonction objective. Cependant comme cette valeur n'est généralement pas disponible, à priori, on utilise une valeur sous-estimée de $z^*$. Avec tous ces éléments nous pouvons trouver la nouvelle allocation $f = f + t_n \gamma$. Si le vecteur $f$ est réalisable ($\sum_k f_{ij}^k \leq b_{ij}$) nous adoptons ce nouveau point, sinon on enclenche la deuxième étape. Dans cette étape, on essaie de trouver un point $\overline{f}$ (le plus proche possible de $f$) en faisant une projection de la nouvelle allocation sur la région réalisable $\left\{ f^1, \ldots, f^K \middle| \sum_k f^k \leq b, f \geq 0 \right\}$. Mathématiquement cela revient à résoudre le problème d'optimisation suivant:

$$\text{Minimiser} \sum_k \sum_{(i,j)\in E} (f_{ij}^k - \overline{f}_{ij}^k)^2$$

$$\sum_k f_{ij}^k \leq b_{ij} \quad \forall k = 1, K,$$

$$f_{ij}^k \geq 0 \quad \forall (i, j) \in E, \forall k = 1, K.$$

Ce problème se décompose en $M$ (nombre d'arcs) programmes quadratiques indépendants pour lesquels on détermine la solution optimale de façon explicite, en utilisant les

conditions de Khün et Tucker. L'algorithme d'optimisation par la méthode de sous-gradient prend alors la forme générale suivante:

*Algorithme de décomposition par les ressources utilisant une technique de sous-gradient*

*Pas 0 : Initialisations*

Fixer $\varepsilon > 0$ (pour évaluer la qualité de la solution) ainsi qu'en entier $T > 0$ désignant le nombre maximum d'allocations à calculer.
Déterminer une allocation réalisable $f_0 = (f_0^1, \ldots, f_0^K)$ et une borne inférieure $BI$ (on pourra prendre $BI = \sum_k z^k(b)$ en résolvant (II.12) avec $f^k = b$).
Poser $t$ = 1 (compteur d'itérations) et $z^* = \infty$.

*Pas 1 : Résolution des sous-problèmes*

Résoudre pour tout $k$, le problème de flot simple:

$$z^k = \min \left\{ c^k x^k \middle| Ax^k = r^k,\ 0 \le x^k \le f^k \right\}.$$

Si $\sum_k z^k \le z^*$ stocker la solution, poser $z^* = \sum_k z^k$ et aller au pas 2. Sinon aller au pas 3.

*Pas 2 : Test d'arrêt*

Si $\frac{(z^* - BI)}{BI} \le \varepsilon$ ou $t = T$ arrêter, sinon aller au pas 3.

*Pas 3 : Nouvelle allocation*

Poser $t = t + 1$, déterminer une nouvelle allocation réalisable et retourner au pas 1.

L'idée d'appliquer un algorithme de sous-gradient pour minimaliser la fonction $z(f)$ apparaît dans Held Wolfe et Crowder [1974] dans le cas du problème de multiflot maximum. La première mise en œuvre de la méthode est due à Kennington et Shalaby [1977] qui proposent une extension pour résoudre le problème de multiflot de coût minimum. Cependant, ces auteurs se contentent d'utiliser le minorant $\sum_k z^k(b)$, lequel peut être très éloigné de l'optimum. Une approche similaire est également proposée par Gratzer et Steiglitz [1972]. Minoux [1977] présente une amélioration par rapport à Kennington et Shalaby, en utilisant dans son algorithme un meilleur minorant (comme solution approchée du problème dual, obtenue également par une méthode de sous-gradient). Ceci lui permet de réaliser un meilleur encadrement de la solution optimale et d'accélérer la convergence de l'algorithme de sous-gradient. Il existe d'autres méthodes

pour le calcul des bornes inférieures, notamment celle de Zangwill [1969a] et celle de Gersht et Schulman [1987]. Assad [1980b] présente également une approche de décomposition par les ressources basée sur une méthode de sous-gradient.

Avant de clôturer cette discussion, précisons que toutes ces approches utilisent une direction de déplacement correspondant à un sous-gradient de la fonction $z(f)$. Mais toujours étant donnée la convexité de cette fonction, on pourrait tout aussi bien utiliser la "meilleure direction", au lieu d'un simple sous-gradient. D'un point de vue théorique ceci peut se faire grâce à des méthodes de montée duale (voir Grinold [1972] et Lasdon [1970]). Il reste que, sur un plan pratique, ces techniques peuvent induire certaines lourdeurs au niveau de l'implémentation.

### II.1.2.3. Décomposition mixte

Les deux types de décomposition précédents ont mis en valeur le principal objectif de la décomposition, à savoir la réduction de la taille du problème. Mais ce n'est pas là le seul objectif justifiant des recherches dans ce domaine. Effectivement, on peut évoquer la *décentralisation* à la fois des informations et des décisions optimales qui doit permettre aux sous-systèmes de préserver leur autonomie jusqu'à la solution optimale et cela, sans l'aide d'un niveau de coordination centralisateur. En fait, les deux types de décomposition précédents ne permettaient pas une décentralisation des sous-problèmes. On entend par décentralisation la capacité de chaque sous-système de calculer et d'identifier sa propre solution optimale en résolvant les sous-problèmes de flot simple avec le vecteur prix optimal ou l'allocation optimale (respectivement pour la décomposition par les prix et la décomposition par les ressources.) Ainsi, dans la décomposition par les prix nous avons appliqué une méthode duale consistant à traiter les contraintes couplantes par relaxation lagrangienne. En termes de programmation mathématique, le *coordinateur* doit donc rechercher la solution d'un problème dual dont la fonction objective n'est généralement pas différentiable en $w^*$. Il en est de même dans la décomposition par les ressources, où l'on résout un problème d'allocation dont la fonction objective n'est pas différentiable en $f^*$. Par conséquent les décisions ne peuvent pas être totalement décentralisées par ces méthodes. Autrement dit, les sous-systèmes n'ont aucun moyen de calculer et d'identifier la solution optimale. C'est dans ce but que sera introduit ici, un nouvel algorithme de décomposition mixte complètement décentralisé. Mis à part le gain introduit par la simplification de la coordination, l'objectif de décentralisation a des conséquences intéressantes sur l'organisation du processus décisionnel, dans le cas des systèmes

économiques notamment (cf. Atkins [1974] et Jennergren[1973]). Ce schéma de décomposition a été proposé par Mahey [1986] dans le cadre plus général des modèles à structure bloc-angulaire. Nous présentons ici, une spécialisation de cette approche au cas des multiflots. Pour simplifier la présentation, nous supposerons $u_{ij}^k = +\infty$ pour tout $(i,j) \in E$ et tout $k$. Réécrivons alors le problème (I.1) sous une forme matricielle plus compacte:

$$\text{Minimiser} \sum_k c^k x^k \tag{II.14a}$$

sous les contraintes

$$\sum_k E_k x^k \leq b, \tag{II.14b}$$

$$x^k \in V_k = \left\{ x^k \in R^M \middle| Ax^k = r^k,\ x^k \geq 0 \right\} \ \forall\, k = 1,\ K. \tag{II.14c}$$

Où les $E_k$ sont des matrices identité de taille $M$. Nous supposerons les hypothèses initiales suivantes: $A$ est de rang $N$ et $V_k$ est borné pour tout $k$. De plus la solution optimale du problème (II.14) existe, est unique et non dégénérée. Ceci implique que le problème dual de (II.14) possède une solution unique et non dégénérée.

II.1.2.3.1. Analyse de la dégénérescence des sous-problèmes

On a vu que les deux types de coordination précédents ne pouvaient permettre la décentralisation totale des décisions optimales locales. Montrons comment ce fait est introduit par des dégénérescences introduites dans les sous-problèmes par la décomposition.

Avec les hypothèses faites au départ, on sait que la solution optimale contient exactement $(M + KN)$ variables de base strictement positives. Prenons alors les sous-problèmes associés à la décomposition par les prix. Leur dimension étant $N$, la solution de base constituée par les $K$ solutions locales contiendra au plus $KN$ variables positives. D'autre part, comme la solution optimale globale de (II.14) satisfait les conditions d'optimalité locales de chaque sous-problème on peut affirmer qu'une partie des sous-problèmes (au moins un et au plus $M$) possèdent une infinité de solutions et par conséquent ne peuvent être décentralisés. On est donc en présence d'une dégénérescence duale. Dans le cas des sous-problèmes associés à la décomposition par les ressources, la dimension de la solution de base constituée par les $K$ solutions locales est $(KN + KM)$. A l'optimum, il y aura donc

nécessairement *(K-1)M* variables dégénérées (en base et nulles) et cette dégénérescence portera sur au moins *(K-1)* sous-problèmes.

On s'aperçoit donc que, *si l'on veut maintenir les problèmes linéaires*, un algorithme décentralisé devra allouer *exactement* $M$ contraintes dans les sous-problèmes. C'est dans ce but que nous allons introduire une partition des contraintes de couplage en $K$ sous-ensembles (un certain nombre de ces sous-ensembles peuvent être vides). L'allocation consistera alors en une bijection entre ces $K$ sous-ensembles disjoints et les sous-problèmes. Chaque bloc $E_k$ est partitionné en $K$ sous-blocs, de même pour $b$ :

$$E_k = \begin{bmatrix} E_{1k} \\ \cdot \\ \cdot \\ \cdot \\ E_{kk} \\ \cdot \\ \cdot \\ \cdot \\ E_{Kk} \end{bmatrix}, \qquad b = \begin{bmatrix} b_1 \\ \cdot \\ \cdot \\ \cdot \\ b_k \\ \cdot \\ \cdot \\ \cdot \\ b_K \end{bmatrix}$$

Les contraintes $\sum_{l=1}^{K} E_{kl} x^k \leq b_k$ seront allouées au sous-problème $k$, qui recevra des allocations de prix associées aux *(K-1)* blocs de contraintes restants. La preuve de l'existence d'une partition qui mène à des solutions locales non dégénérées dépend du résultat suivant:

***Lemme II.8.*** *Étant donnée une matrice carrée non singulière et une partition quelconque des colonnes (reps. des lignes) de cette matrice, il existe au moins une partition des lignes (resp. des colonnes) faisant apparaître des blocs diagonaux non singuliers.*

Pour la preuve voir Mahey [1986].

II.1.2.3.1. Décomposition décentralisée

Étant donnée une partition des contraintes en $K$ sous-ensembles, réécrivons le problème (II.14) en rajoutant les variables $f = (f^1, ..., f^K)$ telles que $f \in R^M$ et $f^k$ a la même dimension que $b_k$ :

$$\text{Minimiser} \sum_k c^k x^k \qquad \text{(II.15a)}$$

$$E_{kk}x^k - f^k = 0, \tag{II.15b}$$

$$f^k + \sum_{l \neq k} E_{kl}x^l \leq b_k \quad \forall\, k = 1,\ K, \tag{II.15c}$$

$$x^k \in V_k. \tag{II.15d}$$

On effectue alors une relaxation lagrangienne des contraintes de couplage (II.15c). On obtient alors un lagrangien en $x$, $w$ et $f$:

$$L(x, w, f) = \sum_k \left[ c^k x^k + w_k \left( \sum_{l \neq k} E_{kl}x^k + f^k - b_k \right) \right].$$

Pour un couple $(\overline{w}, \overline{f})$, on a les sous-problèmes:

$$p_k(\overline{w}, \overline{f}) = \min \left[ c^k x^k + \sum_{l \neq k} \overline{w}_l E_{kl} x^k \right] \tag{II.16a}$$

$$E_{kk}x^k = \overline{f}^k, \tag{II.16b}$$

$$x^k \in V_k. \tag{II.16c}$$

Le coordinateur doit rechercher un point-selle de la fonction:

$$p(w, f) = \sum_{k=1}^{K} \left[ p_k(w, f^k) + w_k(f^k - b_k) \right]$$

$p(w, f)$ est convexe en $f$ et concave en $w$. A l'intérieur du domaine de $p$, on peut définir le sous-différentiel de $p$ en $(w, f)$ (Rockafellar [1970], p. 374):

$$\partial p(w, f) = \partial_w p(w, f) \times \partial_f p(w, f).$$

Soient $X_i(w, f)$ et $\Pi_i(w, f)$, les ensembles de solutions optimales primales et duales de (II.16), on a alors:

$$\partial_{fk} p(w, f) = \left\{ -\pi_i + p_i \middle| \pi_i \in \Pi_i(w, f) \right\}$$

$$\partial_{wk} p(w, f) = \left\{ \sum_{l \neq k} E_{kl}x^l + f^k - b_k \middle| x^l \in X_l(w, f) \right\}.$$

On a vu que la possibilité de décentralisation requiert la non dégénérescence des sous-problèmes. Montrons qu'il existe une décomposition telle que chaque sous-problème (II.12) a une solution unique non dégénérée au voisinage de l'optimum.

***Théorème II.9.*** (Mahey [1986]) *Il existe une décomposition des contraintes de couplage telle que chaque sous-problème* (II.16) *a une solution unique non dégénérée pour* $w_l = \overline{w}_l,\ \forall\, l \neq k$ *et* $f^k = \overline{f}^k$ *où* $\overline{f}^k = b_k - \sum_{l \neq k} E_{kl} \overline{x}^k$. *Cette solution est égale à* $(\overline{x}^k,\ \overline{w}^k)$ *et* $(\overline{w},\ \overline{f})$ *est l'unique point-selle de* $p(w, f)$.

On ne pourra donc parler de décentralisation que dans un voisinage de l'optimum où il est possible de caractériser une partition des contraintes favorable. Pour cela, on doit se poser deux problèmes distincts:

- Comment obtenir une telle partition?
- Une fois connue une partition favorable, comment obtenir le point-selle $(\overline{w},\ \overline{f})$?

Répondons en premier lieu à la deuxième question: la forme particulièrement symétrique des sous-problèmes (II.16) suggère le schéma itératif suivant: à l'itération $t$, on résout les sous-problèmes (II.16) séquentiellement. Appelons $x_t^k$, la solution primale optimale et $\pi_t^k$ le vecteur des multiplicateurs optimaux associé à l'allocation $\overline{f}^k$ du sous-problème $k$. Ces informations sont transmises aux autres sous-problèmes qui modifient leurs allocations primales et duales par:

$$\left.\begin{array}{rl} & \overline{w}_l = \pi_t^l, \qquad l = 1,\ \ldots,\ k \\ & \overline{w}_l = \pi_{t-1}^l, \qquad l = k+2,\ \ldots,\ K \\ \text{sous - problème} & \\ \underline{k+1} \qquad & \overline{f}^{k+1} = b_{k+1} - \sum_{l=1}^{k} E_{k+1,l} x_t^l \\ & \qquad - \sum_{l=k+2}^{K} E_{k+1,l} x_{t-1}^l \end{array}\right\}$$

Observons que ce schéma itératif est équivalent à résoudre $\partial p(w, f) = 0$ par un algorithme de type point fixe. Plus précisément, si on considère la base optimale de (II.14) dont les lignes sont arrangées de manière à faire apparaître les bases optimales de chaque sous-problème (II.16) on peut en déduire que l'algorithme proposé équivaut à appliquer une méthode de Gauss-Seidel par blocs à cette matrice. Sur la Figure II.6, on a noté $\tilde{E}_{kj}$ *et* $\tilde{A}_k$ les parties basiques des sous-matrices $E_{kj}$ *et* $A_k$. Par conséquent l'algorithme converge

vers le point-fixe $(\bar{w}, \bar{f})$ si le rayon spectral de la matrice $(D-E)^{-1}F$ est inférieur à l'unité.

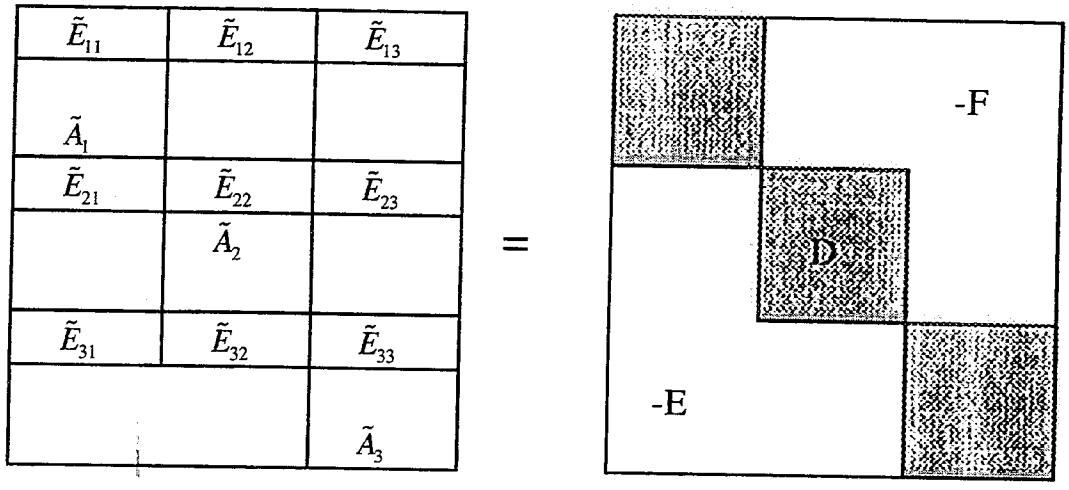



**Figure II.6** Forme bloc-diagonale de la base optimale.

Le principal avantage de cette décomposition est donc l'élimination du niveau coordinateur dans le voisinage de l'optimum où la partition favorable des contraintes a pu être identifiée. Le problème de la recherche de la partition favorable est plus difficile. Il est pourtant crucial pour construire des algorithmes compétitifs avec les méthodes classiques (application du théorème II.9 et convergence des itérations de point fixe). Une réponse partielle à ce problème est exposée dans l'article de Mahey [1986] où l'introduction de quelques règles heuristiques se sont avérées suffisantes sur les exemples numériques traités (voir également Cohen [1980] et Morin [1993]).

### II.1.3. MÉTHODES DE PARTITIONNEMENT

Comme nous l'avons constaté au chapitre I, le problème de multiflot est un programme linéaire qui met en œuvre deux types de contraintes: (1) des contraintes de conservation de flot indépendantes pour chaque produit, (2) des contraintes couplantes, relatives au partage de la capacité par tous les produits. C'est cette structure sous-jacente de flots dans les réseaux qui est à la base des méthodes de partitionnement. En effet, ces méthodes sont des spécialisations de la méthode du simplexe, où la base courante est partitionnée de façon à exploiter les propriétés particulières des flots dans les réseaux. Nous présentons dans ce qui suit deux formes de partitionnement, à savoir le partitionnement dual et le partitionnement primal.

### II.1.3.1. Partitionnement dual

Cette première approche se base sur le fait qu'en pratique, seul un sous-ensemble réduit des contraintes de capacité est "saturé" à l'optimum, de telle sorte que la plupart de ces contraintes peuvent être relaxées sans perte d'optimalité. Nous présenterons dans ce qui suit une technique duale, pouvant être interprétée comme une forme de relaxation. Pour simplifier la présentation, nous supposerons $u_{ij}^k = +\infty$ pour tout $(i,j) \in E$ et tout $k$ dans la formulation (I.1). Réécrivons alors le problème de multiflot de coût minimum sous une forme matricielle plus compacte en introduisant des variables d'écart:

$$\text{Minimiser} \sum_k c^k x^k \tag{II.17a}$$

sous les contraintes

$$\sum_k E_k x^k + s = b \tag{II.17b}$$

$$Ax^k = r^k \quad \forall\, k = 1,\, K, \tag{II.17c}$$

$$s \geq 0,\; x^k \geq 0 \quad \forall\, k = 1,\, K. \tag{II.17d}$$

Où les $E_k$ sont des matrices identité de taille $M$. L'observation précédente a motivé l'idée de diviser les contraintes de couplage en deux groupes: les contraintes courantes (à variables d'écart nulles ou proches de zéro) et les contraintes secondaires, soit respectivement:

$$\sum_k E_k^c x^k + s^c = b^c,\;\; \sum_k E_k^s x^k + s^s = b^s.$$

Supposons que $A$ soit de plein rang, et partitionnons la en $[B, N]$ en référence aux parties basiques et non basiques de $A$. Autrement dit $\det(B) \neq 0$ et $B^{-1}r^k \geq 0$. (Si cela est impossible le problème original n'a pas de solution. ) Le problème (II.17) peut alors s'écrire comme suit:

$$\text{Minimiser} \sum_k (c_B^k x_B^k + c_N^k x_N^k) \tag{II.18a}$$

sous les contraintes

$$\sum_k (E_k^{c^B} x_B^k + E_k^{c^N} x_N^k) + s^c = b^c, \tag{II.18b}$$

$$\sum_k E_k^s x^k + s^s = b^s, \tag{II.18c}$$

$$x_B^k = B^{-1}r^k - B^{-1}Nx_N^k \qquad \forall\, k = 1,\, K, \tag{II.18d}$$

$$s^c \geq 0,\;\; x_N^k \geq 0 \qquad \forall\, k = 1,\, K, \tag{II.18e}$$

$$s^s \geq 0,\ x_B^k \geq 0 \qquad \forall\, k = 1,\ K. \tag{II.18f}$$

Toutes les partitions sont compatibles avec $A = [B, N]$.

Grigoriadis et White [1972a, 1972b] proposent une procédure de relaxation où les contraintes (II.18f) sont relaxées. Alors (II.18d) peut être utilisé pour éliminer $x_B^k$ du problème (II.18), ce qui donne lieu au problème relaxé suivant:

$$\text{Minimiser} \sum_k \left\{ c_B^k B^{-1} r^k + (c_N^k - c_B^k B^{-1} N) x_N^k \right\} \tag{II.19a}$$

sous les contraintes

$$\sum_k (E_k^{c^N} + E_k^{c^B} B^{-1} N) x_N^k + s^c = b^c - \sum_k E_k^{c^B} B^{-1} r^k, \tag{II.19b}$$

$$s^c \geq 0,\ x_N^k \geq 0 \qquad \forall\, k = 1,\ K, \tag{II.19c}$$

La stratégie de base consiste alors à résoudre le problème relaxé (II.19) en $x_N^k$ pour tout $k$. Si les contraintes relaxées sont satisfaites par la solution courante, on est à l'optimum. Sinon le partitionnement en variables de base et hors base est révisé de telle sorte qu'une variable négative de $x_B^k$ soit remplacée par une variable positive de $x_N^k$. La preuve qu'un tel échange est possible est donnée dans Rosen [1964]. Si les contraintes $s^c \geq 0$ sont violées, le partitionnement de $E$ en $[E^c, E^s]$ est modifié en translatant une contrainte violée de $E^s$ dans $E^c$. Cette procédure peut être interprétée comme une application particulière de la méthode duale simplexe. Sauf que, au lieu de manipuler l'inverse de la base pour (II.17), on se restreint à l'inverse de (II.19b).

L'application d'une telle procédure répond à deux motivations principales: premièrement, la structure de réseau est préservée, en ce sens que toutes les opérations de multiplication par $B^{-1}$ sont mises à jour par des techniques de marquage issues de la théorie des graphes, deuxièmement, si seulement quelques contraintes de capacité sont saturées à l'optimum, les matrices $E_k^c$ auront des dimensions assez petites tout au long de la procédure. Par conséquent les opérations de pivotage seront effectuées sur un système de taille réduite. L'approche souffre cependant du fait qu'étant dualisée, la faisabilité n'est atteinte qu'à l'optimum.

### II.1.3.2. Partitionnement primal

Tel qu'il a été formulé en (I.1), le problème de multiflot de coût minimum se présente comme un programme linéaire à structure bloc-angulaire. Cette approche permet

d'implémenter la méthode du simplexe en tirant profit de cette structure particulière et en exploitant certaines propriétés des flots dans les réseaux.

Considérons une base $B$ quelconque du programme linéaire (I.1), où chaque colonne correspond à un flot de produit quelconque sur un des arcs du réseau, les colonnes basiques forment donc un ensemble de sous-graphes. Chacun de ces sous-graphes est associé à un des produits en circulation. Soit $M^k$ la sous-matrice de $B$ correspondant au produit $k$. Notons que chaque sous-matrice $M^k$ contient une base $B^k$ de la matrice incidence sommets-arcs $A$. Or nous savons que toute base $B^k$ correspond à un arbre $T^k$ du graphe associé au réseau (voir Ahuja, Magnanti et Orlin [1993] chap. 11). Par conséquent les sous-graphes associés à une base quelconque sont composés de ces arbres et de certains arcs additionnels. Par ailleurs le problème de multiflot présentant une structure bloc-angulaire, la base $B$ reproduira cette même structure en prenant la forme indiquée sur la Figure II.7(a). Plutôt que de travailler sur la base $B$, nous allons procéder à un changement de variables. Le but de ce changement est de transformer la base $B$ en une base $B'$ de façon à pouvoir faire ressortir la structure sous-jacente de flots dans les réseaux, et ainsi accomplir les opérations du simplexe d'une manière plus efficace. Les contraintes de capacité sont présentées ici sous la forme (I.1 c'). Si $x_B$ représente le vecteur des variables de base et $h$ le vecteur second membre $(h = h^1, ..., h^K)$, on peut écrire le système $Bx_B = h$. Remarquons au passage que les variables de base incluent à la fois des variables flot et des variables d'écart.

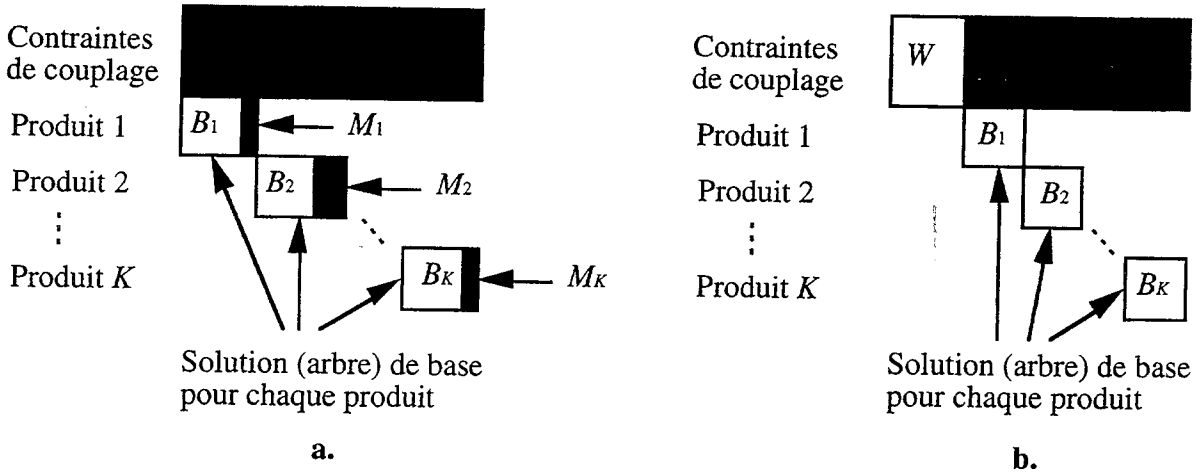



**Figure II.7** Partitionnement de la base pour un problème de multiflot: (a) base initiale; (b) base après le changement de variables et le réarrangement des colonnes.

Pour simplifier le système $Bx_B = h$, supposons le changement de variables suivant: $Dy_B = x_B$ pour une matrice $D$ non singulière choisie de manière appropriée. Par substitution de $x_B$ dans le système $Bx_B = h$, en posant $B' = BD$, on obtient le système $B(Dy_B) = (BD)y_B = B'y_B = h$. Supposons que nous puissions faire ce changement de façon à ce que la matrice $B'$ prenne la forme particulière indiquée sur la Figure II.7(b). Dans cette forme, seules les variables qui correspondent à la base $B^k$ apparaissent dans les contraintes de conservation du flot $k$.

Montrons donc comment s'effectue ce changement de variables $Dy_B = x_B$. Remarquons pour cela que ce changement revient à éliminer toute colonne $M_{ij}^k$ appartenant à $M^k$ et non à $B^k$ dans les contraintes de conservation du flot $k$. Ainsi les seules variables figurant dans ces équations sont celles correspondant à $B^k$. Par ailleurs notons que ces colonnes basiques sont associées aux arcs $(i,j)$ correspondants, lesquels forment avec l'arbre $T^k$ un circuit unique. Dès lors le changement de variables peut se faire par une simple opération de marquage. Supprimons l'indice $k$ momentanément pour ne pas surcharger la présentation, et posons $\delta_{ij}^{pq}$ égal à $\pm 1$ ou à 0, selon que l'arc $(p, q)$ de l'arbre $T$ appartient ou non à ce circuit (le signe $\pm$ dépend de l'orientation de l'arc $(p, q)$ relativement à l'arc $(i,j)$). Pour transformer les variables, il suffit de poser $x_{pq} = y_{pq} - \sum \delta_{ij}^{pq} y_{pq}$, avec la somme portant sur les indices $(i,j)$ de toutes les colonnes $M_{ij}$ destinées à être éliminées. Sans oublier $x_{ij} = y_{ij}$ pour tous les arcs $(i,j)$ qui ne sont pas dans l'arbre $T$. En supposant ces transformations réalisées, nous pouvons faire les remarques suivantes:

1. Comme le changement de variables correspond à des opérations sur les colonnes de la matrice $B$, la non singularité est maintenue dans $B'$ et $W$ est aussi non singulière (comme sous-matrice de $B'$).

2. De même les opérations sur les colonnes ne changeant pas la solution du système $\pi B = c_B$, le vecteur des multiplicateurs du simplexe est le même que celui déterminé par le système $\pi B' = c_{B'}$ ($c_B$ et $c_{B'}$ étant les coûts basiques avant et après transformation).

A présent nous pouvons implémenter la méthode du simplexe. Pour ce faire, rappelons les deux types d'opérations requis par cette méthode:

1. Résoudre un système de la forme $Bx_B = h$ ou $Bx_B = A_j$ pour une colonne $A_j$ à introduire en base.

2. Résoudre un système de la forme $\pi B = c_B$, pour mettre à jour les variables duales $\pi$.

Plutôt que de faire les calculs sur la matrice $B$, nous utiliserons la matrice $B'$. Partitionnons le vecteur $\pi = (\pi^0, \pi^1, ..., \pi^K)$, où $\pi^0$ contient les variables duales rattachées aux contraintes de couplage, et $\pi^k$ celles rattachées aux contraintes de conservation de flot pour le produit $k$. Soit $c_W$ le sous-vecteur de $c_B$ correspondant aux colonnes de $W$. Nous pouvons alors résoudre le système $\pi B' = c_{B'}$ par une procédure de substitutions successives. Résolvons en premier lieu le sous-système $\pi_0 W = c_W$ qui met en jeu seulement $M$ équations, puis substituons ces valeurs dans le système $\pi B' = c_{B'}$. Le système se décompose alors en sous-systèmes indépendants (pour tout $k$) avec des coûts modifiés $\hat{c}_{B'}$. Par ailleurs, ces calculs correspondant à un calcul de variables duales pour un problème de flot simple, ils sont mis à jour par des techniques classiques (Ahuja, Magnanti et Orlin [1993] chap. 11) à travers des opérations de graphes qui permettent d'exploiter la structure d'arbre associée à chaque base.

Le même scénario est repris pour résoudre le système $Bx_B = h$. le vecteur $y_B$ est partitionné sous la forme $y_B = (y^0, y^1, ..., y^K)$ en correspondance avec les colonnes de $W$, $B^1$, ..., $B^K$ respectivement dans $B'$. Les sous-systèmes indépendants $B^k y^k = h^k$ sont (on le sait maintenant) des problèmes de flot simple pour lesquels il existe des techniques de résolution efficaces. En substituant par la suite les valeurs des variables $y^1$, ..., $y^K$ dans le système $B'y = h$, on tombe sur un système réduit de la forme $Wy^0 = b'$, ($b'$ représente la capacité modifiée après injection des variables flot dans les contraintes de capacité.) Enfin, après avoir résolu tous ces systèmes, nous pouvons utiliser la transformation $Dy_B = x_B$ pour évaluer les variables $x_B$.

Récapitulons à présent les simplifications réalisées par cette approche. Il est clair que toutes les opérations du simplexe sont accomplies à partir d'une base plus compacte $W$, le reste des opérations relevant de la théorie des graphes. En effet plutôt que de résoudre le système $Bx_B = h$ à partir d'une matrice de taille $(M+K)\times(M+K)$, nous faisons $K$ calculs de flot simple et nous résolvons un système réduit $(M \times M)$, soit $Wy^0 = b'$. D'une manière analogue, au lieu de résoudre un système $(M+K)\times(M+K)$ de la forme $\pi B = c_B$, nous réalisons $K$ calculs de flot simple et nous résolvons un système de taille réduite $(M \times M)$, soit $\pi_0 W = c_W$ . Le gain en temps de calcul est d'autant plus substantiel qu'il arrive que seulement $p$ arcs $(p < M)$ soient assortis d'une contrainte de capacité.

Le reste des détails pratiques rentrant dans l'implémentation du simplexe, notamment les entrées et sorties de colonnes, est décrit en détails dans différents articles en rapport avec les techniques de partitionnement primal. Dans ce contexte, Saigal [1967] a été le premier à appliquer cette méthode, dite aussi méthode de représentation compacte de l'inverse de la base, au cas des multiflots. Une approche similaire est décrite par Hartman et Lasdon [1972]. L'algorithme qu'ils ont développé est une spécialisation d'une technique de la programmation linéaire connue sous le nom de Generalized Upper Bounding (G.U.B.), adaptée aux problèmes à structure bloc-angulaire. Leur algorithme montre qu'on peut réaliser les calculs du simplexe à partir d'une base encore plus compacte de taille $(s+1)\times(s+1)$ (où $s$ représente le nombre d'arcs saturés à l'étape courante). Maier [1974] propose un autre schéma de partitionnement où il tombe sur la même base $W$. Mais contrairement à son prédécesseur, il préconise de maintenir explicitement $W^{-1}$, au lieu d'une sous-matrice de $W^{-1}$. Kennington [1977] propose une spécialisation de l'algorithme de Hartman et Lasdon pour un problème de transport en multiflots. McCallum [1977] présente une approche de G.U.B. pour un problème de communication. D'autres techniques de partitionnement sont également proposées par Graves et McBride [1974, 1976].

## II.1.4. COMPARAISON DES MÉTHODES ET NOUVELLES APPROCHES

Nous avons vu précédemment que le problème de multiflot pouvait être résolu soit directement (en tant que programme linéaire) par des techniques de programmation linéaire, soit par des méthodes spécialisées. A partir de là, se sont développés de nombreux codes en rapport avec les trois approches de base que sont: la décomposition par les prix, la décomposition par les ressources et les techniques de partitionnement, utilisées classiquement pour résoudre des problèmes de multiflots. Nous présenterons donc ces différents codes pour donner une idée sur les méthodes les plus prometteuses, et sur leur efficacité relative par rapport à des codes plus généraux de programmation linéaire. Puis nous donnerons un aperçu sur les nouvelles approches récemment mises au point pour résoudre ce genre de problème.

### II.1.4.1. Comparaison des codes de résolution

Les trois approches principales utilisées pour la résolution des problèmes de multiflot sont la décomposition par les prix, par les ressources et les méthodes de partitionnement. Nous donnons dans le tableau suivant (Farvolden, Powell et Lustig [1993]) un résumé des

principaux codes qui ont été développés, avec la taille des plus grands problèmes résolus et la méthode utilisée.

| Référence | Nombre de produits | Taille du problème Nombre de Lignes × colonnes | Nombre de contraintes de couplage |
|---|---|---|---|
| | Programmation Linéaire | | |
| Ali et al., 1980 | 10 | 1.190×2.203 | 180 |
| Ho et Loute, 1983 | 6 | 5.899×12.048 | 594 |
| | Décomposition de Dantzig-Wolfe | | |
| Ali et al., 1980 | 10 | 1.190×2.203 | 180 |
| Assad, 1976 | 18 | 1.698×3.672 | 98 |
| Ho et Loute, 1983 | 6 | 5.899×12.048 | 594 |
| | Décomposition par les ressources | | |
| Ali et al., 1980 | 10 | 1.190×2.203 | 180 |
| Assad, 1976 | 18 | 1.698×3.672 | 98 |
| Held et al.,1974 | 10 | 1.170×8.700 | 8.780 |
| | Partitionnement | | |
| Ali et al., 1980 | 10 | 1.190×2.203 | 180 |
| Kennington, 1977 | 13 | 507×2.704 | 208 |
| McCallum, 1977 | 1.857 | N/M | N/M |
| Grigoriadis et White, 1972 | 10 | 309×1.078 | 98 |

**Figure** II.8 Problèmes de multiflots résolus par des codes spécialisés.
(Farvolden, Powell et Lustig [1993]).

Pour plus de rigueur dans la comparaison de ces différents codes, on gagnerait à utiliser les mêmes bases de données; ce travail a été réalisé par Ali, Helgason, Kennington et Lall [1980] dont le groupe a mis au point trois codes classiques qui ont ensuite été testés sur un même jeu de données, et comparés à des codes plus généraux de programmation linéaire. Le premier code appelé PDD est basé sur la méthode de décomposition de Dantzig-Wolfe. Les sous-problèmes sont résolus par le code de Helgason et Kennington [1976]. La procédure de réinversion de la base du problème maître est basée sur le travail de Hellerman et Rarick [1971]. La solution de base réalisable initiale est mise à jour par une méthode des deux phases (voir Orchard-Hays [1968]). Le deuxième code appelé RDD est une version améliorée d'un premier code développé par Kennington et Shalaby [1977] qui se limitait au cas des problèmes de transport. Les sous problèmes sont toujours résolus par

le code de Helgason et Kennington [1976], et le pas de déplacement utilisé dans l'algorithme est identique à celui développé par Held-Wolfe et Crowder [1974]. Enfin, le troisième code appelé PP est une version améliorée du code développé par Kennington [1977], comme généralisation d'un cas particulier de problème de transport.

Les trois codes ont été écrits en Fortran et ont tourné sur un CDC Cyber 72. Certains problèmes test ont été générés aléatoirement en utilisant un code similaire à NETGEN de Klingman, Napier et Stutz [1974]. Une description complète de ce générateur ainsi que des paramètres d'input peut être trouvée dans Ali et Kennington [1977]. Comme les codes PDD et PP sont basés sur l'algorithme du simplexe, ils héritent du même critère d'arrêt, c'est pourquoi on dit que ces méthodes sont exactes. La méthode PDD quant à elle, ne possède pas réellement de critère d'optimalité, elle est donc considérée comme une technique heuristique. Le critère d'arrêt utilisé dans ce code consiste à arrêter si l'écart entre la solution courante et une borne inférieure calculée, est inférieure (en valeur relative) à $\varepsilon$. Cette borne inférieure a été calculée en relaxant les contraintes de couplage et en résolvant des problèmes de flot simple. L'écart $\varepsilon$ a été fixé à 10% et comme son obtention n'est pas garantie, on a également fixé un nombre maximum d'itérations. Les résultats obtenus par ce groupe montrent que les codes de décomposition par les prix et de partitionnement sont presque équivalents. Le code de décomposition par les ressources pouvait tourner deux fois plus rapidement que les deux autres codes. Néanmoins, à l'inverse des deux premiers, aucune garantie de convergence n'est assurée par ce dernier.

Par ailleurs Ali et al. [1984] présentent une étude sur trois modèles réels de multiflots dans laquelle ils expérimentent un code spécialisé basé sur une technique de partitionnement primal. Leur analyse porte sur un ensemble de problèmes test dont la taille varie entre 400 à 600 lignes et 700 à 1000 colonnes. La différence majeure entre ces problèmes réside dans le nombre de contraintes de couplage en comparaison avec le nombre de contraintes individuelles de conservation de flot, ainsi que dans l'intervalle de variation de la capacité pour les contraintes de couplage. Leurs résultats confirment la supériorité des méthodes spécialisées en comparaison avec les méthodes générales de programmation linéaire. En effet, leur code s'avéra être trois fois plus rapide que des logiciels de programmation linéaire et ce gain de temps serait fonction du nombre de contraintes saturées à l'optimum.

### II.1.4.2. Nouvelles approches

Les résultats numériques exposés précédemment montrent que les méthodes classiques ne permettent de prendre en compte qu'un nombre réduit de produits (cf. tableau II.8), c'est

pourquoi on assiste actuellement à la recrudescence de nouvelles approches visant essentiellement à résoudre efficacement des problèmes de multiflots impliquant un plus grand nombre de flots. De même que les approches de résolution classiques étaient soit des méthodes générales de programmation linéaire, soit des méthodes spécialisées de décomposition ou de partitionnement, les nouvelles approches comptent également des nouvelles méthodes générales de programmation linéaire et des approches beaucoup plus personnalisées. Ces dernières consistent généralement soit à combiner des approches classiques, soit à mettre en œuvre des procédures d'optimisation non linéaires.

Ainsi, les problèmes de multiflots, toujours en tant que problèmes linéaires, peuvent faire appel à des méthodes de résolution basées sur le nouveau concept des algorithmes de points intérieurs inventé par Karmarkar [1984] pour la programmation linéaire. Mais même si ces algorithmes sont les seuls algorithmes polynomiaux connus pour ce genre de problème, une implémentation efficace de ces méthodes reste encore un sujet de recherche future. Dans ce contexte, Tardo's [1986] résout le problème de multiflot en un temps fortement polynomial; la meilleure borne en temps pour ce genre de problème est due à Vaidya [1989]. Carolan et al. [1991] présentent des résultats numériques pour plusieurs problèmes de multiflots de très grande taille (105.127 lignes) résolus par le système KORBX (Cheng et al. [1989]) et par MPSX (IBM corporation [1979]). Le système KORBX est constitué par un environnement hardware de processeurs parallèles ayant un grand potentiel mémoire, et un environnement software basé sur le principe des algorithmes de points intérieurs. Le système MPSX est basé, quant à lui, sur l'algorithme du simplexe. Ces auteurs ont trouvé, sur des problèmes de multiflots, que le système KORBX était plus rapide que le MPSX tournant sur un IBM 3081D. Jusqu'à présent il paraît clair que ces méthodes peuvent être plus rapides que le simplexe sans pour autant pouvoir rivaliser avec des méthodes spécialisées dans la résolution des problèmes de multiflots.

Parmi les nouvelles approches spécialisées dans la résolution des multiflots, on peut citer l'approche de Farvolden, Powell et Lustig [1993] dont la motivation a été de résoudre des problèmes issus d'un système de distribution concernant le transport de moteurs. Ce modèle dynamique implique une origine-destination spécifique pour chaque produit, d'où la taille importante des programmes linéaires mis en jeu, même sur des petits exemples. Ainsi un exemple impliquant seulement 10 terminaux et 700 livraisons sur un horizon de 18 périodes nous met en présence d'un programme linéaire de 133.700 lignes et 490.000 colonnes. C'est pourquoi une formulation arcs-chemins est utilisée par Farvolden pour

déboucher sur un schéma de résolution combinant les techniques de partitionnement primal et de génération de colonnes. En attendant des tests plus rigoureux, l'expérience de Farvolden a permis de résoudre des problèmes de multiflots de taille très importante ne rentrant pas dans la catégorie des problèmes résolus par les méthodes classiques. D'autre part, la comparaison de son code avec des codes de programmation linéaire basés sur le simplexe (MINOS, Murtagh et Saunders [1983]) et sur les algorithmes de points intérieurs (OB1, Marsten et Shanno [1991]), confirme le potentiel de cette méthode et sa capacité à entrer en compétition avec d'autres approches spécialisées.

Gersht et Schulman [1987] proposent une nouvelle approche pour résoudre des problèmes de multiflots mettant en jeu un grand nombre de produits. Leur approche permet également de prendre en compte certaines extensions du modèle de base, notamment les contraintes générales de ressources, les contraintes de capacité sur les nœuds et les restrictions de circulation des flots individuels sur des sous-graphes partiels. De plus cette approche se généralise à des problèmes de multiflots dont les contraintes de capacité ou la fonction objective sont des fonctions non linéaires. Cette approche est basée sur les méthodes de barrières et de pénalités. Le problème de multiflot original est ainsi approximé par une séquence de problèmes de flots non linéaires, où les contraintes de capacité sont remplacées par des fonctions barrières et pénalités appropriées. Ces fonctions auxiliaires quelque peu modifiées sont définies à la fois à l'intérieur et à l'extérieur du domaine de réalisabilité associé à toutes les allocations possibles de capacité. De fait, elles se comportent comme des fonctions barrières à l'intérieur et des fonctions pénalités à l'extérieur. Cette modification permet de combiner les avantages des méthodes barrières et pénalités et de construire des algorithmes plus simples et plus efficients. La méthode débouche sur un algorithme itératif procédant en deux étapes: dans la première, la solution converge vers un voisinage de l'optimum sans que les flots ne soient évalués, dans la seconde, les flots sont calculés sur la base de fonctions indicatrices de plus courts chemins. Cette approche est spécialement attractive dans le cas où le nombre de flots est important. En effet, en reportant le calcul des flots à la deuxième étape, cette méthode permet de réaliser un gain de temps appréciable. De plus cette méthode ne nécessite pas de trouver une solution initiale réalisable. Les résultats numériques obtenus montrent qu'on peut obtenir par cette approche une bonne solution approchée pour des problèmes de multiflots de grande taille, et ce relativement rapidement. De plus, comme la solution obtenue est une approximation de la solution optimale, Gersht et Schulman proposent un nouvel algorithme pour calculer des bornes inférieures nécessaires à l'évaluation de la

qualité de leur solution. Ces bornes inférieures sont obtenues par résolution d'un programme linéaire basé sur les $q$ chemins de coût minimum.

Pinar et Zenios [1992] développent une approche basée sur les fonctions de pénalités linéaires quadratiques pour résoudre des problèmes de multiflots de très grande taille. Leur approche débouche sur un schéma de décomposition avec des sous-problèmes de flot simple et un problème maître non linéaire. Dans ce contexte les concepts de vectorisation et de parallélisme peuvent être exploités à travers des calculs d'algèbre linéaire.

Enfin, d'autres approches sont également disponibles pour ce genre de problème grâce à: Barnhart [1988], Armstrong, Qi et Zenios [1990], Brown et al. [1990], Schultz et Meyer [1991], Schneur [1991], McBride et Mamen [1993], et également Matsumoto, Nishizeki et Saito [1986] qui exhibent un algorithme combinatoire pour résoudre des problèmes de multiflots sur des graphes planaires, en un temps polynomial. Les performances de ces approches confirment, d'une part, que les approches spécialisées sont supérieures aux approches générales de programmation linéaire sur des problèmes aussi structurés que les problèmes de multiflots (notamment les approches de points intérieurs), et d'autre part, que les techniques de résolution gagneraient à utiliser le principe de décomposition pour améliorer les performances d'un simplexe plus spécialisé.

## II.2. CAS ENTIER

Comme nous l'avons précisé au premier chapitre, la contrainte d'intégralité sur les flots est une contrainte qui intervient dans la majorité des applications de multiflots. Mais malgré cette importance, les problèmes de multiflots en nombres entiers n'ont pas suscité autant d'effort de recherche que leurs homologues en nombres réels. Ceci est dû au fait que, dans certains cas pratiques, la solution continue a constitué une bonne approximation de la solution optimale entière. Mais surtout au fait que les problèmes de multiflots entiers sont autrement plus compliqués à résoudre que les problèmes de multiflots continus. De fait, les problèmes de multiflot maximal et de coût minimum sont des problèmes *NP-difficiles* (Even, Itai et Shamir [1976] ou encore Grötschel, Lovász, et Schrijver [1988]). Nous pourrions alors envisager, pour les résoudre, des méthodes générales de programmation linéaire en nombres entiers (telles que la méthode des coupes ou les méthodes d'évaluation et de séparation). Cependant, il existe d'autres méthodes plus appropriées, dans ce contexte particulier, qui consistent à approcher la solution de ces problèmes de façon plus personnalisée. Parmi ces techniques heuristiques, nous avons retenu deux approches essentielles. La première procède par relaxations successives pour résoudre le problème de

multiflot compatible, avec une extension pour le multiflot de coût minimum. La deuxième approche est une méthode d'approximation déterministe du problème de multiflot maximal qui a été construite à partir d'une version d'approximation probabiliste, basée sur une technique d'arrondi aléatoire.

## II.2.1. MÉTHODE DE RELAXATION POUR LE MULTIFLOT COMPATIBLE ET DE COÛT MINIMUM

L'étude qui va suivre a été réalisée par Minoux [1975] dans le cadre de l'élaboration de programmes de routages pour le réseau de télécommunication français (les détails de l'application peuvent être trouvés dans Minoux [1974].) Le graphe $H=(V,U)$ associé à ce réseau est un graphe non orienté, où les sommets représentent les centres de commutation ou de modulation, et les arêtes, les artères de transmission entre paires de nœuds. Chaque artère (arête) porte des moyens de transmission (faisceau hertzien, câble coaxial, guide d'onde...) de capacité fixée. Les demandes entre couple de nœuds sont représentées par la demande en circuits entre ces deux nœuds. Partant de là, il fallait définir à court terme un programme de routages pour l'année à venir en partant de celui de l'année en cours et en tenant compte des nouvelles capacités installées et des nouvelles demandes en circuits. On appelle routage d'une demande $d^k$ entre deux sommets $i$ et $j$, l'opération qui consiste à affecter à la demande $k$, $d^k$ circuits (au total) pris sur une chaîne joignant $i$ à $j$ dans $H$, pour l'uniroutage, sur plusieurs chaînes pour le multiroutage. Dans ce cas on se heurte au problème de l'admissibilité: comment savoir s'il existe un routage compatible et comment le mettre en évidence. Le problème posé est un problème de multiflot compatible avec une contrainte d'intégralité sur les flots.

### II.2.1.1 Heuristique pour le multiflot compatible

La formulation du problème de multiflot compatible utilisée, est équivalente à la formulation arcs-chemins (I.2) du multiflot de coût minimum, moyennant les modifications suivantes:

- Chaque arc $(i,j)$ est remplacée par son arête correspondante $m$ $(m=1, M)$, $\mathbf{P}^k$ représente l'ensemble des chaînes élémentaires entre $s_k$ et $t_k$, et les chemins $P$ sont remplacés par des chaînes élémentaires.

- les coûts sont nuls.

- on rajoute la contrainte d'intégrité sur les flots.

Introduisons dans chaque contrainte de capacité (I.2c) une variable d'écart $s_m$ et une variable artificielle $y_m$ ( non négatives), et écrivons (I.2c) sous la forme:

$$\sum_k \sum_{P \in \mathbf{P}^k} \delta_m(P) f(P) + s_m - y_m = b_m,$$

$s_m$ représente le nombre de circuits excédentaires sur l'arête $m$ et $y_m$ le nombre de circuits manquants. Dans le cas continu, un routage pour lequel la somme des variables artificielles est nulle constitue donc une solution du problème de l'admissibilité. Pour déterminer une telle solution, on voit qu'il suffit de rechercher une solution optimale du programme linéaire suivant:

$$\text{Minimiser } r = \sum_m y_m \tag{II.20a}$$

sous les contraintes

$$\sum_k \sum_{P \in \mathbf{P}^k} \delta_m(P) f(P) + s_m - y_m = b_m \quad \forall\, m = 1,\, M, \tag{II.20b}$$

$$\sum_{P \in \mathbf{P}^k} f(P) = d^k \qquad \forall\, k = 1,\, K, \tag{II.20c}$$

$$f(P) \geq 0 \qquad \forall\, P \in \mathbf{P}^k,\ \forall\, k = 1,\, K, \tag{II.20d}$$

$$s_m,\, y_m \geq 0 \qquad \forall\, m = 1,\, M. \tag{II.20e}$$

Si la solution $r^*$ de (II.20) est strictement positive , le réseau donné n'est pas compatible ou admissible. L'idée est alors la suivante: minimiser la fonction $r$ (somme des circuits manquants), par relaxations successives, en recherchant à chaque étape le meilleur routage pour une demande $d^k$ $(1 \leq k \leq K)$, les routages des $(K-1)$ autres demandes étant fixés. Deux cas sont possibles: l'uniroutage et le multiroutage. Il est commode par souci de clarté de commencer par l'uniroutage; on rappelle qu'un uniroutage est un routage dans lequel chaque flot s'écoule sur une chaîne unique. Si on se donne pour chaque flot $k$, un routage il est facile de calculer la valeur de $r$ correspondante. Soit $\bar{b}_m$ les capacités résiduelles des arêtes après routage des flots 1 à $K$ sur les chaînes $(R^1, R^2, \ldots, R^K)$ alors:

$$\begin{cases} \bar{b}_m = b_m - \sum_k R^k(m) d^k, \\ r = -\sum_{m/\bar{b}_m < 0} \bar{b}_m. \end{cases}$$

Où $R^k(m)$ est un identificateur arête-chaîne et $r$ peut être considérée comme une fonction $\Phi(R^1, R^2, \ldots, R^K)$ des routages. Mais la recherche du minimum absolu de $\Phi$ est un

problème très combinatoire (même pour un graphe de taille modérée, il existe des milliers de chaînes entre deux nœuds donnés). Par contre, il est facile de trouver le minimum de chacune des différentes restrictions $\Phi^k$ de $\Phi$ à un flot $k$ unique .

Exemple: recherchons le minimum de $\Phi^1(R^1) = \Phi(R^1, R_0^2, ..., R_0^K), R_0^2, ..., R_0^K$ fixés. Les capacités résiduelles $\bar{b}_m$ après routage des flots 2 à $K$ sont:

$$\bar{b}_m = b_m - \sum_{k=2}^{K} R_0^k(m) d^k , \tag{II.21}$$

et la valeur correspondante de $r$ est $r_0 = -\sum_{m/\bar{b}_m<0} \bar{b}_m$.

Cherchons maintenant le meilleur routage possible pour le flot 1. Alors pour une arête $m$ quelconque de $H$:

- si la capacité résiduelle $\bar{b}_m$ calculée en (II.21) est telle que $\bar{b}_m \geq d^1$ alors $r = r_0$;

- si par contre $0 \leq \bar{b}_m < d^1$ alors la valeur de $r_0$ sera augmentée de la quantité $(d^1 - \bar{b}_m)$;

- enfin si $\bar{b}_m < 0$, la valeur de $r_0$ sera augmentée de $d^1$.

Au total le routage du flot 1 sur une chaîne $P$ de $\mathbf{P}^1$ donne:

$$r = r_0 + \sum_{m \in P} \rho_m ,$$

avec:

$$\begin{aligned} &\rho_m = 0 \text{ si } \bar{b}_m \geq d^1, \\ &\rho_m = d^1 - \bar{b}_m \text{ si } 0 \leq \bar{b}_m < d^1, \\ &\rho_m = d^1 \text{ si } \bar{b}_m < 0. \end{aligned} \tag{II.22}$$

Par suite, le meilleur routage du flot 1 est celui qui minimise $\sum_{m \in P} \rho_m$. C'est donc la plus courte chaîne de $s_k$ à $t_k$, les arêtes de $H$ étant munies des longueurs $\rho_m$.

*Algorithme de résolution (cas uniroutage)*

*Pas 1:* Partir d'un routage initial pour chacun des flots $k = 1, K$,
Poser $t = 0$ (compteur d'itérations).

*Pas 2:* $t = t+1$.

*Pas 3:* Pour chaque flot $k = 1, K$

- calculer les capacités résiduelles $\bar{b}_m$ pour tout $m$.
- calculer les longueurs $\rho_m$ pour tout $m$.
- le meilleur routage du flot $k$ est la plus courte chaîne entre $s_k$ et $t_k$ au sens des $\rho_m$.

*Pas 4:* si tous les flots ont été examinés, aller au pas 5 sinon retourner au pas 3.

*Pas 5:* si la valeur de $r$ a été améliorée retourner au pas 2, sinon un optimum (local) de $r$ a été atteint, la procédure s'arrête.

On a évidemment intérêt à choisir un routage de départ donnant à $r$ une valeur aussi faible que possible. Le choix retenu par Minoux [1975] a consisté à examiner les flots les uns après les autres dans l'ordre croissant. Chaque flot $k$ est alors routé sur la plus courte chaîne (en nombre d'arêtes) joignant $s_k$ et $t_k$. Lorsqu'il existe deux chaînes de même longueur, la chaîne de capacité résiduelle maximale est retenue. La convergence de la méthode est facile à démontrer: les capacités et les valeurs des flots étant des nombres entiers, la fonction $r$ est entière à chaque itération. Il suffit alors de remarquer que $r$ est monotone, strictement décroissante au cours de itérations (tant que le critère d'arrêt n'a pas été atteint) et bornée inférieurement par 0 (la solution obtenue n'est évidemment qu'un optimum local de $r$).

### II.2.1.2. Fractionnement des flots et résultats obtenus

La restriction à un uniroutage est une sérieuse limitation. Ainsi au lieu de faire passer les flots sur une seule chaîne, on peut les fractionner sur plusieurs chaînes (multiroutage). Le fractionnement conduit à une meilleure utilisation des capacités et on peut s'attendre par conséquent à obtenir des meilleures valeurs de $r$. Deux types de fractionnement ont été expérimentés:

- le premier décompose un flot de valeur $n$ en $n$ flots de valeur 1 (résultats plus fins mais temps de calcul accru).
- le deuxième est le fractionnement en puissances de 2. Il permet d'avoir moins de fractions. Par exemple pour un flot de 25, on obtient: 1, 2, 4, 8 et 10.

Remarquons que le fractionnement n'entraîne aucune modification de l'algorithme de résolution dans le cas de l'uniroutage: il suffit de traiter chaque fraction de flot comme un flot indépendant.

L'heuristique a été programmée en FORTRAN sur un calculateur BGE 6000. Elle a été expérimentée pour déterminer un plan de routage annuel sur le réseau interurbain français. Le graphe $H$ comportait 87 nœuds, 150 arêtes et 365 flots. Les principales caractéristiques des résultats fournis sont:

- un nombre d'itérations égal à 6 pour obtenir l'admissibilité. Plus de 95% du gain sont obtenus dès la première itération.
- un pourcentage des demandes routées au plus court chemin (en nombres d'arêtes) de 80%.
- un nombre de flots uniroutés: 327, biroutés: 30 et triroutés: 8.

Indépendamment des résultats obtenus sur quelques problèmes réels, une série de 5 problèmes-témoins a été construite sur un graphe comprenant 12 nœuds et 25 arêtes. Les expériences effectués avaient pour objet l'étude de la rapidité de convergence et l'influence du fractionnement et de l'ordre d'examen des demandes sur la qualité du résultat fourni. Pour la convergence on constate que le nombre d'itérations croît avec le fractionnement mais que dans tous les cas ce nombre reste faible (inférieur à 10). Pour ce qui est de la qualité de la solution, les écarts entre la solution fournie et la solution optimale en continu sont assez faibles (voir résultats numériques sur Minoux [1975]).

### II.2.1.3. Extension aux problèmes de multiflots compatibles avec coûts

**a-** Le principe heuristique qui a été présenté se généralise sans difficulté au cas où l'on attacherait plus d'importance à vérifier les contraintes de capacité sur certaines arêtes que sur d'autres. Ceci se fait en affectant à chaque arête $m$, un poids $\pi_m$ et en recherchant:

$$\text{Min } r' = -\sum_{m/\bar{b}_m<0} \pi_m \bar{b}_m .$$

Pour cela il suffit de rechercher pour chaque flot $k$ une plus courte chaîne au sens des longueurs $(\pi_m \rho_m)$, où $\rho_m$ défini en II.22.

**b-** Une autre extension serait d'affecter un coût $c_m > 0$ de passage d'une unité de flot sur l'arête $m$, c'est le problème de multiflot à coût minimum. Pour ce faire, on décide de pénaliser une capacité manquante ($\bar{b}_m < 0$), d'une certaine quantité $c_m + \rho$ (où $\rho$ est un paramètre positif) et on cherche à minimiser la fonction:

$$\text{Minimiser } r'' = \sum_{m/x_m \le b_m} c_m x_m + \sum_{m/x_m > b_m} c_m x_m + \rho(x_m - b_m).$$

Il suffit alors de calculer, à chaque examen de flot, une plus courte chaîne en fonction des nouvelles longueurs $\rho_m$. Par exemple pour le flot $k = 1$:

$$\rho_m = c_m d^1 \text{ si } \bar{b}_m \geq d^1,$$
$$\rho_m = c_m d^1 + \rho(d^1 - \bar{b}_m) \text{ si } 0 \leq \bar{b}_m < d^1,$$
$$\rho_m = (c_m + \rho) d^1 \text{ si } \bar{b}_m < 0.$$

Remarquons que si l'on cherche à résoudre un problème de multiflot de coût minimum classique (c'est à dire sans contrainte d'intégrité) cette heuristique peut fournir une bonne solution de départ à l'algorithme du simplexe.

## II.2.2. APPROXIMATION DÉTERMINISTE DU PROBLÈME DE MULTIFLOT MAXIMUM

Étant donné un graphe $H = (V, U)$ non orienté, $k$ paires de sommets source-puits et des capacités positives , le problème de multiflot maximal consiste à trouver des flots entiers pour autant de paires de sommets source-puits que possible et tels que le flot total transporté des sources aux puits soit maximisé. Tout en sachant que le flot total passant à travers chaque arête $m = \{i, j\}$ appartenant à $U$ est au plus égal à $b_m$. Introduisons des variables $x_{ij}^k$ et $x_{ji}^k$, où $x_{ij}^k$ représente le flot de produit $k$ sur l'arête $\{i, j\}$ de $i$ vers $j$, et inversement. La formulation de ce problème se présente alors comme suit:

$$\begin{aligned} &\max \sum_k \sum_{\{i \in V : \{t_k, i\} \in U\}} x_{t_k i}^k - x_{i t_k}^k \\ &\sum_k x_{ij}^k + \sum_k x_{ji}^k \leq b(\{i, j\}) \quad \forall \{i, j\} \in U, \\ &\sum_{\{i \in V : \{i, j\} \in U\}} x_{ij}^k = \sum_{\{i \in V - \{s_k, t_k\} : \{i, j\} \in U\}} x_{ij}^k \\ &\qquad \forall k, \ \forall j \in V - \{s_k, t_k\}. \end{aligned} \tag{II.23}$$

Les résultats d'approximation précédents concernaient uniquement les flots en 0-1. La meilleure approximation est donnée, dans ce cas, par Raghavan[1987] qui met en évidence un facteur constant et implicite d'approximation lorsque $c_{min} = \Omega(log|E|)$, où $c_{min}$ représente la capacité minimale. L'algorithme d'approximation déterministe que nous présentons ici (Srivastav et Stangier [1993b]), s'applique au cas entier général et montre que pour une large classe d'instances du problème, une assez bonne approximation des multiflots entiers peut être construite en un temps polynomial, en mettant à jour des facteurs d'approximation constants et explicites.

### II.2.2.1. Technique d'arrondi aléatoire

La solution fractionnaire $f_R$ (correspondant à la relaxation continue du problème en nombres entiers) peut être construite en un temps polynomial par les algorithmes non standard de programmation linéaire (voir Grötschel, Lovász et Schrijver [1988]). Nous représentons dans ce qui suit les flots fractionnaires (et plus tard les flots entiers) par des chemins orientés. Soit $\Gamma = \{P_1, ..., P_S\}$ l'ensemble des chemins, où chaque chemin $P \in \Gamma$ est un $(s_k - t_k)$ chemin pour le produit $k$. A chaque chemin $P \in \Gamma$, on associe $\lambda(P)$, un entier positif ou un nombre rationnel dans le cas des flots fractionnaires. La valeur du flot de produit $k$ est égale à la somme des $\lambda(P)$ pour lesquels $P$ est un $(s_k - t_k)$ chemin. Ces chemins peuvent être construits en un temps polynomial grâce à des méthodes classiques (voir Malhotra, Kumar et Maheshwari [1978] ou Raghavan et Thompson [1987]).

Posons $\Gamma_k$ l'ensemble des chemins représentant le produit $k$, on a alors $\Gamma = \mathbf{U}_k \Gamma_k$. L'algorithme suivant montre que pour chaque chemin on introduit au plus $O(\frac{1}{\varepsilon^2} log|E|)$ variables en 0-1. Avec comme entrées: l'ensemble des chemins $\Gamma$, les valeurs associées $\lambda(P)$, $P \in \Gamma$ et un nombre rationnel $0 < \varepsilon \leq \frac{1}{2}$. En sortie nous récupérons les nouvelles valeurs associées aux chemins: $\lambda(P)$, $\lambda_0(P)$, ..., $\lambda_{N(P)}(P)$ pour chaque $P \in \Gamma$. Cette stratégie d'arrondi judicieuse a été introduite par Srivastav et Stangier [1993c].

*Algorithme VALCHEMIN* ( $\varepsilon$)

*Pas 0:* Poser $c_\varepsilon = \frac{6(2-\varepsilon)}{\varepsilon^2} \lceil log 2|U| \rceil$, $m = 0$

Pour chaque $P \in \Gamma$ poser: $\begin{cases} \lambda_0(P) = \lambda(P) - \lfloor \lambda(P) \rfloor, \\ \lambda(P) = \lambda(P) - \lambda_0(P) \text{ et } N(P) = 0. \end{cases}$

*Pas 1:* $m = m + 1$

Si $b_m - \sum_{m \in P \in \Gamma} \lambda(P) < c_\varepsilon$ alors:

Choisir $k$, $P \in \Gamma_k$ avec $m \in P$ et $\lceil \lambda(P) \rceil \geq 1$.

poser $\lambda(P) = \lambda(P) - 1$, $\lambda_{N(P)+1}(P) = \lambda_{N(P)+2}(P) = 0.5$ et $N(P) = N(P) + 2$.

Sinon aller au pas 2.

*Pas 2:* Si $m = M$, arrêter sinon retourner au pas 1.

Cet algorithme donne une représentation du multiflot avec au plus $O(|U||K|\varepsilon^{-2})$ valeurs de chemins et la propriété supplémentaire que $\lambda(P)$ initial est décrémenté de façon à ce que pour tout $m$:

$$b_m - \sum_{m \in P \in \Gamma} \lambda(P) \geq c_\varepsilon \qquad \text{(II.23)}$$

Pour arrondir aléatoirement, on procède comme suit:

*Algorithme ARRONDI* ( $\varepsilon$)

Soient $0 < \varepsilon \leq \frac{1}{2}$, $P \in \Gamma$, $\lambda(P)$ et les $\lambda_i(P)$ $(i = 0, 1, ..., N(P))$ générés par *VALCHEMIN* ( $\varepsilon$).

$\forall\, P \in \Gamma$, $\forall\, i = 0, 1, ..., N(P)$, faire indépendamment:

1- Poser:

- $\lambda_i^I(P) = 1$ avec la probabilité $(1 - \frac{\varepsilon}{2})\lambda_i(P)$.

- $\lambda_i^I(P) = 0$ avec la probabilité $1 - (1 - \frac{\varepsilon}{2})\lambda_i(P)$.

2- $\forall\, P \in \Gamma$ poser $\lambda^I(P) = \lambda(P) + \sum_{i=0}^{N(P)} \lambda_i^I(P)$.

Le flot total généré par cet algorithme est $f = \sum_{P \in \Gamma} \lambda^I(P)$. Nous allons prouver que ce flot est réalisable (au sens de la contrainte de capacité) et que c'est une bonne approximation du flot optimal $f_{opt}$ et ce, en faisant appel à l'inégalité de Angluin-Valiant [1979].

***Lemme II.10.*** (McDiarmid [1989]) *Soient $a_1, ..., a_n$ des nombres réels tels que $0 \leq a_j \leq 1$ pour tout $j$, $\psi_1, ..., \psi_n$ des variables aléatoires indépendantes en 0-1.*
*Soient* $\tilde{p}_j = E(\psi_j)$, $\tilde{q}_j = 1 - \tilde{p}_j$, $\psi = \sum_{j=1}^{n} a_j \psi_j$, $p = \frac{1}{n} E(\psi)$, $q = 1 - p$ *et* $0 < \beta < 1$.
*Alors on a:*

(i) $Prob(\psi > (1+\beta)np) \leq e^{-\frac{\beta^2 np}{3}}$.

(ii) $Prob(\psi < (1-\beta)np) \leq e^{-\frac{\beta^2 np}{2}}$.

Le lemme suivant montre que le flot est réalisable (Srivastav et Stangier [1993b]).

***Lemme II.11.*** *Soit* $0 < \varepsilon \leq \frac{1}{2}$ *et supposons que* $c_{min} \geq \frac{6(2-\varepsilon)}{\varepsilon^2}\lceil log 2|U| \rceil$. *Alors pour chaque* $P \in \Gamma$, *ARRONDI(* $\varepsilon$*) trouve une valeur entière du chemin* $\lambda^I(P)$ *telle que pour tout m/* $\sum_{m \in P \in \Gamma} \lambda^I(P) \leq b_m$ *et ce, avec une probabilité supérieure ou égale à* $\frac{1}{2}$.

*Preuve* Nous arrondissons seulement une partie de chaque valeur de chemin, par conséquent nous devons réduire les capacités sur les arêtes. On définit $b_m^{red}$ par:

$$b_m^{red} = b_m - \sum_{m \in P \in \Gamma} \lambda(P) \qquad \text{(II.24)}$$

Pour chaque arête $m$, on a alors la variable aléatoire suivante:

$$\chi_m = \sum_{m \in P \in \Gamma} \sum_{i=0}^{N(P)} \lambda_i^I(P).$$

Le calcul de l'espérance mathématique montre que:

$$E(\chi_m) \leq (1 - \tfrac{\varepsilon}{2}) b_m^{red}.$$

En prenant $\beta = \frac{\varepsilon}{2-\varepsilon}$, en utilisant (II.24) ainsi que l'inégalité d'Angluin-Valiant (Lemme II.10(i)), on obtient:

$$Prob(\chi_m > b_m^{red}) = Prob(\chi_m > (1+\beta)(1-\tfrac{\varepsilon}{2}) b_m^{red}) \leq \frac{1}{2|U|}$$

Cette inégalité en conjonction avec (II.24) impliquent qu'avec une probabilité de au moins $\frac{1}{2}$:

$$\begin{aligned} \sum_{m \in P \in \Gamma} \lambda^I(P) &= \sum_{m \in P \in \Gamma} \lambda(P) + \chi_m \\ &\leq \sum_{m \in P \in \Gamma} \lambda(P) + b_m^{red} \\ &= b_m. \end{aligned}$$

Prouvons maintenant la garantie d'approximation (Srivastav et Stangier [1993b]):

***Lemme II.12.*** *Soit* $0 < \varepsilon \leq \frac{1}{2}$. *Alors avec une probabilité supérieure ou égale à* $\frac{3}{4}$ *ARRONDI(* $\varepsilon$*) trouve pour chaque* $P \in \Gamma$, *une valeur de chemin entière* $\lambda^I(P)$ *telle que pour le flot total* $f = \sum_{P \in \Gamma} \lambda^I(P)$, on a $f \geq (1-\varepsilon) f_{opt}$.

*Preuve* Soit $g_{red}$ la partie résiduelle fractionnaire du flot total définie par:

$$g_{red} = f_R - \sum_{P \in \Gamma} \lambda^I(P)$$

Définissons la variable aléatoire représentant le flot arrondi par:

$$g = \sum_{P \in \Gamma} \sum_{i=0}^{N(P)} \lambda_i^I(P)$$

Nous avons alors:

$$E(g) = (1 - \tfrac{\varepsilon}{2}) g_{red}.$$

Avec $\lambda = \sqrt{\frac{4}{(2-\varepsilon)g_{red}}}$ et l'inégalité d'Angluin-Valiant (Lemme II.10(ii)), on montre que:

$$Prob(g < (1-\varepsilon)g_{red}) \leq \frac{1}{4}.$$

Ceci implique qu'avec une probabilité supérieure ou égale à $\frac{3}{4}$:

$$f = \sum_{P \in \Gamma} \lambda^I(P) = \sum_{P \in \Gamma} \lambda(P) + g$$
$$\geq (1-\varepsilon) f_R$$

En combinant les lemmes (II.11) et (II.12) on a:

***Théorème II.13.*** *Soit* $0 < \varepsilon \leq \frac{1}{2}$ *et supposons que* $c_{min} \geq \frac{6(2-\varepsilon)}{\varepsilon^2} \lceil log 2|U| \rceil$. *Alors avec une probabilité supérieure ou égale à* $\frac{1}{4}$ *ARRONDI(* $\varepsilon$*) trouve un multiflot entier F, qui respecte la capacité b et dont la valeur totale f est telle que:* $f \geq (1-\varepsilon) f_{opt}$.

### II.2.2.2. Génération déterministe du flot

Nous allons maintenant donner une version déterministe du théorème II.4. Cette version s'inspire essentiellement de la version déterministe algorithmique de l'inégalité d'Angluin-Valiant. Dans le cas des multiflots en 0-1, Raghavan [1988] a déjà établi une telle version en construisant ce que l'on appelle des estimateurs pessimistes de certaines probabilités conditionnelles.

Le principal inconvénient de cette version apparaît dans le cas où la somme des variables de Bernouilli est pondérée par des nombres rationnels. Dans ce cas, en effet, le calcul des estimateurs pessimistes requiert l'exponentiation de nombres rationnels à des puissances elles-mêmes rationnelles, ce qui ne peut donner lieu à un algorithme efficace.

Srivastav et Stangier [1993b] contournent ce problème en construisant efficacement des polynômes qui constituent une nouvelle classe d'estimateurs pessimistes. Le théorème utilisé peut être trouvé dans Srivastav, Stangier [1993a].

Soient $X_1, \ldots, X_n$, $n$ variables de Bernouilli définies par: $Prob(X_i = 1) = \tilde{x}_i$ et $Prob(X_i = 0) = 1 - \tilde{x}_i$ pour un nombre rationnel $0 \le \tilde{x}_i \le 1$. Soient $w_{ij}$ un ensemble de poids rationnels, $1 \le i \le m$, $1 \le j \le n$, $0 \le \omega_{ij} \le 1$, et notons $\psi_i$ les variables aléatoires suivantes:

$$\psi_i = \sum_{j=1}^{n} \omega_{ij} X_j$$

Soient $p_i = \frac{E(\psi_i)}{n}$, $q_i = 1 - p_i$ et $\beta_i$ un nombre rationnel tel que $0 \le \beta_i \le 1$, $\forall\, 1 \le i \le m$. Notons $E_i^+$ et $E_i^-$ les événements respectifs:

" $\psi \le (1+\beta_i)E(\psi_i)$ " et " $\psi \ge (1-\beta_i)E(\psi_i)$ ".

Et par suite posons $E = E_1 \wedge \ldots \wedge E_m$ , où $E_i$ est $E_i^+$ ou $E_i^-$.

A chaque $i$, associons un nombre rationnel $\delta_i$, $0 < \delta_i \le 1$ avec la propriété que:

Si $E_i$ est l'événement " $\psi \ge (1+\beta_i)E(\psi_i)$ " alors $\exp\left(-\frac{\beta^2 E(\psi_i)}{3}\right) \le \delta_i$.

Si $E_i$ est l'événement " $\psi \le (1-\beta_i)E(\psi_i)$ " alors $\exp\left(-\frac{\beta^2 E(\psi_i)}{2}\right) \le \delta_i$.

L'inégalité d'Angluin-Valiant montre alors que l'événement $E$ est réalisé avec une probabilité d'au moins $1 - \sum \delta_i$. Le problème qu'on se pose alors revient à trouver un vecteur $x \in \{0,1\}^n$ de façon déterministe, en un temps polynomial, et tel que l'événement $E$ soit réalisé. Pour le prouver, nous construisons itérativement un vecteur $(x_1, \ldots, x_n)$ et des fonctions $U_1(x_1), \ldots, U_n(x_1, \ldots, x_n)$ calculables en un temps polynomial et telles que:

(a) $U_1(x_1) \ge U_2(x_1, x_2) \ge \ldots \ge U_n(x_1, \ldots, x_n)$

(b) $\Pr ob(E^c | x_1, \ldots, x_k) \le U_k(x_1, \ldots, x_k)$

(c) $U_1(x_1) < 1$.

De telles fonctions sont appelées estimateurs pessimistes (Raghavan [1988]). On voit alors que étant données ces estimateurs pessimistes, le vecteur $x = (x_1, \ldots, x_n)$ est le vecteur recherché, suite aux conditions (a), (b) et (c):

$$\Pr ob(E^c | x_1, \ldots, x_n) < 1,$$

par conséquent

$$\Pr ob(E^c | x_1, \ldots, x_n) = 0.$$

Ce qui implique que $x \in E$.

Les candidats potentiels pour de tels estimateurs sont des fonctions exponentielles qui paraissent dans l'inégalité d'Angluin-Valiant (voir McDiarmid [1989]) et qui prennent la forme suivante:

Soit le vecteur $(x_1, \ldots, x_l) \in \{0,1\}^l$. Définissons $V_l(x_1, \ldots, x_l)$ comme suit:

Si $E_i$ est l'événement " $\psi \geq (1+\beta_i)E(\psi_i)$ " alors définissons $s_i = \frac{q_i(1+\beta_i)}{q_i - p_i\beta_i}$ et $V_l(x_1, \ldots, x_l) = e^{-(1+\beta_i)p_i n \ln s_i} e^{\sum_{j=1}^{l} w_{ij} x_j \ln s_i} \times \prod_{j=l+1}^{n} \left[\tilde{p}_j e^{w_{ij} \ln s_i} + 1 - \tilde{p}_j\right]$.

Si $E_i$ est l'événement " $\psi \leq (1-\beta_i)E(\psi_i)$ " alors définissons $s_i = \frac{q_i + p_i\beta_i)}{q_i(1-\beta_i)}$ et $V_l(x_1, \ldots, x_l) = e^{-(1-\beta_i)p_i n \ln s_i} e^{\sum_{j=1}^{l} w_{ij} x_j \ln s_i} \times \prod_{j=l+1}^{n} \left[\tilde{p}_j e^{-w_{ij} \ln s_i} + 1 - \tilde{p}_j\right]$.

On peut voir que ces fonctions ne sont pas calculables de manière efficace. On construit alors $U_1, \ldots, U_n$ en deux étapes:

1. L'étape facile consiste à montrer que les fonctions $V_1, \ldots, V_n$ définies par $V_l = \min(V_l(x_1, \ldots, x_{l-1}, 1), V_l(x_1, \ldots, x_{l-1}, 0))$ possèdent les propriétés (a), (b) et (c).

2. Le problème principal consiste alors à approximer $V_1, \ldots, V_n$ par des polynômes de faible degré et de construire les $U_i$ en utilisant ces polynômes de façon à satisfaire les conditions (a), (b) et (c).

Le passage de l'aléatoire au déterministe se fait alors par le théorème de base de Srivastav et Stangier [1993a]:

***Théorème II.14.*** *Soit l'événement* $E = E_1 \wedge \ldots \wedge E_m$ *où* $E_i$ *dénote soit* $E_i^+$ *soit* $E_i^-$ *(définis plus haut). Soit* $\delta$ *un nombre rationnel tel que*: $0 < \delta < 1$ et $\delta + \sum_{i=1}^{m} \delta_i < 1$.

*Supposons que* $\beta_i$, $p_i \leq \frac{n-1}{n}$, $\forall\, i$. *Alors on peut construire un vecteur* $x \in \{0,1\}^n$ *en un temps* $O\left(mn^2 \log \frac{m}{\delta}\right)$ *et tel que l'événement E soit réalisé.*

A partir de là, la version déterministe du théorème II.13 s'écrit alors:

***Théorème II.15.*** *Soit* $0 < \varepsilon \leq \frac{1}{2}$ *avec* $c_{min} \geq \frac{6(2-\varepsilon)}{\varepsilon^2}\lceil log\, 2|U| \rceil$. *Alors nous pouvons, en un temps polynomial, construire un multiflot F entier qui respecte b et dont la valeur totale f est telle que* $f \geq (1-\varepsilon) f_{opt}$.

Pour la preuve voir Srivastav et Stangier [1993b].

Et pour $\varepsilon = \frac{1}{2}$:

***Corollaire II.16.*** *Soit* $c_{min} \geq 36\lceil log\, 2|U| \rceil$ *alors on peut trouver en un temps fortement polynomial un multiflot F entier qui respecte la capacité b et dont la valeur totale f est telle que* $f \geq \frac{1}{2} f_{opt}$.

## II.3. TRANSFORMATION D'UN MULTIFLOT EN FLOT SIMPLE

Nous avons vu dans les précédents développements que les opérations de pivotage du simplexe qu'on peut effectuer directement sur des réseaux de flot simple (sans nécessité d'inversion explicite de matrice), ne peuvent plus être reproduites sur des réseaux de multiflots. Par ailleurs, le problème de multiflot en nombres entiers est NP-difficile comme nous l'avons souligné en § II.2. De fait, la structure totalement unimodulaire de la matrice des contraintes (chaque sous matrice ayant un déterminant égal à +1, -1, ou 0) dans le cas du flot simple, n'est pas conservée pour le multiflot, d'où la quasi-prépondérance de solutions non entières (voir Ford et Fulkerson [1962]). Il paraît donc très avantageux de pouvoir transformer des problèmes de multiflots en problèmes de flot simple afin de pouvoir les résoudre en tant que tels. Nous présentons dans ce qui suit une condition suffisante établie par Evans [1978], permettant de reformuler un problème de multiflot de coût minimum en problème de flot simple. Cette condition est basée sur la structure topologique du réseau et permet d'unifier des résultats antérieurs dans une structure théorique plus générale. Il s'agit en l'occurrence des résultats de Evans, Jarvis et Duke [1975] qui ont montré que le problème de transport à 2 sources, $n$ puits et $r$ produits ($n \geq 2$, $r \geq 2$) avaient une représentation équivalente en flot simple (une extension de ces résultats est donnée dans Evans [1976a]). Des résultats similaires pour certains problèmes

de multiflot concernant la planification de la production sur des réseaux dynamiques, avaient été établis par Evans [1976b, 1977].

## II.3.1. CONDITION SUFFISANTE

Reprenons le graphe orienté $G = (V, E)$ défini précédemment. Pour chaque produit $k$ nous avons une partition des sommets de $G$ en trois sous-ensembles $S_k$, $I_k$ et $T_k$ dénotant respectivement les sources, les nœuds intermédiaires et les destinations du flot $k$. Pour $i \in V$ posons $d^k(i)$ comme le montant de produit $k$ au nœud $i$. Il s'en suit que $d^k(i) > 0$ si $i \in S_k$, $d^k(i) = 0$ si $i \in I_k$ et $d^k(i) < 0$ si $i \in T_k$. Supposons, sans perte de généralité, que pour chaque produit, l'offre totale est égale à la demande totale i.e. $\sum_{i \in V} d^k(i) = 0$. Nous pouvons alors reformuler le problème de multiflot de la façon suivante:

$$\text{Minimiser} \sum_k \sum_{(i,j) \in E} c^k(i,j) x^k(i,j) \qquad \text{(II.25a)}$$

sous les contraintes

$$x^k(i,V) - x^k(V,i) = d^k(i) \qquad \forall\, i \in V,\ \forall\, k, \qquad \text{(II.25b)}$$

$$\sum_k x^k(i,j) + s(i,j) = b(i,j) \qquad \forall\, (i,j) \in E, \qquad \text{(II.25c)}$$

$$x^k(i,j),\ s(i,j) \geq 0 \qquad \forall\, (i,j) \in E,\ \forall\, k. \qquad \text{(II.25d)}$$

Avec $x^k(i,V) = \sum_{j \in V} x^k(i,j)$. Soit $i_0 \in V$, définissons $E(i_0)$ comme l'ensemble des arcs $\{(i,j) \in E \mid i \text{ ou } j = i_0\}$, $\overline{E}(i_0) = E - E(i_0)$ et écrivons un nouveau problème comme suit:

$$\text{Minimiser} \sum_k \sum_{(i,j) \in E} c^k(i,j) x^k(i,j) \qquad \text{(II.26a)}$$

sous les contraintes

$$\sum_k x^k(i_0,j) + s(i_0,j) = b(i_0,j) \qquad \forall\, (i_0,j) \in E(i_0), \qquad \text{(II.26b)}$$

$$-\sum_k x^k(j,i_0) - s(j,i_0) = -b(j,i_0) \qquad \forall\, (j,i_0) \in E(i_0), \qquad \text{(II.26c)}$$

$$x^k(i,V) - x^k(V,i) = d^k(i) \qquad \forall\, i \neq i_0 \text{ et } i \in V,\ \forall\, k, \qquad \text{(II.26d)}$$

$$s(i,V) - s(V,i) = b(i,V) - b(V,i) - \sum_k d^k(i) \qquad \forall\, i \neq i_0 \text{ et } i \in V, \qquad \text{(II.26e)}$$

$$x^k(i,j),\ s(i,j) \geq 0 \qquad \forall\ (i,j) \in E,\ \forall\ k. \tag{II.26f}$$

Le problème (II.26) est un problème de flot simple dans la mesure où chaque variable figure dans exactement deux contraintes avec des signes opposés. Par ailleurs (II.26) est une relaxation de (II.25) dans la mesure où les contraintes suivantes ne sont pas explicitées:

$$\sum_k x^k(i,j) + s(i,j) = b(i,j) \quad \forall\ (i,j) \in \overline{E}(i_0) \tag{II.27}$$

$$x^k(i_0, V) - x^k(V, i_0) = d^k(i_0) \qquad \forall\ k. \tag{II.28}$$

Il est évident qu'une solution de (II.26) est aussi solution de (II.25) si et seulement si les équations (II.27) et (II.28) sont satisfaites. Nous pouvons alors énoncer le théorème suivant:

***Théorème II.17.*** *Une solution optimale de* (II.26) *résout* (II.25) *s'il existe une permutation des arcs de* $\overline{E}(i_0)$, $(i_1, j_1)$, ..., $(i_n, j_n)$ *telle que:*

*(i)* $(i_1, j_1)$ *est incident à un nœud dont le restant des arcs incidents est dans* $E(i_0)$ *uniquement et*

(ii) $(i_p, j_p)$ *est incident à un nœud dont le restant des arcs incidents est dans* $E(i_0) \cup J_p$ pour $p = 2, \ldots, n$, où $J_p = \{(i_q, j_q) |\ q < p\}$.

*Dans ce cas on dit que* (II.25) *et* (II.26) *sont équivalents.*

Pour la preuve consulter Evans [1978]. Les conditions établies dans le théorème sont suffisantes mais non nécessaires. Une solution du problème (II.26) peut satisfaire l'équation (II.27), l'équation (II.28) étant toujours vérifiée pour tout $k$ (en sommant les équations (II.26d) sur tous les sommets $i \neq i_0$ et en utilisant l'hypothèse que $\sum_{i \in V} d^k(i) = 0$), sans que les conditions (i) et (ii) ne soient vérifiées. A ce moment là la solution optimale du problème (II.26) est aussi solution optimale du problème (II.25). Néanmoins si les conditions (i) et (ii) ne sont pas vérifiées, les flots optimaux $x^{k^*}(i,j)$ de (II.26) sont aussi solution du problème suivant:

$$\text{Minimiser} \sum_k \sum_{(i,j) \in E} c^k(i,j) x^k(i,j) \tag{II.29a}$$

sous les contraintes

$$x^k(i,V) - x^k(V,i) = d^k(i) \qquad \forall\ i \in V,\ \forall\ k, \tag{II.29b}$$

$$\sum_k x^k(i_0, j) \le b(i_0, j) \quad \forall\, (i_0, j) \in E(i_0), \tag{II.29c}$$

$$\sum_k x^k(j, i_0) \le b(j, i_0) \quad \forall\, (j, i_0) \in E(i_0), \tag{II.29d}$$

$$\sum_k x^k(i, j) \le \sum_k x^{k^*}(i, j) \quad \forall\, (i, j) \in \overline{E}(i_0), \tag{II.29e}$$

$$x^k(i, j) \ge 0 \quad \forall\, (i, j) \in E. \tag{II.29f}$$

Ceci se rapproche des résultats de Everett [1963] concernant les multiplicateurs généralisés de Lagrange et peut être utilisé dans l'évaluation de l'irréalisabilité surtout dans les cas où les valeurs des capacités sont incertaines ou sujettes à contrôle.

### II.3.2. Algorithme de caractérisation

Nous pouvons facilement remarquer que les caractérisations données précédemment dans le théorème II.8 suggèrent des transformations directes dans le cas d'un problème de transport avec 2 sources, $n$ puits, $r$ produits (Figure II.9), ainsi que dans celui d'un problème de planification dynamique de la production sur un horizon à une seule période (Figure II.10).

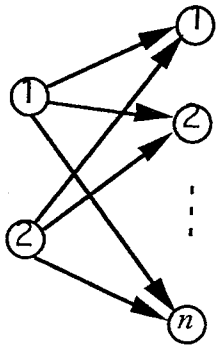



**Figure II.9** Réseau de transport à 2 sources.

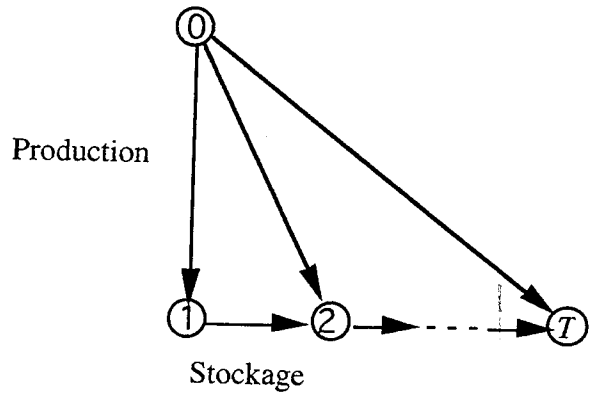



**Figure II.10** Réseau de production dynamique à une seule période

Dans le premier cas $i_0$ peut être représenté par l'une ou l'autre des deux sources, dans le deuxième cas par le sommet 0. Ceci nous amène alors à nous poser la question suivante: comment savoir si un réseau arbitraire satisfait les conditions du théorème II.17?

L'algorithme suivant donne alors la réponse, en offrant l'avantage d'être facile à implémenter.

*Algorithme de caractérisation d'un multiflot en flot simple*

*Pas 0 : (Initialisations)*
Tous les nœuds et arcs n'étant pas marqués, choisir un nœud avec le plus grand nombre d'arcs incidents (sinon on pourra choisir un nœud quelconque), appeler ce nœud $i_0$ et marquer tous les nœuds et arcs incidents à ce sommet.

*Pas 1 : (Itérations)*
Choisir arbitrairement un sommet marqué. Si au plus un arc incident est non marqué, marquer cet arc et le deuxième nœud qui lui est incident et répéter l'itération. Si cette condition n'est pas vérifiée choisir un autre nœud marqué et répéter. Si aucun nœud marqué ne vérifie cette condition, arrêter (les conditions du théorème ne sont pas satisfaites). Dans ce cas supprimer $i_0$ de la liste des sommets à examiner, choisir un autre sommet parmi les sommets restants et recommencer à partir du pas 0. Toutes ces opérations se répètent jusqu'à ce qu'un sommet vérifie les conditions du théorème II.17, ou que tous les sommets aient été examinés.

Le pas 0 consiste à marquer tous les arcs de l'ensemble $E(i_0)$. La première application de l'étape 1 recherche un arc satisfaisant la condition (i). Si aucun arc n'existe alors (i) n'est pas satisfaite. La répétition du pas 1 correspond à la recherche d'arcs satisfaisant la condition (ii). Nous devons juste montrer que le choix des nœuds marqués au pas 1 peut se faire des façon arbitraire. En fait ceci découle du fait que chaque fois qu'un arc est marqué, le nombre d'arcs non marqués incidents à un nœud quelconque ne peut augmenter. Donc par induction, la procédure est justifiée.

## II.4. CONCLUSION

L'importance des problèmes de multiflots se confirme, encore ici, à travers la variété des approches proposées pour les résoudre. Nous avons envisagé ces approches dans les deux cas de figure possibles, en l'occurrence le cas continu et le cas entier. Dans le cadre des multiflots fractionnaires, nous avons présenté les techniques classiques de résolution que sont: l'approche de décomposition par les prix, l'approche de décomposition par les ressources et les méthodes de partitionnement. Nous avons présenté en plus, un autre algorithme de décomposition mixte adapté au cas particulier des multiflots. Sur un plan

pratique, les deux approches les plus prometteuses sont représentées par les deux approches de décomposition par les prix et de partitionnement. Les codes développées dans le cadre des techniques spécialisées pour les multiflots peuvent être deux à cinq fois plus rapides que les codes généraux de programmation linéaire. Précisons tout de même que les performances du simplexe ont été largement améliorées par l'utilisation d'une bonne base de départ (Bixby [1991]). En dehors de ces techniques de résolution classiques, nous avons évoqué de nouvelles approches de résolution destinées surtout à résoudre des problèmes de multiflots de taille importante (impliquant un plus grand nombre de flots de produits). Mais pendant que nous assistons à la recrudescence de ces nouvelles approches, de récents développements mettent l'accent sur un autre aspect de l'optimisation des multiflots, à savoir l'influence de la formulation sur l'efficacité de certaines méthodes de décomposition. Ces recherches redonnent leur crédibilité aux méthodes de décomposition, qui peuvent être appliquées si le "bon" problème est résolu, en utilisant la technologie d'optimisation actuellement disponible.

Dans le cas des multiflots entiers, les techniques de résolution sont beaucoup moins performantes. Actuellement, le meilleur algorithme d'approximation met en évidence un facteur d'approximation déterministe de $\frac{1}{2}$ pour le problème de multiflot maximal.

Finalement, toute cette discussion a permis de confirmer le fait établi que les problèmes de multiflots sont beaucoup plus difficiles à résoudre que les problèmes de flots simple. Ceci a donné une bonne motivation à la dernière partie de ce chapitre, consacrée notamment à encourager les procédures d'identification d'un problème de multiflot en problème de flot simple, surtout si on garde à l'esprit l'efficacité relative des algorithmes de flot simple par rapport aux algorithmes de multiflots.

# Chapitre III

# CAS PARTICULIER : LE BIFLOT

## III.0. INTRODUCTION

Le problème de biflot constitue le cas particulier des problèmes de multiflots avec seulement deux flots indépendants. Comme application, on peut considérer le cas des réseaux de télécommunication où le téléphone et le télex, par exemple, partagent les mêmes lignes de transmission. Le flot maximum indique alors le nombre maximum de bits d'information qui peuvent passer à travers le réseau. L'intérêt du biflot réside dans le fait que d'une part, il permet de modéliser de nombreux cas réels (on peut citer, entre autres, le modèle de biflot élaboré par Helgason, Kennington et Wong [1981], Kirby, Hager et Wong [1982] pour un problème d'aménagement de forêts, ou encore celui de Finke, Claus et Gunn [1984] pour le problème du voyageur de commerce), et que d'autre part, la spécificité de sa structure donne lieu à des résultats théoriques qu'on ne peut pas toujours étendre aux autres cas de multiflots. Ce chapitre sera donc consacré à l'étude du problème de biflot dans sa version non orientée puis orientée, et ce, en termes de formulations, de méthodes de résolution et de résultats théoriques. Précisons qu'en parallèle, nous donnerons quelques généralisations de certains de ces résultats à d'autres problèmes de multiflots, notamment sur des réseaux à structure très particulière.

## III.1. BIFLOT NON ORIENTÉ

Nous allons envisager les deux problèmes d'optimisation que sont le problème de biflot maximal, puis de coût minimum. Nous donnerons, dans les deux cas, une formulation

mathématique du problème, une méthode de résolution associée et les principaux résultats théoriques existant pour chaque problème. Dans le cadre du biflot de coût minimum, nous aborderons les conditions de réalisabilité pour ce type de problème, puis nous finirons par quelques notions de complexité concernant le problème de biflot en nombres entiers.

## III.1.1. BIFLOT MAXIMAL

### III.1.1.1. Formulation et algorithme de résolution

On définit un problème de biflot non orienté en se donnant:

- un graphe non orienté $G=(V, E)$ $(|V| = N,\ |E| = M)$ sans arêtes parallèles et sans boucles; une arête entre $i$ et $j$ est notée $[i, j]$,
- une fonction capacité $b$ sur les arcs (réels non négatifs); on notera $b[i, j]$ la capacité de l'arête $[i, j]$,
- deux sommets $s_1$ et $s_2$ (pas nécessairement distincts) appelés sources; deux sommets $t_1$ et $t_2$ (pas nécessairement distincts) appelés destinations.

Comme le graphe n'est pas orienté $[i, j] = [j, i]$. Cependant les flots et les chemins sont orientés. On utilisera la notation (., .) pour les paires ordonnées.

Le problème consiste alors à trouver deux fonctions flot réalisables, telles que le flot total soit maximisé; soit la formulation suivante:

$$\text{Maximiser } d^1 + d^2 \tag{III.1a}$$

sous les contraintes

$$|x^1(i,j)| + |x^2(i,j)| \le b[i,j] \qquad \forall\ (i,j) \in E, \tag{III.1b}$$

$$\sum_{i \in V} x^k(i,j) = 0 \qquad \forall\ k = 1,2 \text{ et } \forall\ j \in V - \{s_k, t_k\}. \tag{III.1c}$$

Avec $x^k(i, j)$ dénotant la quantité de flot $k$ $(k = 1, 2)$ passant à travers l'arête $(i, j)$; notons que si le flot passe de $i$ à $j$, $x^k(i, j) > 0$ et $x^k(j, i) = -x^k(i, j) < 0$. Le flot total est défini par $d^k = \sum_{j \in V} x^k(s_k, j)$ pour $k = 1, 2$.

On retrouve la contrainte de capacité (III.1b) avec les notations $|x^1(i, j)|$, $|x^2(i, j)|$ qui désignent la valeur absolue du flot. En effet, dans le cas non orienté, le flot est de signe quelconque et les différents flots s'ajoutent le long d'une arête indépendamment de leur sens de circulation. La contrainte de conservation du flot pour chaque produit $k$ prend la

forme (III.1c) exprimant que la somme des flots rentrants, en chaque sommet, est égale à celle des flots sortants.

Pour maximiser $d^1 + d^2$, on maximise d'abord $d^1$ indépendamment de $x^2$ ($x^2 = 0$). On augmente ensuite $d^2$ sans changer $d^1$ (même si $x^1$ peut changer sur certaines arêtes). L'algorithme qui en découle (Itai [1978]) s'inspire de celui de Hu [1963] et procède en trouvant des paires de chemins augmentants.

Pour tout $(i,j) \in E$, on définit $\alpha(i,j) = \frac{1}{2}\left(b[i,j] - x^1(i,j) - x^2(i,j)\right)$, $2\alpha(i,j)$ est alors la borne supérieure sur le flot additionnel qu'on peut envoyer de $i$ à $j$ à travers l'arête $[i,j]$. Un chemin "aller" est un $(s_2 - t_2)$ chemin simple $\pi_\alpha = (s_2 = i_0, \ldots, i_r = t_2)$ tel que:

$$\alpha(i_l, i_{l+1}) > 0,\ l = 0, \ldots, r-1.$$

La capacité résiduelle d'un chemin aller est $\alpha(\pi_\alpha) = \min\left\{\alpha(i,j)/(i,j) \in \pi_\alpha\right\}$.

De la même manière définissons $\beta(i,j) = \frac{1}{2}\left(b[i,j] - x^1(i,j) + x^2(i,j)\right)$, $2\beta(i,j)$ est alors la borne supérieure sur le flot additionnel qui peut être envoyé de $i$ à $j$ pour $x^1$ et de $j$ à $i$ pour $x^2$. Un chemin "retour" est un $(t_2 - s_2)$ chemin simple $\pi_\beta = (t_2 = i_0, \ldots, i_q = s_2)$ tel que:

$$\beta(i_l, i_{l+1}) > 0,\ l = 0, \ldots, q-1.$$

La capacité résiduelle sur un chemin retour est $\beta(\pi_\beta) = \min\left\{\beta(i,j)/(i,j) \in \pi_\beta\right\}$.

Une paire de chemins augmentants $(\pi_\alpha, \pi_\beta)$ est constituée par une paire $\pi_\alpha$ et $\pi_\beta$ de chemins aller et retour. La capacité résiduelle d'une paire de chemins augmentants est:

$$\gamma(\pi_\alpha, \pi_\beta) = \min\left\{\alpha(\pi_\alpha),\ \beta(\pi_\beta)\right\}.$$

L'algorithme proposé trouve des paires de chemins augmentants $\pi_\alpha$ et $\pi_\beta$ à travers lesquels $\gamma$ unités de $x^1$ sont redistribuées, permettant ainsi une augmentation du deuxième flot de $\gamma$ unités de $s_2$ à $t_2$ sur chacun des chemins $\pi_\alpha$ et $\pi_\beta$.

*Algorithme (biflot maximum)*

*Pas 1 :* Poser $x^2 = 0$ initialement et trouver un flot maximum $x^1$ de $s_1$ à $t_1$.

*Pas 2 :* Trouver une paire de chemins augmentants $(\pi_\alpha, \pi_\beta)$. Si une telle paire n'existe pas, arrêter, le biflot maximum est atteint.

*Pas 3 :* $\forall (i,j) \in \pi_\alpha$ augmenter $x^1$ et $x^2$ de $\gamma$ unités (dans le sens du chemin).

*Pas 4 :* $\forall (i,j) \in \pi_\beta$ augmenter $x^1$ de $\gamma$ unités et diminuer $x^2$ de la même quantité.

*Pas 5 :* Aller au pas 2.

### III.1.1.2. Validité et propriétés de l'algorithme

Pour prouver la validité de l'algorithme, il faut montrer qu'il est fini et que le biflot trouvé est réalisable (i.e., satisfaisant les contraintes de capacité et de conservation du flot) et maximum. La notion de réalisabilité est exprimée par le lemme suivant:

***Lemme III.1.*** *Si on commence par un flot réalisable, à la fin de chaque itération, le flot résultant est réalisable.*

Pour la preuve, se référer à Itai [1978].

En supposant que l'algorithme s'arrête, nous allons montrer que le biflot trouvé est maximal. Pour ce faire, rappelons certaines notions de coupes minimums.

Soit un ensemble de sommets $X \subseteq V$ $(\overline{X} = V - X)$, l'ensemble des paires ordonnées $(X, \overline{X}) = \{(i,j) / i \in X, j \in \overline{X}, (i,j) \in E\}$ est une coupe. Sa valeur (ou sa capacité) est $b(X, \overline{X}) = \sum_{(i,j) \in (X, \overline{X})} b[i,j]$. Une coupe est dite minimale si sa capacité est minimale. Posons $\Gamma(i_1, i_2; j_1, j_2)$ comme étant la valeur de la coupe minimum $b(X, \overline{X})$ qui met $i_1$, $i_2$ dans $X$ et $j_1, j_2$ dans $\overline{X}$, et $v(i_1, \ldots, i_m; j_1, \ldots, j_m)$ comme étant la valeur de la coupe minimum qui sépare $i_l$ de $j_l$ $(l = 1, m)$.

Dans le cas d'un biflot (resp. d'un multiflot), une coupe est définie comme étant un sous-ensemble $C$ de $E$, pour lequel chaque chemin entre $s_k$ et $t_k$ contient au moins une arête de $C$, et est minimal pour cette propriété, $\forall k = 1,2$ (resp. $k = 1, K$).

Compte tenu des définitions précédentes, la capacité de la coupe minimale pour un problème de biflot correspondrait à $v(s_1, s_2; t_1, t_2)$. Ford et Fulkerson montrent que pour un flot simple: flot maximum = coupe minimale, autrement dit $d^1 = v(s_1; t_1)$. Notons qu'en général $\Gamma(i_1, i_2; j_1, j_2) \geq v(i_1, i_2; j_1, j_2)$ dans la mesure où $v(i_1, i_2; j_1, j_2)$ peut être réalisé pour une coupe $(X, \overline{X})$ telle que $i_1, j_2 \in X$ et $i_2, j_1 \in \overline{X}$. D'où le lemme suivant:

***Lemme III.2.*** *(Hu [1963]) Dans un graphe non orienté, on a:*

$$v(s_1, s_2; t_1, t_2) = \min \{\Gamma(s_1, s_2; t_1, t_2);\ \Gamma(s_1, t_2; s_2, t_1)\}.$$

***Lemme III.3.*** *(Itai [1978]) Si, pour un quelconque flot réalisable $(x^1, x^2)$ trouvé par l'algorithme, il n'existe pas de paires de chemins augmentants, alors:*

$$d^1 + d^2 = v(s_1, s_2; t_1, t_2).$$

*Preuve.* Considérons deux cas:

*(a) Il n'existe pas de chemin aller:*

Posons $X = \{i / i \in V$, et il existe un chemin $(s_2 = i_0, \ldots, i_r = i)$ de $s_2$ à $i$ tel que $\alpha(i_{l-1}, i_l) > 0,\ l = 1, r\}$.

Par définition $s_2 \in X$ et comme il n'existe pas de chemin aller, $t_2 \in \overline{X}$. Pour tout $(i, j) \in (X, \overline{X})$, $\alpha(i, j) = \frac{1}{2}\left(b[i, j] - x^1(i, j) - x^2(i, j)\right) = 0$. Donc $b[i, j] = x^1(i, j) + x^2(i, j)$. Comme $b[i, j] \geq |x^1(i, j)| + |x^2(i, j)|$, alors $x^1(i, j),\ x^2(i, j) \geq 0$.

- Si $x^1(i, j) = 0\ \forall\ (i, j) \in (X, \overline{X})$ alors $x^2(i, j) = b[i, j]$. Le flot total est égal à $b(X, \overline{X}) \geq v(s_2; t_2)$. Par conséquent, conformément au théorème flot maximum coupe minimum, $x^2$ est maximal (indépendamment de $x^1$) et $d^2 = v(s_2; t_2)$.
Comme l'algorithme commence avec $d^1$ maximum et que ce dernier reste inchangé tout au long de l'algorithme, $d^1 = v(s_1; t_1)$. Donc $d^1 + d^2 = v(s_1; t_1) + v(s_2; t_2) \geq \Gamma(s_1, s_2; t_1, t_2)$.

- Autrement, il existe une paire $(i, j) \in (X, \overline{X})$ telle que $x^1(i, j) > 0$. Alors $s_1 \in X$, $t_1 \in \overline{X}$ (autrement, pour conserver le flot, il existerait une paire $(p, q) \in (X, \overline{X})$ telle que $x^1(p, q) < 0$). Donc $d^1 + d^2 = b(X, \overline{X}) \geq \Gamma(s_1, s_2; t_1, t_2)$.

*(b) Il n'existe pas de chemin retour:*

On conclut de la même manière que $d^1 + d^2 \geq \Gamma(s_1, t_2; s_2, t_1)$.

En combinant (a) et (b), on obtient $d^1 + d^2 \geq \min \{\Gamma(s_1, s_2; t_1, t_2), \Gamma(s_1, t_2; s_2, t_1)\} = v(s_1, s_2; t_1, t_2)$. Comme par ailleurs $d^1 + d^2 \leq v(s_1, s_2; t_1, t_2)$, il vient:

$$d^1 + d^2 = v(s_1, s_2; t_1, t_2).$$

On peut alors énoncer le corollaire suivant:

***Corollaire III.4.*** *Quand l'algorithme s'arrête le flot trouvé est maximal.*

Le dernier point concerne la convergence de l'algorithme. Revenons au pas 2 de l'algorithme, où il est question de trouver des chemins augmentants. Ces chemins doivent être convenablement choisis sinon le nombre d'itérations pourrait devenir exponentiel. Pour ce faire, on peut opter pour la démarche de Edmonds et Karp [1972] concernant le choix des plus courts chemins (pour le flot simple), mais la démarche de Dinic [1970], Even et Tarjan [1975], réalise une meilleure borne sur le temps d'exécution. Grâce à cette démarche, l'algorithme requiert au plus $O(|V|^2|E|)$ unités de temps (pour plus de détails, consulter Itai [1978]). Karzanov [1974] propose une autre méthode qui permet d'améliorer l'algorithme avec un temps d'exécution en $O(|V|^3)$. Nous pouvons à présent dégager les principales propriétés de l'algorithme, à savoir:

(1) Le biflot maximum est atteint avec le premier flot ayant sa valeur maximale.

(2) Si les capacités sont entières, le flot obtenu à la fin de l'algorithme est en demi-unités (par définition de $\alpha$ et $\beta$).

(3) Le biflot maximum obtenu est égal à la coupe minimum $v(s_1, s_2; t_1, t_2)$. La preuve est basée sur le fait qu'après un nombre fini d'étapes, il n'existe plus de paires de chemins augmentants et à partir du lemme III.3, le flot réalisé est égal à la coupe minimale.

(4) On peut utiliser l'algorithme de biflot maximum pour résoudre un autre problème en relation avec ce dernier: c'est le problème de biflot avec besoins. Étant données deux constantes $R^1$ et $R^2$ représentant les besoins respectifs en $x^1$ et $x^2$, le problème revient à trouver un biflot réalisable tel que $d^k \geq R^k$ pour $k = 1,2$. Pour ce faire, on rajoute deux nouveaux sommets sources $\bar{s}_1$, $\bar{s}_2$ et deux arêtes $[\bar{s}_1, s_1]$, $[\bar{s}_2, s_2]$ de capacités respectives $R^1$ et $R^2$. Le biflot maximum dans le nouveau réseau est égal à $R^1 + R^2$ si et seulement si il existe un biflot réalisable dans le réseau original tel $d^1 \geq R^1$ et $d^2 \geq R^2$ .

Hu [1963] a été le premier à établir les deux résultats fondamentaux pour le problème du biflot non orienté, à savoir le théorème concernant le biflot maximum et la coupe minimale, et celui concernant l'intégralité du biflot maximal. Il faut préciser que sa preuve (pour l'égalité entre flot maximum et coupe minimum) est également algorithmique, mais ne fonctionne que dans le cas où les capacités sont entières ou rationnelles. L'amélioration introduite dans la recherche des chemins augmentants a permis finalement, de généraliser cette preuve au cas des capacités quelconques. Les deux principaux théorèmes sont les suivants:

***Théorème III.5.*** *Soient* $d^{1*}$, $d^{2*}$ *les flots optimaux et* $v(s_1, s_2; t_1, t_2)$ *la capacité de la coupe minimale pour le problème de biflot maximal non orienté, alors:*

$$d^{1*} + d^{2*} = v(s_1, s_2; t_1, t_2).$$

***Théorème III.6.*** *Si les capacités correspondant au biflot maximal non orienté sont des entiers pairs, alors il existe un optimum entier.*

Ces résultats sont redémontrés par Sakarovitch [1973] à travers la théorie de la programmation linéaire. En même temps, l'auteur donne un algorithme de résolution pour le biflot maximal qui consiste essentiellement à appliquer deux fois l'algorithme de flot maximum de Ford et Fulkerson [1962].

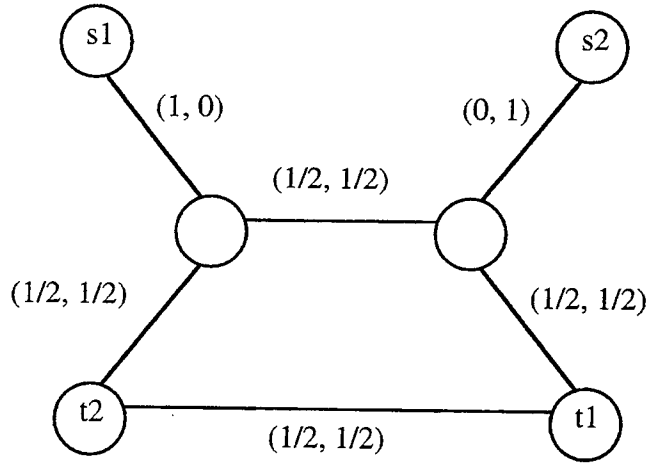



**Figure III.1** Biflot maximum, avec flots fractionnaires.
(Toutes les capacités sont égales à 1).

Il faut noter que le théorème III.5 ne garantit pas l'intégralité des flots même si $v(s_1, s_2; t_1, t_2)$, $d^{1*}$, $d^{2*}$ sont des entiers. Le problème de biflot de la Figure III.1 vérifie le théorème III.5, avec $v(s_1, s_2; t_1, t_2) = 2$, $d^{1*} = d^{2*} = 1$, mais avec des flots fractionnaires sur certaines arêtes.

De plus ce théorème ne peut être étendu, ni au cas du biflot orienté (cf. § III.2.), ni à celui des autres multiflots (voir l'exemple de Fulkerson [1963] en Figure III.2.)

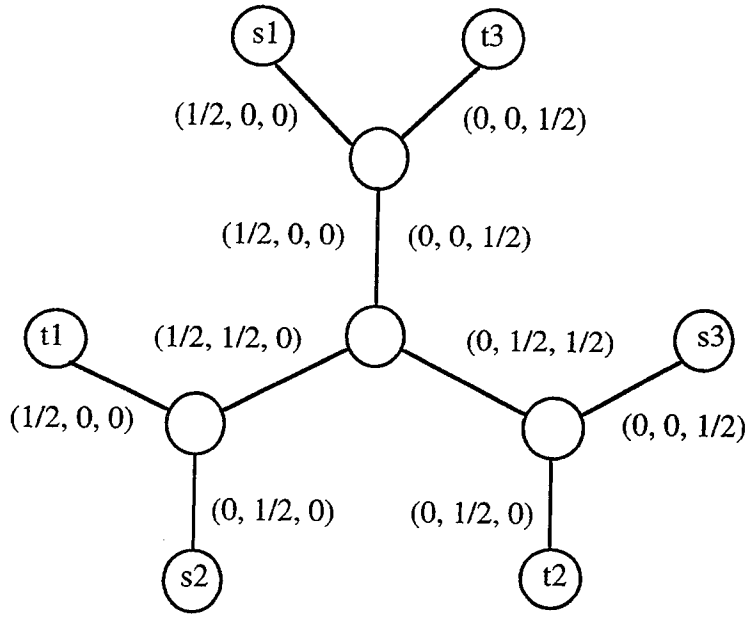



**Figure III.2** Triflot maximum (toutes les capacités sont égales à 1.)
Le flot maximum est de 1.5, la coupe minimum est de 2.

Quelques exceptions sont à noter cependant, puisque pour $K = 3$ Rothfarb et Frish [1969] prouvent qu'un triflot sur un graphe non orienté, ne comportant pas de nœuds intermédiaires (tous les nœuds sont soit des sources soit des destinations), le flot maximum est égale à la coupe minimum. Partant de là, ces auteurs donnent un algorithme de flot maximum. De même, Tang [1965] montre que la classe des réseaux "bipath" vérifie flot maximum = coupe minimum, et ce, pour tout $K$. Rothfarb, Shein et Frish [1968] présentent un algorithme pour maximiser $\sum_k \omega^k d^k$, où les $\omega^k$ sont des poids décroissants et tous les flots ont une destination commune. Kleitman [1971] présente un algorithme de flot maximum lorsque chaque nœud est un puits pour tous les produits, sauf un.

## III.1.2. BIFLOT DE COÛT MINIMUM

### III.1.2.1. Formulation et décomposition

Dans le cas non orienté, le problème de biflot de coût minimum peut se formuler de la façon suivante:

$$\text{Minimiser } c^1x^1 + c^2x^2 \tag{III.2a}$$

sous les contraintes

$$|x^1| + |x^2| \leq b \tag{III.2b}$$

$$Ax^1 = r^1 \tag{III.2c}$$

$$Ax^2 = r^2. \tag{III.2d}$$

Dans cette formulation matricielle, les deux flots sont représentés par les $M$-vecteurs $x^1$ et $x^2$. $b$ est un vecteur de dimension $M$ désignant les capacités sur les différents arcs du réseau, et $A$ est la matrice d'incidence sommets-arcs du graphe associé au réseau. $r^1$ est le vecteur ressource (de dimension $N$), il est égal à $+d^1$ à la source, à $-d^1$ à la destination et à 0 ailleurs; $r^2$ est défini de la même manière, mais avec le montant $d^2$. Remarquons qu'ici les montants $d^1$ et $d^2$ ne sont plus des variables de décision (biflot maximal) mais sont des constantes connues d'avance, on cherche ici à minimiser le coût total de transport: $c^1$ et $c^2$ sont des $M$-vecteurs représentant les coûts unitaires de transport pour les deux flots.

On retrouve la contrainte de capacité (III.2b), la somme, en valeur absolue, des flots ne devant pas dépasser la capacité disponible. Et les deux blocs de contraintes (III.2c) et (III.2d) correspondant à la conservation du flot pour $x^1$ et $x^2$.

Le problème ainsi présenté peut, par un changement de variables adéquat, se décomposer en deux problèmes de flot simple orientés à coûts minimaux. La substitution est proposée par Sakarovitch [1973]:

$$\begin{cases} x^1 = y^1 + y^2 - b \\ x^2 = y^1 - y^2 \end{cases}$$

Le problème (III.2) prend alors la forme suivante:

$$\text{Minimiser } (c^1 + c^2)y^1 + (c^1 - c^2)y^2 - c^1 b \qquad \text{(III.3a)}$$

sous les contraintes

$$Ay^1 + Ay^2 = r^1 + Ab \qquad \text{(III.3b)}$$

$$Ay^1 - Ay^2 = r^2 \qquad \text{(III.3c)}$$

$$0 \le y^1 \le b \qquad \text{(III.3d)}$$

$$0 \le y^2 \le b. \qquad \text{(III.3e)}$$

En remplaçant (III.3b) et (III.3c) par la somme, respectivement la différence des deux équations, on obtient:

$$\text{Minimiser } (c^1 + c^2)y^1 + (c^1 - c^2)y^2 - c^1 b \qquad \text{(III.4a)}$$

sous les contraintes

$$Ay^1 = \frac{1}{2}\left(r^1 + r^2 + Ab\right) \qquad \text{(III.4b)}$$

$$Ay^2 = \frac{1}{2}\left(r^1 - r^2 + Ab\right) \qquad \text{(III.4c)}$$

$$0 \le y^1 \le b \qquad \text{(III.4d)}$$

$$0 \le y^2 \le b. \qquad \text{(III.4e)}$$

Ainsi présenté, le problème (III.4) devient décomposable en deux problèmes de flots simples à coûts minimaux. Soient (III.4') et (III.4") ces deux sous-problèmes:

$$\text{Minimiser } (c^1 + c^2)y^1 \qquad \text{(III.4'a)}$$

sous les contraintes

$$Ay^1 = \frac{1}{2}\left(r^1 + r^2 + Ab\right) \qquad \text{(III.4'b)}$$

$$0 \le y^1 \le b. \qquad \text{(III.4'c)}$$

$$\text{Minimiser } (c^1 - c^2)y^2 \qquad \text{(III.4"a)}$$

sous les contraintes

$$Ay^2 = \frac{1}{2}\left(r^1 - r^2 + Ab\right) \qquad \text{(III.4"b)}$$

$$0 \le y^2 \le b. \qquad \text{(III.4"c)}$$

Les conditions d'intégralité du biflot découlent alors directement des équations de conservation du flot pour les deux sous-problèmes de flot simple (III.4') et (III.4"). Rappelons que pour un problème de flot simple, la condition d'intégralité est que les capacités ainsi que les montants impliqués dans les vecteurs ressource soient entiers. Les montants impliqués dans les sous-problèmes (III.4') et (III.4") ($\frac{1}{2}(r^1 + r^2 + Ab)$ $\frac{1}{2}(r^1 - r^2 + Ab)$ respectivement), étant en général réels (ou en demi-unités pour des valeurs entières de $r^1$, $r^2$ et $b$), on introduit deux définitions avant d'énoncer les conditions d'intégralité pour un problème de biflot non orienté de coût minimum.

***Définition III.7.*** *On dit qu'un nœud est pair si la somme de toutes les capacités des arêtes ayant pour extrémité ce nœud est un entier pair.*

***Définition III.8.*** *Un réseau est dit d'Euler si toutes ses capacités sont entières et que ses nœuds sont tous pairs.*

Remarquons que si un réseau est d'Euler alors le vecteur $(Ab)$ a toutes ses composantes entières et paires. Si de plus les montants $d^1$ et $d^2$ sont des entiers pairs, les quantités $\frac{1}{2}(r^1 + r^2 + Ab)$ et $\frac{1}{2}(r^1 - r^2 + Ab)$ seront entières. Et le résultat suivant est direct:

***Théorème III.9.*** *(conditions suffisantes d'intégralité)*
*Soient $d^1$ et $d^2$, deux entiers pairs et un réseau d'Euler. Si le problème de biflot de coût minimum admet une solution réalisable, alors il existe un optimum entier.*

La matrice $A$ intervenant dans le problème (III.3) est une matrice totalement unimodulaire, en tant que matrice d'incidence sommets-arcs (i.e., toute sous-matrice de $A$ a un déterminant égal à 0 ou $\pm 1$). Sakarovitch [1973] montre que si $d^1$ et $d^2$ sont les variables de décision pour le problème de biflot maximal, alors la matrice des contraintes est également totalement unimodulaire. Le second membre étant égal à $\frac{1}{2}(Ab)$ dans ce cas, on peut dire que le problème de biflot maximal admet un optimum entier si les capacités sont des entiers pairs. Ainsi on dispose d'une autre preuve pour le théorème III.6 de Hu. Un résultat équivalent serait de dire (conformément aux définitions données plus haut):

***Théorème III.10.*** *(Rothschild et Whinston [1966a])*
*Si le réseau associé au problème de biflot maximal non orienté est d'Euler, alors il existe un optimum entier.*

Sakarovitch [1973] montre que ce théorème est également applicable dans le cas où les sources et les destinations ne sont pas des nœuds pairs, et donne d'autres conditions de

réalisabilité pour le cas entier. Des conditions similaires existent également dans Rothschild et Whinston [1966b].

### III.1.2.2. Conditions de réalisabilité

Il reste à savoir si, disposant des capacités du réseau, on peut faire passer les montants voulus à travers les différents arcs sans en violer la capacité. Le théorème suivant répond à cette question en donnant une condition nécessaire et suffisante pour avoir une solution réalisable.

***Théorème III.11.*** *(Hu [1963])*
*Le problème de biflot non orienté à coût minimum possède une solution réalisable si et seulement si:* $v(s_1;t_1) \geq d^1$, $v(s_2;t_2) \geq d^2$ *et* $v(s_1,s_2;t_1,t_2) \geq d^1 + d^2$.

Dans le cas où le nombre de flots mis en jeu est supérieur à deux, le théorème III.11 ne peut plus être appliqué. Si on considère l'exemple présenté par Hu [1964] (cf. Figure III.3), avec $d^1 = 4$, $d^2 = 2$ et $d^3 = 1$, on se rend compte que les conditions du théorème sont satisfaites, en effet:

$v(s_1;t_1) = 4 \geq d^1$, $v(s_2;t_2) = 6 \geq d^2$, $v(s_3;t_3) = 6 \geq d^3$,
$v(s_1,s_2;t_1,t_2) = 6 \geq d^1 + d^2$, $v(s_1,s_3;t_1,t_3) = 6 \geq d^1 + d^3$, $v(s_2,s_3;t_2,t_3) = 8 \geq d^2 + d^3$ et
$v(s_1,s_2,s_3;t_1,t_2,t_3) = 8 \geq d^1 + d^2 + d^3$, pourtant on ne peut pas trouver de solution réalisable à ce problème.

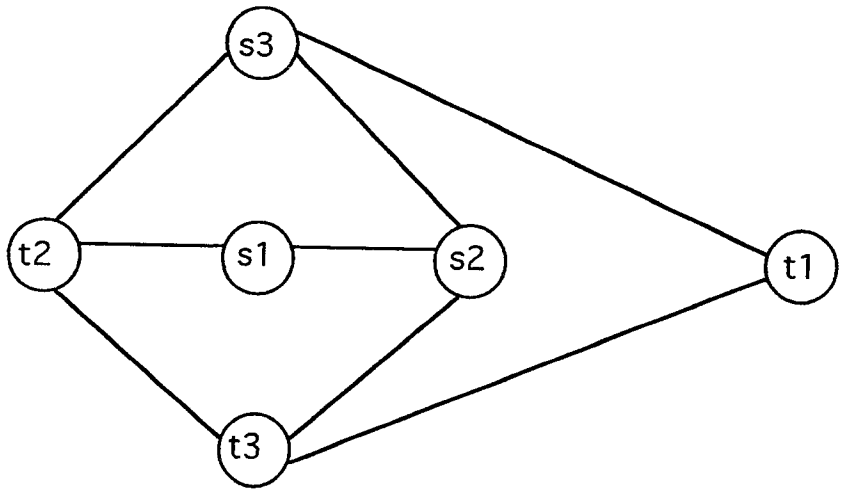



**Figure III.3** Multiflot à coût minimum.
(les capacités sont égales à 2).

Naniwada [1969] présente des conditions nécessaires pour le problème général. Des conditions nécessaires et suffisantes sont également données par Onaga [1970] mais le temps de calcul requis pour les vérifier est prohibitif. Des conditions similaires basées sur la dualité en programmation linéaire sont mises à jour par Iri [1971].

Dans le cas entier général, le problème de biflot compatible (existence d'un biflot réalisable en nombres entiers) est un problème NP-complet (Even Itai et Shamir [1976]). Il en de même pour les autres problèmes de multiflots en nombres entiers. Cependant, il existe des cas très particuliers où ces problèmes peuvent être résolus polynomialement. Ainsi par exemple, lorsque toutes les capacités sont égales à 1 (à ce moment là, le problème de multiflot s'identifie au problème de la recherche de chemins arcs-disjoints dans un graphe, entre un ensemble de paires de sommets source-destination), que toutes les demandes sont égales à 1 et que le nombre de flots est borné par un entier fixé, le problème devient polynomial, conformément au résultat de Robertson et Seymour [1990]. Un autre cas particulier est celui des multiflots sur des graphes planaires, pour lesquels on a longtemps essayé de savoir s'il était possible de les résoudre polynomialement. Un résultat récent de Middendorf et Pfeiffer [1990] prouvent que le problème est NP-complet. Cependant si le nombre de flots est borné par un entier $K$ fixé arbitrairement (toujours en supposant la planarité), Sebö [1993] présente un algorithme polynomial qui permet de mettre en évidence un multiflot compatible ou de prouver qu'il n'en existe pas. Pour $K = 2$, ce problème avait déjà été étudié par Seymour [1981] (cf. également Lomonosov [1983] et Sebö [1987]), et pour $K = 3$ par Korach [1982]. Un article de synthèse est disponible grâce à Frank [1990].

## III.2. BIFLOT ORIENTÉ

Dans sa version orientée, le flot possède une direction et une seule: celle de l'arc qui le véhicule. C'est pourquoi dans la version orientée on impose la contrainte de non-négativité des flots (non contraints dans le cas précédent). Ceci se répercute sur la contrainte de couplage dans laquelle ne figurent plus les valeurs absolues. Un biflot réalisable est alors décrit par le système de contraintes suivant (conservation du flot et respect de la capacité):

$$\begin{cases} Ax^1 = r^1 \\ Ax^2 = r^2 \\ x^1 + x^2 \leq b \\ x^1 \geq 0,\ x^2 \geq 0. \end{cases}$$

La contrainte de capacité (III.2b) associée au biflot non orienté est remplacée ici par les trois dernières inéquations. Cette formulation va entraîner une certaine difficulté du biflot orienté par rapport à son prédécesseur non orienté. En effet:

(1) la décomposition proposée par Sakarovitch, dans le cadre du biflot non orienté de coût minimum, n'est plus valable dans ce cas, suite à la contrainte de non-négativité des flots.

(2) Le théorème stipulant flot maximum = coupe minimum n'est plus applicable dans le cas orienté. Dans la Figure III.4, le flot maximum est obtenu pour $d^1 = 1/2$, $d^2 = 1$, ce qui est inférieur à la coupe minimale, égale à 2 (toutes les capacités sont égales à 1.)

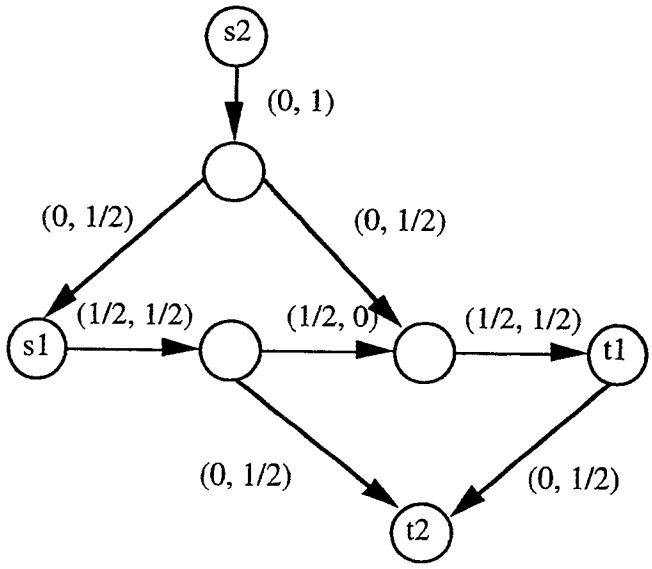



**Figure III.4** Le flot maximum est inférieur à la coupe minimale.

(3) Même si les capacités sont entières, le biflot maximum peut être rationnel, et pas nécessairement en demi-unités. (Voir Figure III.5.)

(4) Le biflot maximum n'est pas toujours réalisé pour le premier flot égal à sa valeur maximale. Dans la Figure III.4 , si $d^1 = 1$ alors $d^2 = 0$ et $d^1 + d^2 = 1$, ce qui est inférieur au flot maximum.

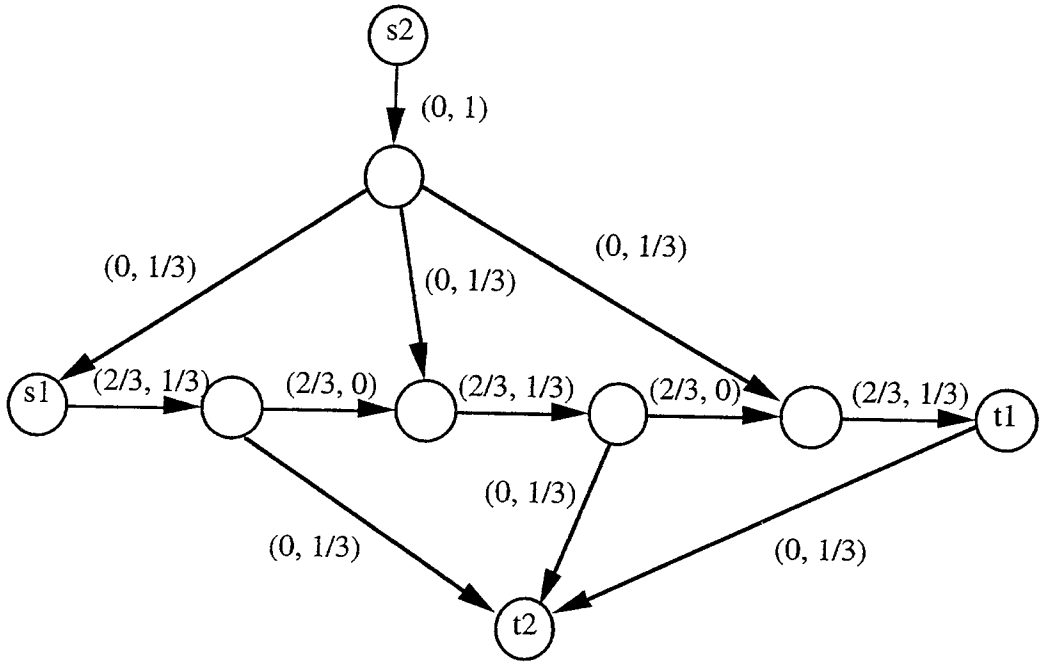



**Figure III.5** Biflot à valeurs rationnelles (les capacités sont égales à 1.)

Précisons qu'il existe certains cas où le multiflot maximal est égal à la coupe minimale. Ces cas de figure sont illustrés par le théorème de Sakarovitch [1966]:

***Théorème III.12.*** *le multiflot maximum sur un graphe complètement planaire est entier et vérifie: flot maximum = coupe minimum.*

Un graphe planaire est un graphe qui peut être représenté sur un plan sans comporter aucun croisement d'arêtes, excepté au niveau des nœuds.

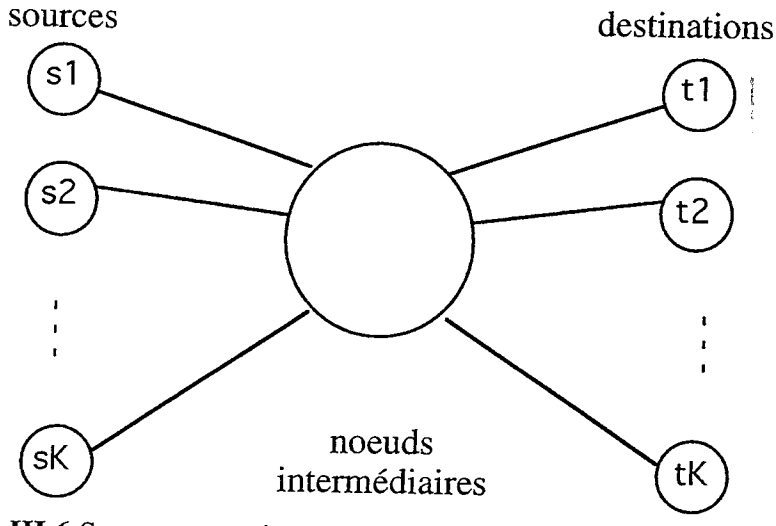



**Figure III.6** Structure requise pour un multiflot complètement planaire.

Un graphe est complètement planaire si, l'ayant mis sous la forme présentée dans la Figure III.6, il constitue un graphe planaire.

Remarque: un multiflot peut vérifier flot maximum = coupe minimum sans être entier. L'exemple de la Figure III.7 (Sakarovitch [1966]) en est une parfaite illustration.

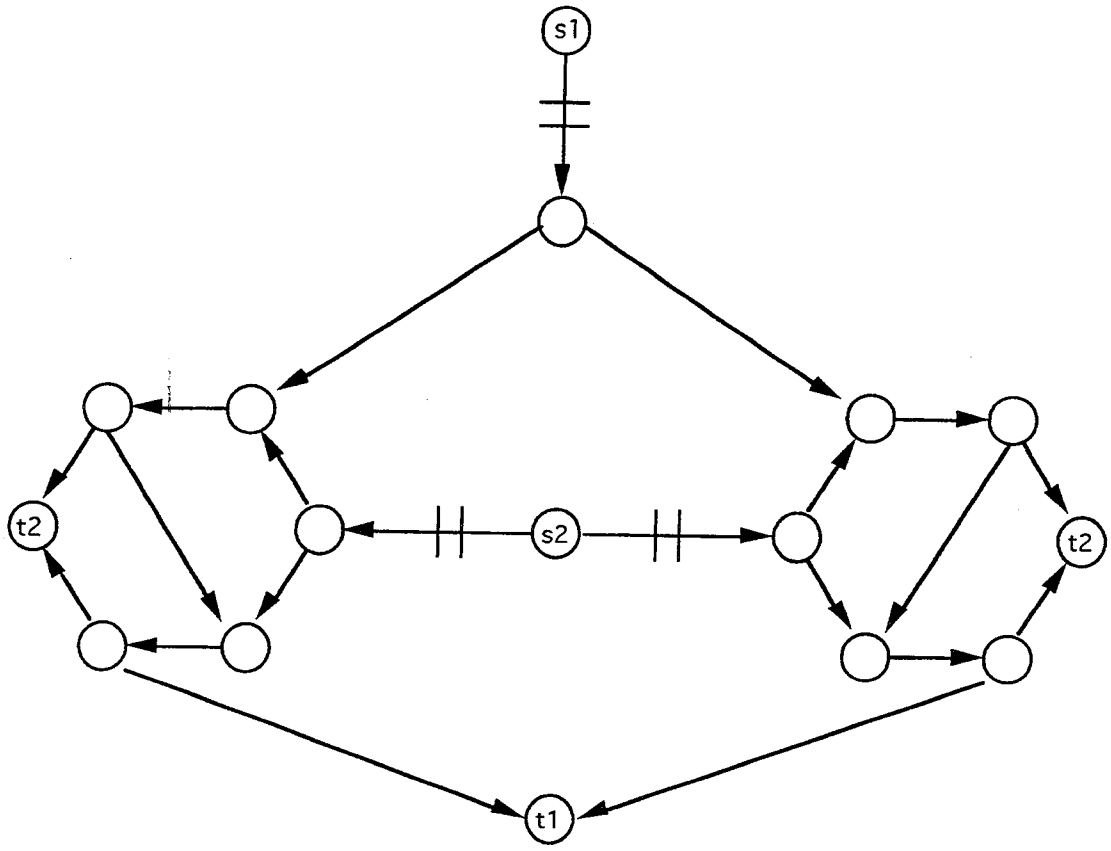



**Figure III.7** Le flot maximum est réalisé pour $d^1 = 1$, $d^2 = 2$, la capacité de la coupe minimale est égale à 3 (toutes les capacités sont égales à 1). Mais les flots sur les arcs sont de 0,5.

D'un point de vue complexité, dans le cas continu, le problème de biflot ainsi que les autres problèmes de multiflots (orientés) peuvent être considérés comme des cas particuliers de la programmation linéaire (Ford et Fulkerson [1962]). Dans le cas entier par contre, le problème de biflot orienté devient NP-complet (Even, Itai et Shamir [1976]); et c'est aussi le cas de tous les autres multiflots en nombres entiers. Des algorithmes polynomiaux existent cependant pour des problèmes très particuliers de biflot (Nach-Williams [1960]) ou de triflot (Ibaraki et Poljak [1991]) sur des graphes eulériens, avec toutes les capacités égales à 1. Les résultats concernant ce genre de problèmes de

Williams [1960]) ou de triflot (Ibaraki et Poljak [1991]) sur des graphes eulériens, avec toutes les capacités égales à 1. Les résultats concernant ce genre de problèmes de multiflots, correspondant en l'occurrence à la recherche de chemins arcs-disjoints dans un graphe, peuvent être trouvés dans Frank [1990] et également Vygen [1992].

## II.3. CONCLUSION

Les deux résultats fondamentaux existant pour les flots simples, en l'occurrence, le théorème des valeurs entières et celui concernant le flot maximum et la coupe minimale, ne se généralisent pas aux multiflots, sauf dans le cas très particulier des biflots sur des graphes non orientés, ou des graphes planaires. Le problème de biflot, dans sa version non orientée, reste alors plus facile à appréhender que son homologue en version orientée. Ainsi, le problème de biflot de coût minimum (non orienté) admet une décomposition qui permet de le scinder en deux problèmes de flot simple orientés. De cette façon, la solution optimale de ce problème peut être reconstituée efficacement, à partir des deux solutions individuelles de flot simple. Malheureusement, cette décomposition n'est plus applicable dans le cas du biflot orienté qui peut seulement être résolu en tant que programme linéaire. Mais la difficulté majeure intervient lorsque l'on impose une contrainte d'intégralité sur les flots. A ce moment là, le problème de biflot compatible (dans sa version orientée ou non orientée) devient NP-complet. Et par généralisation, tous les autres problèmes de multiflots présenteront la même complexité.

Tous ces résultats auront permis de confirmer, une fois de plus, la difficulté des problèmes de multiflots, et d'expliquer, en partie, la nature des diverses techniques mises en œuvre pour les résoudre.

# Chapitre IV

# BIFLOT ENTIER DE COÛT MINIMUM: MÉTHODE DE RÉSOLUTION PAR DÉCOMPOSITION DÉCENTRALISÉE.

## IV.0. INTRODUCTION

Nous avons souligné, dans le chapitre précédent, l'intérêt suscité par le problème du biflot, d'une part à travers les résultats théoriques qu'il met en jeu, et d'autre part à travers la possibilité qu'il offre de modéliser un certain nombre de problèmes réels. Plus précisément, nous nous intéresserons, ici, au problème de biflot entier et de coût minimum. Rappelons que la contrainte d'intégralité, sur les flots, est une contrainte quasiment implicite dans ce genre de problèmes. Malheureusement, cette contrainte nous met en présence d'un problème NP-difficile, qui pourra être résolu soit de manière heuristique en exploitant sa structure particulière, soit en utilisant des méthodes générales de programmation en nombres entiers.

Jusqu'à présent, nous avons assisté à un déséquilibre énorme entre l'effort de recherche consacré à la résolution des problèmes de multiflots en continu, et leurs homologues en nombres entiers. Ces derniers n'ayant bénéficié que de quelques rares approches plus ou moins personnalisées. De plus, parmi les méthodes heuristiques les plus récentes, la méthode d'approximation de Srivastav et Stangier [1993] peut induire, au pire des cas, une déperdition du flot de 50% pour le problème de multiflot maximal, ce qui est assez défavorable.

Toute cette problématique a donc motivé l'idée d'élargir le panorama des méthodes approchées existant pour ce genre de problèmes. Cette idée s'est développée dans le cadre particulier du biflot entier de coût minimum. Nous présenterons donc le problème, de façon détaillée, en termes de formulation et de complexité. Puis nous proposerons une méthode de résolution basée essentiellement sur l'approche de décomposition décentralisée développée par Mahey [1986], dans le cadre de la programmation linéaire. Il est à noter que même si cette approche a fait, dans ce même cadre, l'objet de plusieurs études et applications (Morin [1993]), son adaptation au cas particulier du biflot entier constitue une première qui confère à la méthode un caractère de pionnière. Sur un plan pratique, nous avons développé un logiciel permettant d'une part, l'élaboration de problèmes de biflot aléatoires, et d'autre part, la mise en œuvre de la méthode en question.

En utilisant une méthode de décomposition mixte, on évite certes le niveau de coordination, mais d'autres difficultés sont mises à jour, notamment en ce qui concerne la notion de convergence. Toutes ces difficultés sont mises en évidence, et partiellement résolues dans le cadre des principales heuristiques développées. Nous présenterons, pour finir, les résultats numériques découlant de toutes ces applications. Puis nous ferons une analyse comparative entre les différentes heuristiques, en même temps que nous expliciterons quelques aspects de la convergence.

## IV.1. PRÉSENTATION DU PROBLÈME

On se donne un réseau quelconque représenté par un graphe orienté $G=(V,E)$, avec $|V|=N$ et $|E|=M$. Pour chaque flot $x^k$ $(k=1,2)$, nous avons une partition des sommets de $G$ en trois sous-ensembles $S_k$, $I_k$ et $D_k$, dénotant respectivement, les sources, les nœuds intermédiaires et les destinations du flot $k$. Pour $i \in V$ posons $d^k(i)$ comme le montant du produit $k$ au nœud $i$. Il s'en suit que $d^k(i)>0$ si $i \in S_k$, $d^k(i)=0$ si $i \in I_k$, et $d^k(i)<0$ si $i \in D_k$. Supposons, sans perte de généralité, que l'offre totale est égale à la demande totale et étudions le problème de biflot de coût minimum, en nombres entiers, suivant:

$$\text{Minimiser } c(x^1 + x^2) \tag{IV.1a}$$

sous les contraintes

$$Ax^1 = r^1 \tag{IV.1b}$$

$$Ax^2 = r^2 \tag{IV.1c}$$

$$x^1 + x^2 \leq b \tag{IV.1d}$$

$$x^1, x^2 \text{ entiers non négatifs.} \tag{IV.1e}$$

Où $x^1, x^2$ sont des $M$-vecteurs représentant les deux flots en circulation sur le réseau,

$c$ est un $M$-vecteur entier non négatif représentant les coûts de transport sur les réseau,

$r^1, r^2$ sont des $N$-vecteurs entiers dont les composantes s'identifient aux montants $d^k(i)$ pour $i \in V$, respectivement pour $k = 1, 2$,

$b$ est un $M$-vecteur entier positif représentant les capacités sur les arcs du réseau,

et $A$ est la matrice d'incidence sommets-arcs du graphe $G$ associé au réseau.

Dans ce problème de biflot à coût minimum, on cherche évidemment à minimiser le coût total de transport (IV.1a). Mais contrairement aux formulations précédentes, nous avons fait le choix, ici, de prendre les mêmes coûts de circulation pour les deux flots, représentés en l'occurrence par l'unique vecteur $c$. Sur un plan pratique, ce choix peut être largement justifié dans la mesure où les coûts de transport sont assez fréquemment indépendants de la nature du flot. Par ailleurs, ce choix rend le problème plus compliqué en induisant une "concurrence" plus importante entre les deux flots, quant au choix des arcs et des chemins. On retrouve dans ce problème de biflot les classiques contraintes de conservation du flot (IV.1b) et (IV.1c), respectivement pour les flots $x^1$ et $x^2$. On retrouve également la contrainte de couplage (IV.1d) traduisant le respect des capacités, ainsi que la contrainte de positivité sur les flots indiquant que le réseau est orienté. En plus de toutes ces contraintes largement étudiés dans les développements précédents, nous avons introduit une contrainte d'intégralité sur les flots $x^1$ et $x^2$, qui nous met effectivement en présence d'un biflot entier de coût minimum. Ainsi défini, le problème (IV.1) est un problème NP-difficile (Even, Itai et Shamir [1976]).

## IV.2. MÉTHODE DE RÉSOLUTION

Le problème (IV.1) étant un problème NP-difficile, nous avons choisi de le résoudre d'une manière heuristique, en lui adaptant une méthode de résolution appropriée. Cette méthode de résolution s'inspire fondamentalement de l'approche de décomposition décentralisée proposée par Mahey [1986], dans le cadre de la programmation linéaire (cf. chap. II, § II.1.2.3).

Cette approche de décomposition est une approche primale-duale, permettant d'une part de faire une allocation de capacité sur certains arcs, et d'autre part de pénaliser une trop grande utilisation de la capacité par l'un des deux flots, et ce, par l'intermédiaire de variables duales (variables "prix") introduites dans la fonction objective.

Dans la formulation du biflot entier de coût minimum, la contrainte de couplage (IV.1d) est donc relaxée, en fixant itérativement un flot puis l'autre. On s'aperçoit alors que dans le cas particulier du biflot, l'allocation de capacité se fait de façon dichotomique: soit on décide d'allouer une partie de la capacité au premier flot, le deuxième ne peut alors que disposer de la capacité résiduelle, soit on privilégie le deuxième flot et cette fois c'est le premier flot qui se verra utiliser la capacité restante, les réarrangements des flots se faisant par l'intermédiaire des transformations introduites dans la fonction objective par le biais des variables prix. L'adaptation de cette approche de décomposition mixte au cas du biflot débouche alors sur le schéma de résolution suivant:

Étant donnée une partition $H$ sur les arcs $(H \cap \overline{H} = \varnothing,\ H \cup \overline{H} = E)$, on peut décomposer le problème de biflot entier de coût minimum en deux sous-problèmes symétriques de flot simple à coût minimum. Soient (IV.1') et (IV.1") ces deux sous-problèmes associés respectivement aux flots $x^1$ et $x^2$:

$$\text{Minimiser} \sum_{(i,j)\in E} c_{ij}x_{ij}^1 + \sum_{(i,j)\in \overline{H}} \overline{w}_{ij}^2 x_{ij}^1 \qquad \text{(IV.1'a)}$$

sous les contraintes

$$x_{ij}^1 \le b_{ij} - \overline{x}_{ij}^2 \qquad \text{pour } (i,j)\in H \qquad \text{(IV.1'b)}$$

$$x_{ij}^1 \le b_{ij} \qquad \text{pour } (i,j)\in \overline{H} \qquad \text{(IV.1'c)}$$

$$Ax^1 = r^1 \qquad \text{(IV.1'd)}$$

$$x^1 \ge 0. \qquad \text{(IV.1'e)}$$

$$\text{Minimiser} \sum_{(i,j)\in E} c_{ij}x_{ij}^2 + \sum_{(i,j)\in H} \overline{w}_{ij}^1 x_{ij}^2 \qquad \text{(IV.1"a)}$$

sous les contraintes

$$x_{ij}^2 \le b_{ij} - \overline{x}_{ij}^1 \qquad \text{pour } (i,j)\in \overline{H} \qquad \text{(IV.1"b)}$$

$$x_{ij}^2 \le b_{ij} \qquad \text{pour } (i,j)\in H \qquad \text{(IV.1"c)}$$

$$Ax^2 = r^2 \qquad \text{(IV.1"d)}$$

$$x^2 \ge 0. \qquad \text{(IV.1"e)}$$

Les conventions de notation adoptées pour le sous-problème (IV.1') sont les suivantes:

$x_{ij}^1$ : variable de décision désignant la quantité de flot $x^1$ passant sur l'arc $(i,j)$, $(i,j)\in E$.

$c_{ij}$ : coût unitaire de circulation sur l'arc $(i,j)$, $(i,j)\in E$.

$b_{ij}$ : capacité de l'arc $(i,j)$, $(i,j)\in E$.

$\overline{x}_{ij}^2$ : quantité de flot $x^2$ circulant sur l'arc $(i,j)$, $(i,j)\in E$.

$\overline{w}_{ij}^2$ : variable duale du flot $x^2$ associée à la contrainte de capacité sur l'arc $(i,j)$, $(i,j)\in E$.

L'écriture $\overline{x}^2, \overline{p}^2$ indique que ces variables (mises à jour dans le sous-problème (IV.1")), seront considérées comme constantes dans le sous-problème (IV.1').

Pour le sous-problème (IV.1") (symétrique de (IV.1')), toutes les notations sont maintenues. Simplement les indices sont inversés ainsi que le rôle de la partition.

Étant donné que $\bar{x}, \bar{p}$ sont des constantes, nous sommes bien en présence de deux problèmes de flot simple à coût minimum (avec des contraintes classiques de capacité et de conservation du flot). Ces deux problèmes de flot mettent en jeu des coûts et des capacités modifiés sur des sous-ensembles d'arcs complémentaires.

Grâce à ce schéma de décomposition, la contrainte d'intégralité des flots devient implicite. De fait, la solution au problème de biflot initial consistera en la combinaison de deux flots simples, entiers par définition. Rappelons que pour un flot simple, la condition suffisante d'intégralité porte sur l'intégralité des capacités et des montants concernant les ressources et les demandes en flot, ce qui sera toujours le cas dans les problèmes (IV.1') et (IV.1"). Ce résultat se déduit de la propriété de totale-unimodularité de la matrice d'incidence sommets-arcs $A$ associée au réseau (voir Magnanti, Ahuja et Orlin [1993], chap. 11, § 11.12).

Le processus de résolution sera donc itératif et consistera a osciller entre (IV.1') et (IV.1") avec un échange d'informations entre les deux sous-problèmes (sans aucun niveau de coordination). L'algorithme qui en découle prend la forme simplifiée suivante:

*Algorithme de résolution (biflot entier de coût minimum.)*

*Pas 0 : Initialisations*

Définir la partition $H$.
Fixer $T > 0$ (nombre d'itérations maximum).
Poser $t = 0$ (compteur d'itérations),

$$\left.\begin{array}{l}(\overline{w}_{ij}^2)_t = 0 \\ (\bar{x}_{ij}^2)_t = 0\end{array}\right\} \text{ pour } (i,j) \in E.$$

*Pas 1 : Résolution du sous-problème 1*

Poser $t = t+1$.

Si $t > T$ alors arrêter, sinon
résoudre (IV.1') et déterminer $(x_{ij}^1)_t$ et $(w_{ij}^1)_t$ pour $(i,j) \in E$.

Si $t = 1$ alors aller au pas 2, sinon
si $(x_{ij}^1)_t = (x_{ij}^1)_{t-1}$ et $(w_{ij}^1)_t = (w_{ij}^1)_{t-1}$ pour $(i,j) \in E$ alors arrêter, sinon
aller au pas 2.

*Pas 2 : Résolution du sous-problème 2*

Résoudre (IV.1") et déterminer $(x_{ij}^2)_t$ et $(w_{ij}^2)_t$ pour $(i,j) \in E$.

Si $t = 1$ alors aller au pas 1, sinon
si $(x_{ij}^2)_t = (x_{ij}^2)_{t-1}$ et $(w_{ij}^2)_t = (w_{ij}^2)_{t-1}$ pour $(i,j) \in E$, alors arrêter, sinon
aller au pas 1.

Tel qu'il se présente, l'algorithme met en jeu un critère d'arrêt classique qui consiste en une convergence des variables primales et duales. Mais cette convergence n'étant pas garantie, on fait intervenir un deuxième critère qui consiste à s'arrêter au bout d'un certain nombre $T$ d'itérations. Incontestablement, les principales difficultés soulevées par l'algorithme, sont le choix de la partition et la notion de convergence. En effet, nous ne connaissons pas la bonne partition et nous ne savons pas si la méthode va diverger ou converger, et le cas échéant, sous quelle forme. Nous aborderons ces deux questions un peu plus en détails dans la partie heuristiques et applications (cf. § IV.4.)

## IV.3. DESCRIPTION DU LOGICIEL

La méthode ainsi présentée, requiert un algorithme efficace pour la résolution des deux sous-problèmes de flot simple (IV.1') et (IV.1"). Pour cela nous avons fait appel au code établi par Ahrens et Finke [1980]. Il existe bien d'autres codes, notamment le code NETFLOW développé par Kennington et Helgason [1980], ou plus récemment encore, celui de Bertsekas et Tseng [1988a] et celui de Chang et Chen [1989], mais le code que nous avons retenu s'avère suffisant pour une implémentation efficace de la méthode.

Partant de là, ce code de flot simple a été enrichi par des procédures diverses permettant soit de générer aléatoirement des problèmes de biflot, soit de rentrer directement des données réelles ou imposées, et de mettre ainsi en œuvre la méthode présentée. Nous

allons donc envisager une première partie pour présenter le logiciel de flot simple, puis une deuxième partie pour mettre en évidence son adaptation au problème du biflot.

## IV.3.1. OPTIMISATION DU FLOT SIMPLE

Le code utilisé est écrit en FORTRAN et consiste essentiellement en une routine NET qui permet d'optimiser un problème de flot simple de coût minimum (des adaptations existent pour d'autres problèmes de flot simple, tels que le flot maximum et les problèmes d'affectation et de transport).

L'algorithme utilisé dans ce logiciel est de type primal, les détails sur l'adaptation de la méthode du simplexe sont donnés dans Ahrens et Finke [1980]. Jusqu'à présent, les résultats de Srinivasan et Thompson [1973], et indépendamment, Glover, Karney, Klingman et Napier [1974], ont été vérifiés par nombre de chercheurs; ils ont établi que le code primal était substantiellement plus rapide que les codes mettant en jeu des algorithmes de type primal-dual ou out-of-kilter. De fait, des études récentes conduites par Grigoriadis [1986], et Kennington et Wang [1990], indiquent qu'en pratique, les algorithmes de simplexe dans les réseaux et les algorithmes de relaxation, sont les deux algorithmes les plus rapides dont on dispose actuellement pour résoudre des problèmes de flot simple à coût minimum.

Avant d'aborder la description du logiciel, à proprement parler, nous donnons, pour commencer, quelques notions de flot simple, indispensables à la compréhension de tous les développements ultérieurs.

### IV.3.1.1. Notions de flot simple

Reprenons le graphe orienté $G=(V,E)$ défini précédemment avec $|V|=N$, $|E|=M$. A chaque nœud $i \in V$, on associe un montant $d_i$ d'un certain produit. Si $d_i > 0$, le nœud est un point ressource, si $d_i < 0$, le nœud est un point demande. Les nœuds pour lesquels $d_i = 0$ sont appelés nœuds intermédiaires. Les arcs $(i,j)$ sont munis de coûts de transport $c_{ij}$ et de capacités $b_{ij}$ entiers et positifs (les coûts peuvent être nuls). L'objectif est de déterminer le flot à coût minimum à travers le réseau ainsi décrit. Le problème se formule sous forme d'un programme linéaire, soit le problème (IV.2):

$$\text{Minimiser} \sum_{(i,j)\in E} c_{ij} x_{ij} \qquad \text{(IV.2a)}$$

sous les contraintes

$$\sum_j x_{ij} - \sum_j x_{ji} = d_i \qquad i \in V, \qquad \text{(IV.2b)}$$

$$0 \le x_{ij} \le b_{ij} \qquad (i,j) \in E. \qquad \text{(IV.2c)}$$

L'équation (IV.2b) exprime la conservation du flot au niveau de chaque nœud $i$. L'hypothèse est faite que $\sum_i d_i = 0$. Si $\sum_i d_i < 0$, il n'y a pas de solution réalisable. Si $\sum_i d_i > 0$, on introduit un nœud additionnel avec une demande de $-\sum_i d_i$ et des arcs le reliant à tous les points ressource. Définissons $u_i$ et $w_{ij}$ comme étant les variables duales du problème (IV.2), associées respectivement aux contraintes (IV.2b) et (IV.2c). Le problème dual de (IV.2) peut alors s'écrire comme suit:

$$\text{Maximiser} \sum_i d_i u_i - \sum_{(i,j)\in E} b_{ij} w_{ij} \qquad \text{(IV.3a)}$$

sous les contraintes

$$u_i - u_j - w_{ij} \le c_{ij} \quad (i,j) \in E, \qquad \text{(IV.3b)}$$

$$w_{ij} \ge 0 \qquad (i,j) \in E. \qquad \text{(IV.3c)}$$

***Théorème IV.1.*** *Une solution de base du problème* (IV.2) *est associée à un arbre maximal du graphe G associé au réseau décrit.*

Pour la preuve, consulter Ahrens et Finke [1980].

A l'optimum nous avons $x$ admissible pour (IV.2), $(u, w)$ admissible pour (IV.3) et les conditions du théorème des écarts complémentaires satisfaites, soit :

$$x_{ij}(c_{ij} - u_i + u_j + w_{ij}) = 0 \qquad \text{(IV.4)}$$

$$w_{ij}(b_{ij} - x_{ij}) = 0. \qquad \text{(IV.5)}$$

En définissant les coûts réduits par

$$\bar{c}_{ij} = c_{ij} - u_i + u_j, \qquad \text{(IV.6)}$$

On peut facilement vérifier les conditions d'optimalité suivantes, pour une paire de solutions optimales primale et duale $x$ et $u$:

$$0 < x_{ij} < b_{ij} \text{ implique } \bar{c}_{ij} = 0, \tag{IV.7}$$

$$\text{Si } x_{ij} = 0 \text{ alors } \bar{c}_{ij} \geq 0, \text{ et} \tag{IV.8}$$

$$\text{Si } x_{ij} = b_{ij} \text{ alors } \bar{c}_{ij} \leq 0. \tag{IV.9}$$

Étant donnée une solution de base de (IV.2b) et (IV.2c), on peut déterminer les variables $u_i$ à partir des équations (IV.7), i.e. $\bar{c}_{ij} = 0$ ou $u_i - u_j = c_{ij}$ pour les arcs $(i, j)$ en base. Après avoir attribué une valeur quelconque à l'une des variables, la solution est déterminée de manière unique.

Un arc hors base pour lequel:

$$x_{ij} = 0 \quad \text{et} \quad \bar{c}_{ij} < 0, \quad \text{ou} \tag{IV.10}$$

$$x_{ij} = b_{ij} \quad \text{et} \quad \bar{c}_{ij} > 0 \tag{IV.11}$$

est candidat à entrer en base.

Les arcs sortants sont déterminés par le cycle unique généré par l'arc entrant, introduit dans l'arbre de base. Un arc atteignant l'une des deux bornes supérieure ou inférieure (respectivement $b_{ij}$ et 0) est sélectionné pour quitter la base.

### IV.3.1.2. Base de données

Le réseau à entrer contient, en plus de ses sommets et arcs propres, un sommet et des arcs artificiels introduits pour déterminer la solution arbre initiale. Le sommet artificiel est relié aux autres sommets par les arcs artificiels. La convention adoptée est que les arcs artificiels (au nombre des sommets propres) sont orientés du sommet artificiel vers les sommets demande ou les sommets intermédiaires, et le contraire pour les sommets ressource. Tous les arcs artificiels ont des coûts et des capacités infinis. De cette façon on peut identifier une solution irréalisable par simple consultation du coût total, infini dans ce cas. Les premiers paramètres à saisir sont donc:

- Le nombre de sommets (y compris le sommet artificiel), soit NNODES.

- Le nombre total d'arcs (y compris les arcs artificiels), soit NARCS.

- Le nombre de sommets à examiner dans l'opération de recherche de pivots, soit NEXAM.

- Un paramètre test LARGE tel que: si $|c_{ij} - u_i + u_j| \geq \text{LARGE}$, aucun sommet supplémentaire ne soit examiné dans la recherche de pivots.

- La valeur commune de $\infty$ attribuée aux coûts et aux capacités des arcs artificiels, soit INFIN.

Comme les réseaux de flot sont rarement "denses" (i.e. ne contiennent pas tous les arcs entre toutes les paires de sommets), les arcs existant avec leurs coûts et leurs capacités sont stockés dans des listes séparées. Les arcs sont numérotés de 1 jusqu'à NARCS et ordonnés par groupes de même extrémité. Ainsi nous trouverons dans le premier groupe tous les arcs dont l'extrémité est 1, dans le deuxième tous ceux dont l'extrémité est 2, etc., jusqu'à NNODES (comprenant le sommet artificiel). Les listes associés aux arcs sont alors T, C et K qui représentent respectivement, les sommets origine, les coûts et les capacités des différents arcs. L'ordre établi sur les arcs permet de définir une première liste associée aux sommets, cette fois, et permettant d'identifier les sommets extrémité des différents groupes d'arcs, soit H. A cette liste s'ajoute la liste X des montants $d_i$ disponibles au niveau de chaque sommet $i$, de dimension NNODES. La dimension de H est NNODES+1; si l'on appelle $N_i$ l'ensemble des arcs d'extrémités $i$, alors $\text{H(i)} = \sum_{j=1}^{i-1} |N_j| + 1$ avec H(1)=1. De cette façon, les origines des arcs de $N_i$ sont stockées entre les positions H(i) et H(i+1)-1 de la liste T; d'où H(NNODES+1) est égal à NARCS+1.

L'algorithme primal procède essentiellement en passant d'une solution de base réalisable à une autre. Par conséquent, toute l'information peut être retenue sur l'arbre de base. En sortie nous récupérons donc les listes I, H, X, P, W, D, Y et U, où I représente la liste des sommets, X celle des variables flot, P celle des prédécesseurs, W celle des successeurs, D la liste donnant la profondeur des sommets dans l'arbre de base, Y la liste des arcs et U celle des variables duales associées aux sommets (conservation du flot). Il est alors facile d'extraire la solution de base en relevant le flot X(I) sur les arcs allant de I à P(I) si Y(I) est positif et le contraire si Y(I) est négatif. Quant aux variables hors base saturées, elles sont identifiées grâce à la liste K. En effet tout arc hors base saturé apparaît dans cette liste avec sa capacité en valeur négative. La procédure NET de ce logiciel permet donc de donner la solution optimale à un problème de flot simple à coût minimum ainsi que les variables duales associées aux différents sommets.

## IV.3.2. GÉNÉRATION ALÉATOIRE ET PROCÉDURES ANNEXES

Cette partie sera consacrée à l'adaptation du logiciel au problème du biflot. De fait, nous avons introduit de nouvelles listes, de nouvelles grandeurs et de nouvelles procédures permettant à la fois, d'implémenter la méthode proposée, et de réaliser des tests numériques sur des problèmes de biflot construits ou générés aléatoirement.

Pour la génération aléatoire, trois principales listes de paramètres sont à saisir. La première se rapporte à des paramètres de réseau et d'optimisation, nous y comptons:

- Un chiffre aléatoire : ZS (nécessaire au générateur aléatoire, compris de préférence entre $10^2$ et $10^4$).

- Le nombre de sommets du réseau : ZN.

- Le nombre d'arcs : ZA.

- Les deux paramètres de flot simple : NEXAM et LARGE.

- Trois paramètres ZI, ZJ, ZJ1 représentant des grandeurs infinies, qu'on explicitera ultérieurement.

La deuxième liste comprend des paramètres de réseau et des paramètres propres au premier flot:

- Le nombre de nœuds ressource : ZPOS.

- Le nombre de nœuds demande : ZNEG.

- La borne supérieure sur les coûts : ZC.

- Les bornes inférieure et supérieure sur la capacité : ZK1, ZK.

- Le flot total à transporter : ZF (qui doit être supérieur au maximum entre le nombre de nœuds demande et le nombre de nœuds ressource).

Enfin la troisième liste met en jeu les paramètres du deuxième flot, à savoir:

- Le nombre de nœuds ressource : MPOS.

- Le nombre de nœuds demande : MNEG.

- Le flot total à transporter : MF.

La génération d'un problème de biflot aléatoire se fait d'abord par la génération d'un réseau aléatoire, puis par l'attribution de montants à transporter sur ce même réseau, et ce pour les deux flots. A cet effet, six listes sont mises à jour : T, C, K, H, X et X1. Les trois premières sont dimensionnées sur le nombre d'arcs et représentent respectivement: les origines des arcs, leurs coûts et leurs capacités. Elles sont faciles à générer grâce au générateur de nombres aléatoires RAN. Ce générateur aléatoire donne l'avantage au code de regénérer le même problème de biflot, si tous les paramètres d'input sont les mêmes.

Les trois listes suivantes sont associées aux sommets du réseau et renseignent, indirectement pour H, sur les extrémités des arcs ainsi que sur les montants liés aux ressources des deux flots, pour X et X1. Leur génération se fait de façon plus élaborée grâce à la procédure ODEAL. Cette dernière utilise la fonction RAN et permet de générer aléatoirement des montants (positifs et négatifs) dont les sommes respectives correspondent au flot total à transporter, et ce pour chacun des deux flots. Par ailleurs, elle permet de, judicieusement, mettre à jour H, en maintenant un ordre croissant sur les valeurs de cette liste.

En même temps que le réseau est généré, une troisième procédure EXT permettra de rendre son appréhension plus conviviale pour l'utilisateur. En fait, cette procédure permet, par l'intermédiaire de la liste H, de calculer les extrémités des différents arcs. Outre la simplicité qu'elle confère à la représentation du réseau, la liste des extrémités entre en jeu, en même temps que la liste T, dans le calcul des variables duales (associées aux arcs), indispensables à l'application de la méthode.

Par la suite, et tout au long de l'optimisation, nous travaillerons sur un réseau augmenté où tous les arcs artificiels ont été dupliqués. Cette opération est réalisée par la procédure DUPLIC. Effectivement, cela revient doubler le nombre d'arcs artificiels, qui passe ainsi du nombre de nœuds à deux fois ce nombre. Ceci vient du fait que les arcs artificiels tels qu'ils ont été définis initialement, donnent une forme de rigidité au réseau puisque leur orientation change en fonction des montants impliqués pour chacun des flots. Pour pallier

à cet inconvénient, on dédouble chacun des arcs artificiels en les orientant dans le sens inverse à leur premier sens défini. Cette duplication des arcs artificiels (avec des coûts et des capacités toujours infinis) définit le réseau définitivement et de manière univoque, quelque soient les montants mis en jeu par les deux flots. Ce qui nous permet de travailler sur un réseau unique dont les paramètres sont stockés dans le fichier RESALEADUP.

Les deux flots ainsi mis en jeu sont optimisés indépendamment l'un de l'autre au sens du coût minimum, en utilisant la procédure NET. Dans la plupart des cas, les deux flots simples ne sont pas réalisables. La procédure PERMUT permet alors de les rendre admissibles (quand cela est possible), en effectuant une série de permutations sur les montants associés aux sommets du réseau. Cette permutation a pour effet de changer uniquement le cheminement des flots, le réseau restant inchangé. Il s'en suit que la série de permutations peut être parfois très longue (ou même infini) avant l'obtention de solutions admissibles. Pour éviter cet inconvénient, un nombre de permutations maximum peut être saisi par l'utilisateur. Au delà de ce nombre, une option permet de rentrer un nombre de permutations supérieur, ou de revenir à la génération d'un nouveau réseau mieux conditionné.

Pour finir, le résultats de cette double optimisation sont superposés puis soumis à une confrontation (au sens des capacités conjointement utilisées), avant d'être stockés dans des fichiers différents, en rapport avec l'un des deux cas possibles suivants:

- Les deux flots libres ne se gênent pas, et le biflot résultant est optimal. La solution est stockée dans un fichier (BIFLOTOPT) qui pourra être utilisé pour exhiber un biflot "irréalisable" (au sens de optimisable), moyennant certaines modifications judicieuses des coûts ou des capacités. Dans ce fichier, nous disposons également des arcs, à flot positif, communs aux deux solutions de flot simple, avec les flots correspondants et la capacité disponible sur ces arcs. Ce qui nous permet de modifier plus facilement le fichier RESALEADUP et de relancer l'optimisation.

- Les deux flots se gênent en dépassant la capacité sur certains arcs. La solution correspondante ainsi que les arcs "violés" sont enregistrés dans un fichier (CAPAVIOLES), et l'optimisation de l'exemple peut commencer.

Les mêmes étapes sont observées dans le cas où un problème réel, ou construit, est saisi par l'utilisateur. Toutes les données sont stockées dans le fichier RESNONALEADUP.

Pour donner un aspect plus concret à tous ces développements, nous présentons un exemple pour illuster la représentation conventionnelle d'un réseau quelconque.

**Exemple:** soit le réseau suivant

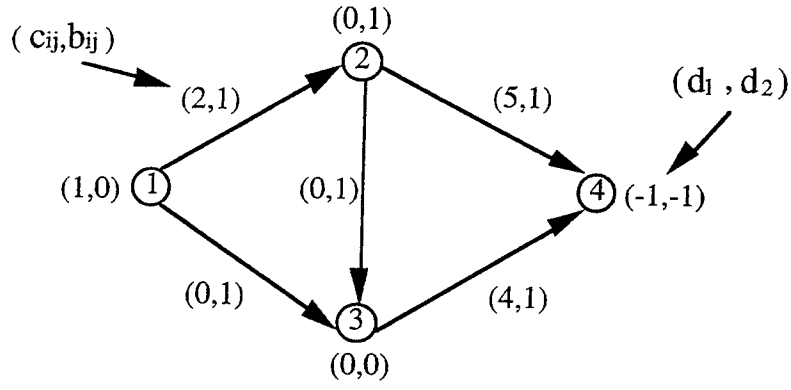



**Figure IV.1** Exemple de biflot.

Le réseau ci dessus sera représenté par les paramètres et les listes suivantes:

NNODES = 5; NARCS = 13
NEXAM = 1; LARGE =100
INFIN =1000

| J | T(J) | C(J) | K(J) |
|---|---|---|---|
| 1 | 5 | 1000 | 1000 |
| 2 | 1 | 2 | 1 |
| 3 | 5 | 1000 | 1000 |
| 4 | 1 | 0 | 1 |
| 5 | 2 | 0 | 1 |
| 6 | 5 | 1000 | 1000 |
| 7 | 2 | 5 | 1 |
| 8 | 3 | 4 | 1 |
| 9 | 5 | 1000 | 1000 |
| 10 | 1 | 1000 | 1000 |
| 11 | 2 | 1000 | 1000 |
| 12 | 3 | 1000 | 1000 |
| 13 | 4 | 1000 | 1000 |

| I | H(I) | X(I) | X1(I) |
|---|---|---|---|
| 1 | 1 | 1 | 0 |
| 2 | 2 | 0 | 1 |
| 3 | 4 | 0 | 0 |
| 4 | 7 | -1 | -1 |
| 5 | 10 | 0 | 0 |
| 6 | 14 | | |

Le sommet 5 est le sommet artificiel, il est relié à tous les autres sommets par deux arcs (un arc rentrant, un autre sortant) dont les coûts et les capacités sont égaux à INFIN=1000. La liste T donne les origines des arcs, les extrémités peuvent se déduire de la liste H. Ainsi, le 4ème arc correspond à l'arc $1 \rightarrow 3$, le 5ème à l'arc $2 \rightarrow 3$, etc....

Enfin, pour récapituler, nous donnons un résumé des procédures utilisées dans notre logiciel:

- RAN : permet de générer des nombres aléatoires entre 0 et 1.

- ODEAL : rentre dans la génération des listes H et X et utilise la fonction RAN.

- EXT : permet de mettre à jour les extrémités des arcs du réseau à partir de la liste H.

- NET : permet d'optimiser un problème de flot simple, à coût minimum en l'occurrence.

- DUPLIC : permet de dupliquer les arcs artificiels du réseau traité afin d'optimiser le biflot sur un réseau unique.

- PERMUT : permet d'effectuer une permutation sur les éléments d'un vecteur, ici le vecteur des montants associés aux sommets.

En marge de ce travail, nous avons développé un programme permettant de mettre à jour, extérieurement au logiciel, la formulation (en programme linéaire) du problème de biflot généré aléatoirement, ou saisi par l'utilisateur. Le programme linéaire correspondant est ainsi stocké dans un fichier PL. Ce fichier sera destiné à être optimisé par des logiciels de

programmation linéaire, en vue de l'obtention d'une borne inférieure (en continu) pour le problème entier. Nous utiliserons en l'occurrence le logiciel CPLEX [1993].

Les procédures mises en jeu dans ce programme sont:

- SUPRESS : permet de supprimer toute la construction artificielle du réseau.

- PL : permet d'écrire le programme linéaire en continu associé au biflot traité.

- CHART : rentre dans la performance de la procédure PL, en permettant le passage au mode caractère.

## IV.4. HEURISTIQUES ET APPLICATIONS

La mise en œuvre de l'algorithme proposé pour la résolution du problème de biflot requiert un choix préalable de la partition $H$, qui reste somme toute assez difficile, puisque l'on ne connait pas, à priori, de bonne partition. Nous proposons deux idées principales pour tester la bonne partition (Rezig et Finke [1992]). La première respecte l'esprit de la méthode, en ce sens que la partition est fixée initialement et reste inchangée tout au long de l'algorithme: c'est l'approche statique. La seconde est une innovation, du fait que la partition est changée à chaque étape d'optimisation des sous-problèmes (IV.1') et (IV.1"): c'est l'approche dynamique. Nous verrons un peu plus loin la justification théorique d'une telle approche.

Par souci de clarté, nous ordonnerons les flots traités en leur affectant la désignation définitive de flot1 (f1), et flot2 (f2). Nous verrons plus loin l'utilité de cela pour la dénomination des différentes heuristiques.

### IV.4.1. APPROCHE STATIQUE

Dans l'approche statique, la partition $H$ est fixée définitivement tout au long de l'optimisation. Nous avons fait trois choix essentiels qui ont donné lieu aux trois applications suivantes (Rezig et Finke [1992]).

#### 1ère application

La partition a été choisie comme suit: les deux flots sont optimisés indépendamment avec les capacités réelles du réseau; si en les combinant, les capacités sont respectées, on dispose d'un biflot optimal. Sinon on retiendra comme arcs de $H$ les arcs sur lesquels les

deux flots se gênent. Pratiquement cela revient à diminuer la capacité sur les arcs dont la capacité est violée, soit $H$, et à augmenter le coût sur les autres arcs, et ce pour le premier flot optimisé. Le flot suivant, comme l'indique l'algorithme, sera traité exactement à l'inverse. Comme nous disposons de deux flots, nous pouvons commencer par f1 ou par f2. Ce qui donne lieu aux deux premières heuristiques suivantes:

- **Sf1cap** : S, pour statique; f1, parceque l'on commence l'optimisation par le flot1 et cap, parceque dans ce cas, on diminue la capacité sur les arcs violés, pour le premier flot optimisé, soit f1.

- **Sf2cap** : définie de la même manière que Sf1cap mais en commençant, cette fois, par le deuxième flot f2. Au risque de se répéter, disons simplement que les capacités sont diminuées sur les arcs violés, que les coûts sont augmentés sur le reste des arcs, et ce pour f2 (alors que pour f1, nous adopterons le scénario inverse.)

Les coûts modifiés sont mis à jour de la façon suivante:

On calcule d'abord les variables duales associées aux contraintes de capacité (sur les arcs). Seuls les arcs hors base saturés sont assortis de variables duales $w$ positives (nulles en cas de dégénérescence), tous les autres arcs mettant en jeu des variables égales à zéro. Pour le montrer, revenons aux notions de flot simple exposées au § IV.3.1.1. En effet, étant donné le coût réduit $\bar{c}_{ij} = c_{ij} - u_i + u_j$, on peut réécrire la contrainte (IV.3b) du problème dual (IV.3) comme:

$$w_{ij} \geq -\bar{c}_{ij}. \tag{IV.12}$$

Par ailleurs, la fonction objective (IV.3a) étant une fonction de minimisation, et les coefficients des variables $w_{ij}$ étant $-b_{ij}$, toute solution optimale de (VI.3) se verra attribuer la plus petite valeur possible à $w_{ij}$. Cette observation, en coordination avec (IV.3c) et (IV.12) impliquent que:

$$w_{ij} = \max\left\{0, -\bar{c}_{ij}\right\}. \tag{IV.13}$$

En reprenant les conditions d'optimalité pour le problème de flot simple (IV.7) à (IV.9) et l'équation (IV.13), on peut déduire le calcul suivant pour les variables $w_{ij}$:

$$\begin{cases} w_{ij} = u_i - u_j - c_{ij} & (i,j) \text{ saturé hors base} \\ w_{ij} = 0 & \text{ailleurs.} \end{cases} \tag{IV.14}$$

Les nouveaux coûts $c'$ sont alors donnés par la formule:

$$c'_{ij} = c_{ij} + w_{ij}. \qquad \text{(IV.15)}$$

Où $c_{ij}$ est le coût d'optimisation, et $w_{ij}$ est calculée à partir de la formule (IV.14), les variables $u$ étant directement données par le logiciel.

Sur un plan pratique, deux problèmes d'ordre technique se sont posés:

(a) Le logiciel ne fonctionne pas avec des capacités nulles. Or les capacités étant diminuées au fur et à mesure des itérations, on peut se retrouver avec une ou plusieurs capacités égales à zéro. Dans ce cas, pour simuler une capacité annulée, nous avons introduit une deuxième grandeur infinie appelée infin2 (en plus de l'infini des arcs artificiels que l'on désignera par infin1). Infin2 est une valeur assez grande qui permet, étant attribuée comme coût à un arc dont la capacité devient nulle, d'en empêcher l'utilisation tout en lui associant sa capacité réelle (sauf si le flot n'a pas d'autre choix de passage). Toutefois infin2 doit rester largement inférieur à infin1, de façon à privilégier les arcs réels par rapport aux arcs artificiels.

(b) dans le même ordre d'idée, les coûts augmentés peuvent atteindre des proportions infinies en dépassant infin1. Comme dans ce cas les arcs de la construction artificielle risquent d'être favorisés, on fait suivre le calcul des coûts augmentés par le test suivant:

$$\text{Si } c'_{ij} > \text{infin2} \quad \text{alors} \quad c'_{ij} = \text{infin2}. \qquad \text{(IV.16)}$$

Mais pour permettre une plus grande flexibilité, nous avons transformé ce test en faisant intervenir une troisième grandeur infinie (infin3):

$$\text{Si } c'_{ij} > \text{infin2} \quad \text{alors} \quad c'_{ij} = \text{infin3}. \qquad \text{(IV.17)}$$

Nous pourrons ainsi étudier l'influence de ces paramètres sur la qualité des solutions obtenues. Remarquons au passage que le premier test est restitué en prenant infin2=infin3.

### 2ème application

Cette deuxième application prend comme partition $H$, les arcs sur lesquels les flots libres ne se gênent pas par superposition. Mais pour simplifier la compréhension, nous pouvons dire, par abus de langage, qu'elle maintient la même partition $H$ de la 1ère application, mais en inverse simplement le rôle. De cette manière, en commençant par le flot1 nous

pouvons définir une troisième heuristique "duale" de Sf1cap. De même qu'en commençant par le flot2, nous pouvons définir une quatrième heuristique "duale" de Sf2cap. Soient Sf1coût et Sf2coût ces deux heuristiques respectives:

- **Sf1coût** : toujours S, en référence à statique; f1 parceque le premier flot optimisé est le flot1 et coût, pour préciser que cette fois, c'est le coût qui est augmenté sur les arcs violés, et non la capacité qui est diminuée).

- **Sf2coût** : définie d'une manière analogue à Sf1coût, à la différence près que le premier flot optimisé est f2.

D'une manière analogue, nous reprendrons les mêmes remarques et calculs précédents, sans les mentionner ici.

**3ème application**

Cette troisième application est surtout introduite pour étudier l'influence du choix de la partition sur la qualité de la solution. De fait la partition *H* est choisie ici aléatoirement. Les quatre heuristiques précédemment définies sont ainsi mises en jeu avec une partition fixée aléatoirement par l'utilisateur. Dans ce groupe d'heuristiques deux paramètres sont à saisir: le nombre d'arcs de la partition et les arcs eux-mêmes. On retrouve donc les quatre heuristiques précédentes avec les dénominations suivantes: AlSf1coût, AlSf2coût, etc., avec la désignation "Al" signifiant le choix aléatoire de la partition.

## IV.4.2. APPROCHE DYNAMIQUE

Malgré toutes les applications mises en jeu par l'approche statique, le problème du choix de la partition reste encore posé. D'une part les deux premières applications mettent en jeu deux partitions qui ne sont pas forcément les meilleures. D'autre part la troisième application ne permet d'explorer qu'un nombre fini de partitions. Partant de là, nous avons voulu enrichir les choix de partitions précédents en développant une nouvelle approche: l'approche dynamique (Rezig et Finke [1992]).

L'idée d'une telle approche vient de l'observation suivante: les modifications de coûts n'interviennent que sur les arcs saturés hors base, lesquels mettent en jeu des variables prix positives (cf. § IV.4.1). Par conséquent, il paraît intuitif de vouloir d'une part pénaliser les arcs saturés (hors base), en augmentant leur coût par l'intermédiaire des variables prix. Et d'autre part, de faire une allocation de capacité sur les arcs de base, dont

les variables prix sont nulles, mais qui véhiculent un flot positif (nul en cas de dégénérescence).

Ainsi en optimisant chaque sous-problème (SP1) et (SP2), on obtient une solution de flot simple mettant en jeu des arcs de base et des arcs hors base. Pour la construction du sous-problème (SP2) on prendra comme sous ensemble $\overline{H}$ l'ensemble des arcs de base, et comme sous ensemble $H$ l'ensemble des arcs hors base; ces arcs étant mis en évidence par (SP1). Réciproquement, pour construire (SP1) on identifiera $H$ aux arcs de base et $\overline{H}$ aux arcs hors base; ces arcs étant mis en évidence par (SP2).

Comme la solution de base change au fur et à mesure des itérations, on s'explique bien la dénomination de dynamique, par opposition à l'approche statique où la partition reste inchangée tout au long de l'optimisation.

Le principe de l'approche dynamique est alors simple: augmenter le coût sur les arcs hors base (effectivement les arcs saturés), et diminuer la capacité sur les arcs de base. Comme nous pouvons commencer par f1 ou par f2 nous définirons les deux heuristiques suivantes:

- **Df1** : D, pour dynamique et f1 parceque le premier flot optimisé est le flot1.

- **Df2** : définie de la même manière que Df1 mais en initialisant l'optimisation par le flot2.

Ces heuristiques possèdent les mêmes caractéristiques que leurs homologues statiques, en ce qui concerne le calcul et la mise à jour des coûts et des capacités.

### IV.4.3. RÉSULTATS NUMÉRIQUES

Pour tester toutes ces idées d'heuristiques, nous avons généré un premier échantillon aléatoire de problèmes de biflot. La taille de ces problèmes varie entre 12 à 100 sommets et 24 à 250 arcs. Ce premier échantillon d'exemples, de taille réduite, nous a permis, entre autres, de faire une première évaluation de nos heuristiques et en même temps de visualiser quelques aspects de la convergence. Par ailleurs la majorité de ces exemples ont été choisis de façon à ne pas donner lieu à des solutions triviales, correspondant par exemple à l'une des solutions de coût minimum pour l'un des deux flots.

Pour ce qui est de l'évaluation, nous avons pu disposer d'une borne de comparaison, correspondant à la solution du problème en continu. Cette borne inférieure, calculée à

partir du logiciel de programmation linéaire CPLEX, correspondrait, étant donnée l'intégralité des flots, à la solution optimale. Précisons au passage que d'autres bornes ont été évaluées mais non prises en compte. Il s'agit en l'occurrence de la borne correspondant à la somme des coûts minimaux des deux flots simples optimisés indépendamment. La deuxième borne est celle associée au coût minimum, obtenu par assimilation des deux flots à un flot simple (les deux flots sont ainsi mélangés). Dans le premier cas, la borne reste assez éloignée de l'optimum en continu, et dans le deuxième cas le résultat peut être encore plus mauvais, dans la mesure où, si les montants des deux flots sont opposés, la borne est tout simplement égale à zéro.

Les résultats obtenus, par la mise en œuvre de nos heuristiques, sont donnés dans le tableau ci dessous (voir Figure IV.2). Dans ce tableau, nous indiquons pour chaque problème, la taille (en nombre de sommets et nombre d'arcs), ainsi que l'erreur relative par rapport à l'optimum continu.

| Problème | Taille (sommets-arcs) | Erreur relative |
|---|---|---|
| 1 | (16-52) | 0,043 |
| 2 | (16-54) | 0,002 |
| 3 | (20-110) | 0,013 |
| 4 | (60-120) | 0,002 |
| 5 | (50-140) | 0,007 |
| 6 | (45-150) | 0,002 |
| 7 | (75-220) | 0,007 |
| 8 | (102-245) | 0,004 |
| 9 | (100-250) | 0,009 |

**Figure IV.2** Erreurs relatives entre solutions heuristiques et solutions continues.

Malgré la taille des problèmes traités, ces résultats permettent de dégager un optimisme certain quant aux performances générales de la méthode . En effet, on peut constater que l'erreur relative est assez faible (5% au pire des cas, et une moyenne d'erreur de 1%), ce qui montre que la solution heuristique est finalement très proche de l'optimum.

Nous pouvons passer à présent à l'étude comparative des différents groupes d'heuristiques. Les résultats de la Figure IV.3 mettent en confrontation l'approche dynamique et l'approche statique. L'approche statique est représentée par les deux dernières colonnes sous forme d'approche statique critique et statique aléatoire. Cette différenciation est en rapport avec les trois applications évoquées dans l'approche statique (cf. § IV.4.1). Ainsi l'approche statique critique se rapporte aux deux premières applications de l'approche statique, mettant en jeu un sous-ensemble d'arcs critique (les arcs de capacité violée). L'approche statique aléatoire est en rapport avec la troisième application mettant en œuvre une partition choisie aléatoirement.

| Problème | Taille (sommets-arcs) | Approche dynamique | Approche statique critique | Approche statique aléatoire |
|---|---|---|---|---|
| 1 | (50-140) | 0,007 | 0,01 | pas de solution |
| 2 | (60-140) | 0,004 | optimum | pas de solution |
| 3 | (45-150) | 0,002 | 0,028 | 0,035 |
| 4 | (75-220) | 0,007 | 0,025 | 0,01 |
| 5 | (90-190) | optimum | 0,003 | 0,1 |
| 6 | (102-245) | 0,003 | 0,015 | 0,007 |
| 7 | (102-245) | 0,004 | 0,013 | pas de solution |
| 8 | (100-250) | 0,009 | 0,041 | 0,08 |

**Figure IV.3** Comparaison des différents groupes d'heuristiques (1er échantillon).

Les résultats exposés permettent de déduire une supériorité relative de l'approche dynamique par rapport à l'approche statique. Sur les 7 problèmes présentés, 6 d'entre eux associent une meilleure solution à l'approche dynamique, et la différence peut aller jusqu'à 3%. Notons que cette supériorité se traduit également par l'avantage qu'offre l'approche dynamique en exhibant en moyenne un plus grand nombre de solutions réalisables.

Par ailleurs, ces résultats nous permettent de noter l'influence de la partition sur la qualité de la solution. Ainsi, sur le problème 5, nous pouvons détériorer la résolution en passant d'une solution à 0.3% (approche statique critique) à une solution à 10% (approche statique aléatoire). Dans le pire des cas (problèmes 1, 2, 7), un simple changement de partition peut induire l'absence totale de solutions, alors que les approches dynamique et statique critique mettent en évidence de très bonnes solutions. D'un autre côté, nous pouvons également améliorer la solution en passant d'un sous-ensemble d'arcs critique à un autre sous-ensemble d'arcs (problèmes 4 et 6). Ainsi nous confirmons l'importance du choix de la partition dans la mise en œuvre de nos heuristiques statiques, et nous soulignons, par là même, l'intérêt et le succès de l'approche dynamique.

Nous allons, à présent, aborder les notions d'allocation de capacité et de convergence. En effet, les heuristiques mettent en évidence, à travers la plupart des problèmes traités, une réelle allocation de capacité. La situation que nous avons observée est la suivante: étant donné (sans perte de généralité) un arc saturé par les deux flots optimaux, optimisés séparément, cet arc est toujours saturé par les deux flots dans la solution heuristique. Mais surtout, la répartition de la capacité entre les deux flots est non triviale. Autrement dit, il y a une réelle tendance à répartir la capacité entre les deux flots. Une autre allocation consisterait, d'une façon plus simple, à attribuer la totalité de la capacité à l'un ou à l'autre des deux flots. Les différences de coût entre les différentes heuristiques sont souvent en rapport avec cette différence d'allocation. De fait, lorsque les deux types d'allocation sont mis en évidence par différentes heuristiques, le deuxième type d'allocation produit presque toujours le plus grand coût. Par ailleurs, ces aspects de saturation sur les arcs critiques pourraient amener la conjecture suivante: étant donné un arc saturé par les deux flots simples de coût minimum, cet arc sera toujours saturé dans la solution du biflot optimal. Malheureusement, on s'aperçoit très vite que cette conjecture est faussée par l'aspect d'admissibilité de la solution, prioritaire dans ce genre de problème par rapport à l'aspect de minimisation du coût. La difficulté est double: assurer la réalisabilité, puis trouver la meilleure solution réalisable au sens du coût minimum. Le contre exemple suivant (Figure IV.4) en donne une parfaite illustration:

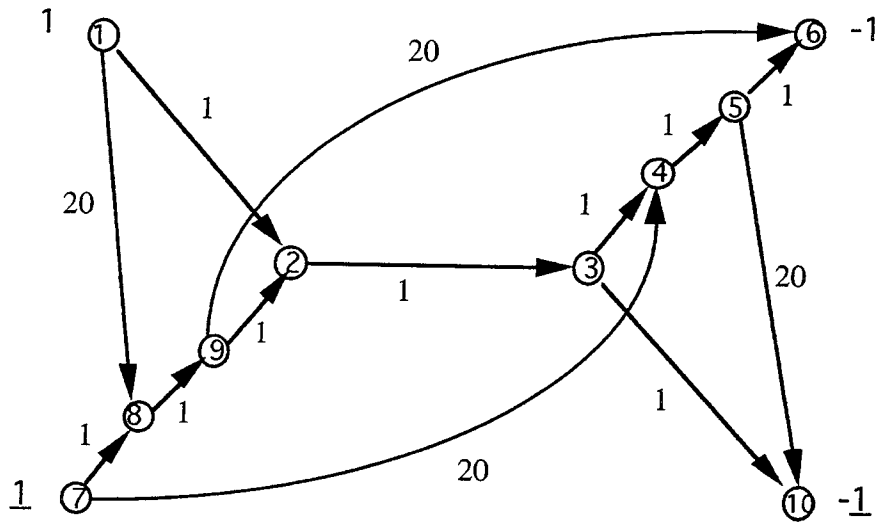



**Figure IV.4** Exemple de biflot.

Dans cet exemple, les coûts sont donnés en regard des arcs, toutes les capacités sont égales à 1. Les demandes, indiquées en regard des sommets, sont de 1 pour les deux flots (la notation soulignée désignant le flot2). Dans la solution de coût minimum, le flot1 emprunte le chemin $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 6$, et le flot2 le chemin $7 \rightarrow 8 \rightarrow 9 \rightarrow 2 \rightarrow 3 \rightarrow 10$. La superposition des deux flots est donc irréalisable puisque l'arc $2 \rightarrow 3$ est saturé à 1 par les deux flots. La solution optimale de ce problème consiste alors à envoyer, pour chaque flot, une unité sur les chemins respectifs suivants: $1 \rightarrow 8 \rightarrow 9 \rightarrow 6$ et $7 \rightarrow 4 \rightarrow 5 \rightarrow 10$. Ces deux chemins sont certes beaucoup plus coûteux mais correspondent à la seule solution réalisable, et l'arc $2 \rightarrow 3$ ne sera donc pas saturé.

En ce qui concerne la convergence, elle se manifeste sous forme d'un cycle de solutions, de "longueur" variable. Autrement dit, la méthode ne s'arrête pas sur la solution optimale, ou de meilleure coût, mais y revient après avoir parcouru un ensemble de solutions pouvant être réalisables ou non réalisables. C'est ce que l'on désignera par le cyclage (voir exemple décrit en Figure IV.5).

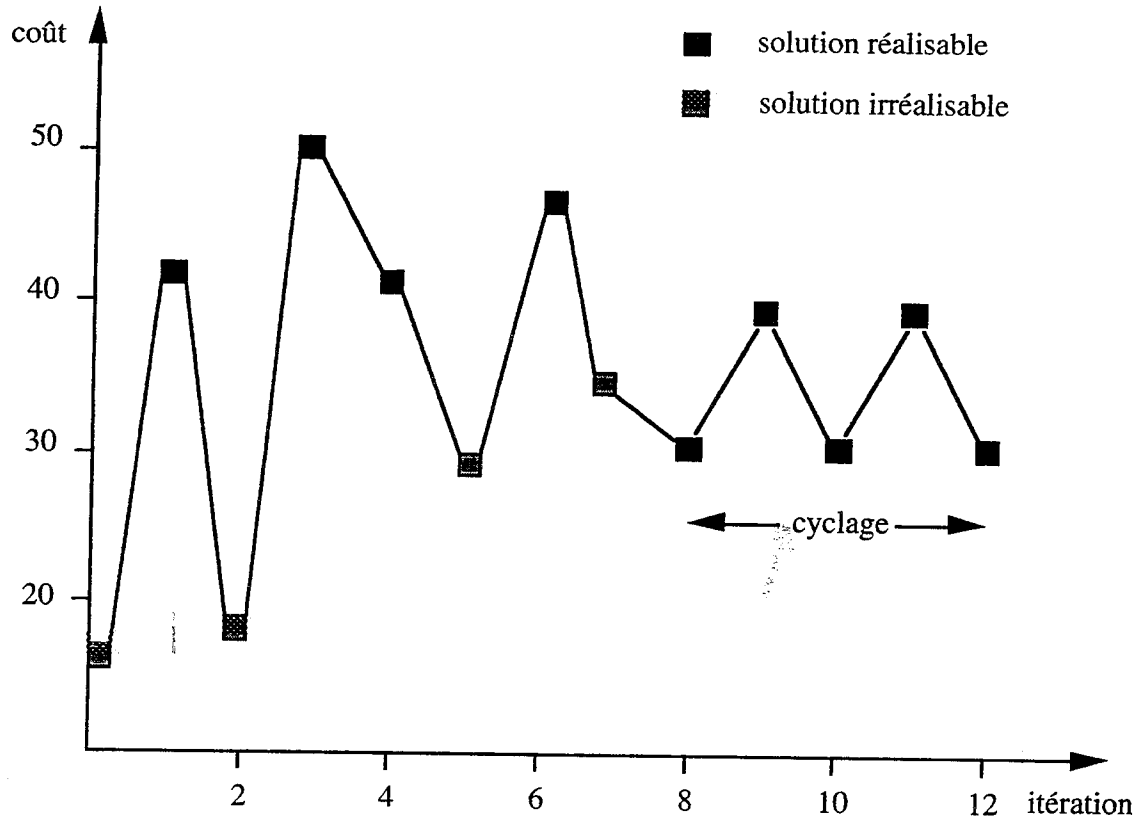



**Figure IV.5** Exemple de cyclage.

Dans l'exemple de la Figure IV.5, le calcul reproduira, à partir de la 8ème itération, les deux mêmes solutions réalisables, l'une de coût 30 (meilleure solution), l'autre de coût 39. Cette situation de reproduction des mêmes solutions est appelée cyclage. Sur la totalité des exemples traités, on observe ce cyclage, avec un diagramme en dents de scie où le calcul oscille entre des solutions de coûts variables, puis cycle sur un ensemble de solutions contenant la solution réalisable de meilleur coût (voir Figure IV.5). On retrouve là, la tendance oscillatoire des techniques de relaxation lagrangienne. Ce qui est assez compréhensible, puisque dans la décomposition mixte, nous faisons à la fois de la décomposition par les prix, et de la décomposition par les ressources. Le nombre de solutions contenues dans un cycle peut varier en moyenne entre 4 et 15. L 'arrêt sur une solution unique (cycle contenant une seule solution) a pu être observé dans 18% des cas. Ce qu'il faut remarquer, c'est que dans le cycle on peut passer d'une solution de moindre coût à une solution de coût supérieur, ou d'une solution réalisable à une solution irréalisable.

Partant de là, nous pouvons réactualiser le critère d'arrêt mis en jeu par l'algorithme du biflot (cf. § IV.2), en faisant intervenir cette notion de cyclage. Informatiquement cela se traduit par le stockage de la première solution réalisable de meilleur coût, ainsi que de toutes les solutions intermédiaires, entre l'occurrence de cette première solution et le retour à cette dernière. l'identification d'un cyclage permet alors d'arrêter l'algorithme. Il s'en suit que cette identification ne peut avoir lieu si le calcul cycle sur une solution différente de celle ayant été stockée (cette solution serait évidemment de même coût). Ou encore, si le cycle de solutions n'est pas élémentaire. Mais à ce moment là, on fait intervenir le critère du nombre maximum d'itérations.

Dans un deuxième temps, nous avons voulu analyser ces aspects de convergence et de performance sur des problèmes test de taille beaucoup plus importante. Pour ce faire, nous avons généré un deuxième échantillon d'exemples dont la taille peut aller jusqu'à 500 sommets et 7000 arcs. Dans le tableau ci dessous (Figure IV.6), nous analysons les résultats obtenus par les différents groupes d'heuristiques, en termes d'erreur relative par rapport à l'optimum continu. Nous remarquons que les tendances observées sur le premier échantillon sont maintenues: les erreurs sont très faibles (< 1%). De même l'approche dynamique reste quantitativement et qualitativement meilleure que l'approche statique (ie. une différence pouvant aller jusqu'à 6% et une mise en jeu d'un plus grand nombre de solutions réalisables). La sensibilité de la méthode au choix de la partition est encore confirmé ici. Pour ce qui est de la convergence, elle se manifeste toujours par un cyclage sur la meilleure solution, avec une moyenne de 20 itérations, et 4 à 15 solutions par cycle (le nombre de solutions contenues dans le cycle est indépendant de la taille des exemples). Les heuristiques ont donc tourné avec un nombre maximum d'itérations égal à 100, en induisant des temps d'exécution très faibles (suite à l'efficacité de la routine de flot simple.)

| Problème | Taille (sommets-arcs) | Approche dynamique | Approche statique critique | Approche statique aléatoire |
|---|---|---|---|---|
| 1 | (229-2145) | 0,002 | 0,003 | pas de solution |
| 2 | (321-2514) | 0,002 | 0,010 | 0,013 |
| 3 | (314-3214) | 0,007 | 0,058 | 0,050 |
| 4 | (341-4125) | 0,009 | 0,010 | 0,004 |
| 5 | (420-5000) | 0,009 | 0,042 | 0,068 |
| 6 | (168-5874) | 0,003 | 0,004 | 0,050 |
| 7 | (165-6000) | 0,011 | 0,039 | 0,022 |
| 8 | (325-6954) | 0,008 | 0,030 | 0,010 |
| 9 | (500-7000) | 0,013 | 0,035 | pas de solution |

**Figure IV.6** Comparaison des différents groupes d'heuristiques (2ème échantillon).

Pour finir, nous donnons quelques remarques concernant la sensibilité des heuristiques à certains choix de paramètres. Ainsi, les différentes heuristiques ne semblent pas sensibles à de petites variations des paramètres infin2 et infin3. Une bonne appréciation de infin2 se situe dans l'intervalle suivant: [borne supérieure sur les coûts - infin1/20], infin3 est pris égal à infin2. Une série de tests a été effectuée en prenant comme valeur pour infin2, les valeurs des variables duales associées aux arcs hors base saturés en optimisation libre. En dépit de quelques essais prometteurs, on s'aperçoit très vite que la démarche est erronée, dans la mesure où le test sur les coûts (IV.17) peut les ramener à des valeurs inférieures à leurs propres coûts réels.

D'un autre côté, les heuristiques sont sensibles à la solution initiale, puisque commencer par f1 ou par f2 ne donne pas toujours la même solution finale. Ce qui s'explique aisément par la différence des informations primales et duales, dans les deux cas.

Enfin, les essais d'amélioration, par changement de partition, sont très fréquemment associés à des partitions consistant en des combinaisons d'arcs appartenant à l'ensemble critique. Nous avons également testé quelques variantes de nos heuristiques par rapport à la mise à jour des nouveaux coûts. Pour ce faire, nous avons réalisé des tests en modifiant les coûts calculés, de manière cumulative; la formule (IV.15) devenant alors: $c'_{ij} = c'_{ij} + w_{ij}$. Dans un deuxième temps, nous avons supprimé le test sur les coûts (IV.17). Dans les deux cas, les résultats ont été, la plupart du temps, beaucoup moins appréciables.

## IV.5. CONCLUSION

L'utilisation d'une méthode de décomposition mixte dans le cas du biflot entier de coût minimum a été satisfaisante à plus d'un titres. En effet, cette méthode nous a permis de déboucher sur un schéma de résolution efficace, mettant en jeu deux problèmes de flot simple à coût minimum. Ce schéma de résolution se répercute directement sur le temps d'exécution de la méthode qui reste dominé par les temps de calcul correspondant à la résolution d'un problème de flot simple (quelques secondes). Par ailleurs, les difficultés soulevées par cette méthode, notamment le choix de la partition, ont suscité des analyses intéressantes, tant sur le plan théorique que sur le plan pratique. Ces analyses nous ont permis de développer deux approches essentielles: l'approche statique et l'approche dynamique. Sur un plan pratique, ces approches ont prouvé leur efficacité de par les performances réalisées sur des problèmes test aléatoires dont la taille pouvait aller jusqu'à 500 sommets et 7000 arcs. En effet, les expériences de calcul effectuées permettent d'affirmer que les résultats fournis s'écartent très peu (en valeur relative) des résultats obtenus par programmation linéaire (en relaxant la contrainte d'intégrité) et par conséquent, sont proches de l'optimalité en nombres entiers. En termes de comparaison, nous avons enregistré une supériorité relative de l'approche dynamique, par rapport à l'approche statique. En effet l'approche dynamique met en évidence des résultats quasiment meilleurs, et offre l'avantage d'un plus grand nombre de solutions réalisables. Enfin, cette supériorité de l'approche dynamique est, d'autant plus satisfaisante, qu'elle constitue pour nous un véritable moyen de pallier à un des problèmes majeurs de la décomposition mixte: le problème du choix de la partition.

Un autre problème demeure cependant: le problème de la convergence. De fait, la convergence de la méthode semble assez complexe, et jusqu'à présent, les conclusions acquises ne sont qu'empiriques. Nous avons observé, sur la totalité des problèmes test, une convergence sous forme de cyclage sur un sous ensemble de solutions contenant la solution de moindre coût. Mais, même si aucune divergence n'a été observée, nous restons très prudents quant aux conclusions concernant cette délicate et cruciale notion de convergence.

# Chapitre V

# VALIDATION DES HEURISTIQUES SUR UN MODÈLE DE BIFLOT POUR LE PROBLÈME DU VOYAGEUR DE COMMERCE.

## V.0. INTRODUCTION

Après la validation de nos heuristiques sur des problèmes de biflot purs, nous passons ici à un modèle de biflot plus structuré. Le modèle en question a été proposé par Finke et al. [1984], pour la formalisation du problème du voyageur de commerce. Ce chapitre présente donc une nouvelle technique de résolution pour le problème du voyageur de commerce, basée sur une approche de décomposition mixte, via une formulation par les flots. Étant donnés des coûts sur les arcs d'un graphe, le problème du voyageur de commerce consiste à trouver un circuit hamiltonien de coût minimum. Les différentes applications du problème du voyageur de commerce, notamment dans les domaines de l'ordonnancement et du routage, ainsi que les techniques de résolution associées, sont décrites dans de nombreux articles et ouvrages de base. Parmi ceux là, on peut citer, ceux de Bellmore et Nemhauser [1968], Burkard [1979], Christofides [1979], Domschke [1982] et Lawler et al. [1985]. Le problème du voyageur de commerce est NP-complet. Par conséquent, la plupart des techniques de résolution exactes consistent à procéder par séparation et évaluation. Les performances de ces méthodes dépendent essentiellement de la qualité des bornes inférieures utilisées. Ces bornes proviennent généralement des relaxations du problème de voyageur de commerce, en particulier les problèmes d'affectation linéaire (voir Balas et Christofides [1981], Bellmore et Malone [1971], Carpaneto et Toth [1980], Smith, Srinivasan et Thompson [1977]), de 1-arborescences (voir Bazaraa et Goode

[1977], Hansen et Krarup [1974], Held et Karp [1970, 1971], Smith et Thompson [1977]), de n-plus courtes chaines, et les problèmes de couplage pour le cas symétrique.

Nous aborderons ce chapitre par la présentation du modèle de biflot élaboré par Finke et al. [1984] pour formaliser le problème de voyageur de commerce. Dans ce contexte, nous détaillerons les deux types de relaxation qui en ont été faites. En fait, la première relaxation a été proposée par les auteurs du modèle et a donné lieu à une résolution par la programmation linéaire. Nous proposons donc une deuxième relaxation (Rezig et Finke [1995]) qui permet de déboucher sur un autre type d'approche, en l'occurrence, une approche de résolution par les flots, mettant en œuvre notre logiciel de biflot. Nous verrons, à travers les résultats numériques obtenus, que les caractéristiques du graphe de support du biflot peuvent être raisonnablement exploitées pour obtenir une solution au problème du voyageur de commerce. De fait, nous nous sommes concentrés, par la suite, sur la mise en évidence d'un circuit hamiltonien, à partir du graphe partiel engendré par la solution du biflot ainsi relaxé. Pour ce faire, nous avons opté pour une méthode exacte permettant de détecter un ou plusieurs circuits hamiltoniens dans un graphe, méthode proposée par Martello [1983]. Très vite on s'aperçoit que la plupart des graphes de support du flot ne contiennent quasiment pas de circuits hamiltoniens, mais plutôt une structure de chemins et de circuits. C'est pourquoi nous proposons une méthode heuristique permettant de construire un premier circuit à partir du graphe de support du flot, lequel circuit sera ensuite élargi par des techniques d'insertion jusqu'à l'obtention d'un circuit hamiltonien. Il est clair que le but recherché, ici, n'est pas de rivaliser avec la technologie de résolution actuelle du problème de voyageur de commerce, mais essentiellement de prouver que les techniques d'insertion classiques peuvent tirer profit de la structure du graphe de support du flot. Une évaluation de cette stratégie d'optimisation sera réalisée, par rapport aux techniques constructives classiques, sur un échantillon de problèmes test générés aléatoirement, à la fois pour le cas symétrique et le cas asymétrique.

## V.1. MODÈLE DE BIFLOT POUR LE PROBLÈME DU VOYAGEUR DE COMMERCE

Avant d'aborder à proprement parler le modèle de biflot associé au problème du voyageur de commerce, nous donnons une première formulation du problème à travers le problème d'affectation linéaire en vue d'établir, entre autres, une comparaison entre les deux formulations (il existe d'autres formulations possibles, voir Padberg et Sung [1991]). Nous donnerons ensuite la formulation proposée par Finke et al. [1984] sous forme d'un

problème de biflot spécifique avec des contraintes d'intégralité. Ce sont ces contraintes d'intégralité qui rendent ce problème de biflot difficile, restituant ainsi la complexité du problème de voyageur de commerce dont il est la modélisation. Par conséquent, la résolution de ce problème de biflot se fera à partir d'une relaxation de la formulation exacte. Une première relaxation a été proposée par les auteurs du modèle dans le cadre de la programmation linéaire. En particulier, il a été établi que cette relaxation donnait lieu à une meilleure borne inférieure que celle associée au problème d'affectation linéaire. Par ailleurs, il a été prouvé qu'on pouvait associer à toute solution du biflot ainsi relaxé, une solution au problème d'affectation linéaire qui satisfaisait une version relaxée des contraintes d'élimination des sous-tours. La deuxième relaxation que nous proposons (Rezig et Finke [1995]) permet d'expérimenter une autre approche que la programmtion linéaire, en l'occurrence notre approche par les flots basée sur un principe de décomposition mixte (cf. chapitre IV).

### V.1.1. FORMULATION VIA L'AFFECTATION LINÉAIRE

On se donne un graphe orienté $G=(V,E)$ avec des coûts $c_{ij}$ associés aux arcs $(i,j)\in E$. Avec $n=|V|$, les sommets étant numérotés $1,2,...,n$; et $m=|E|$. Le problème d'affectation linéaire peut être défini de la manière suivante (notez que toutes les sommes se font sur les arcs existants dans $G$)

$$\text{Minimiser } z_A=\sum_i\sum_j c_{ij}x_{ij} \tag{V.1a}$$

sous les contraintes

$$\sum_j x_{ij}=1 \quad \text{et} \quad \sum_j x_{ji}=1 \qquad \forall\, i\in V, \tag{V.1b}$$

$$x_{ij}\geq 0 \qquad \forall\,(i,j)\in E. \tag{V.1c}$$

Pour formuler le problème du voyageur de commerce, il faut imposer en plus la condition d'intégralité suivante:

$$x_{ij}\in\{0,1\} \qquad \forall\,(i,j)\in E, \tag{V.1d}$$

ainsi que les $2^n-2(n+1)$ contraintes d'élimination des sous tours

$$x(W,V-W)=\sum_{i\in W}\sum_{j\in V-W}x_{ij}\geq 1, \tag{V.1e}$$

ou alors de manière équivalente,

$$x(W) = \sum_{i \in W} \sum_{j \in W} x_{ij} \leq |W| - 1, \qquad \text{(V.1 e')}$$

et ce, pour tous les sous ensembles $W \subseteq V$ de cardinalité $2 \leq |W| \leq n-2$.

### V.1.2. FORMULATION PAR UN MODÈLE DE BIFLOT

Cette approche par le biflot (Finke et al. [1984]) peut être considérée comme une tentative de combiner les relaxations associées aux problèmes d'affectation et de 1-arborescence. On peut également noter un rapport avec l'approche du lagrangien réduit de Balas et Christofides [1981]. Choisissons arbitrairement un sommet $s \in V$. Pour chaque $i \in V$, les montants $p_i$ d'un flot de produit $P$ et les montants $q_i$ d'un second flot de produit $Q$ sont définis comme ci dessous:

$$p_i = \begin{cases} N & \text{pour } i = s \\ -1 & \text{sinon} \end{cases} \quad \text{et} \quad q_i = \begin{cases} -N & \text{pour } i = s \\ 1 & \text{sinon} \end{cases} \qquad \text{(V.2)}$$

avec $N = n-1$. Par convention, les nombres positifs sont associés aux points ravitaillement, les nombres négatifs aux points demande. Supposons que notre voyageur de commerce traverse un circuit hamiltonien en démarrant du sommet $s$. Il pourra transporter avec lui les $N$ unités de produit $P$, disponibles en $s$. Au prochain sommet du circuit, il abandonnera une unité de $P$ et récupérera une unité de $Q$. Par conséquent, il transportera encore un flot combiné de $N$ unités sur le prochain arc (i.e. $(N-1)$ unités de $P$ et 1 unité de $Q$). L'échange d'une unité de chaque produit se poursuit jusqu'à ce que $N$ unités arrivent en $s$. Considérons alors le programme d'optimisation suivant:

$$\text{Minimiser } z_{VC} = \frac{1}{N} \sum c_{ij} (x_{ij}^P + x_{ij}^Q) \qquad \text{(V.3a)}$$

sous les contraintes

$$\sum_j x_{ij}^P - \sum_j x_{ji}^P = p_i \quad \forall\, i \in V, \qquad \text{(V.3b)}$$

$$\sum_j x_{ij}^Q - \sum_j x_{ji}^Q = q_i \quad \forall\, i \in V, \qquad \text{(V.3c)}$$

$$\sum_j (x_{ij}^P + x_{ij}^Q) = N \quad \forall\, i \in V, \qquad \text{(V.3d)}$$

$$x_{ij}^P + x_{ij}^Q \in \{0, N\} \quad \forall\, (i,j) \in E, \qquad \text{(V.3e)}$$

$$x_{ij}^P \geq 0,\ x_{ij}^Q \geq 0 \quad \forall\, (i,j) \in E. \qquad \text{(V.3f)}$$

Les contraintes (V.3b), (V.3c) et (V.3f) définissent des flots réalisables $x^P$, $x^Q$, respectivement pour les produits $P$ et $Q$. Les contraintes (V.3d) et (V.3e) garantissent qu'il existe exactement un arc sortant de chaque sommet et transportant un flot combiné égal à $N$ unités. Comme les équations (V.3b) et (V.3c) ont pour somme zéro, il existe aussi exactement un arc entrant en chaque sommet avec un flot total combiné de $N$ unités. De plus, il existe obligatoirement un chemin joignant $s$ à tout autre sommet $j$, et un chemin partant de $j$ et arrivant sur $s$, suite à la configuration des points demande et ravitaillement. Finalement les contraintes (V.3b) à (V.3f) traduisent la représentation mathématique d'un circuit hamiltonien de coût $z_{VC}$, et nous sommes bien en présence d'un problème de voyageur de commerce.

Si on introduit des variables 0-1, on peut réécrire le problème (V.3) sous la forme standard d'un programme en nombres entiers:

$$\text{Minimiser } z_{VC} = \sum c_{ij} x_{ij} \qquad \text{(V.3 a')}$$

sous les contraintes

$$\text{(V.3b), (V.3c), (V.3d), (V.3f) et}$$

$$x_{ij}^P + x_{ij}^Q = N x_{ij} \quad \text{avec } x_{ij} \in \{0,1\}. \qquad \text{(V.3 e')}$$

### V.1.2.1. Relaxation par programmation linéaire (PL)

Nous pouvons obtenir une borne inférieure au problème du voyageur de commerce en remplaçant la condition d'intégralité (V.3e) par:

$$x_{ij}^P + x_{ij}^Q \leq N \qquad \text{(V.3 e'')}$$

Cette contrainte devient alors redondante suite à la présence de la contrainte (V.3d). Par conséquent, on est en présence de la relaxation PL suivante:

$$\text{Minimiser } z_{PQ} = \frac{1}{N} \sum c_{ij} (x_{ij}^P + x_{ij}^Q) \qquad \text{(V.3 a'')}$$

sous les contraintes

$$\text{(V.3b), (V.3c), (V.3d), (V.3f).}$$

La comparaison des bornes inférieures rattachées à ce biflot particulier, $\min z_{PQ}$, et au problème d'affectation, $\min z_A$, met en évidence les résultats suivants (Finke et al. [1984]):

***Théorème V.1.*** *Les deux bornes inférieures associées au problème voyageur de commerce satisfont l'inégalité* $\min z_{PQ} \geq \min z_A$. *Les deux bornes coïncident si les deux produits sont confondus,* $P = Q$.

*Preuve* Considérons le flot combiné par unité:

$$x_{ij} = (x_{ij}^P + x_{ij}^Q)/N \tag{V.4}$$

La contrainte (V.3d) est équivalente à $\sum_j x_{ij} = 1$ et les contraintes (V.3b), (V.3c) impliquent que $\sum_j x_{ji} = 1$. Il s'en suit que $x$ est une solution réalisable du problème d'affectation (V.1a) à (V.1c), d'où $\min z_{PQ} \geq \min z_A$.

Supposons maintenant que nous ayons des flots de produits $P$ et $Q$ identiques. Soit $x$ une solution entière du problème d'affectation. Les arcs associés aux variables $x_{ij} = 1$ forment une collection de circuits. Comme $P$ et $Q$ sont interchangeables, les montants du flot associés à chaque sommet sont nuls. Par conséquent il est facile de trouver des nombres $x_{ij}^P$ et $x_{ij}^Q$ satisfaisant $x_{ij}^P + x_{ij}^Q = N$ lorsque $x_{ij} = 1$ et qui remplissent les conditions (V.3b), (V.3c) et (V.3f). Il s'en suit que $\min z_{PQ} = \min z_A$ pour des flots $P$ et $Q$ identiques.

Considérons une solution réalisable $x^P$ et $x^Q$ de la relaxation PL, et définissons le graphe de support du flot associé $G^X = (V, E^X)$, où $E^X = \{(i,j) \in E / x_{ij}^P + x_{ij}^Q > 0\}$.

***Théorème V.2.*** *Le graphe* $G^X = (V, E^X)$ *est fortement connexe.*

*Preuve* Soient $i, j \in V$. Le sommet $i \neq s$ est un point de ravitaillement avec un montant de une unité, pour le produit $Q$. Comme le sommet $s$ est l'unique point demande pour $Q$, il existe obligatoirement un chemin dans $G^X$ entre $i$ et $s$. De façon identique, le sommet $j \neq s$ présente une demande égale à une unité de produit $P$. Le sommet $s$ étant l'unique point ravitaillement pour le produit $P$, il existe aussi un chemin dans $G^X$ de $s$ à $j$. Par conséquent, on obtient un chemin de $i$ à $j$ pour toutes les paires de sommets.

***Théorème V.3.*** *Soient* $x^P$ et $x^Q$ *une solution réalisable du problème de biflot relaxé. Alors la solution associée au problème d'affectation* $x = (x^P + x^Q)/N$, *satisfait la version relaxée des contraintes d'élimination des sous-tours suivante*

$$x(W, V-W) \geq \min \{|W|, |V-W|\}/N \tag{V.5}$$

$$x(W) \leq |W| - \min \{|W|, |V-W|\}/N \tag{V.6}$$

pour tout $W \subseteq V$ avec $2 \leq |W| \leq n-2$.

*Preuve* Soit la partition $(W, V-W)$, et supposons que le sommet particulier $s \in W$. L'ensemble $V-W$ requiert $|V-W|$ unités de produit $P$. Par conséquent $|V-W|$ unités sont envoyées de $W$ à $V-W$, i.e. $x(W, V-W) \geq |V-W|/N$. De façon analogue, si $s \in V-W$, un montant ravitaillement total d'au moins $|W|$ unités de produit $Q$ doit être évacué de $W$. Par conséquent $x(W, V-W) \geq |W|/N$, d'où la condition (V.5). L'inégalité (V.6) est une conséquence directe de (V.5).

Une solution optimale (entière) du problème d'affectation correspond à une collection de circuits ou sous-tours. Une propriété caractéristique de cette configuration, incluant le cas d'un unique circuit (circuit hamiltonien), est la symétrie:

$$x(W, V-W) = x(V-W, W) \qquad \forall\, W \subseteq V. \tag{V.7}$$

***Théorème V.4.*** *La solution* $x = (x^P + x^Q)/N$ *associée au problème d'affectation présente la propriété de symétrie.*

*Preuve* Le théorème est vrai pour les cas triviaux où $|W| \leq 1$ et $|W| \geq n-1$. Pour les autres cas, on peut supposer que $s \in V-W$. Il y a alors exactement $|W|$ unités du produit $P$, de plus, arrivant à $W$, qu'il n'y en a quittant $W$. De la même manière, il y a un excès de $|W|$ unités de $Q$ quittant $W$. Par conséquent, nous avons la même quantité de flot totale circulant dans les deux sens.

Tous ces théorèmes tendent à mettre en évidence une structure très riche du graphe $G^x$ de support du flot. Cette structure définit en (V.4) une solution (fractionnaire) au problème d'affectation qui contient également des 1-arborescences. Le graphe $G^x$ peut éventuellement contenir un circuit hamiltonien qui n'est pas nécessairement optimal. Il existe une certaine similitude entre ce graphe et le graphe $G_0$ construit par Balas et Christofides [1981]. Ces auteurs démarrent avec une solution optimale du problème

d'affectation et génèrent au fur et à mesure un graphe fortement connexe $G_0$ en incorporant des contraintes d'élimination de sous-tours (V.1e) ou (V.1e′).

Sur un plan pratique, la relaxation de ce modèle de biflot a été résolue par l'intermédiaire d'un logiciel standard de programmation linéaire. Les résultats sont obtenus sur la base d'un ensemble de problèmes test générés aléatoirement, et dont la taille varie de 10 à 50 villes (voir Finke et al. [1984]). Ces résultats se rapportent aux améliorations de la borne inférieure (par rapport à la relaxation de l'affectation linéaire), ainsi qu'aux tailles moyennes de $G^X$ et de $X$ (ensemble de variables non nulles). Dans le cas asymétrique, les valeurs moyennes sont comme suit: $|G^X| = 1,35n$ et $|X| = 2,16n$. Précisons qu'en théorie la taille du graphe support du flot $G^X$ est telle que $n \leq |G^X| \leq 3n-2$, et le nombre $|X|$ de variables non nulles est tel que $2n-2 \leq |X| \leq 3n-2$. Les configurations associées au plus haut degré de dégénérescence, soit $|G^X| = n$ et $|X| = 2n-2$, caractérisent précisément des circuits hamiltoniens. Dans le cas symétrique, par contre, les résultats sont moins performants puisque $|G^X| = 2,14n$ et $|X| = 2,63n$. Il ressort de cette analyse que l'on peut espérer trouver un "bon" circuit hamiltonien à partir du graphe $G^X$, et que par conséquent, cette approche par le biflot peut fournir une bonne solution heuristique de départ pour le problème du voyageur de commerce, en particulier dans le cas asymétrique.

### V.1.2.2. Relaxation par le flot

De la même manière que précédemment, on relaxe les contraintes d'intégralité (V.3e), sous la forme (V.3e″), soit $x_{ij}^P + x_{ij}^Q \leq N$. Grâce à la condition (V.3d) ces contraintes deviennent redondantes, mais cette condition reste elle-même difficile à prendre en compte dans le cadre d'une approche par les flots. C'est pourquoi nous relaxons les contraintes (V.3d) qui prennent la forme:

$$\sum_j x_{ij}^P + x_{ij}^Q \leq N \qquad \forall\, i \in V. \tag{V.3d′}$$

Les contraintes (V.3e″) restent toujours redondantes, mais nous pouvons les considérer comme des contraintes de capacité classiques pour le problème de biflot ainsi relaxé. De même, les contraintes (V.3d′) peuvent être également assimilées à des contraintes de capacité, moyennant certaines transformations techniques du graphe. Dans le cas précis, cela reviendrait à doubler le nombre de sommets de $G$ et à rajouter $n$ arcs de capacité $N = n-1$, comme l'indique la figure ci dessous:

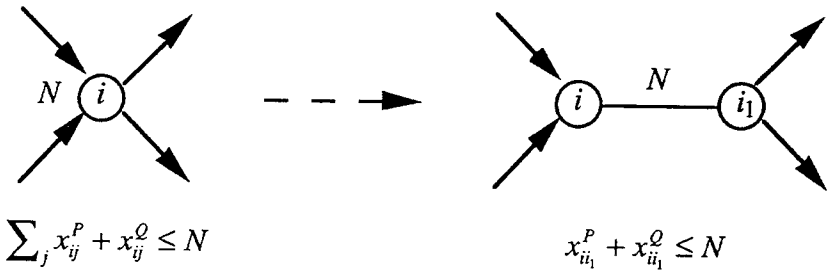



**Figure V.1** Transformation de la contrainte de flot sur les sommets en contrainte de capacité sur les arcs.

La contrainte de couplage est ainsi établie entre les deux flots, via la capacité $N$ admise sur tous les arcs du réseau ainsi augmenté. Finalement nous sommes bien en présence d'un problème de biflot ayant pour objectif la minimisation du coût (V.3a), avec des contraintes de conservation du flot (V.3b) et (V.3c), des contraintes implicites et explicites de capacité (V.3 d′), (V.3 e″), en plus des contraintes de positivité (V.3f) traduisant la version orientée du problème.

L'inconvénient de ce schéma de relaxation réside dans le fait que tous les résultats théoriques présentés précédemment ne sont plus valables. Entre autres, le graphe $G^X$ n'est plus fortement connexe mais seulement connexe, de plus la borne inférieure associée à cette nouvelle relaxation est moins bonne que celle associée au problème d'affectation. Néanmoins le but recherché est de déboucher sur un graphe de support du flot $G^X$ présentant une bonne configuration. La priorité étant donnée ici à l'utilisation potentielle du graphe $G^X$ pour la mise à jour d'un circuit hamiltonien.

Des résultats statistiques sont obtenus dans les cas asymétrique et symétrique sur des problèmes test générés aléatoirement (voir Figures V.2 et V.3 respectivement).

| Villes $n$ | Graphe du flot $\lvert G^X \rvert / n$ | Variables $\lvert X \rvert / n$ |
|---|---|---|
| 10 | 1,36 | 1,80 |
| 20 | 1,30 | 1,92 |
| 30 | 1,39 | 1,96 |
| 40 | 1,39 | 1,95 |
| 50 | 1,42 | 1,97 |

**Figure V.2** Approche par le biflot pour le cas asymétrique du problème de voyageur de commerce.
(moyennes sur 10 problèmes aléatoires, coûts $c_{ij} \in [0,1000]$)

| Villes $n$ | Graphe du flot $\lvert G^X \rvert / n$ | Variables $\lvert X \rvert / n$ |
|---|---|---|
| 10 | 1,65 | 1,88 |
| 20 | 1,85 | 1,91 |
| 30 | 1,84 | 1,94 |
| 40 | 1,9 | 1,95 |
| 50 | 1,83 | 1,97 |

**Figure V.3** Approche par le biflot pour le cas symétrique du problème de voyageur de commerce.
(moyennes sur 10 problèmes aléatoires, coûts $c_{ij} \in [0,1000]$)

Précisons que nous générons ici des graphes complets. Toutefois, la modélisation par le biflot permet d'envisager à la fois le cas des graphes complets et le cas des graphes quelconques (moins denses).

Il ressort de ces expériences les résultats suivants (en moyenne): pour le cas asymétrique, $|G^X| = 1,37n$ et $|X| = 1,92n$; pour le cas symétrique, $|G^X| = 1,81n$ et $|X| = 1,93n$. Ce qui montre que, malgré une relaxation plus forte, les résultats ne sont que moyennement altérés (avec toujours une supériorité pour le cas asymétrique). Partant de là, on peut raisonnablement envisager l'exploitation de cette structure pour détecter, ou alors construire, un circuit hamiltonien.

## V.2. RECHERCHE D'UN CIRCUIT HAMILTONIEN

Cette partie sera consacrée à la mise en évidence d'un circuit hamiltonien dans un graphe orienté. Le graphe en question étant le graphe de support du biflot relaxé, $G^X$. Il apparaît en effet que ce graphe reste de taille raisonnable (cf. résultats de la partie V.2.1.2), et s'apprête donc à l'application d'une méthode énumérative exacte pour la recherche d'un circuit hamiltonien. Nous ferons appel à l'algorithme de Martello [1983] pour détecter un ou plusieurs circuits hamiltoniens dans le graphe $G^X$. Malheureusement l'occurrence de tels circuits est très rare, la structure de $G^X$ étant plutôt dominée par la présence de chemins et de sous-tours. Pour pallier à cet inconvénient, nous proposons une méthode heuristique permettant d'exploiter la structure de ce graphe et de construire, par des techniques d'insertion de sommets, un circuit hamiltonien.

### V.2.1. ALGORITHME DE MARTELLO

Reprenons le graphe orienté $G = (V, E)$ défini en (V.1.1) avec $V = \{1, 2, ..., n\}$ l'ensemble des $n$ sommets, et $E$ l'ensemble des $m$ arcs $(i, j)$ appartenant à $G$. Un circuit hamiltonien est une permutation $(s_i)$ des sommets telle que $(s_i, s_{i+1}) \in E$ pour $i = 1, ..., n$ et $(s_n, s_1) \in E$. On admet, sans perte de généralité, que $(i, i) \notin E$, $\forall i \in V$.

Le problème qui consiste à trouver un ou plusieurs circuits hamiltoniens dans un graphe donné, ou réciproquement à déterminer si ce graphe n'en contient pas, est NP-complet. Par conséquent, les méthodes de recherche de tels circuits sont soit des méthodes énumératives, soit des méthodes heuristiques. L'algorithme que nous allons présenter (Martello [1983]) permet de résoudre, précisément, le problème *(P)* suivant (soit de manière exacte, soit de manière heuristique)

*(P)* étant donné un graphe orienté $G=(V,E)$ et une valeur $h$, $(1 \le h \le \infty)$, trouver $h$ circuits hamiltoniens distincts de $G$, ou alors si $G$ ne possède pas $h$ circuits hamiltoniens, trouver tous les circuits hamiltoniens de $G$.

La méthode la plus efficace pour résoudre *(P)* est celle proposée par Christofides [1975] et Selby [1970], basée sur un schéma d'énumération de Roberts et Flores [1966]. L'algorithme de Martello est une amélioration de cette méthode essentiellement par l'introduction de techniques heuristiques pour la détermination des branches à suivre dans l'arbre de décision. Cet algorithme est basé sur une stratégie de branchement en profondeur d'abord. Cette stratégie permet d'étendre un chemin élémentaire $S=\{(s_1,s_2),\ (s_2,s_3),\ \ldots,\ (s_{k-1},s_k)\}$ (où $k$ dénote le niveau courant de l'arbre de branchement pour enregistrer l'information relative à chaque nœud actif de l'arbre) jusqu'à ce que l'extension ne soit plus possible ou que le chemin contienne $(n-1)$ arcs (avec éventuellement la possibilité de donner un circuit hamiltonien si $(s_k,s_1)\in E$). Dans les deux cas, l'algorithme procède à un *retour arrière* consistant à effacer le dernier arc $(s_{k-1},s_k)$ de $S$. Cette opération est suivie par une opération de *séparation* qui consiste à rajouter à $S$ un nouvel arc émanant de $s_{k-1}$. A chaque *séparation*, une phase de *réduction* élimine de $E$ tous les arcs inutiles (arcs qui ne seront jamais dans $S$), et insère des arcs *impliqués* dans un ensemble $I$ (arcs qui seront tôt ou tard inclus dans $S$). L'algorithme sera décrit dans sa version exacte. Les notations suivantes seront utilisées:

$R_i=\{j/(i,j)\in E\}$;
$C_j=\{i/(i,j)\in E\}$.

*Algorithme de recherche d'un ou plusieurs circuits hamiltoniens dans un graphe orienté.*

**Pas 0:** on procède aux initialisations, $I=\varnothing$, $k=1$, $s_1=r$. Le sommet $r$ est choisi tel que $C_r=\max_{i\in V}\{|C_i|\}$ (s'il y a plusieurs candidats, choisir $r$ tel que $|R_r|$ soit minimum); cette stratégie permet d 'augmenter la probabilité de trouver un circuit hamiltonien à travers l'arc $(s_n,s_1\equiv r)$ lorsque $S$ inclut $(n-1)$ arcs.

**Pas 1:** on recherche les arcs impliqués, en examinant les nœuds ayant un unique arc rentrant ou sortant. Soit $(i,j)$ un arc de $E$ tel que $R_i=\{j\}$ ou $C_j=\{i\}$. Pour tout arc vérifiant cela, le plus long chemin incluant $(i,j)$ et les précédents arcs impliqués est construit; soit $\bar{I}$ ce chemin. Si $|\bar{I}|<n-1$, $(i,j)$ est inséré dans $I$, tous les arcs émanant de $i$ ou aboutissant à $j$, ainsi que l'arc reliant les premier et dernier sommets de $\bar{I}$, sont

supprimés de $E$, (si cette opération met à jour un sommet $q$ tel que $R_q = \varnothing$ ou $C_q = \varnothing$, un retour arrière (Pas 5) est effectué). Si $|\bar{I}| = n-1$, soit un circuit hamiltonien est trouvé (si un arc entre les premier et dernier sommets de $\bar{I}$ existe), soit un retour arrière doit être effectué. Le Pas 1 est répété jusqu'à ce qu'aucun arc ne puisse plus être rajouté à $I$.

**Pas 2:** on vérifie si un arc de $I$ émane de $s_k$ (soit $(s_k, \bar{s}) \in I$ ). Si $\bar{s} \equiv r$ et $k < n$ un retour arrière est effectué; sinon $S$ est étendu en posant $k = k+1$, $s_k = \bar{s}$ (si $k = n$, un circuit hamiltonien est trouvé ou alors un retour arrière doit être effectué). Le Pas 2 est itéré jusqu'à ce qu'aucun autre arc impliqué ne puisse plus être rajouté à $S$.

**Pas 3:** on procède à la phase de séparation. Le prochain arc $(s_k, \bar{s})$ à rajouter à $S$ est choisi parmi les arcs non encore testés, tel que $min\{|R_{\bar{s}}|, |C_{\bar{s}}|\}$ soit minimum (en cas d'égalité, on choisit $(s_k, \bar{s})$ tel que $|R_{\bar{s}}| + |C_{\bar{s}}|$ soit minimum); de cette manière les sommets les plus "critiques" sont insérés en priorité dans $S$ (augmentant ainsi la probabilité d'obtenir un circuit hamiltonien). S'il n'existe aucun arc réalisable, un retour arrière est effectué. Autrement, tous les arcs émanant de $s_k$ et arrivant sur $\bar{s}$, de même que l'arc $(\bar{s}, r)$ sont éliminés de $E$, $S$ est étendu en posant $k = k+1$, $s_k = \bar{s}$. Si cette opération a pour effet de mettre à jour un sommet $q$, tel que $R_q = \varnothing$ ou $C_q = \varnothing$, un retour arrière est effectué, sinon on revient au Pas 1.

**Pas 4:** on vérifie que le nombre de circuits hamiltoniens requis est atteint, chaque fois qu'un circuit hamiltonien est trouvé. Si oui, on arrête, sinon on procède à un retour arrière.

**Pas 5:** on procède à l'opération retour arrière. L'arc $(s_{k-1}, s_k)$ est effacé de $S$. Si $k = 1$, toutes les possibilités ont été examinées, l'algorithme s'arrête. Si $(s_{k-1}, s_k) \in I$ (autrement dit, si l'arc a été rajouté à $S$ au Pas 2), on pose $k = k-1$ et on procède de nouveau à un retour arrière. Autrement, (l'arc a été rajouté à $S$ au Pas 3) tous les arcs effacés de $E$ au niveau $k$ sont réinsérés dans $E$, tous les arcs insérés dans $I$ sont retirés de $I$, on pose $k = k-1$ et une nouvelle phase de séparation est mise en œuvre (Pas 3).

La version heuristique de l'algorithme est obtenue en imposant une borne supérieure sur le nombre de retours arrière effectué. L'algorithme a été implémenté en Fortran, et testé sur des exemples dont la taille peut aller jusqu'à 250 sommets. Les détails d'implémentation peuvent être trouvés dans Martello [1981].

## V.2.2. MÉTHODE HEURISTIQUE

La méthode que nous proposons permet de pallier au côté restrictif de l'algorithme de Martello, au cas où le graphe $G^X$ n'est pas hamiltonien. Cette méthode est une méthode constructive qui procède en deux étapes. La première étape consiste, à partir du graphe $G^X$ de support du flot, à construire selon un principe heuristique un circuit initial. Si ce circuit contient tous les sommets du graphe, on dispose d'un circuit hamiltonien. Autrement, les sommets manquants sont annexés à ce circuit en utilisant des techniques d'insertion classiques. On admettra ici que le graphe $G$ est complet.

*Algorithme de construction d'un circuit hamiltonien*

*Étape 1: Construction d'un circuit initial* $C_0$ *à partir de* $G^X$

**Pas 1:** Construire, en démarrant du sommet 1, un chemin élémentaire maximal, en empruntant les arcs véhiculant le plus grand flot (combiné), et aller au Pas 2.

**Pas 2:** On rajoute l'arc joignant le sommet final du chemin, au sommet initial (sommet 1).Si le circuit construit, $C_0$, contient $n$ arcs, l'algorithme s'arrête. Sinon aller à l'étape 2.

*Étape 2: Insertion des sommets manquants*

**Pas 1:** Prendre un sommet manquant, et élargir séquentiellement le circuit $C_0$ en insérant ce sommet au meilleur emplacement possible (en minimisant le coût du circuit courant).

La complexité de cette heuristique correspond à la compléxité de la technique d'insertion utilisée. Nous considérerons, ici, deux techniques d'insertion, en l'occurrence l'insertion séquentielle selon un ordre quelconque en $O(n^2)$, et l'insertion au moindre coût en $O(n^2 \log n)$. Cette dernière technique consiste à examiner tous les sommets manquants à travers leurs différents emplacements, et à insérer au fur et à mesure celui qui induit le plus petit coût. Elle est certes plus coûteuse que les autres techniques, mais pas forcément meilleure (voir Lawler et al. [1985]).

Pour illuster cet algorithme de construction d'un circuit hamiltonien, nous présentons deux exemples d'application tirés de la littérature classique du problème de voyageur de commerce.

***Exemple 1***

Considérons la matrice de distances du problème de voyageur de commerce suivant (Murty [1976]).

| | | | | | |
|---|---|---|---|---|---|
| ∞ | 27 | 43 | 16 | 30 | 26 |
| 7 | ∞ | 16 | 1 | 30 | 25 |
| 20 | 13 | ∞ | 35 | 5 | 0 |
| 21 | 16 | 25 | ∞ | 18 | 18 |
| 12 | 46 | 27 | 48 | ∞ | 5 |
| 23 | 5 | 5 | 9 | 5 | ∞ |

La solution optimale associée au problème d'affectation met en évidence la matrice de coûts réduits suivante:

| | | | | | |
|---|---|---|---|---|---|
| ∞ | 7 | 23 | 0 | 10 | 11 |
| 0 | ∞ | 11 | 0 | 25 | 25 |
| 13 | 8 | ∞ | 34 | 0 | 0 |
| 3 | 0 | 9 | ∞ | 2 | 7 |
| 0 | 36 | 17 | 42 | ∞ | 0 |
| 16 | 0 | 0 | 8 | 0 | ∞ |

La résolution du biflot relaxé, par l'approche de décomposition mixte, met à jour le graphe $G^X$ suivant:

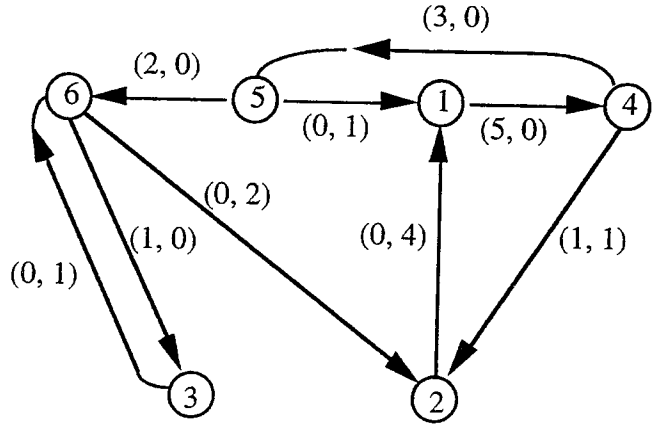



**Figure V.4** Solution du biflot.

L'algorithme de Martello met en évidence, comme on peut le constater sur la Figure V.4, l'absence de circuit hamiltonien. Nous passons dès lors à la phase de construction d'un

circuit hamiltonien. L'algorithme que nous proposons donne lieu (suite à l'étape 1) au chemin $\{1-4-5-6-2\}$, et par extension, au circuit $C_0 = (1-4-5-6-2-1)$.

La deuxième étape procède à l'insertion des sommets manquants, soit le sommet 3. Le meilleur coût correspond alors à l'insertion du sommet 3 en troisième position, ce qui donne le circuit hamiltonien $C = (1-4-3-5-6-2-1)$, de coût 63. Ce circuit correspond précisément à la tournée optimale.

***Exemple 2***

Cet exemple met en jeu neuf villes; il est donné par Balas et Christofides [1981]. Le graphe $G^X$ obtenu est le suivant:

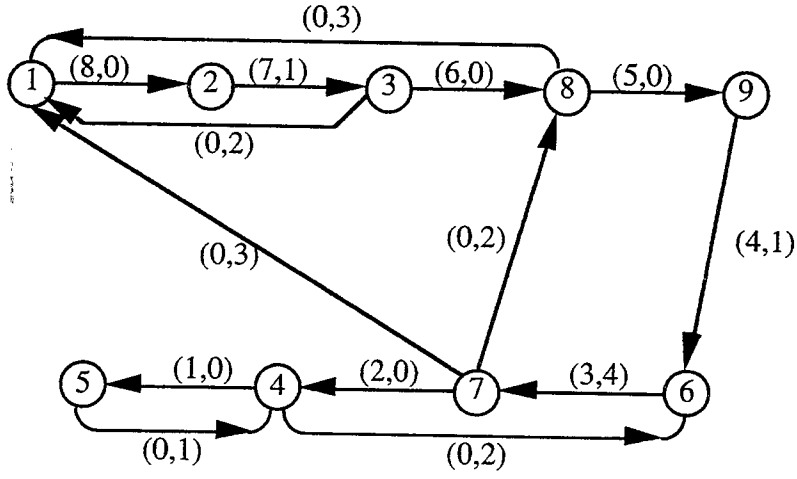



**Figure V.5** Graphe $G^X$.

Le graphe obtenu n'étant pas hamiltonien, on applique notre algorithme de construction d'un circuit hamiltonien. L'algorithme s'arrête à l'issue de l'étape 1, puisqu'on obtient un chemin contenant tous les sommets (9 sommets). De fait, en suivant la règle du flot décroissant, on obtient le chemin $\{1-2-3-8-9-6-7-4-5\}$. Par extension, le circuit associé est le circuit hamiltonien $C = (1-2-3-8-9-6-7-4-5-1)$ qui correspond au circuit optimal.

## V.3. DÉTAILS D'IMPLÉMENTATION ET RÉSULTATS OBTENUS

Toutes les phases d'optimisation précédentes ont été implémentées. Nous les présenterons sous forme d'un schéma d'optimisation global, dont nous détaillerons les différentes étapes, en termes de procédures et de paramètres de calcul. Enfin, nous présenterons une évaluation de ce schéma d'optimisation par le biais de résultats numériques obtenus à partir de problèmes test aléatoires.

### V.3.1. DÉTAILS D'IMPLÉMENTATION

D'une façon globale la résolution du problème du voyageur de commerce prend la forme schématique suivante:

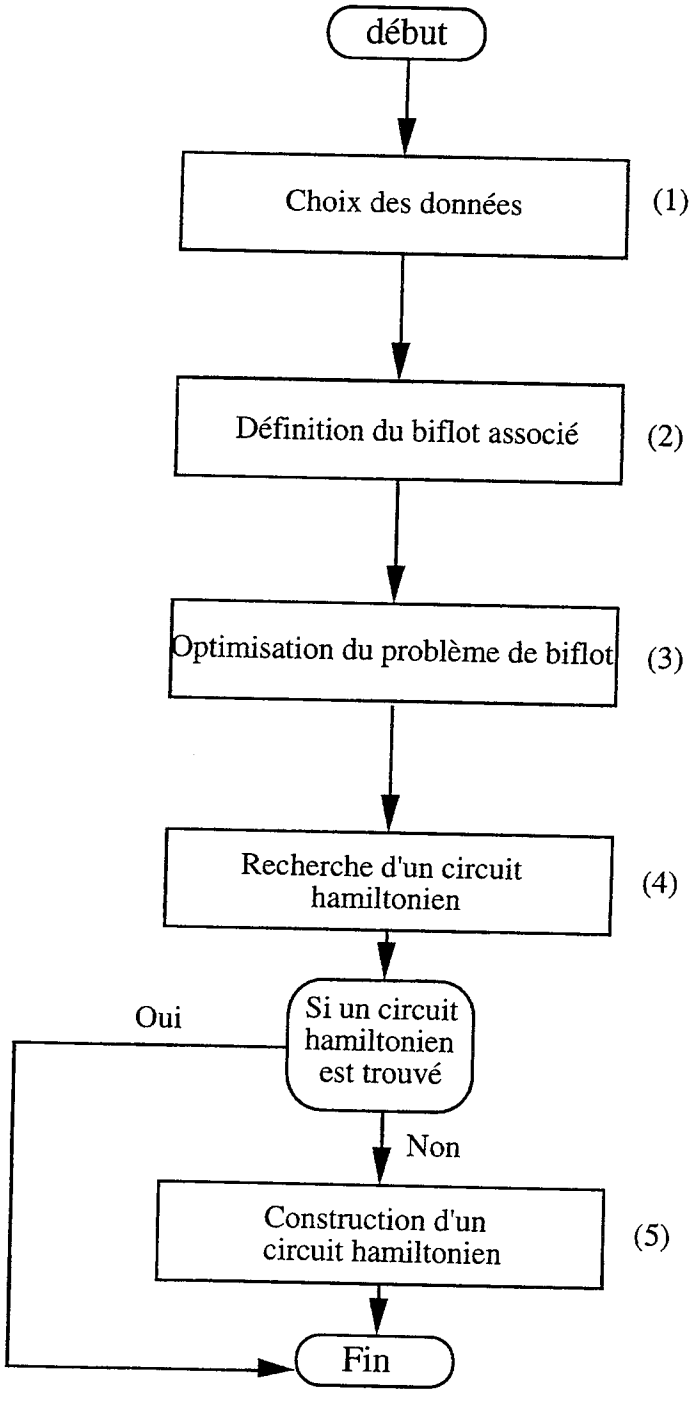



**Figure V.6** Schéma global d'optimisation du problème de voyageur de commerce.

Explicitons les différentes étapes:

- La phase (1) consiste à saisir les données, avec un choix préalable sur le type de données à entrer. L'utilisateur peut opter pour le cas de données réelles (ou rapportées), ou pour le cas d'une génération aléatoire. Dans les deux cas, l'utilisateur doit entrer le nombre de villes N ainsi que les paramètres concernant l'optimisation du biflot: NEXAM, LARGE, INFIN, INFIN2, INFIN3. Ces paramètres ont été explicités dans le chapitre précédent (voir Chapitre IV, § IV.3.1.2). Dans le cas des données réelles, on doit entrer la matrice de distances intervilles représentées par les coûts $c_{ij}$; les coûts $c_{ii}$ sont $\infty$ (cette valeur étant identifiée à INFIN). Dans le cas de la génération aléatoire, l'utilisateur doit entrer un paramètre aléatoire ZS, et une borne supérieure sur les coûts à générer ZC. Les coûts sont alors générés aléatoirement grâce à la fonction RAN. Dans un cas comme dans l'autre, ces données que nous résumons au nombre de villes et aux coûts intervilles, sont stockées dans un fichier *coûts*.

- La phase (2) consiste à traiter les données saisies par l'utilisateur, de façon à pouvoir les traduire en données propres à notre modèle de biflot relaxé, mais également compatibles avec notre logiciel d'optimisation de ce problème. Deux traitements sont envisagés pour tenir compte de cela. Le premier consiste à résoudre le problème d'affectation associé aux données initiales (N, $c_{ij}$). Ceci est réalisé par la procédure LSAP qui permet d'obtenir les coûts réduits qui seront utilisés lors de l'optimisation du biflot. Ces coûts réduits sont stockés dans le fichier *cou.red.* Le deuxième traitement opéré concerne, cette fois, la structure du réseau associé à notre modèle de biflot. Rappelons en effet, qu'une modification du réseau est nécessaire pour tenir compte de la contrainte de flot sur les sommets (V.3.d') (cf. Figure V.1). Par ailleurs il faudra tenir compte des conventions d'optimisation du biflot, en particulier la construction artificielle imposée par notre procédure d'optimisation (arcs et sommet artificiels, voir Chapitre IV, § IV.3.1.2). Ce deuxième traitement est pris en compte dans la procédure VC qui met à jour les listes T, C, K, H, X, X1, préalables à l'application de notre procédure d'optimisation du biflot (pour les définitions, voir Chapitre IV, § IV.3.1.2). Retenons que ces listes sont stockées dans le fichier RESALEADUP.

- La phase (3) consiste à optimiser le problème de biflot défini à l'étape précédente. Cette optimisation se fait par une approche de décomposition mixte via les différentes

heuristiques que nous avons développées au chapitre IV. Il est clair qu'on retrouvera, ici, les mêmes fichiers induits par cette application, de même que les différentes procédures d'optimisation intermédiaires que nous ne détaillerons donc pas à ce niveau. A partir de là, on peut dégager les caractéristiques du graphe de support du flot, en termes de rapports $|G^X|/N$, $|X|/N$. Ceci est concrètement réalisé par la procédure STAT. Par ailleurs, la procédure GX permet de définir les inputs nécessaires à l'application des différents algorithmes intervenant dans les étapes suivantes.

- La phase (4) correspond à la recherche d'un circuit hamiltonien par l'algorithme de Martello. On utilise la procédure HC dont les détails d'implémentation se trouvent dans Martello [1981]. Nous avons appliqué cette procédure dans sa version exacte, avec une recherche de tous les circuits hamiltoniens; on retiendra le circuit hamiltonien de coût minimum. La procédure HC, utilisée une première fois sur le graphe $G^X$, peut être réutilisée sur le même graphe enrichi. Notamment dans le cas symétrique, où l'on peut rajouter à tous les arcs de $G^X$, leurs homologues en sens inverse.

- La phase (5) est enclenchée en cas d'échec de la phase (5), autrement dit si $G^X$ n'est pas hamiltonien. La méthode heuristique que nous avons développée au § V.2.2 est alors mise en jeu, à travers deux procédures essentielles: les procédures CHEMIN et INSERT (INSERT1). Les procédures sont associées, comme leurs noms l'indiquent, aux étapes 1 et 2 de l'algorithme de construction d'un circuit hamiltonien (cf. § V.2.2). Remarquons que INSERT correspond à une insertion séquentielle des sommets, pendant que INSERT1 correspond à une insertion au moindre coût.

### V.3.2. RÉSULTATS OBTENUS

L'évaluation de ce schéma d'optimisation du problème de voyageur de commerce se fera dans le contexte des techniques heuristiques d'insertion classiques, en particulier les techniques d'insertion utilisées en seconde phase de notre méthode heuristique (ie. insertion séquentielle et insertion au moindre coût).

Pour ce faire, nous avons généré un échantillon de problèmes test dont la taille varie de 10 à 50 villes. Les résultats obtenus donnent, d'une part, le pourcentage des sommets inclus dans le circuit initial $C_0$, obtenu par l'algorithme de construction d'un circuit hamiltonien, à l'issue de l'étape 1; d'autre part, on présente la variation relative entre la solution

heuristique obtenue $z_{hr}$, et la solution correspondant aux techniques d'insertion classiques $z_{ins}$. Ces résultats sont représentés dans la Figure (V.7) pour le cas asymétrique.

| Villes $n$ | Nombre de sommets inclus dans $C_0$ (%) | Différence relative* (%) |
|---|---|---|
| 10 | 82 | 18,7 |
| 20 | 52 | 12,3 |
| 30 | 70 | 14,0 |
| 40 | 45 | 14,5 |
| 50 | 38 | 18,6 |

**Figure V.7** Approche par le biflot pour le voyageur de commerce asymétrique.
(moyennes sur 10 problèmes aléatoires, coûts $c_{ij} \in [0,1000]$)
*) $(z_{ins} - z_{hr})/z_{ins}$.

Dans le cas symétrique nous obtenons les résultats suivants:

| Villes $n$ | Nombre de sommets inclus dans $C_0$ (%) | Différence relative* (%) |
|---|---|---|
| 10 | 75 | 17,6 |
| 20 | 53 | 09,6 |
| 30 | 44 | 16,8 |
| 40 | 36 | 13,5 |
| 50 | 35 | 13,2 |

**Figure V.8** Approche par le biflot pour le voyageur de commerce symétrique.
(moyennes sur 10 problèmes aléatoires, coûts $c_{ij} \in [0,1000]$)
*) $(z_{ins} - z_{hr})/z_{ins}$.

Ces résultats mettent en valeur des différences relatives satisfaisantes, en moyenne, 15,6% pour le cas asymétrique, et 14,1% pour le cas symétrique (au pire des cas, cette différence peut atteindre jusqu'à 28%). Par ailleurs, le pourcentage de sommets inclus dans $C_0$ est en moyenne de 57% et 49% pour les cas asymétrique et symétrique, respectivement. Tout ceci nous permet de valider l'utilisation du graphe de support du flot, comme préalable aux techniques d'insertion classiques. Ce qu'il faut mentionner en plus, et qui ne transparait pas dans ces résultats, c'est la fréquence d'obtention d'un circuit hamiltonien par l'algorithme de Martello. En fait, les expérimentations réalisées mettent en valeur une occurrence de ces circuits dans 7% des cas. Toutefois, cette faible fréquence est contrebalancée par la qualité, très satisfaisante, des circuits obtenus. Le cas asymétrique enregistre de meilleures performances que le cas symétrique. En l'occurrence, les circuits optimaux, mis en évidence, sont tous associés au cas asymétrique.

## V.4. CONCLUSION

A travers la relaxation du modèle de biflot pour le problème du voyageur de commerce, les heuristiques, que nous avons développées, trouvent une application tout à fait originale. Cette originalité se précise encore par l'exploitation brute du graphe de support du biflot relaxé, pour détecter ou pour construire un circuit hamiltonien. En effet, ces graphes présentent une taille moyenne tout à fait raisonnable quant à l'utilisation d'une méthode exacte pour la recherche de circuits hamiltoniens. Nous avons utilisé l'algorithme de Martello qui a permis de mettre en évidence des circuits hamiltoniens de très bonne qualité. Malheureusement l'occurrence de tels circuits reste tout à fait épisodique. Pour ne pas en rester là, nous avons décidé de construire un circuit hamiltonien, directement à partir du graphe de support du flot. Nous avons proposé une technique heuristique facile à mettre en œuvre, permettant de construire un circuit de départ, lequel pourra ensuite être élargi par des techniques classiques d'insertion de sommets. Les résultats obtenus permettent de confirmer qu'il est toujours plus rentable, dans le contexte des techniques constructives classiques, d'exploiter l'information contenue dans le graphe de support du flot.

# CONCLUSION

Le travail présenté dans cette thèse a pour thème général les problèmes de multiflots. Ces problèmes, qui sont une généralisation des problèmes de flot simple, n'héritent cependant pas des propriétés intéressantes de ces derniers, notamment le théorème du flot maximum et de la coupe minimum, ou encore le théorème des valeurs entières. Par contre, sur un plan pratique, nous avons prouvé que les problèmes de multiflots bénéficiaient d'une aussi grande popularité que leurs homologues en flot simple. En effet, nous avons présenté une large panoplie d'applications des problèmes de multiflots où pouvaient se côtoyer des domaines aussi variés que la communication, le management, la planification de production, jusqu'à certaines applications militaires.

Sur le plan des techniques de résolution, nous avons envisagé le cas des multiflots continus et celui, plus compliqué encore, des multiflots entiers. Les deux certitudes que nous avons acquises à l'issue de cette analyse sont, d'une part, la grande difficulté des problèmes de multiflots, et d'autre part, le profond déséquilibre entre le nombre de techniques de résolution consacrées au cas continu, et celles relatives au cas entier. Ainsi dans le cas continu, les problèmes de multiflots bénéficient d'un large éventail de méthodes de résolution, essentiellement représentées par les techniques de décomposition par les prix, les techniques de décomposition par les ressources ainsi que les techniques de partitionnement. Précisons tout de même que certaines de ces techniques de résolution mettent en jeu des critères d'arrêt heuristiques, illustrant encore une fois la difficulté relative des problèmes de multiflots par rapport aux problèmes de flot simple. Dans le cas entier, la situation est beaucoup moins privilégiée, aussi bien sur le plan qualitatif que sur le plan quantitatif. De fait, la complexité des problèmes de multiflots en nombres entiers, les classe parmi les problèmes NP-difficiles. Actuellement le meilleur algorithme d'approximation peut induire, simplement pour le problème de multiflot maximal, une déperdition du flot pouvant atteindre jusqu'à 50%.

Enfin, pour finir sur une note optimiste, nous avons montré qu'il était parfois possible de contourner la difficulté intrinsèque des problèmes de multiflots, en identifiant un réseau

de multiflots à un réseau de flot simple. Il est clair que cette condition d'identification dépend exclusivement de la structure topologique du réseau.

L'essentiel de notre contribution s'illustre par le développement d'une méthode heuristique, particulièrement adaptée à la résolution d'un cas particulier des problèmes de multiflots : le problème du biflot. La méthode que nous avons présentée est destinée à résoudre précisément le problème du biflot entier à coût minimum. Cette méthode est basée sur une approche de décomposition mixte permettant de coupler les effets des méthodes classiques de décomposition par les prix et de décomposition par les ressources. Cette approche de résolution est d'autant plus appropriée qu'elle induit, dans le cas du biflot, un principe d'allocation de ressources tout à fait dichotomique. En d'autres termes, si une partie de la capacité est affectée au premier flot, le deuxième flot dispose de la capacité restante, et réciproquement. Par ailleurs, cette approche de décomposition mixte nous a permis de déboucher sur un schéma de résolution efficace mettant en jeu deux problèmes de flots simples spécifiques, supprimant ainsi tout niveau de coordination.

En contrepartie de tous ces avantages, nous avons hérité des inconvénients récurrents au thème de la décomposition mixte. Il s'agit, en l'occurrence, des notions de choix de partitions et de convergence de la méthode. Nous avons réussi à remédier au problème du choix de la partition en développant des techniques heuristiques innovantes et parfaitement efficaces sur des problèmes de biflot purs. Malheureusement le problème de la convergence reste toujours posé sur un plan théorique, même si empiriquement, nous avons pu mettre en évidence une convergence sous forme d'un cyclage sur un ensemble de solutions.

Sur un plan pratique, nous avons développé un code permettant d'une part, l'élaboration de problèmes de biflot aléatoires, et d'autre part, l'implémentation de toutes les heuristiques proposées. La génération aléatoire comprend la génération automatique de réseaux et de données concernant le problème du biflot.

Pour finir, nous avons envisagé une application particulière de nos heuristiques sur un modèle de biflot en variables discrètes, proposé pour la formalisation du problème de voyageur de commerce. Nous avons résolu une relaxation de ce modèle grâce à notre approche de décomposition mixte pour le problème du biflot. Les résultats obtenus, à la fois pour les cas symétrique et asymétrique, permettent d'illustrer, dans le contexte des techniques d'insertion classiques, qu'il est toujours plus avantageux d'exploiter la structure du graphe de support du flot.

A présent, force est de constater que ce travail n'est qu'un point de départ sur plusieurs plans. Les concepts sous-jacents et les travaux réalisés ouvrent à moyen et à long terme, des perspectives intéressantes d'études théoriques, d'applications à d'autres problèmes et de réflexions méthodologiques. Parmi tous ces aspects, soulignons quelques directions qu'il nous semble devoir privilégier.

- L'utilisation d'une méthode de décomposition mixte pose un problème de convergence. Les conclusions acquises dans ce contexte ne sont qu'empiriques. Il serait fort intéressant d'exploiter ces conclusions, et de soumettre cette notion de convergence à une étude théorique approfondie.

- La méthode de décomposition mixte que nous avons développée est destinée à résoudre un cas particulier des problèmes de multiflots, à savoir le problème du biflot. Une adaptation de cette méthode aux autres problèmes de multiflots serait très enrichissante, aussi bien pour lui donner un caractère plus général, que pour en élargir la classe des applications.

- Enfin, nous pensons qu'il serait judicieux de reproduire l'expérience, menée dans le cadre du problème du voyageur de commerce, sur une relaxation plus avantageuse du modèle de biflot. En l'occurrence, la relaxation proposée par les auteurs du modèle qui débouche sur une résolution par la programmation linéaire. L'utilisation de la programmation linéaire dans un contexte de multiflots sera alors compensée par une structure plus riche du graphe de support du flot.

# BIBLIOGRAPHIE


AASHTIANI, H. A., and T. L. MAGNANTI. 1976. Implementing primal-dual network flow algorithms. Technical Report OR 055-76, Operations Research Center, MIT, Cambridge, MA.

AHRENS, J. H., G. FINKE. 1980. Primal transportation and transshipment algorithms. *Zeitshrift für Operations Research* **24,** 1-32.

AHUJA, R., T. MAGNANTI, and J. ORLIN. 1993. *Network Flows.* Prentice Hall, Englewood Cliffs, NJ.

ALI, A. I., D. BARNETT, K. FARHANGIAN, J. L. KENNINGTON, B. PATTY, B. SHETTY, B. MCCARL, and P. WONG. 1984. Multicommodity network problems: Applications and computations. *IIE Transactions* **16,** 127-134.

ALI, A. I., R. V. HELGASON, and J. L. KENNINGTON. 1982. An air force logistics decision support system using multicommodity network models. Technical Report 82-OR-1, Department of Operations Research, Southern Methodist University.

ALI, A. I., R. V. HELGASON, J. L. KENNINGTON, and H. LALL. 1980. Computationnal comparison among three multicommodity network flow algorithms. *Operations Research* **28,** 995-1000.

ALI, A. I., and J. L. KENNINGTON. 1977. MNETGEN program documentation. Technical Report IEOR 77003, Department of Industrial Engineering and Operations Research, Southern Methodist University, Dallas, Texas.

ANGLUIN, D. and L. G. VALIANT. 1979. Fast probabilistic algorithms for hamiltonion circuits and matchings. *Journal of Computer and System Sciences* **18,** 155-193.

ARMSTRONG, P. K., R. QI, and S. A. ZENIOS. 1990. Coercion methods for decomposition of network structured problems. Report 90-11-09, Decision Sciences Department, The Wharton School, University of Pennsylvania.

ASSAD, A. A. 1976. Solution techniques for the multicommodity flow problem. Masters' Thesis, Massachusets Institute of Technology, Cambridge, MA.

ASSAD, A. A. 1978. Multicommodity network flows: A survey. *Networks* **8,** 37-91.

ASSAD, A. A. 1980a. Models for rail transportation. *Transportation Research* **14A,** 205-220.

ASSAD, A. A. 1980b. Solving linear multicommodity flow problems. *Proceedings of the IEEE International Conference on Circuits and Computers*, pp. 157-161.

ATKINS, D. 1974. Managerial decentralisation and decomposition in mathematical programming. *Operational Research Quarterly* **25,** 615-624.

BALAS, E. 1966. An infeasibility-pricing decomposition method for linear programs. *Operations Research* **14,** 847-873.

BALAS, E., and N. CHRISTOFIDES. 1981. A restricted lagrangean approach to the traveling salesman problem. *Mathematical Programming* **21,** 19-46.

BARNETT, D., J. BINKLEY, and B. McCARL. 1982. The effects of U.S. port capacity constraints on national and world grain shipments. Technical Paper, Purdue Agricultural Experiment Station , Purdue University.

BARNHART, C. 1988. A network-based primal-dual solution methodology for the multicommodity network flow problem. Ph.D. dissertation, Department of Civil Engineering, MIT, Cambridge, MA.

BAZARAA, M. S. An infeasibility pricing algorithm for the multicommodity minimum cost flow problem. Working Paper, School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta.

BAZARAA, M. S., and J. L. GOODE. 1977. The traveling salesman problem: A duality approach. *Mathematical Programming* **13,** 221-237.

BAZARAA, M. S., and J. J. JARVIS. 1977. *Linear Programming and Network Flows.* Wiley, New York.

BAZARAA, M. S., J. J. JARVIS, and H. D. SHERALI. 1990. *Linear Programming and Network Flows*, 2nd ed. Wiley, New York.

BELLMORE, M., G. BENNIGTON, and S. LUBORE. 1971. A multivehicle tanker scheduling problem. *Transportation Science* **5,** 36-47.

BELLMORE, M., and J. C. MALONE. 1971. Pathology of traveling salesman subtour elimination algorithms. *Operations Research* **19,** 278-307.

BELLMORE, M., and G. L. NEMHAUSER. 1971. The traveling salesman problem: A survey. *Operations Research* **16,** 538-558.

BERTSEKAS, D. P., and P. TSENG. 1988a. The relax codes for linear minimum cost network flow problem. In *FORTRAN codes for Network Optimization*, edited by B. Simeone, P. Toth, G. Gallo, F. Maffioli, and S. Pallottino. *Annals of Operations Research* **13,** 125-190.

BERTSEKAS, D. P., and P. TSENG. 1988b. Relaxation methods for minimum cost ordinary and generalized network flow problems. *Operations Research* **36,** 93-114.

BIXBY, R. E. 1991. The simplex method: It keeps getting better. Presented at the *14th International Symposium on Mathematical Programming*, Amsterdam, The Netherlands

BODIN, L. D., B. L. GOLDEN, A. D. SCHUSTER, and W. ROWING. 1980. A model for the blockings of trains. *Transportation Research* **14B,** 115-120.

BRADLEY, S. P. 1965. Solution techniques for the traffic assignement problem. ORC 65-35, Operations Research Center, University of California, Berkeley.

BRADLEY, G. H., G. G. BROWN, and G. W. GRAVES. 1977. Design and implementation of large scale primal transshipment algorithms. *Management Science* **21,** 1-38.

BROWN, G., G. GRAVES, H. LANGE, C. STANIEC, and R. K. WOOD. 1990. Dual decomposition methods for solving multicommodity flow problems. Manuscript, Naval Postgraduate School.

BURKARD, R. E. 1971. Traveling salesman problem and assignment problems: A survey. *Annals of Discrete Mathematics* **4,** 193-217.

CAROLAN, W. J., J. E. HILL, J. L. KENNINGTON, S. NIEMI, and S. J. WICHMANN. 1990. An empirical evaluation of the KORBX algorithms for military airlift applications. *Operations Research* **38,** 240-248.

CARPANETO, G., and P. TOTH. 1980. Some new branching and bounding criteria for the asymmetric traveling salesman problem. *Management Science* **26,** 736-743.

CARTER, E. C., and J. R. STOWERS. 1963. Models for funds allocation for urban highway systems capacity improvements. *Highway Res. Rec.* **20,** 84-102.

CHANG, M. D., and C. J. CHEN. 1989. An improved simplex variant for pure processing networks. *ACM Transactions on Mathematical Software* **15,** 64-78.

CHARNES, A., and W. W. COOPER. 1961. Multicommodity traffic network models. *Theory of Traffic Flow*, edited by Robert Herman, Elsevier Publishing Co., Amsterdam, pp. 85-96.

CHEN, Y. L., and Y. H. CHIN. 1992. Multicommodity network flows with safety considerations. *Operations Research* **40,** 48-55.

CHENG, Y., D. HOUCH, J. LIU, M. MEKETON, L. SLUTSMAN, R. VANDERBEI, and P. WANG. 1989. The AT&T KORBX system. *AT&T Tech. J.* **68,** 7-19.

CHIEN, R.T., R. E. GOMORY, and T. C. HU. 1964. Communication networks. *IEE Trans. Circuit Theory* **11,** 19-22.

CHRISTOFIDES, N. 1975. *Graph Theory - An algorithmic approach.* Academic Press.

CHRISTOFIDES, N. 1979. The traveling salesman problem. In *Combinatorial Optimization*, edited by Christofides, Mingozzi, Toth and Sandi. Wiley, New York, pp. 131-150.

CLARKE, S., and J. SURKIS. 1968. An operations research approach to racial desegregation of school systems. *Socio-Economic Planning Sciences* **1,** 259-272.

COHEN, G. 1980. Auxiliary problem and decomposition of optimization problems. *J. Optimization Theory and App.* **32,** 277-305.

CPLEX Optimization Inc. 1993. *Using the CPLEX optimizer, 1.2 edition* (Incline Village, NV).

CRAINIC, T., J. A. FERLAND, and J. M. ROUSSEAU. 1984. A tactical planning model for rail freight transportation. *Transportation Science* **18,** 165-184.

CREMEANS, J. E., R. A. SMITH, and G. R. TYNDALL. 1970. Optimal multicommodity network flows with resource allocation. *Naval Research Logistics Quarterly* **17,** 269-280.

DANTZIG, G. B., and P. WOLFE. 1960. Decomposition principle for linear programs. *Operations Research* **8,** 101-111.

DANTZIG, G. B., and P. WOLFE. 1961. The decomposition algorithm for linear programs. *Econometrica* **29,** 767-778.

DEMMY, W. S. 1969. Multi-commodity flows in generalized networks. Memorandum Report No. 9, Advanced Logistics Systems Center Headquarters, Air Force Logistics Command, Dayton, Ohio.

DINIC, E. A. 1970. Algorithm for solution of a problem of maximum flow with power estimation. *Soviet Mathematics Doklady* **11,** 1277-1280.

DOMSCHKE, W. 1982. *Rundreisen und touren.* Oldenbourg Verlag, München-Wien.

EDMONDS, J., and R. M. KARP. 1972. Theoretical improvements in algorithm efficiency for network flow problems. *Journal of ACM* **19,** 248-264.

ELMAGHRABY, S. E. 1970. *Some Network Models in Management Science.* Springer-Verlag, New York.

EVANS, J. R. 1976a. A combinatorial equivalence between a class of multicommodity flow problems and the capacitated transportation problem. *Mathematical Programming* **10,** 141-144.

EVANS, J. R. 1976b. A single commodity network model for a class of multicommodity dynamic planning problems. Working Paper, Department of Quantitative Analysis, University of Cincinnati.

EVANS, J. R. 1977. Some network flow models and heuristics for multiproduct production and inventory planning. *AIIE Transactions* **10,** 75-81.

EVANS, J. R. 1978. A single-commodity transformation for certain multicommodity networks. *Operations Research* **26,** 673-680.

EVANS, J. R., J. J. JARVIS, and R. A. DUKE. 1975. Matroids, unimodularity, and the multicommodity transportation problem. Presented at the *ORSA/TIMS Conference*, Chicago.

EVEN, S., A. ITAI, and A. SHAMIR. 1976. On the complexity of timetable and multi-commodity flow problems. *SIAM Journal on Computing* **5,** 691-703.

EVEN, S.and R. E. TARJAN. 1975. Network flow and testing graph connectivity. *SIAM Journal on Computing* **4,** 507-518.

EVERETT, H., III. 1963. Generalized lagrange multiplier method for solving problems of optimum allocation of resources. *Operations Research* **11,** 399-417.

FARVOLDEN, J. M., W. B. POWELL, and I. J. LUSTIG. 1993. A primal partitioning solution for the arc-chain formulation of a multicommodity flow problem. *Operations Research* **41,** 669-693.

FINKE, G., A. CLAUS, and E. GUNN. 1984. A two-commodity network flow approach to the traveling salesman problem. *Congressus Numerantium* **41,** 167-178.

FORD, L. R., and D. R. FULKERSON. 1958. A suggested computation for maximal multicommodity network flows. *Management Science* **5,** 97-101.

FORD, L. R., and D. R. FULKERSON. 1962. *Flows in Networks*. Princeton University Press, Princeton, NJ.

FRANK, A. 1990. Packing paths, circuits and cuts-a survey. *Paths, Flows, and VLSI-layout*, Springer-Verlag, Korte, Lovász, Prömel, Schrijver editors, pp. 47-100.

FRATTA, L., M. GERLA, and L. KLEINROCK. 1973. The flow deviation method: An approach to store-And forward communication network design. *Networks* **3,** 97-133.

FULKERSON, D. R. 1963. Flows in networks. In *Recent Advances in Mathematical Programming*, edited by R. L. Graves and P. Wolfe. McGraw-Hill, New York, pp. 319-332.

GERSHT, A., and A. SHULMAN. 1987. A new algorithm for the solution of the minimum cost multicommodity flow problem. *Proceedings of the IEEE Conference on Decision and Control* **26,** 748-758.

GEOFFRION, A. M. 1970. Primal resource-directive approaches for optimizing nonlinear decomposable systems. *Operations Research* **18,** 375-403.

GEOFFRION, A. M., and G. W. GRAVES. 1974. Multicommodity distribution system design by benders decomposition. *Management Science* **20,** 822-844.

GLOVER, F., D. KARNEY, and D. KLINGMAN. 1974. Implementation and computational comparisons of primal, dual and primal-dual computer codes for minimum cost network flow problem. *Networks* **4,** 191-212.

GLOVER, F., D. KARNEY, D. KLINGMAN, and A. NAPIER. 1974. A computation study on start procedures, basis change criteria, and solution algorithms for transportation problems. *Management Science* **20,** 793-813.

GOLDEN, B. L. 1975. A minimum cost multicommodity network flow problem concerning imports and exports. *Networks* **5,** 331-356.

GONDRAN, M., et M. MINOUX. 1979. *Graphes et Algorithmes*. Eyrolles, Paris.

GRATZER, F. J., and K. STEIGLITZ. 1972. A heuristic approach to large multicommodity flow problems. *Proceeding of the Symposium on Computer Communication Networks and Teletraffic of Polytechnic Institute of Brooklyn*, pp. 311-324.

GRAVES, G. W., and R. D. MCBRIDE. 1974. Linear programming with linked lists ad dynamic factorization. Presented at the *National Meeting of ORSA*, San Juan, Puerto Rico.

GRAVES, G. W., and R. D. MCBRIDE. 1976. The factorization approach to large scale linear programming. *Mathematical Programming* **10,** 91-110.

GRIGORIADIS, M. D. 1986. An efficient implementation of the network simplex method. *Mathematical Programming Study* **26,** 83-111.

GRIGORIADIS, M. D., and W. W. WHITE. 1972a. A partitioning algorithm for the multicommodity network flow problem. *Mathematical Programming* **3,** 157-177.

GRIGORIADIS, M. D., and W. W. WHITE. 1972b. Computational experience with a multicommodity flow algorithm. In *Optimization Methods for Resource Allocation.* Edited by R. Cottle, and J. Krarup, English Universities Press, pp 205-225.

GRINOLD, R. C. 1972. Steepest ascent for large scale linear programs. *SIAM Review* **14,** 447-464.

GRÖTSCHEL, M., L. LOVÁSZ, and A. SCHRIJVER. 1988. *Geometric algorithms and combinatorial optimisation.* Springer-Verlag.

HAKIMI, L. S. 1962. Simultaneous flows through a communication network. *IRE Trans. Circuit Theory* **9,** 169-175.

HARTMAN, J. K., and L. S. LASDON. 1972. A generalized upper bounding algorithm for multicommodity flow problems. *Networks* **1,** 333-354.

HELD, M., and R. M. KARP. 1970. The traveling salesman problem and minimum spanning trees. *Operations Research* **18,** 1138-1162.

HELD, M., and R. M. KARP. 1971. The traveling salesman problem and minimum spanning trees. Part II, *Mathematical Programming* **1,** 6-25.

HELD, M., and J. KRARUP. 1974. Improvements of the Held-Karp algorithm for the symmetric traveling salesman problem. *Mathematical Programming* **7,** 87-96.

HELD, M., P. WOLFE, and H. CROWDER. 1974. Validation of subgradient optimization. *Mathematical Programming* **6,** 62-88.

HELGASON, R. V., and J. L. KENNINGTON. 1976. NETFLO program documentation. Technical Report IEOR 76011, Department of Industrial Engineering and Operations Research, Southern Methodist University, Dallas, Texas.

HELGASON, R. V., J. L. KENNINGTON, and P. WONG. 1981. An application of network programming for national forest planning. Technical Report OR 81006, Departement of Operations Research, Southern Methodist University.

HELLERMAN, E., and D. RARICK. 1971. Reinversion with the preassigned pivot procedure. *Mathematical Programming* **1,** 195-216.

HO, J. K., and E. LOUTE. 1983. Computational experience with advanced implementation of decomposition algorithms for linear programming. *Mathematical programming* **27,** 283-290.

HU, T. C. 1963. Multi-commodity network flows. *Operations Research* **11,** 344-360.

HU, T. C. 1964. On the feasibility of simultaneous flows in a network. *Operations Research* **12,** 359-360.

IBARAKI, T., and S. POLJAK. 1991. Weak three-linking in eulerian digraphs. *SIAM Journal on Discrete Mathematics* **4,** 84-98.

IBM CORPORATION. 1971. IBM mathematical programming system extended (MPSX)-linear and separable programming program description. Program Manual SH20-0968-0, International Business Machines Corporation, White Plains, New York.

IRI, M. 1971. On an extension of the maximum-flow minimum-cut theorem to multicommodity flows. *Journal of the Operations Research Society of Japan* **13,** 129-135.

ITAI, A. 1978. Two-commodity flow. *Journal of ACM* **25,** 596-611.

JARVIS, J. J. 1969. On the equivalence between the node-arc and arc-chain formulations of the multi-commodity maximal flow problem. *Naval Research logistics Quarterly* **16,** 525-529.

JENNERGREN, L. P. 1973. A price-schedules decomposition algorithm for linear programming problems. *Econometrica* **41,** 965-980.

JEWELL, W. S. 1957. Warehousing and distribution of a seasonal product. *Naval Research Logistics Quarterly* **4,** 29-34.

JEWELL, W. S. 1966. Multi-commodity network solutions. ORC 66-23, Operations Research Center, University of California, Berkeley.

JONES, K. L., I. L. LUSTIG, J. M. FARVOLDEN, and W. B. POWELL. 1993. Multicomodity network flows: The impact of formulation on decomposition. *Mathematical Programming* **62,** 95-117.

JORGENSON, N. O. 1963. Somme aspects of the urban traffic assignement problem. Graduate Report, Institute of Transportation and Traffic Ingineering, University of California, Berkeley.

KALABA, R. E., and M. L. JUNCOSA. 1957. Optimal design and utilization of communication networks. *Management Science* **3,** 33-44.

KAPLAN, S. 1973. Readiness and the optimal redeployment of resources. *Naval Research Logistics Quarterly* **20,** 625-638.

KARMARKAR, N. 1984. A new polynomial-time algorithm for linear programming. *Combinatorica* **4,** 373-395.

KARZANOV, A. V. 1974. Determining the maximal flow in a network by the method of preflow. *Soviet Mathematics Doklady* **15,** 434-437.

KENNINGTON, J. L. 1977. Solving multicommodity transportation problems using a primal partitioning simplex technique. *Naval Research Logistics Quarterly* **24,** 309-325.

KENNINGTON, J. L. 1978. A survey of linear cost multicommodity network flows. *Operations Research* **26,** 209-236.

KENNINGTON, J. L., and R. V. HELGASON. 1980. *Algorithms for Network Programming*. Wiley-Interscience, New York.

KENNINGTON, J. L., and M. SHALABY. 1977. An effective subgradient procedure for minimal cost multicommodity flow problems. Technical Report IEOR 750010, Department of Industrial Engineering and Operations Research, Southern Methodist University.

KENNINGTON, J. L., and Z. WANG. 1990. The shortest augmenting path algorithm for the transportation problem. Technical Report 90-CSE-10, Southern Methodist University, Dallas, TX.

KIRBY, M., W. HAGER, and P. WONG. 1982. Simultaneous planning of wild landmanagement and transportation alternatives. Technical Report, Forest Service, US Department of Agriculture, Berkeley, California.

KLEITMAN, D. J. 1971. An algorithm for certain multi-commodity flow problems. *Networks* **1,** 75-90.

KLINGMAN, D., A. NAPIER, and J. STUTZ. 1974. NETGEN: A program for generating large scale capacitated assignment, transportation, and minimum cost flow network problems. *Management Science* **20,** 814-821.

KORACH, E. 1982. Packing of T-cuts, and other aspects of dual integrality. Ph.D. thesis, Waterloo University.

KORTE, B. 1988. Applications of combinatorial optimization. Technical Report 88541-OR, Institute für Okonometrie und Operations Research, Bonn, Germany.

LANGLEY, R. W., J. L. KENNINGTON, and C. M. SHETTY. 1974. Efficient computation devices for the capacitated transportation problem. *Naval Research Logistics Quarterly* **21,** 637-647.

LASDON, L. S. 1970. *Optimization Theory for Large Systems*. Macmillan, New York.

LAWLER, E. L., J. K. LEENSTRA, A. H. G. RINNOOY KAN, and D. B. SHMOYS. 1985. *The Traveling Salesman Problem*. Wiley, New York.

LOMONOSOV, M. W. 1983. On the planar integer two-flow problem. *Combinatorica* **3,** 207-218.

LUSTIG, I. J., R. E. MARSTEN, and D. F. SHANNO. 1992. On implementing Mehrotra's predictor-corrector interior point method for linear programming. *SIAM Journal on Optimization* **2,** 435-449.

MAHEY, P. 1986. Méthodes de décomposition et décentralisation en programmation linéaire. *R.A.I.R.O.* **20,** 287-306.

MALHOTRA, V. M., M. P. KUMAR, and S. N. MAHESHWARI. 1978. An $O(|V^2|)$ algorithm for finding maximum flows in networks. *Information Processing Letters* **7,** 277-278.

MARTELLO, S. 1981. An enumerative algorithm for finding hamiltonian circuits in a directed graph. Research Report, Istituto di Automatica, University of Bologna, Italy.

MARTELLO, S. 1983. An enumerative algorithm for finding hamiltonian circuits in a directed graph. *ACM Transactions on Mathematical Software* **9,** 131-138.

MATSUMOTO, K., T. NISHIZEKI, and N. SAITO. 1985. An efficient algorithm for finding multicomodity flows in planar networks. *SIAM Journal on Computing* **14,** 289-302.

MCBRIDE, R. D., and J. W. MAMEN. 1993. Solving multicommodity flow problem with a primal embedded network simplex algorithm. Presented at the *ORSA/TIMS Conference*, Phoenix.

MCCALLUM, C. J. JR. 1977. A generalized upper bounding approach to a communication network planing problem. *Networks* **7,** 1-23.

MCDIARMID, C. 1989. On the method of bounded differences. In *Surveys in Combinatorics*, edited by J. Siemons. London Math. Soc. Lecture Notes, Series 141, Cambridge University Press, Cambridge, England.

MIDDENDORF, M., and F. PFEIFFER. 1990. On the complexity of the disjoint paths problem. In *Polyhedral Combinatorics*, DIMACS 1, AMS, ACM, W. Cook and P. Seymour editors, pp. 171-178.

MINOUX, M. 1974. Planification à court terme d'un réseau de télécommunication. *Annales Télécommunications* **29**.

MINOUX, M. 1975. Résolution des problèmes de multiflots en nombres entiers dans les grands réseaux. *R.A.I.R.O.* **3,** 21-40.

MINOUX, M. 1977. Amélioration d'un algorithme de Kennington et Shalaby par résolution approchée du problème dual. Papier non publié.

MORIN, M. H. 1993. Modélisation et décomposition des problèmes de transbordement dynamiques: application à la répartition des wagons Fret/SNCF. Thèse de Doctorat, Université de Grenoble.

MULVEY, J. 1978. Pivot strategies for primal-simplex network codes. *Journal of ACM* **25,** 266-270.

MURTAGH, B. A., and M. A. SAUNDERS. 1983. MINOS 5.0 user's guide. Technical Report SOL-83-20, Stanford University, Stanford, California.

MURTY, K. G. 1976. *Linear and Combinatorial Programming*. Wiley, New York.

NACH-WILLIAMS, C. St. J. A. 1960. On orientations, connectivity and odd vertex pairings in finite graphs. *Canad. J. Math.* **12,** 555-567.

NANIWADA, M. 1969. Multicommodity flows in a communication network. *Electronics communication Japan* **52A,** 34-41.

ONAGA, J. 1970. A multi-commodity flow theorem. *Electronics communication Japan* **53A,** 16-22.

ORCHARD-HAYS, W. 1968. *Advanced Linear-Programming Computing Techniques.* McGraw-Hill, New York.

PADBERG, M. and T-Y. SUNG. 1991. An analytical comparison of different formulations of the traveling salesman problem. *Mathematical Programming* **52,** 315-357.

PANAGIOTAKOPOULOS, D. 1976. Multicommodity multi-transformed networks flows with an application to residuals management. *Management Science* **22,** 874-882.

PINAR, M. C., and S. A. ZENIOS. 1992. Parallel decomposition of multicommodity network flows using a linear-quadratic penalty algorithm. *ORSA Journal on Computing* **4,** 235-249.

POTTS, R. B., and R. M. OLIVER. 1972. *Flows in Transportation Networks*. Academic Press, New York.

POWELL, W. B., and Y. SHEFFI. 1983. The load planning problem of motor carriers: Problem description and a proposed solution approach. *Transportation Research* **17A,** 471-480.

POWELL, W. B., and Y. SHEFFI. 1989. Design and implementation of an interactive optimization system for network design in the motor carrier industry. *Operations Research* **37,** 12-29.

PRAGER, W. 1954. Problems of traffic and transportation. *Proceedings of the Symposium on Operations Research in Business and Industry*, Kansas City, Mo., pp. 105-113.

RAGHAVAN, P. 1988. Probabilistic construction of deterministic algorithms: Approximating packing integer programs. *Journal of Computer and System Sciences* **37,** 130-143.

RAGHAVAN, P., and C. D. THOMPSON. 1987. Randomized rounding: A technique for provably good algorithms and algorithmic proofs. *Combinatorica* **7,** 365-373.

RAO, M. R. 1969. A column generation sheme for the deterministic multicommodity warehousing model with cash liquidity constraints. CORE Discussion Paper No. 6902, ICMS.

RAO, M. R., and S. ZIONTS. 1968. Allocation of transportation units to alternative trips. *Operations Research* **16,** 52-63.

REZIG, W. et G. FINKE. 1995. Problèmes de multiflots continus et entiers. Présenté aux *Rencontres Francophones de Recherche Opérationnelle (FRANCORO)*, Mons, Belgique.

REZIG, W. et G. FINKE. 1992. Une méthode de décomposition pour le problème du biflot. Présenté à la *3ème Rencontre Franco-Algérienne de Recherche Opérationnelle*, Alger, Algérie.

ROBACKER, J. T. 1956. Notes on linear programming: Part XXXVII concerning multicommodity networks. RM-1799, The Rand Corporation, Santa Monica, California.

ROBERTS, S. M., and B. FLORES. 1966. Systematic generation of hamiltonian circuits. *Comm. ACM* **9,** 690-694.

ROBERTSON, N., and P. SEYMOUR. 1990. An outline of a disjoint paths algorithm. *Paths, Flows, and VLSI-layout*, Springer-Verlag, Korte, Lovász, Pr ömel, Schrijver editors, pp. 47-100.

ROCKAFELLAR, R. T. 1984. *Network Flows and Monotropic Optimization.* Wiley, New York.

ROSEN, J. B. 1964. Primal partition programming for block diagonal matrices. *Numeriche Mathematics* **6,** 250-260.

ROTHFARB, B., and I. T. FRISCH. 1969. On the 3-commodity flow problem. *SIAM journal on Applied Mathematics* **17,** 46-58.

ROTHFARB, W., N. P. SHEIN, and I. T. FRISCH. 1968. Common terminal multicommodity flow problem. *Operations Research* **16,** 202-205.

ROTHSCHILD, B., and A. WHINSTON. 1966a. On two commodity network flows. *Operations Research* **14,** 377-387.

ROTHSCHILD, B., and A. WHINSTON. 1966b. Feasibility of two commodity network flows. *Operations Research* **14,** 1121-1129.

SAIGAL, R. 1967. Multicommodity flows in directed networks. ORC 67-38, Operations Research Center, University of California, Berkeley.

SAKAROVITCH, M. 1966. The multi-commodity maximal flow problem. ORC Report 66-25, Operations Research Center, University of California, Berkeley.

SAKAROVITCH, M. 1973. Two commodity network flows and linear programming. *Mathematical Programming* **4,** 1-20.

SCHNEUR, R. 1991. Scaling algorithms for multicommodity flow problems and network flow problems with side constraints. Ph.D. dissertation, Department of Civil Engineering, MIT, Cambridge, MA.

SCHULTZ, G. L., and R. R. MEYER. 1991. An interior point method for block angular optimization. *SIAM Journal on Optimization* **1,** 583-602.

SEBÖ, A. 1987. Packing of odd cuts and plane multicommodity flows. In *Infinite and Finite Sets, Proceedings of a conference held in Eger*, (Hajnal, T. Sós eds.). North Holland, pp. 453-469.

SEBÖ, A. 1993. Integer plane multiflows with a fixed number of demands. *Journal of Combinatorial Theory* **59B,** 163-173.

SELBY, G. R. 1970. The use of topological methods in computer-aided circuit layout. Ph.D. Thesis, London University.

SMITH, T. H. C., V. SRINIVASAN, and G. L. THOMPSON. 1977. Computational performance of three subtour elimination algorithms for solving asymmetric traveling salesman problems. *Annals of Discrete Mathematics* **1,** 495-506.

SMITH, T. H. C., and G. L. THOMPSON. 1977. A lifo implicit enumeration search algorithm for the symmetric traveling salesman problem using Held and Karp's 1-tree relaxation. *Annals of Discrete Mathematics* **1,** 479-493.

SRINIVASAN, V., and G. L. THOMPSON. 1973. Benefit-cost analysis of coding techniques for the primal transportation algorithm. *Journal of Associating Computer Machinery* **20,** 194-213.

SRIVASTAV, A., and P. STANGIER. 1993a. Algorithmic Chernoff-Hoeffding inequalities in integer programming. Preprint, Research Institute of Discrete Mathematics, University of Bonn. Extended abstract in D.-Z. Du and X.-S. Zhang, eds., *Proceedings of the 5th International Symposium on Algorithms and Computation*, Beijing, Lecture Notes in Computer Science, Springer, Berlin, pp. 226-233.

SRIVASTAV, A., and P. STANGIER. 1993b. Derandomized approximation of maximum integer multiflows. In *Integer Programming and Combinatorial Optimization*, edited by G. Rinaldi and L. Wolsey, pp. 226-234.

SRIVASTAV, A., and P. STANGIER. 1993c. Integer multicommodity flows with reduced demands. In T. Lengauer, ed., *Proceedings of the 1st Annual European Symposium on Algorithms*, Bonn, Lecture Notes in Computer Science, Springer, Berlin, pp. 360-372.

SWOVELAND, C. 1971. Decomposition algorithms for the multi-commodity distribution problem. Working Paper No 184, Western Management Science Institute, University of California, Los Angeles.

SWOVELAND, C. 1973. A two stage decomposition for a generalized multi-commodity flow problem. *INFOR* **11,** 232-244.

TANG, D. T. 1965. Comments on feasibility conditions of simultaneous flows in a network. *Operations Research* **13,** 143-146.

TARDOS, É. 1986. A strongly polynomial algorithm to solve combinatorial linear programs. *Operations Research* **34,** 250-256.

TOMLIN, J. A. 1966a. Minimum-cost multicommodity network flows. *Operations Research* **14,** 44-51.

TOMLIN, J. A. 1966b. A linear programming model for the assignement of traffic. *Proceedings of the Third Conference of The Australian Road Board* **3,** 263-271.

TOMLIN, J. A. 1967. Mathematical programming models for traffic network problems. Unpublished dissertation, Departement of Mathematics, University of Adelaide, Australia.

VAIDYA, P. M. 1989. Speeding up linear programming using fast matrix multiplication. *Proceedings of the 30th Annual Symposium on the Foundations of Computer Science*, pp. 332-337.

VYGEN, J. 1992. *Disjoint Paths in Directed Graphs.* Diploma Thesis (in German), University of Bonn.

WARDROP, J. G. 1952. Some theoritical aspects of road traffic research. *Proceedings of the Institute of Civil Engineers London* **1,** 325-378.

WEIGEL, H. S., and J. E. CREMEANS. 1972. The multicommodity network flow model revised to include vehicle per time period and node constraints. *Naval Research Logistics Quarterly* **19,** 77-89.

WHITE, W. W. 1972a. Mathematical programming, multicommodity flows, and communication nets. *Proceedings of the Symposium of Computer Communications Networks and Teletraffic of Polytechnic Institute of Brooklyn,* pp. 325-334.

WHITE, W. W. 1972b. Dynamic transshipment networks: An algorithm and its application to the distribution of empty containers. *Networks* **2,** 211-259.

WHITE, W. W., and A. M. BOMBERAULT. 1969. A network algorithm for empty freight car allocation. *IBM Systems Journal* **9,** 147-169.

WHITE, W. W., and E. WRATHALL. 1970. A system for railroad traffic scheduling. Technical Report No 320-2993, IBM-Philadelphia Scientific Center.

WOLLMER, R. D. 1972. Multicommodity networks with resource constraints: The generalized multicommodity flow problem. *Networks* **1,** 245-263.

ZANGWILL, W. I. 1969a. *Nonlinear Programming a Unified Approach.* Prentice Hall, Englewood Cliffs, NJ.

ZANGWILL, W. I. 1969b. A backlogging model and a multi-echelon model of dynamic economic lot size production system- A network approach. *Management Science* **15,** 506-527.