

Université Libre de Bruxelles
Faculté des Sciences
Département d' Informatique

Toll Setting Problem :
présentation théorique et développement
d'heuristiques pour un cas particulier.

Mémoire présenté en vue de l'obtention du grade de
Licencié en Informatique
sous la direction de Mme. Martine Labbé
et sous la codirection de Mme. Sophie Dewez

Année académique 2005-2006

Grégory GATHY

Remerciements

Je remercie Mme. Martine Labbé de m'avoir permis de découvrir le sujet passionnant dont traite ce mémoire.

Je remercie également Mme. Sophie Dewez pour son aide permanente, ses encouragements et sa grande disponibilité.

Un très grand merci à ma soeur pour ses conseils avisés, pour les nombreuses relectures effectuées et pour son soutien.

Un grand merci à mes parents pour leur soutien et leur compréhension.

L'aboutissement de ce mémoire est aussi celui des mes études, c'est pourquoi je remercie le CI et plus particulièrement Bastien, Fabian, Guillaume, John, Julien et Nicolas pour leur soutien et pour l'amitié qu'ils m'ont témoigné durant ces dernières années.

Je remercie tout particulièrement Edith pour son réconfort et la tendresse qu'elle m'a témoigné durant tous ces moments difficiles.

Table des matières

1	Introduction	4
2	Programme Linéaire	6
2.1	Programmation Biniveau	7
2.1.1	The price setting problem	8
2.1.2	The toll setting problem	9
2.2	Programme bilinéaire à un niveau	12
3	Le Toll Setting Problem	14
3.1	Exemples	14
3.1.1	Exemple 1 : borne supérieure de la taxe totale du meneur non atteinte	14
3.1.2	Exemple 2 : taxes négatives	15
3.1.3	Exemple 3 : forme de la fonction revenu	16
3.2	Formulations	18
3.2.1	Illustration	18
3.2.2	Reformulation booléenne.	20
3.2.3	Passage d'une formulation bilinéaire à deux niveaux à une formulation bilinéaire à un niveau.	21
3.2.4	Linéarisation	23
3.3	Choix des constantes M_{ij}^k et N_{ij}	24
4	Complexité du Toll Setting Problem	31
5	The Highway Special Case	37
5.1	Introduction et formulation	37
5.1.1	Exemple	37
5.1.2	Formulation du problème	39
5.1.3	Formulation équivalente	44
5.2	Un algorithme exact : K-chemin	47
5.2.1	Calcul du i-ème K-chemin	49
5.3	Heuristiques	55
5.3.1	Heuristique 1	56

5.3.2	Heuristiques 2 et 3	56
5.3.3	Heuristiques 4 et 5	60
5.3.4	Heuristiques 6 et 7	62
5.3.5	Heuristique 8	65
5.3.6	Heuristique 9	66
5.3.7	Heuristique 10	67
5.3.8	Heuristique 11	71
5.3.9	Heuristique 12	75
5.3.10	Heuristique 13	81
5.3.11	Heuristique 14	83
5.3.12	Heuristique 15	84
5.4	Résultats numériques	84
6	Conclusion	91
A	Complexité SAT et 3-SAT	95
A.1	SAT	95
A.2	3SAT	98

Chapitre 1

Introduction

Les gestionnaires de réseaux (télécoms, transports) doivent établir des tarifs sur les services qu'ils proposent. Pour déterminer au mieux les tarifs qu'ils vont imposer, ils doivent tenir compte des réactions des utilisateurs.

Nous distinguons deux niveaux de décisions : la société qui veut maximiser son profit et les utilisateurs qui veulent minimiser leur coût. Une hiérarchie naturelle apparaît. L'avantage va aux gestionnaires qui, s'ils prévoient correctement les réactions des utilisateurs, pourront maximiser leur profit.

Si les objectifs des deux niveaux sont en conflit, il s'agit d'un problème d'optimisation biniveau.

Ce problème d'optimisation est lié au jeu de Stackelberg [17]. Ce jeu met en compétition deux joueurs, l'un d'entre eux (le meneur) joue en premier tout en connaissant le problème d'optimisation du deuxième joueur (le suiveur). Après que le meneur a joué, le suiveur joue en tenant compte des choix qu'a faits le meneur.

Dans ce mémoire, nous étudions le problème de tarification de réseaux routiers. Ce problème appartient à la classe des problèmes bilinéaires à deux niveaux, c'est à dire que la fonction objective de chaque niveau est bilinéaire. Le problème de tarification de réseaux routiers consiste à déterminer les péages à placer sur des tronçons appartenant à une société de façon à ce qu'elle maximise son revenu. Le choix optimal des péages doit se faire en tenant compte des réactions des utilisateurs : les tarifs doivent être assez bas pour que les utilisateurs empruntent les tronçons appartenant à la société mais aussi assez élevés afin que la société génère des revenus importants.

Dans le deuxième chapitre, nous redéfinissons la notion de programme linéaire et nous introduisons la forme générale d'un problème à deux niveaux. Afin de justifier la programmation bilinéaire à deux niveaux, nous donnons deux exemples

d'applications (Price Setting Problem et Toll Setting Problem). Nous terminons ce chapitre par l'introduction du passage d'un programme bilinéaire à deux niveaux à un programme bilinéaire à un niveau.

Le troisième chapitre est consacré à des résultats existants pour le Toll Setting Problem. Nous commençons par donner des exemples illustrant le fait que l'on ne peut pas toujours atteindre la borne supérieure sur le revenu, que les solutions avec des taxes négatives existent et que la solution à ce problème n'est pas unique. Nous posons ensuite quelques hypothèses que nous considérerons comme vérifiées tout au long du mémoire. Nous transformons ensuite le problème en un problème bilinéaire à un niveau comme fait dans Labbé et al [11]. Nous reformulons ensuite le problème bilinéaire à un niveau en un problème linéaire mixte entier comme fait dans S. Dewez [8]. Finalement nous présentons, comme dans S. Dewez [8], des bornes supérieures qui renforcent la formulation linéaire mixte.

Le quatrième chapitre est consacré à l'étude de la complexité du Toll Setting Problem. Nous donnons la preuve de la NP-Complétude du Toll Setting Problem comme effectué dans S. Roch et P. Marcotte [15].

Dans le dernier chapitre, nous étudions un cas particulier du Toll Setting Problem. Ce cas particulier est le cas où les arcs taxés constituent un chemin et chaque arc taxé constitue un chemin unique pour chaque utilisateur. On appelle ce problème The Highway Special Case. Nous reformulons le problème comme dans S. Dewez [8] et nous calculons des bornes supérieures. Cette formulation contient des inégalités triangulaires qui rendent particulièrement difficile la résolution du problème. Nous remplaçons ces inégalités, comme proposé par S. Dewez [8], par d'autres sans changement de la valeur de la solution optimale. Nous présentons ensuite une méthode exacte pour résoudre le problème : l'algorithme de K-Chemins[9] qui fait appel à une résolution inverse du problème : nous connaissons les arcs que l'utilisateur va emprunter, il faut alors adapter les taxes. Finalement nous terminons ce chapitre par le développement d'heuristiques pour le Highway Special Case. Ces heuristiques utilisent différentes méthodes pour résoudre le problème. Lors du choix des tronçons que les utilisateurs vont prendre nous :

- prenons en compte ou non des bornes déjà placées sur les tronçons
- prenons en compte ou non de l'environnement des tronçons
- choisissons les arcs à ne pas emprunter

Nous avons aussi développé une heuristique basée sur l'algorithme des K-Chemins. Nous concluons ce chapitre avec des résultats numériques et en analysant les résultats.

Chapitre 2

Programme Linéaire

Dans ce chapitre nous allons brièvement rappeler quelques notions liées aux programmes linéaires, présenter la programmation biniveau et la programmation bilinéaire à un niveau. Nous présenterons aussi deux problèmes introductifs aux problèmes de taxations.

Définition 2.1 *Un programme linéaire est un problème d'optimisation d'une fonction linéaire sous des contraintes linéaires.*

On peut aussi énoncer le problème comme suit : trouver les valeurs des n variables x_1, x_2, \dots, x_n qui minimisent l'expression :

$$c_1x_1 + c_2x_2 + \dots + c_nx_n$$

appelée **fonction économique** ou **objective**, et qui vérifient les contraintes suivantes :

$$\begin{aligned} (1) \quad & a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n && \leq b_1 \\ & \dots && \\ & a_{i1}x_1 + a_{i2}x_2 + \dots + a_{in}x_n && \leq b_i \\ & \dots && \\ & a_{m1}x_1 + a_{m2}x_2 + \dots + a_{mn}x_n && \leq b_m \\ & x_1, x_2, \dots, x_n && \geq 0, \end{aligned}$$

où $c_1, c_2, \dots, c_n, a_{11}, a_{12}, \dots, a_{mn}, b_1, b_2, \dots, b_m$ sont des nombres donnés. Les notations suivantes peuvent aussi être utilisées :

$$\begin{cases} \min \sum_{j=1}^n c_j x_j \\ \sum_{j=1}^n a_{ij} x_j \leq b_i, & i = 1, 2, \dots, m \\ x_j \geq 0, & \forall j \in \{0, 1, \dots, n\} \end{cases}$$

ou encore :

$$\min CX : AX \leq b, X \geq 0$$

où A, b, C et X sont des matrices de dimension respective $(m \times n)$, $(m \times 1)$, $(1 \times n)$, $(n \times 1)$.

La nature d'un programme linéaire ne dépend pas de la forme particulière qu'on lui donne; par contre, elle dépend de la nature des variables qu'il contient.

Programme linéaire : c'est un problème qui peut se mettre sous la forme (1).

Programme linéaire continu : c'est un programme linéaire où les variables x_i peuvent prendre n'importe quelle valeur réelle.

Programme linéaire booléen : c'est un programme linéaire où les variables x_i ne peuvent prendre que les valeurs 0 ou 1.

Programme linéaire entier : c'est un programme linéaire où les variables x_i ne peuvent prendre que des valeurs entières.

Programme linéaire mixte-booléen : c'est un programme linéaire où certaines variables x_i prennent des valeurs réelles, les autres des valeurs booléennes.

Programme linéaire mixte-entier : c'est un programme linéaire où certaines variables x_i prennent des valeurs réelles, les autres des valeurs entières.

2.1 Programmation Biniveau

Considérons le jeu suivant impliquant 2 joueurs : A et B . Ces deux joueurs veulent optimiser leur fonction objective.

Le joueur A fait son choix en premier, une fois ce choix observé, le joueur B va ensuite faire son propre choix de façon à optimiser sa fonction objective tout en prenant en compte le choix qu'a fait A .

Le problème consiste donc à trouver la stratégie optimale pour le joueur A en prenant en considération le problème d'optimisation lié à B . Ce jeu s'appelle : *Le jeu de Stackelberg* [17].

On suppose que pour chaque décision que le premier joueur fait, il existe une réponse pour le deuxième. C'est un problème hiérarchique où deux niveaux de décision sont en conflit.

Nous appelons le premier joueur le *meneur* et le second joueur le *suiveur*.

Nous allons nous baser sur l'article de Bialas et Karwan [4] pour introduire la forme d'un programme biniveau. Pour une stratégie x choisie par le meneur, la stratégie du suiveur est donnée par :

$$\begin{aligned} & \min_y f(x, y) \\ & \text{s.c. } g(x, y) \leq 0, \end{aligned}$$

où f est la fonction objective du suiveur, $g(x, y)$ est un ensemble de contraintes qui doivent être satisfaites, y est le vecteur contenant les variables de décision du suiveur et x est le vecteur contenant les variables de décision du meneur.

Ce problème est aussi appelé *Problème de deuxième niveau*.

On représente par $\mathcal{S}(x)$ l'ensemble des solutions admissibles pour le problème de deuxième niveau :

$$\mathcal{S}(x) = \{y : g(x, y) \leq 0\}.$$

L'ensemble de solutions optimales pour le deuxième niveau est donné par :

$$\mathcal{R}(x) = \{y^* \in \mathcal{S} : (x, y^*) \in \arg \min_y \{f(x, y) : g(x, y) \leq 0\}\}.$$

Maintenant nous pouvons formuler le problème d'optimisation du meneur.

Ce problème est aussi appelé *Problème de premier niveau* et il prend en compte une solution $y^* \in \mathcal{R}(x)$ du suiveur :

$$\begin{aligned} & \min_{x, y^*} F(x, y^*) \\ \text{s.c. } & G(x, y^*) \leq 0, \\ & y^* \in \mathcal{R}(x), \end{aligned}$$

où F est la fonction objective du meneur et où $G(x, y^*) \leq 0$ est un ensemble de contraintes qui doivent être satisfaites par le premier niveau.

Pour un x donné, $\mathcal{R}(x)$ peut contenir un élément ou plus.

Finalement, l'ensemble :

$$\mathcal{I} = \{(x, y) : G(x, y) \leq 0, y \in \mathcal{R}(x)\}$$

contient les *solutions* réalisables (c'est à dire les points (x, y) satisfaisant les contraintes) et est appelé la région induite.

On peut donc modéliser le problème comme suit :

$$\begin{aligned} & \min_{x, y} F(x, y) \\ \text{s.c. } & G(x, y) \leq 0, \\ & \min_y f(x, y) \\ \text{s.c. } & g(x, y) \leq 0. \end{aligned}$$

Ceci est un problème à deux niveaux puisque nous avons deux niveaux de décision.

2.1.1 The price setting problem

Supposons qu'une entreprise propose de nouveaux services à des clients. Elle percevra un taxe sur chacun de ces services. Le client peut déjà profiter de services non offerts par d'autres compagnies. Il devra donc choisir entre des services

existants et ceux nouvellement proposés par l'entreprise.

Pour modéliser le problème, il faut représenter les quantités de services offerts par l'entreprise qui sont utilisés par les clients. Nous les représenterons comme les entrées du vecteur x .

Comme expliqué ci-dessus, le client a aussi droit à des services qui ne sont pas soumis à la taxe de l'entreprise car ils n'appartiennent pas à cette dernière. Nous les représenterons par les entrées du vecteur y .

Le problème consiste donc à trouver la taxe (représenté par le vecteur t) que l'entreprise doit associer à ses services pour maximiser son revenu. Ceci doit être fait en prenant compte des réactions du client qui, lui, essaie de minimiser son coût.

Aux services taxés (ceux appartenant à l'entreprise), on associe un vecteur de coût c et aux services non-taxés (ceux qui n'appartiennent pas à l'entreprise), on associe un vecteur de coût d .

Le problème peut donc être formulé comme suit :

$$\max_t tx \tag{2.1}$$

$$\min_{x,y} (c + t)x + dy \tag{2.2}$$

$$\text{s.c. } Ax + By = b \tag{2.3}$$

$$x, y \geq 0. \tag{2.4}$$

Le client paiera un coût fixe d s'il prend des services qui n'appartiennent pas à l'entreprise et un coût fixe c majoré d'une taxe t s'il prend un service proposé par l'entreprise. Il veut minimiser son coût, ceci est représenté par (2.2).

2.1.2 The toll setting problem

Ce problème est un cas particulier du Price Setting Problem et c'est celui que l'on va traiter dans ce mémoire, il s'agit d'un problème de péage. Nous considérons un réseau composé de noeuds (villes) et d'arcs (routes). Une partie de ces arcs appartient à une entreprise privée (le meneur) et sont tarifables.

Quelques utilisateurs (les suiveurs) voyagent d'un noeud (origine) à un autre (destination), l'ensemble des utilisateurs ayant la même origine et la même destination est appelé *commodité*.

Le problème consiste à trouver la taxe que doit mettre l'entreprise sur ses arcs pour optimiser son revenu, ceci en prenant en compte la réaction des utilisateurs qui veulent minimiser leurs dépenses.

Le meneur sait comment le suiveur va réagir en fonction des taxes qu'il aura imposées sur ses tronçons routiers. Une fois que l'entreprise aura fixé les taxes sur ses arcs, le client choisit son chemin.

Nous nous basons sur les travaux de Labbé et al. [12] et Roch et al. [16].

On représente par \mathcal{N} l'ensemble des noeuds, par \mathcal{A} l'ensemble des arcs tarifables, par \mathcal{B} l'ensemble des arcs non tarifables et par K l'ensemble des commodités. Pour chaque arc a tarifable, on associe un coût composé d'une partie fixe c_a qui représente le coût (exemple : le temps, la consommation d'essence, \dots) et d'une partie variable t_a qui est la taxe.

D'une façon similaire, on associe un coût fixe d_a pour chaque arc non tarifable a . On représentera aussi une commodité k par $(o_k - d_k)$, où o_k est l'origine de la commodité k et d_k la destination.

Le nombre d'utilisateurs de la commodité k est noté n^k . Le vecteur b^k représente pour chaque noeud $i \in \mathcal{N}$ la demande

$$b_i^k = \begin{cases} n^k & \text{si } i = o_k, \\ -n^k & \text{si } i = d_k, \\ 0 & \text{sinon.} \end{cases}$$

Notons x_a^k la variable correspondant au nombre d'utilisateurs de la commodité k qui utilisent l'arc tarifable a et y_a^k la variable correspondant au nombre d'utilisateurs de la commodité k qui utilisent l'arc non tarifable a .

Si nous négligeons les effets de congestion, si nous estimons que la demande est fixe et que les utilisateurs minimisent le coût général de leur voyage, le *Toll Setting Problem (TSP)* peut être formulé comme un programme à deux niveaux avec des objectifs bilinéaires et avec des contraintes linéaires :

$$(TSP1) \max_{x,y,t} \sum_{a \in \mathcal{A}} t_a \sum_{k \in K} x_a^k \quad (2.5)$$

$$\min_{x,y} \sum_{k \in K} \left(\sum_{a \in \mathcal{A}} (c_a + t_a) x_a^k + \sum_{a \in \mathcal{B}} d_a y_a^k \right) \quad (2.6)$$

$$\text{s.c.} \sum_{a \in i^+} (x_a^k + y_a^k) - \sum_{a \in i^-} (x_a^k + y_a^k) = b_i^k \quad \forall k \in K, \quad \forall i \in \mathcal{N} \quad (2.7)$$

$$x_a^k, y_a^k \geq 0 \quad \forall k \in K, \forall a \in \mathcal{A} \cup \mathcal{B} \quad (2.8)$$

où i^+ représente l'ensemble des arcs qui ont i comme origine et i^- représente l'ensemble des arcs qui ont i comme destination.

Examinons plus en détails les différentes "composantes" du (TSP1) :

- La fonction objective du premier niveau (2.5) modélise le fait que l'entreprise veut maximiser son profit : pour chaque arc tarifable ($a \in \mathcal{A}$) on multiplie la taxe perçue sur l'arc par le nombre d'utilisateurs l'empruntant et ce pour chaque commodité.
- La fonction objective du deuxième niveau (2.6) modélise le fait que l'utilisateur veut minimiser son coût : pour chaque commodité, il faut minimiser le coût du trajet, qu'on utilise des arcs tarifables ou non-tarifables.
- Finalement, la contrainte (2.7) impose que le flot d'utilisateurs d'une même commodité soit le même à l'origine qu'à la destination mais aussi que le flot

entrant dans un noeud soit égal à celui sortant lorsque ce noeud n'est pas l'origine ou la destination de la commodité k . En effet, lorsqu'une commodité "traverse" un noeud, $b_i^k = 0$ ce qui veut dire que la contrainte (2.7) est nulle, chaque opérande de la soustraction doit avoir la même valeur, on a donc bien le flot entrant égal au flot sortant.

Remarquons que le (TSP1) est bien un "price setting problem" : donnons un ordre arbitraire aux arcs de \mathcal{A} , de \mathcal{B} et aux noeuds de \mathcal{N} . Définissons :

- t , le vecteur de dimension $1 \times |\mathcal{A}|$ de composantes t_i qui est la taxe sur le $i^{\text{ème}}$ arc de \mathcal{A} , $i = 1, \dots, |\mathcal{A}|$.
- x , la matrice de dimension $|\mathcal{A}| \times |K|$ où les entrées sont x_i^j qui est le nombre d'utilisateurs de la $j^{\text{ème}}$ commodité qui utilisent le $i^{\text{ème}}$ arc de \mathcal{A} , pour $i = 1, \dots, |\mathcal{A}|$, $j = 1, \dots, |K|$.
- y , la matrice de dimension $|\mathcal{B}| \times |K|$ où les entrées sont y_i^j qui est le nombre d'utilisateurs de la $j^{\text{ème}}$ commodité qui utilisent le $i^{\text{ème}}$ arc de \mathcal{B} , pour $i = 1, \dots, |\mathcal{B}|$, $j = 1, \dots, |K|$.
- c , le vecteur de dimension $1 \times |\mathcal{A}|$ de composantes c_i , où c_i est le coût du $i^{\text{ème}}$ arc de \mathcal{A} , $i = 1, \dots, |\mathcal{A}|$.
- d , le vecteur de dimension $1 \times |\mathcal{B}|$ de composantes d_i , où d_i est le coût du $i^{\text{ème}}$ arc de \mathcal{B} , $i = 1, \dots, |\mathcal{B}|$.
- A , la matrice d'incidence pour les arcs tarifables c'est-à-dire que A est une matrice de dimension $|\mathcal{N}| \times |\mathcal{A}|$ où l'élément A_{ij} , pour $i = 1, \dots, |\mathcal{N}|$, $j = 1, \dots, |\mathcal{A}|$, est défini de la façon suivante :

$$A_{ij} = \begin{cases} 1 & \text{si le noeud } i \text{ est l'origine de l'arc } j \\ -1 & \text{si le noeud } i \text{ est l'extrémité de l'arc } j \\ 0 & \text{sinon} \end{cases}$$

- B , la matrice d'incidence pour les arcs non tarifables c'est-à-dire que B est une matrice $|\mathcal{N}| \times |\mathcal{B}|$ où l'élément B_{ij} , pour $i = 1, \dots, |\mathcal{N}|$, $j = 1, \dots, |\mathcal{B}|$ est défini de la façon suivante :

$$B_{ij} = \begin{cases} 1 & \text{si le noeud } i \text{ est l'origine de l'arc } j \\ -1 & \text{si le noeud } i \text{ est l'extrémité de l'arc } j \\ 0 & \text{sinon} \end{cases}$$

- b , la matrice de dimension $|\mathcal{N}| \times K$ où les entrées sont b_i^j , pour $i = 1, \dots, |\mathcal{N}|$, $j = 1, \dots, K$.
- x^i , le vecteur colonne composé des éléments de la $i^{\text{ème}}$ colonne de x , $i = 1, \dots, |K|$.
- y^i , le vecteur colonne composé des éléments de la $i^{\text{ème}}$ colonne de y , $i = 1, \dots, |K|$.

- b^i , le vecteur colonne composé des éléments de la $i^{\text{ème}}$ colonne de b , $i = 1, \dots, |K|$.

Dès lors le (TSP1) se reformule comme suit :

$$\begin{aligned} & \max_t tx \\ & \min_{x,y} (c+t)x + dy \\ & \text{s.c. } Ax + By = b \\ & \quad x, y \geq 0, \end{aligned}$$

ce qui est la forme d'un "price setting problem".

2.2 Programme bilinéaire à un niveau

Reprenons la formulation (2.1) - (2.4). Nous nous basons sur les travaux de Labbé et al [12] pour reformuler le Price Setting Problem sous une forme bilinéaire à un niveau.

Nous utilisons le Price Setting Problem pour illustrer la transformation d'un programme bilinéaire biniveau à un programme bilinéaire à un niveau. Ce type de formulation permet de faciliter la résolution du problème.

Rappelons rapidement comment nous passons d'un primal à un dual.

Soit le programme linéaire primal suivant :

$$\begin{aligned} & \min c_1x + c_2y \\ & \text{s.c. } Ax + By = b \end{aligned}$$

le dual de ce programme est :

$$\begin{aligned} & \max \lambda b \\ & \text{s.c. } \lambda A \leq c_1 \\ & \quad \lambda B \leq c_2 \end{aligned}$$

Notons que si le vecteur de taxe t est fixé, le deuxième niveau devient linéaire. Nous pouvons donc le remplacer par ses contraintes d'optimalité primales-duales. Prenons le deuxième niveau :

$$\begin{aligned} & \min_{x,y} (c+t)x + dy \\ & \text{s.c. } Ax + By = b \\ & \quad x, y \geq 0. \end{aligned}$$

Si nous nous référons à l'exemple précédent, $(c+t)$ correspond à c_1 et d à c_2 .

Passons au dual :

$$\max_{\lambda} \quad \lambda b \tag{2.9}$$

$$\text{s.c. } \lambda A \leq c + t \tag{2.10}$$

$$\lambda B \leq d \tag{2.11}$$

On trouve les contraintes de complémentarité du programme dual :

$$\begin{aligned}(c + t - \lambda A)x &= 0 \\ (d - \lambda B)y &= 0.\end{aligned}$$

On obtient donc le programme bilinéaire à un niveau suivant :

$$\max_{t,x,y,\lambda} \quad tx \quad (2.12)$$

$$\text{s.c.} \quad Ax + By = b \quad (2.13)$$

$$x, y \geq 0 \quad (2.14)$$

$$\lambda A \leq c + t \quad (2.15)$$

$$\lambda B \leq d \quad (2.16)$$

$$(c + t - \lambda A)x = 0 \quad (2.17)$$

$$(d - \lambda B)y = 0 \quad (2.18)$$

Chapitre 3

Le Toll Setting Problem

Ce chapitre reprend des formulations existantes du "Toll Setting Problem" (introduit au chapitre précédent à la sous-section 2.1.2) que nous développerons en détails.

Rappelons rapidement le principe : le Toll Setting Problem est un problème de péage où une entreprise veut taxer des tronçons routiers qu'elle possède dans le but de maximiser ses revenus tout en connaissant les réactions des utilisateurs qui veulent minimiser le coût de leurs trajets. Pour mieux visualiser le problème et en comprendre la difficulté, nous développerons trois exemples simples.

3.1 Exemples

Les exemples développés ci-après sont issus de Labbé et al. [12] et S. Dewez [8].

3.1.1 Exemple 1 : borne supérieure de la taxe totale du meneur non atteinte

Considérons le réseau représenté à la Figure 3.1 et supposons que les utilisateurs vont du noeud 1 au noeud 5. A chaque arc est associé un coût. Les arcs en pointillés représentent les arcs appartenant à une compagnie (le meneur), ces arcs sont donc sujets à une taxation.

Pour aller du noeud 1 au noeud 5, l'utilisateur (le suiveur) peut choisir entre plusieurs chemins.

Il va choisir le chemin le moins onéreux. L'entreprise va devoir assigner des taxes aux arcs (2,3) et (4,5) dans le cas du réseau ci-dessous (Figure 3.1).

Nous pouvons trouver une borne supérieure pour le revenu du meneur. L'utilisateur ne paiera jamais plus de 22, qui est le coût du chemin non tarifable le plus court depuis le noeud 1 jusqu'au noeud 5 (sans utiliser d'arcs tarifables) :1-3-5.

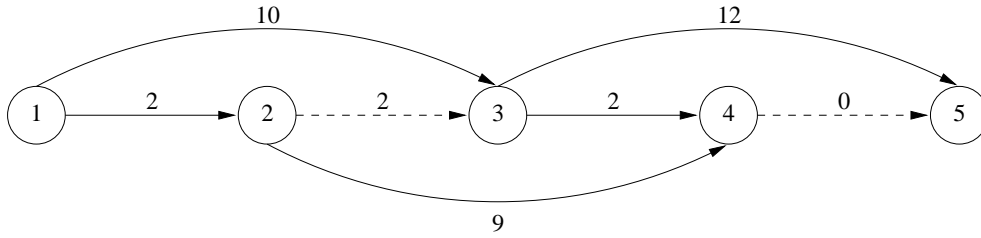


FIG. 3.1 – Exemple d’un réseau où la borne supérieure ne sera jamais atteinte.

Si le meneur met toutes ses taxes à zéro, l’utilisateur choisira alors le chemin 1-2-3-4-5 qui a un coût de 6.

Donc, la taxation totale maximale que peut mettre le meneur est 16 : $22 - 6 = 16$. Malheureusement, cette borne supérieure n’est pas toujours atteinte. Dans notre exemple, la solution optimale consiste à assigner une taxation de 5 sur l’arc (2,3) et une de 10 sur l’arc (4,5), ce qui donnera un revenu de 15.

En effet, l’utilisateur a le choix entre le chemin non tarifable : 1-3-5 qui a un coût de 22 et des chemins tarifables : 1-2-3-4-5, 1-2-3-5, 1-3-4-5 et 1-2-4-5 qui ont respectivement un coût de $6 + t_{23} + t_{45}$, $16 + t_{23}$, $12 + t_{45}$ et $11 + t_{45}$, où les t_{ij} représentent la taxe sur l’arc (i, j) . L’arc tarifable (2,3) sera utilisé seulement si le coût du sous-chemin 1-2-3-4 est inférieur ou égal au coût des sous-chemins non tarifables 1-2-4 et 1-3-4. Donc la valeur de t_{23} est fixée à $5 = 11 - 6$. D’un autre côté, l’arc tarifable (4,5) sera utilisé si le coût du sous chemin 3-4-5 est inférieur ou égal au coût du chemin non tarifable 3-5, nous fixons donc la valeur à 10.

3.1.2 Exemple 2 : taxes négatives

Notre prochain exemple concerne les bornes inférieures sur les taxes qui, nous le verrons grâce à cet exemple, peuvent être négatives.

Considérons le réseau de la Figure 3.2.

Ici nous avons deux utilisateurs : l’un qui part du noeud 1 jusqu’au noeud 2 et l’autre qui part du noeud 3 jusqu’au noeud 4.

Le premier a le choix entre le chemin 1-2 et 1-5-6-2. Si nous fixons la taxe de l’arc (5,6) à 5 ou moins, il prendra le chemin 1-5-6-2.

Le second utilisateur a le choix entre le chemin 3-4 et 3-5-6-4. Si $t_{56} + t_{64} \leq 3$ il choisira le chemin 3-5-6-4.

La solution optimale du revenu est obtenue pour $t_{56} = 5$ et $t_{64} = -2$. La taxe sur l’arc (5,6) est compensée par la taxe négative sur l’arc (6,4) pour le second utilisateur.

$$\begin{aligned} \text{utilisateur 1} & : t_{56} \leq 5 \\ \text{utilisateur 2} & : t_{56} + t_{64} \leq 3. \end{aligned}$$

nous considérerons dans le reste de ce mémoire que les taxes ne prennent jamais de valeurs négatives.

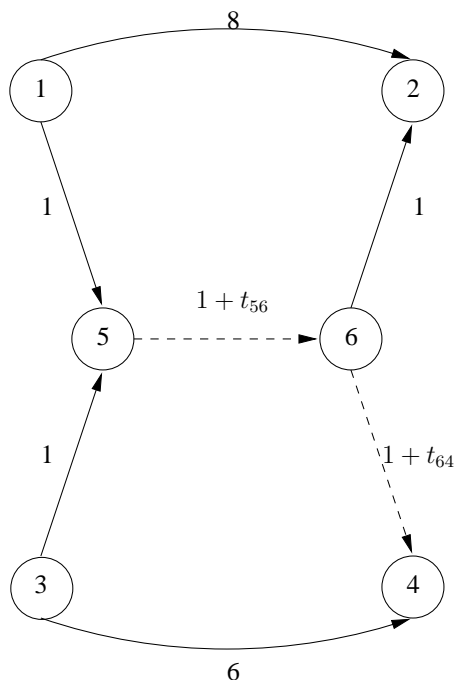


FIG. 3.2 – Exemple d'un réseau avec taxes négatives.

3.1.3 Exemple 3 : forme de la fonction revenu

Maintenant analysons la forme de la fonction revenu pour le réseau représenté par la Figure 3.3. Supposons qu'il existe deux commodités.

La première est composée de deux utilisateurs allant du noeud 1 au noeud 2 (on la représente par (1-2)) et la seconde est composée de quatre utilisateurs allant du noeud 3 au noeud 4 (on la représente par (3-4)).

Comme il n'existe qu'un seul arc tarifable, l'arc t_{56} , intéressons-nous plus en détails à celui-ci. En effet, pour trouver la forme de la fonction revenu ainsi que sa valeur optimale, il suffit d'examiner le problème selon les différentes valeurs possibles pour t_{56} :

1. Tant que la taxe sur l'arc (5,6) n'excède pas 2, les deux commodités vont utiliser cet arc.
Donc, pour $t_{56} \leq 2$ le revenu du meneur (sur cet arc) est une fonction linéaire en t_{56} ($4 * t_{56} + 2 * t_{56}$) et il est maximal pour $t_{56} = 2$ où il vaut 12 ($= 4 * 2 + 2 * 2$).
2. Lorsque $t_{56} \in (2, 5]$, les utilisateurs de la seconde commodité vont emprunter l'arc non tarifable (3,4) tandis que les utilisateurs de la première commodité continueront de l'utiliser. Donc pour $t_{56} \in (2, 5]$, le revenu du meneur est une fonction linéaire en t_{56} ($2 * t_{56}$) qui est maximal en $t_{56} = 5$ et est égal à 10 ($= 2 * 5$).
3. Finalement, pour $t_{56} > 5$, aucune des commodités ne va utiliser l'arc tari-

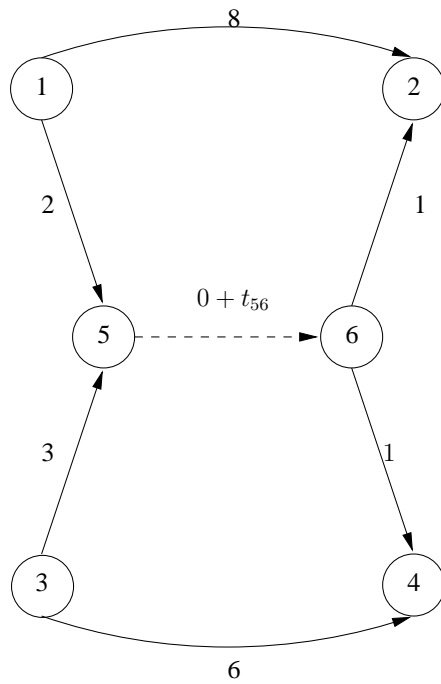


FIG. 3.3 – Exemple général pour le choix d'un arc tarifable.

fable (5,6) et le revenu est égal à 0.

Clairement, la solution optimale est obtenue pour $t_{56} = 2$ et la Figure 3.4 nous donne la forme de la fonction revenu du meneur.

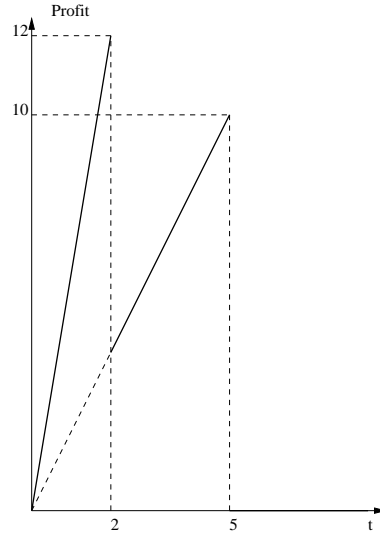


FIG. 3.4 – Représentation de la fonction objective du meneur

3.2 Formulations

Repartons de la formulation TSP1 du Chapitre 2

$$\begin{aligned}
 (\text{TSP1}) \quad & \max_{x,y,t} \sum_{a \in \mathcal{A}} t_a \sum_{k \in K} x_a^k \\
 & \min_{x,y} \sum_{k \in K} \left(\sum_{a \in \mathcal{A}} (c_a + t_a) x_a^k + \sum_{a \in \mathcal{B}} d_a y_a^k \right) \\
 \text{s.c.} \quad & \sum_{a \in i^+} (x_a^k + y_a^k) - \sum_{a \in i^-} (x_a^k + y_a^k) = b_i^k \quad \forall k \in K, \quad \forall i \in \mathcal{N} \\
 & x_a^k, y_a^k \geq 0 \quad \forall k \in K, \forall a \in \mathcal{A} \cup \mathcal{B}
 \end{aligned}$$

3.2.1 Illustration

Considérons le réseau de la Figure 3.5 et supposons qu'il y ait deux commodités. La première est composée de 4 utilisateurs allant du noeud 1 au noeud 3 et la deuxième est composée de 2 utilisateurs allant du noeud 4 au noeud 6.

On a donc :

$$\begin{aligned}
 (x^1)^t &= (x_{15}^1, x_{23}^1, x_{45}^1) \\
 (x^2)^t &= (x_{15}^2, x_{23}^2, x_{45}^2) \\
 (y^1)^t &= (y_{12}^1, y_{13}^1, y_{14}^1, y_{36}^1, y_{42}^1, y_{46}^1, y_{52}^1, y_{56}^1) \\
 (y^2)^t &= (y_{12}^2, y_{13}^2, y_{14}^2, y_{36}^2, y_{42}^2, y_{46}^2, y_{52}^2, y_{56}^2) \\
 t &= (t_{15}, t_{23}, t_{45})
 \end{aligned}$$

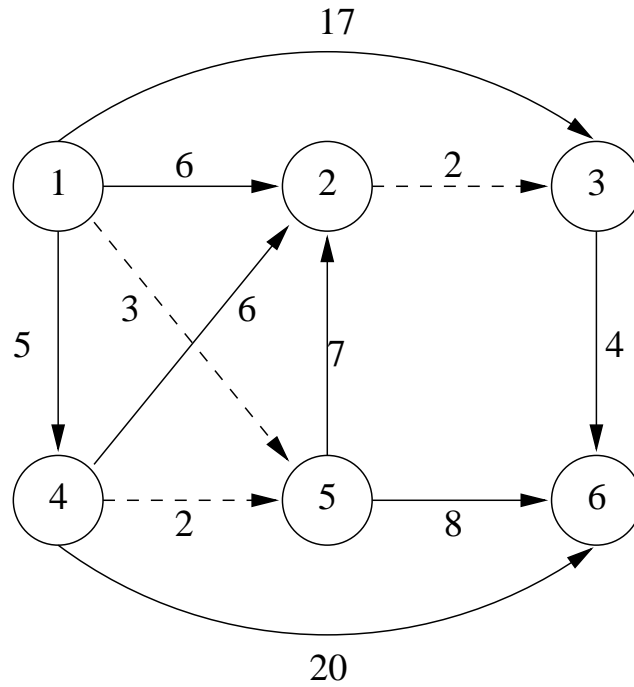


FIG. 3.5 – Exemple avec deux commodités

Les vecteurs de coût fixés associés à x et y sont respectivement :

$$c = (3, 2, 2)$$

$$d = (6, 17, 5, 4, 6, 20, 7, 8)$$

Les demandes des deux commodités sont respectivement :

$$n^1 = 4$$

$$n^2 = 2.$$

Les vecteurs des demandes b^k sont respectivement :

$$(b^1)^t = (4, 0, -4, 0, 0, 0)$$

$$(b^2)^t = (0, 0, 0, 2, 0, -2).$$

Rappelons que la valeur de la $j^{\text{ème}}$ composante du vecteur b^i est la différence entre les flots entrant et sortant du noeud j . Prenons par exemple b^1 , il correspond à la commodité 1 et imaginons que cette commodité passe par le noeud intermédiaire 2 son chemin est alors 1-2-3 :

noeud 1 flot entrant = 0 et flot sortant = 4 (car origine) : $4 - 0 = 4$.

noeud 2 flot entrant = 4 et flot sortant = 4 (car noeud de "passage") : $4 - 4 = 0$.

noeud 3 flot entrant = 4 et flot sortant = 0 (car destination) : $0 - 4 = -4$.

On obtient donc bien les composantes du vecteur b^1 .

3.2.2 Reformulation booléenne.

Précédemment, les variables x_a^k et y_a^k prenaient des valeurs entières. A présent, nous allons passer à une formulation où les variables x_a^k et y_a^k sont booléennes. L'ancienne variable x_a^k est maintenant équivalente à $x_a^k n^k$. Pour cette raison, nous introduisons la demande (n^k) dans la fonction objective :

$$\max_t \sum_{a \in \mathcal{A}} t_a \sum_{k \in K} n^k x_a^k.$$

Le vecteur b^k devient donc un vecteur de booléens et les variables x_a^k et y_a^k prendront les valeurs 0 ou 1. Si x_a^k (resp. y_a^k) prend la valeur 1, tous les utilisateurs de la commodité k passeront par l'arc tarifable (resp. non tarifable) a , sinon aucun utilisateur de la commodité k n'empruntera cet arc.

Le but du meneur est de maximiser son revenu et cette quantité n'augmentera que si les utilisateurs passent par les arcs tarifables. Par exemple pour l'arc (1,5) et la commodité k , on va obtenir un revenu de $t_{15} x_{15}^k n^k$. Nous voyons donc clairement que si les utilisateurs n'utilisent pas cet arc, le revenu est égal à 0.

Récrivons le problème sous forme vectorielle (ce qui allège la formulation) :

$$\max_t \quad t \sum_{k \in K} n^k x^k \quad (3.1)$$

$$\min_{x,y} \quad \sum_{k \in K} \left((c + t)x^k + dy^k \right) \quad (3.2)$$

$$\text{s.c.} \quad Ax^k + By^k = b^k \quad \forall k \in K \quad (3.3)$$

$$x^k, y^k \in \{0, 1\} \quad \forall k \in K \quad (3.4)$$

$$t \geq 0. \quad (3.5)$$

où les matrices A et B sont les mêmes que celles décrites ci-dessus et (3.3) représente toujours la conservation de flot.

Avant de continuer nous avons besoin de faire trois hypothèses :

1. Nous supposons que si nous avons plusieurs chemins avec le même coût mais avec des revenus différents pour le meneur, les utilisateurs choisiront toujours le plus avantageux pour le meneur.
2. Pour éviter les solutions triviales menant à un revenu infini, nous supposons qu'il existe toujours un chemin composé d'arcs non tarifables de chaque origine jusqu'à chaque destination. En d'autres mots, nous supposons que pour chaque commodité k , l'ensemble des solutions $\{y^k \geq 0 : By^k = b^k\}$ est non vide.

3. Nous supposons que le polyèdre défini par les contraintes $\{x^k, y^k \geq 0 : Ax^k + By^k = b^k\}$ est borné et non vide.

Nous reconnaissons la forme d'un programme biniveau. En plus, le terme tx^k rend ce programme non-linéaire. Il faut donc éliminer cette non linéarité. Pour ce faire, on transforme d'abord le programme en un problème à un niveau, comme fait dans Labbé et al [11].

3.2.3 Passage d'une formulation bilinéaire à deux niveaux à une formulation bilinéaire à un niveau.

Remarquons que pour un coût fixé de la taxe t , le second niveau est un programme linéaire. En effet, si t est fixé, il ne s'agit plus que d'un problème de plus court chemin : l'utilisateur connaît le coût de chaque arc et il lui suffit donc de trouver le plus court chemin pour avoir le plus petit coût. Nous pouvons remplacer le deuxième niveau par les conditions d'optimalité primales-duales. Pour obtenir le dual de (3.1) - (3.5), nous appliquons la même méthode que celle expliquée à la Section 2.2.

Nous obtenons le programme dual suivant :

$$\begin{aligned} \max_{\lambda} \quad & \sum_{k \in K} \lambda^k b^k \\ \text{s.c.} \quad & \lambda^k A \leq c + t \quad \forall k \in K \\ & \lambda^k B \leq d \quad \forall k \in K \end{aligned}$$

Les contraintes de complémentarité du dual sont :

$$(c + t - \lambda^k A)x^k = 0 \quad \forall k \in K$$

et

$$(d - \lambda^k B)y^k = 0 \quad \forall k \in K,$$

où λ est le vecteur ligne $1 \times |K|^2$ tel que $\lambda = (\lambda^1 \dots \lambda^{|K|})$ où λ^k est un vecteur de $|K|$ composantes. Dans la suite, nous noterons $\lambda_j^k \quad \forall k = 1 \dots |K|$, la $j^{\text{ème}}$ composante de λ^k pour $j = 1 \dots |K|$.

Les variables duales (λ^k) représentent le coût des chemins $o_k - d_k$ pour la commodité k après avoir placé les taxes : λ_j^k est le coût du chemin $o_k - j$ pour la commodité k après avoir placé les taxes.

Les contraintes de complémentarités sont importantes car elles imposent plusieurs choses.

Considérons la première contrainte de complémentarité et prenons un arc tarifiable (i, j) . Il y a deux cas possibles, soit l'utilisateur passe par l'arc soit il n'y passe pas.

Dans le premier cas, x^k est différent de 0 et cela implique que : $\lambda_j^k = \lambda_i^k + c_{ij} + t_{ij}$. En effet, le coût du chemin dans la solution optimale (c'est à dire le coût minimal) pour aller de o_k à j est égal au coût du chemin pour aller en i (λ_i^k) plus le coût de l'arc (i, j) (c_{ij}) plus la taxe mise sur cet arc (t_{ij}). Ce qui revient à dire que : $\lambda_j^k - \lambda_i^k = c_{ij} + t_{ij}$ et sous forme vectorielle : $c + t = \lambda^k A$. La première contrainte de complémentarité impose donc cette égalité.

Dans le deuxième cas ($x_{ij}^k = 0$), cela implique que $\lambda_j^k < \lambda_i^k + c_{ij} + t_{ij}$ sinon l'utilisateur passerait par l'arc tarifable alors qu'on sait que ce n'est pas le cas.

On peut faire le même type de raisonnement pour la deuxième contrainte de complémentarité.

Finalement on obtient :

$$(TSP2) \quad \max_{x,y,\lambda,t} t \sum_{k \in K} n^k x^k$$

$$\text{t.q } Ax^k + By^k = b^k \quad \forall k \in K \quad (3.6)$$

$$\lambda^k A \leq c + t \quad \forall k \in K \quad (3.7)$$

$$\lambda^k B \leq d \quad \forall k \in K \quad (3.8)$$

$$(c + t - \lambda^k A)x^k = 0 \quad \forall k \in K \quad (3.9)$$

$$(d - \lambda^k B)y^k = 0 \quad \forall k \in K \quad (3.10)$$

$$t \geq 0 \quad (3.11)$$

$$x^k, y^k \in \{0, 1\} \quad \forall k \in K \quad (3.12)$$

Les contraintes (3.9) et (3.10) sont non linéaires car elles contiennent le terme $\lambda^k x^k$ ou $\lambda^k y^k$. Grâce à la dualité forte (c'est à dire que les valeurs optimales des fonctions objectives du primal et du dual sont égales), on a :

$$(c + t)x^k + dy^k = \lambda^k b \quad \forall k \in K \quad (3.13)$$

qui est équivalent aux contraintes (3.9) et (3.10) pour TSP2 car $\lambda^k Ax^k + \lambda^k By^k = \lambda^k b$.

La prochaine étape consiste à linéariser les fonctions bilinéaires en tx^k qui apparaissent dans la formulation. Pour faciliter les choses, nous formulons le problème sous une forme étendue. Nous remplaçons (3.9) et (3.10) par la nouvelle

contrainte (3.13) et nous obtenons le programme bilinéaire à un niveau suivant :

$$\max_{x,y,\lambda,t} \sum_{k \in K} n^k \sum_{(i,j) \in \mathcal{A}} t_{ij} x_{ij}^k \quad (3.14)$$

$$\text{t.q.} \quad \sum_{i:(i,j) \in \mathcal{A}} x_{ij}^k + \sum_{i:(i,j) \in \mathcal{B}} y_{ij}^k - \sum_{l:(j,l) \in \mathcal{A}} x_{jl}^k - \sum_{l:(j,l) \in \mathcal{B}} y_{jl}^k = \begin{cases} 1 & \text{si } j = o_k \\ -1 & \text{si } j = d_k \\ 0 & \text{sinon} \end{cases} \quad (3.15)$$

$$\forall k \in K, j \in N \quad (3.16)$$

$$\lambda_j^k - \lambda_i^k \leq c_{ij} + t_{ij} \quad \forall k \in K, (i,j) \in \mathcal{A} \quad (3.17)$$

$$\lambda_j^k - \lambda_i^k \leq d_{ij} \quad \forall k \in K, (i,j) \in \mathcal{B} \quad (3.18)$$

$$\sum_{(i,j) \in \mathcal{A}} (c_{ij} + t_{ij}) x_{ij}^k + \sum_{(i,j) \in \mathcal{B}} d_{ij} y_{ij}^k = \lambda_{d_k}^k - \lambda_{o_k}^k \quad \forall k \in K \quad (3.19)$$

$$t_{ij} \geq 0 \quad \forall (i,j) \in \mathcal{A}$$

$$x_{ij}^k, y_{ij}^k \in \{0, 1\} \quad \forall k \in K, (i,j) \in \mathcal{A} \cup \mathcal{B}$$

Cette formulation est bien équivalente à la précédente, la contrainte (3.15) est bien la contrainte concernant la conservation des flots, les contraintes (3.17) et (3.18) sont bien équivalentes aux contraintes du dual et (3.19) est bien le coût total d'un trajet pour la commodité k à partir de l'origine jusqu'à la destination.

3.2.4 Linéarisation

Le problème est que la formulation précédente est toujours non linéaire puisque la fonction objective (3.14) et la contrainte (3.19) contiennent le terme bilinéaire $t_{ij} x_{ij}^k$. Nous linéarisons ces deux fonctions par l'introduction de nouvelles variables $t_{ij}^k = t_{ij} x_{ij}^k$ pour toutes les commodités k .

Cette nouvelle variable est le revenu que va réellement percevoir le meneur pour l'arc (i,j) et la commodité k .

En effet, si le flot sur l'arc (i,j) est égal à 0 pour cette commodité, $t_{ij}^k = 0$ même si la valeur de la taxe sur l'arc est strictement positive. Si le flot est égal à 1, le revenu est égal au niveau de la taxe. Nous devons donc considérer deux cas :

1. Quand le flot sur l'arc (i,j) pour la commodité k est égal à 0 ($x_{ij}^k = 0$), nous voulons que t_{ij}^k soit égal à zéro, nous avons la contrainte suivante :

$$t_{ij}^k \leq M_{ij}^k x_{ij}^k,$$

où M_{ij}^k est une constante dépendant de l'arc (i,j) et de la commodité k . Cette constante doit être assez grande pour s'assurer que la contrainte $t_{ij}^k \leq M_{ij}^k x_{ij}^k$ n'est pas violée quand $x_{ij}^k = 1$, ce qui implique que $t_{ij}^k \leq M_{ij}^k$.

2. Quand le flot sur l'arc (i,j) pour la commodité k est égal à 1 ($x_{ij}^k = 1$), nous voulons que t_{ij}^k soit égal à t_{ij} , nous avons la contrainte suivante :

$$t_{ij} - t_{ij}^k \leq N_{ij} (1 - x_{ij}^k),$$

où N_{ij} est une constante dépendante uniquement de l'arc (i, j) . Cette constante doit être assez grande pour s'assurer que la contrainte $t_{ij} - t'_{ij} \leq N_{ij}(1 - x_{ij}^k)$ n'est pas violée quand $x_{ij}^k = 0$, ce qui implique que $t_{ij} \leq N_{ij}^k$.

Finalement nous obtenons un programme linéaire :

$$\max_{x,y,\lambda,t} \sum_{k \in K} n^k \sum_{(i,j) \in \mathcal{A}} t'_{ij} \quad (3.20)$$

$$\text{t.q.} \quad \sum_{i:(i,j) \in \mathcal{A}} x_{ij}^k + \sum_{i:(i,j) \in \mathcal{B}} y_{ij}^k - \sum_{l:(j,l) \in \mathcal{A}} x_{jl}^k - \sum_{l:(j,l) \in \mathcal{B}} y_{jl}^k = \begin{cases} 1 & \text{si } j = o_k \\ -1 & \text{si } j = d_k \\ 0 & \text{sinon} \end{cases} \quad (3.21)$$

$$\forall k \in K, j \in N \quad (3.22)$$

$$\lambda_j^k - \lambda_i^k \leq c_{ij} + t_{ij} \quad \forall k \in K, (i, j) \in \mathcal{A} \quad (3.23)$$

$$\lambda_j^k - \lambda_i^k \leq d_{ij} \quad \forall k \in K, (i, j) \in \mathcal{B} \quad (3.24)$$

$$\sum_{(i,j) \in \mathcal{A}} (c_{ij} x_{ij}^k + t'_{ij}) + \sum_{(i,j) \in \mathcal{B}} d_{ij} y_{ij}^k = \lambda_{d_k}^k - \lambda_{o_k}^k \quad \forall k \in K \quad (3.25)$$

$$t'_{ij} \leq M_{ij}^k x_{ij}^k \quad \forall k \in K, (i, j) \in \mathcal{A} \quad (3.26)$$

$$t_{ij} - t'_{ij} \leq N_{ij}(1 - x_{ij}^k) \quad \forall k \in K, (i, j) \in \mathcal{A} \quad (3.27)$$

$$t'_{ij} \leq t_{ij} \quad \forall k \in K, (i, j) \in \mathcal{A} \quad (3.28)$$

$$t'_{ij}, t_{ij} \geq 0 \quad \forall k \in K, (i, j) \in \mathcal{A} \quad (3.29)$$

$$x_{ij}^k \in \{0, 1\} \quad \forall k \in K, (i, j) \in \mathcal{A} \quad (3.30)$$

$$y_{ij}^k \in \{0, 1\} \quad \forall k \in K, (i, j) \in \mathcal{B} \quad (3.31)$$

Afin de renforcer la formulation nous allons calculer les M_{ij}^k et N_{ij} , ce calcul fait l'objet de la section suivante.

3.3 Choix des constantes M_{ij}^k et N_{ij}

On peut bien entendu attribuer des valeurs arbitrairement grandes aux constantes M_{ij}^k et N_{ij} mais il serait plus judicieux d'avoir des valeurs les plus petites possibles pour ces constantes (tout en respectant les contraintes).

Nous allons développer 4 bornes différentes pour M_{ij}^k comme fait par S. Dewez [8].

Proposition 3.1 *Pour l'arc tarifable (i, j) et la commodité k ,*

$$M_{ij}^k = \max \left\{ 0, \min \{ p_{ij} - c_{ij}, \bar{\lambda}_j^k - \underline{\lambda}_i^k - c_{ij}, \bar{\lambda}_{d_k}^k - (\underline{\lambda}_i^k + c_{ij} + \underline{s}_j^k), \bar{s}_i^k - \underline{s}_j^k - c_{ij} \} \right\}$$

est une constante valide pour la formulation du Toll Setting Problem (3.20)-(3.31) où

- p_{ij} est la valeur du plus court chemin, sans emprunter d'arcs tarifables, du noeud i au noeud j .
- $\bar{\lambda}_i^k$ est la valeur du plus court chemin pour la commodité k , sans emprunter d'arcs tarifables, de l'origine o_k au noeud i .
- $\underline{\lambda}_i^k$ est la valeur du plus court chemin pour la commodité k de l'origine o_k au noeud i où toutes les taxes sont mises à zéro.
- \bar{s}_i^k est la valeur du plus court chemin pour une commodité k , sans emprunter d'arcs tarifables, du noeud i à la destination d_k .
- \underline{s}_i^k est la valeur du plus court chemin pour une commodité k du noeud i à la destination d_k où toutes les taxes sont mises à zéro.

Preuve:

Grâce à la contrainte (3.26), nous pouvons conclure que $t_{ij}^k \leq M_{ij}^k$ lorsque $x_{ij}^k = 1$. M_{ij}^k constitue donc une borne supérieure pour la variable t_{ij}^k . Nous allons montrer que les quatre bornes de la proposition sont valides. Il va de soit que la plus petite borne des quatre est donc aussi valide.

La commodité k peut utiliser l'arc tarifable (i, j) seulement si le coût du chemin le plus court entre i et j , sans emprunter d'arcs tarifables, est plus élevé que le coût total de l'arc tarifable (i, j) .

Nous pouvons donc dire que l'arc (i, j) peut être utilisé par la commodité k si :

$$c_{ij} + t_{ij}^k \leq p_{ij}.$$

Le coût total de l'arc (i, j) est représenté par $c_{ij} + t_{ij}^k$. La première borne supérieure sur t_{ij}^k est donc :

$$M_{ij}^k = p_{ij} - c_{ij}.$$

On peut aussi voir que la commodité k n'utilisera l'arc tarifable (i, j) que si le coût du plus court chemin entre i et j , n'empruntant pas d'arcs tarifables, entre l'origine et j est plus élevé que le coût du chemin le moins cher, entre l'origine et j , passant par l'arc tarifable (i, j) .

Nous pouvons donc dire que l'arc (i, j) peut être utilisé par la commodité k si :

$$\underline{\lambda}_i^k + c_{ij} + t_{ij}^k \leq \bar{\lambda}_j^k.$$

Le chemin le moins cher jusqu'au noeud i est représenté par $\underline{\lambda}_i^k$. Si on y ajoute le coût total de l'arc (i, j) , on a bien le chemin le moins cher entre l'origine et j et passant par l'arc (i, j) pour la commodité k . La deuxième borne supérieure sur t_{ij}^k est donc :

$$M_{ij}^k = \bar{\lambda}_j^k - (\underline{\lambda}_i^k + c_{ij}).$$

Nous avons considéré, dans le cas précédent, le chemin depuis l'origine jusqu'au noeud j , mais nous pouvons aussi considérer le chemin depuis l'origine jusqu'à la destination. En effet, la commodité k n'utilisera l'arc tarifable (i, j) que si le coût du plus court chemin, n'empruntant pas d'arcs tarifables, entre l'origine o_k et la destination d_k est plus élevé que le coût du chemin le moins cher, entre l'origine o_k et la destination d_k , passant par l'arc tarifable (i, j) .

Nous pouvons donc dire que l'arc (i, j) sera utilisé par la commodité k si :

$$\underline{\lambda}_i^k + c_{ij} + t_{ij}^k + \underline{s}_j^k \leq \overline{\lambda}_{d_k}^k.$$

Comme précédemment, le chemin le moins cher entre l'origine et la destination est représenté par $\underline{\lambda}_i^k + c_{ij} + t_{ij}^k$. En ajoutant le coût du plus court chemin entre j et d_k où les taxes sont à zéro, nous avons bien le chemin le moins cher entre l'origine et la destination passant par l'arc (i, j) . La troisième borne supérieure sur t_{ij}^k est donc :

$$M_{ij}^k = \overline{\lambda}_{d_k}^k - (\underline{\lambda}_i^k + c_{ij} + \underline{s}_j^k).$$

Dernièrement, l'arc tarifable (i, j) sera utilisé par la commodité k seulement si le coût du plus court chemin, n'empruntant pas d'arcs tarifables, entre le noeud i et la destination d_k est plus élevé que le coût du chemin le moins cher entre i et d_k qui passe par l'arc tarifable (i, j) .

Nous pouvons dire que l'arc (i, j) sera utilisé par la commodité k si :

$$\underline{s}_j^k + c_{ij} + t_{ij}^k \leq \overline{s}_i^k$$

Le chemin le moins cher entre i et d_k passant par l'arc tarifable (i, j) et ce pour la commodité k est représenté par $\underline{s}_j^k + c_{ij} + t_{ij}^k$. La quatrième borne supérieure est donc :

$$M_{ij}^k = \overline{s}_i^k - (\underline{s}_j^k + c_{ij})$$

□

Aucune des bornes n'est dominée par les autres. Pour prouver ceci, il suffit de donner quatre exemples où chacune des bornes est tour à tour la borne inférieure parmi les quatre.

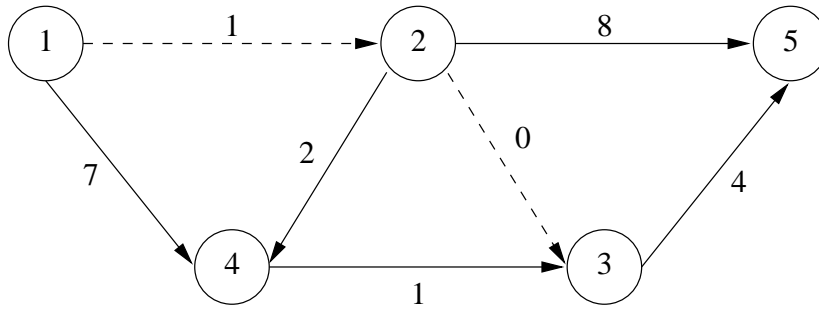


FIG. 3.6 – Exemple 1

Exemple 1 :

Considérons le réseau de la Figure 3.6. Supposons qu'il y a un utilisateur qui se rend du noeud 1 au noeud 5. Nous pouvons alors calculer la borne supérieure sur l'arc (2, 3) car l'utilisateur pourrait utiliser cet arc tarifable.

Pour la première borne, on calcule la différence entre le coût du plus court chemin, n'empruntant pas d'arcs tarifables, entre les noeuds 2 et 3 et le coût total de l'arc (2, 3) :

$$M_1 = p_{23} - c_{23} = 3 - 0 = 2$$

Pour la seconde borne, nous calculons la différence entre le coût du plus court chemin, n'empruntant pas d'arcs tarifables, entre les noeuds 1 et 3 et le coût du plus court chemin entre les noeuds 1 et 3, empruntant l'arc (2, 3) et où les taxes sont mises à zéro :

$$M_2 = \bar{\lambda}_3 - (\underline{\lambda}_2 + c_{23}) = 8 - (1 + 0) = 7$$

Pour la troisième borne, nous calculons la différence entre le coût du plus court chemin, n'empruntant pas d'arcs tarifables, entre les noeuds 1 et 5 et le coût du plus court chemin entre les noeuds 1 et 5, empruntant l'arc (2, 3) et où les taxes sont mises à zéro :

$$M_3 = \bar{\lambda}_5 - (\underline{\lambda}_2 + c_{23} + \underline{s}_3) = 10 - (2 + 0 + 4) = 4$$

Pour la quatrième borne, nous calculons la différence entre le coût du plus court chemin, n'empruntant pas d'arcs tarifables, entre les noeuds 2 et 5 et le coût du plus court chemin entre les noeuds 2 et 5, empruntant l'arc (2, 3) et où les taxes sont mises à zéro :

$$M_4 = \bar{s}_2 - (c_{23} + \underline{s}_3) = 8 - (0 + 4) = 4.$$

La borne M_1 est la meilleure borne parmi les quatre.

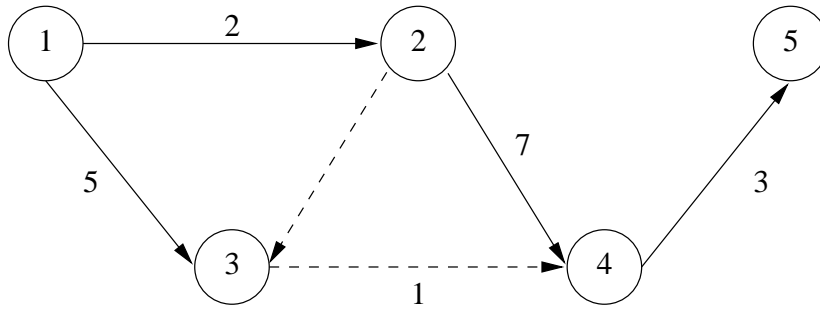


FIG. 3.7 – Exemple 2

Exemple 2 :

Considérons le réseau de la Figure 3.7. Supposons qu'il y a un utilisateur qui se rend du noeud 1 au noeud 5. Nous pouvons alors calculer la borne supérieure sur l'arc (2, 3) .

$$M_1 = \text{pas de chemin non tarifable entre 2 et 3} = \infty$$

$$M_2 = 5 - (2 + 0) = 3$$

$$M_3 = (2 + 7 + 3) - (2 + 0 + (1 + 3)) = 12 - 6 = 6$$

$$M_4 = (7 + 3) - 4 = 10 - 4 = 6$$

La borne M_2 est la meilleure borne parmi les quatre.

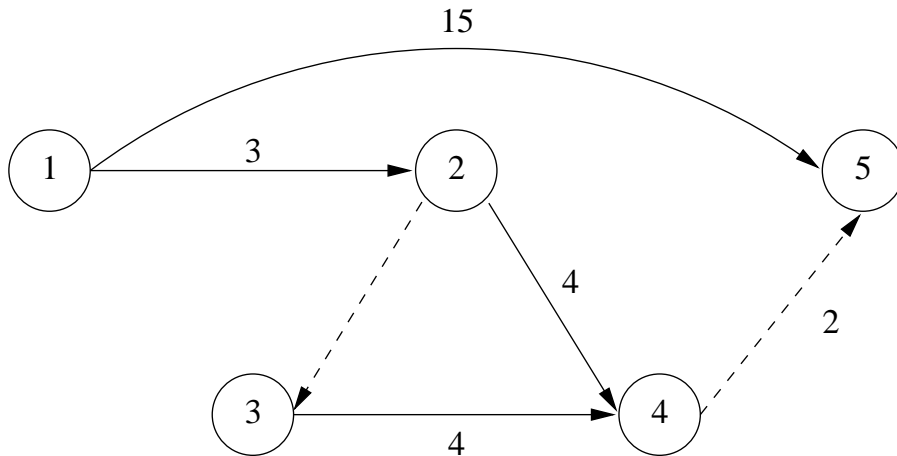


FIG. 3.8 – Exemple 3

Exemple 3 :

Considérons le réseau de la Figure 3.8. Supposons qu'il y a un utilisateur qui se rend du noeud 1 au noeud 5. Nous pouvons alors calculer la borne supérieure sur l'arc (2, 3) .

$$\begin{aligned} M_1 &= \text{pas de chemi non tarifable entre 2 et 3} = \infty \\ M_2 &= \text{pas de chemin sans arc tarifable entre 2 et 3} = \infty \\ M_3 &= 15 - 9 = 6 \\ M_4 &= \text{pas de chemin sans arc tarifable entre 2 et 5} = \infty \end{aligned}$$

La borne M_3 est la meilleure borne parmi les quatre.

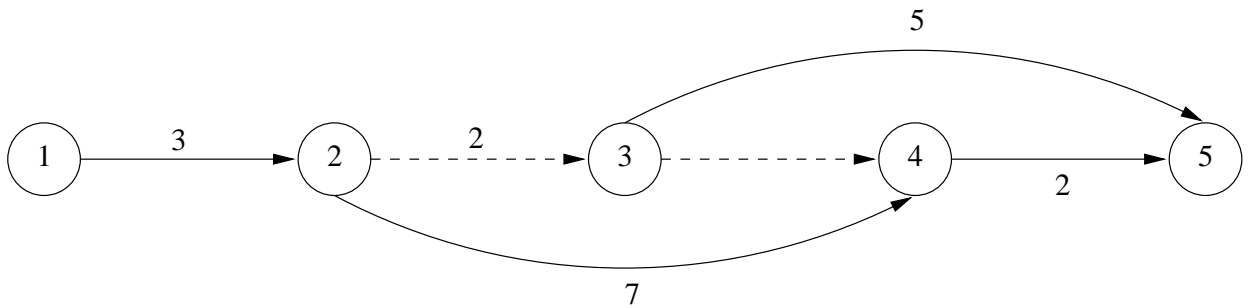


FIG. 3.9 – Exemple 4

Exemple 4 :

Considérons le réseau de la Figure 3.9. Supposons qu'il y a un utilisateur qui se rend du noeud 1 au noeud 5. Nous pouvons alors calculer la borne supérieure sur l'arc (3, 4) .

$$\begin{aligned} M_1 &= \text{pas d'arc non tarifable entre 3 et 4} = \infty \\ M_2 &= 10 - (3 + 2) = 10 - 5 = 5 \\ M_3 &= 12 - ((3 + 2) + 0 + 2) = 12 - 7 = 5 \\ M_4 &= 5 - (0 + 2) = 3 \end{aligned}$$

La borne M_4 est la meilleure borne parmi les quatre.

Nous avons une borne supérieure pour M_{ij}^k mais il faut encore trouver une valeur valide pour N_{ij} .

Proposition 3.2 *Pour tous les arcs tarifables (i, j),*

$$N_{ij} = \max_{k \in K} M_{ij}^k$$

est une constante valide pour la formulation du Toll Setting Problem (3.20)-(3.31).

Preuve:

Grâce à la contrainte (3.27), nous pouvons conclure que N_{ij} est une borne supérieure sur les taxes t_{ij} . On l'obtient très facilement : quand $x_{ij} = 0$ alors t_{ij}^k est égal à 0 par la contrainte (3.26). On a donc :

$$t_{ij} \leq N_{ij}$$

où N_{ij} est bien une borne supérieure sur la taxe t_{ij} . La taxe t_{ij} est la même pour toutes les commodités. Nous devons donc prendre en compte tout ce qui se passe avec toutes les commodités. La taxe t_{ij} est (par définition) au plus égale à la taxe t_{ij}^k la plus élevée parmi toutes les commodités k .

Une valeur valide pour N_{ij} est donc le maximum des M_{ij}^k sur toutes les commodités :

$$N_{ij} = \max_{k \in K} M_{ij}^k.$$

□

S. Dewez [8] a montré que l'utilisation de ces bornes à la place de bornes arbitrairement grandes permet d'améliorer considérablement les performances lors de la résolution du Toll Setting Problem selon la formulation (3.20)- (3.31).

Chapitre 4

Complexité du Toll Setting Problem

Nous allons prouver que le Toll Setting Problem est NP-Complet par réduction à partir de 3-SAT (preuves de la NP-Complétude de SAT et 3-SAT en Annexe). Nous allons y parvenir en suivant la méthode de Roch et Marcotte [15]. Savoir que le Toll Setting Problem est NP-Complet est une chose très importante pour le développement d'algorithmes.

Définition 4.1 Une formule propositionnelle ϕ est satisfaisable ssi il existe une fonction d'interprétation f qui rend vraie ϕ .

Définition 4.2 Une formule ϕ est en forme normale conjonctive si c'est une conjonction de disjonctions de littéraux.

Exemple de formule en forme normale conjonctive :

$$(x_1) \wedge (\neg x_2 \vee x_3) \wedge (\neg x_3)$$

où les $x_i \forall i = 1..3$ sont des littéraux. Nous avons ici trois clause $:(x_1)$, $(\neg x_2 \vee x_3)$ et $(\neg x_3)$. La formule ci-dessous est satisfaite pour $x_1 = True$ et $x_2 = x_3 = False$.

Dans cette preuve nous utilisons le problème 3-SAT pour prouver la NP-Complétude du Toll Setting Problem.

Définition 4.3 3SAT : satisfaisabilité des formules propositionnelles en forme normale conjonctive qui ont exactement 3 littéraux par clause

Théorème 4.0.1 (Roch et Marcotte [15]) Le Toll Setting Problem où tous les arcs taxés ne prennent pas de valeurs négatives est NP-Complet.

Preuve:

La preuve est faite en se basant sur un réseau ne contenant qu'une seule commodité. Premièrement nous décrivons le problème de décision.

Etant donné une instance du Toll Setting Problem et un revenu rationnel Π , existe-t-il un vecteur t tel que $tx \geq \Pi$ et que le couple (x, y) est une "réaction" optimale du niveau inférieur et ce par rapport à t . Nous pouvons bien entendu vérifier en temps polynomial si un vecteur t satisfait cette condition.

Maintenant nous allons montrer qu'il existe un problème $Q \in NP$ tel que Q peut être réduit au TSP. Nous allons, comme dit plus haut, utiliser 3SAT pour y parvenir.

Supposons que pour chaque clause nous construisons un sous-réseau contenant un arc taxé pour chaque littéral de la clause (cf figure 4.1). L'arc taxé correspondant au littéral l_{i1} dans la clause c est représenté par (a_{c1}, b_{c1}) et a un coût fixe de 0. Pour une clause c , o_c et d_c représentent respectivement l'origine et la destination du sous-réseau correspondant à cette même clause. Ces deux noeuds sont liés par un arc non taxé qui a un coût de 1. L'origine o_c d'une clause est liée à chaque origine des arcs taxés par un arc non taxé $(o_c, a_{ci}), i = 1, 2, 3$, d'un coût égal à 0. De façon similaire, il y a un arc non taxé entre chaque destination des arcs taxés, b_{ci} , et d_c .

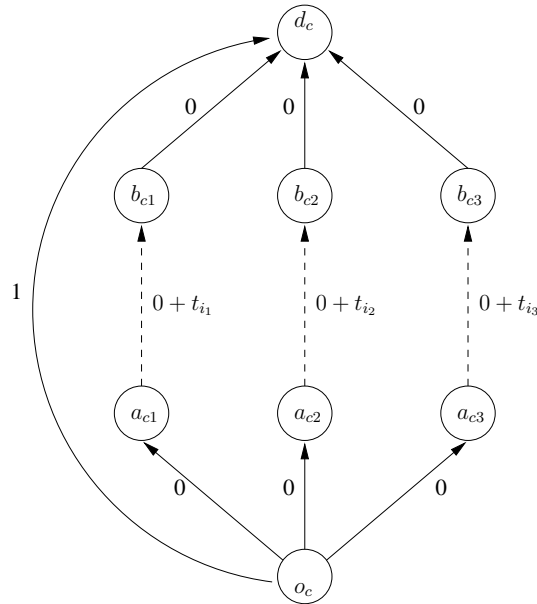


FIG. 4.1 – Représentation d'un triplet

3-SAT est réduit au TSP en utilisant l'idée suivante : si le chemin optimal passe par un arc taxé (i, j) de taxe t_{ij} , le littéral correspondant l_{ij} est Vrai (notons que si $l_{ij} = \bar{x}_k, x_k = \text{Faux}$, càd que la fonction d'interprétation qui évalue 3-SAT évalue à vrai l_{ij}).

Deux clauses consécutives c et $c+1$ sont liées par deux arcs dans les sous-réseaux correspondants : un arc non taxé (d_c, o_{c+1}) de coût 2 et un arc taxé (d_c, o_{c+1}) de coût 0.

Si F (la formule) est satisfaisable, au moins un littéral est Vrai par clause. Comme nous voulons un chemin optimal, l'utilisateur ne sélectionne qu'un arc taxé par sous-réseau. Nous voulons nous assurer que l'utilisateur ne passe pas par un arc dans un sous-réseau et par la négation de ce même arc dans un autre sous-réseau. En résumé, les variables ci et $c'j$ ne peuvent être vraies simultanément si l'une est la négation de l'autre, c'est à dire que le littéral représenté par l'arc correspondant à ci ne peut être vrai dans une clause et sa négation, représentée, elle, par l'arc correspondant $c'j$, aussi mais dans une autre clause. Cette valuation serait totalement impossible car on ne peut pas valuer à vrai une variable et sa négation. Pour éviter ce genre de situation, nous ajoutons un arc non taxé $(b_{ci}, a_{c'j})$ entre les arcs correspondant à une variable et sa négation. Nous mettons un coût de 1 à ce type d'arc. Nous verrons si un utilisateur utilise ce type d'arc car nous n'aurons pas la solution optimale (explications plus bas).

L'origine de la première clause est notée O et la destination de la dernière clause est notée D .

Le plus court chemin entre O et D composé d'arcs non taxés, est celui contenant tous les arcs allant de l'origine d'un sous-réseau à la destination de ce même sous-réseau (ces arcs ont un coût de 1) et contenant les arcs joignant deux sous-réseaux consécutifs. Donc le coût de ce chemin est égal à : $m + 2(m - 1) = 3m - 2$, où m est le nombre de clauses. En effet, nous utilisons au total m arcs internes, un par sous-réseau, de valeur 1 et $m - 1$ arcs liant deux sous-réseaux consécutifs de valeur 2.

Le coût du chemin le plus court entre O et D où les taxations sont égales à 0 est égal à 0.

$3m - 2$ est donc une borne supérieure du revenu.

Nous montrons que F est satisfaisable ssi le revenu optimal est égal à cette borne.

Supposons que le revenu est égal à $3m - 2$. Il est clair que le coût du chemin optimal quand la taxe des arcs est mise à 0 doit être égal à 0.

Pour obtenir ce chemin optimal, l'utilisateur doit choisir un arc taxé par sous-réseau, sur lequel nous avons mis une taxe de 1. Si nous mettons plus que 1, l'utilisateur utilisera l'arc non taxé.

Sur les autres arcs taxés du sous-réseau nous mettons un coût assez grand tel que nous sommes sûrs qu'ils ne seront pas utilisés.

Pour aller d'un sous-réseau à un autre, l'utilisateur utilisera l'arc taxé ayant un coût fixe de 0 et sur lequel nous mettons une taxe de 2.

Le coût d'un tel chemin est égal à $3m - 2$. C'est l'optimum car tous les autres chemins (de par cette construction) ont un coût de $3m - 2$ au minimum. De plus, un chemin optimal ne va jamais contenir une variable et sa négation. En effet, si c'était le cas, l'arc connectant deux sous-réseaux aurait un coût d'au plus 1 à la place de 2 (coût de l'arc reliant une variable et sa négation). Ceci constitue donc une contradiction avec le coût optimal du chemin entre O et D (il serait inférieur à $3m - 2$).

Donc le chemin optimal doit correspondre à une assignation consistante et F est satisfaisable (notons que si une variable ou sa négation n'apparaissent pas dans le chemin optimal, n'importe quelle valeur peut être assignée à cette variable). En effet, on a construit la fonction d'interprétation f telle que $f(p) = 1$ ssi l'arc correspondant au littéral c_i (et $c_i = p$) est utilisé dans le chemin optimal. On aura bien un littéral valué à vrai par clause, F est donc satisfaisable.

Si F est satisfaisable, au moins un littéral par clause est Vrai pour une assignation qui rend F Vrai. Considérons le chemin passant par ces littéraux. Si l'assignation est consistante, le chemin n'inclura pas simultanément une variable et sa négation et aucun arc inter-clause ne limite le revenu.

Donc la borne supérieure $3m - 2$ est atteinte par ce chemin.

□

Figure 4.2 représente le réseau pour la formule : $F = (x_1 \vee x_2 \vee \overline{x_3}) \wedge (\overline{x_2} \vee x_3 \vee \overline{x_4}) \wedge (\overline{x_1} \vee x_3 \vee x_4)$

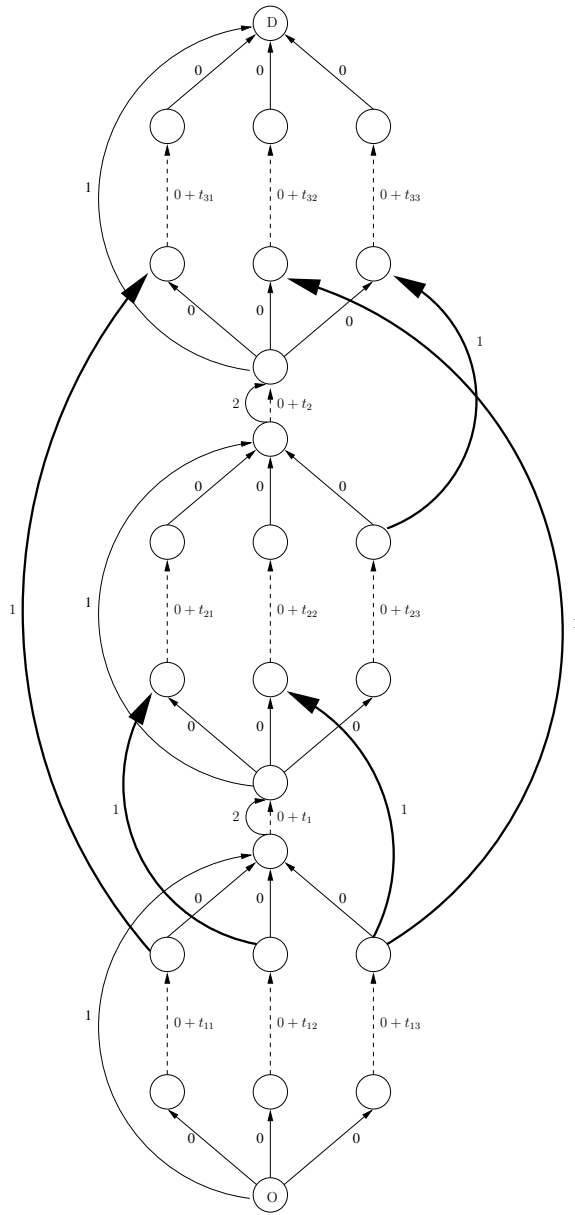


FIG. 4.2 – Exemple de réseau pour $F = (x_1 \vee x_2 \vee \overline{x_3}) \wedge (\overline{x_2} \vee x_3 \vee \overline{x_4}) \wedge (\overline{x_1} \vee x_3 \vee x_4)$

Proposition 4.1 *Le Toll Setting Problem avec un arc tarifable peut être résolu en $O(|K|^2|\mathcal{N}|^2)$ unités de temps.*

Preuve:

Nous présentons la méthode proposée par Brotcorne et al. [6] pour les réseaux ne contenant qu'un seul arc taxé.

Supposons que K_a^* est l'ensemble des commodités $k \in K$ pour lesquelles il existe un chemin de o_k à d_k passant par l'unique arc a du réseau. Nous représentons par $\gamma^k(t_a)$ le coût du plus court chemin de o_k à d_k pour un niveau de taxation t_a de l'arc a . Il s'en suit que le niveau maximum que l'on peut mettre sur l'arc a tel que l'arc sera encore utilisé est :

$$\pi_a^k = \gamma(\infty) - \gamma^k(0).$$

En effet, le coût du plus court chemin si la taxe de l'arc a est infinie moins le coût du plus court chemin si la taxe de l'arc a est nulle, nous donne bien la taxe maximum que l'on peut mettre sur l'arc a et tel que l'utilisation de l'arc a n'utilisera pas d'arc(s) non taxé(s) à la place de a .

Supposons que les valeurs π_a^k est ordonné de façon décroissante :

$$\pi_a^1 \geq \pi_a^2 \geq \dots \geq \pi_a^r,$$

où $r = |K_a^*|$.

Pour une taxation $t_a = \pi_a^i$, les utilisateurs des commodités k telles que $k \leq i$, emprunteront l'arc a .

Le revenu est donc égal à :

$$P(\pi_a^i) = \pi_a^i \sum_{k \leq i} n^k,$$

où n^k est le nombre d'utilisateurs de la commodité k . Nous définissons :

$$i^* = \arg \max_{i \in K} \left\{ \pi_a^i \sum_{k \leq i} n^k \right\}.$$

La taxation donnant le plus grand revenu est donc :

$$t_a = \pi_a^{i^*}.$$

Pour chaque commodité k , nous calculons le plus court chemin de o_k à d_k . Nous avons $|K|$ commodités et l'algorithme de Dijkstra a une complexité de $O(|\mathcal{N}|^2)$, nous allons donc faire $|K|$ fois Dijkstra.

Après, nous devons trouver la plus grande valeur entre $|K|$ éléments, ça peut être fait en $O(|K| - 1)$. Le calcul de $P(\pi_a^i)$ est négligeable. Donc la complexité du problème est bien $O(|K|^2|\mathcal{N}|^2)$. Ceci n'est vrai que parce que nous n'avons qu'un seul arc taxé.

□

Chapitre 5

The Highway Special Case

5.1 Introduction et formulation

Le problème considéré dans ce chapitre et un cas particulier du Toll Setting Problem, nous le formulons comme dans S. Dewez [8].

Dans le cas général, les arcs taxés peuvent apparaître n'importe où dans le réseau tandis qu'ici, ils constituent un chemin.

5.1.1 Exemple

Une compagnie construit deux routes (1,2) et (2,3) entre Ostende et Liège (Figure 5.1). Les noeuds 1,2 et 3 sont les points d'entrée de l'autoroute. Les utilisateurs de chaque commodité doivent sélectionner soit un chemin entièrement non taxé, soit un noeud d'entrée et un noeud de sortie de l'autoroute. Donc le chemin de chaque commodité va être automatiquement déterminé par les deux noeuds sélectionnés. Par exemple, si les utilisateurs voyageant entre Ostende et Bruxelles choisissent les noeuds 1 et 2, le chemin sera Ostende - 1 - 2 - Bruxelles. Dans cet exemple, nous considérons la distance comme le coût. La distance entre deux points taxés consécutifs, entre une ville et un point taxé et entre deux villes est connue. Le problème consiste à trouver la taxation optimale à mettre sur les tronçons taxés afin d'engendrer un revenu maximum pour la compagnie. Pour y parvenir, nous représentons le problème comme un graphe (Figure 5.2). Nous avons une commodité entre chaque paire de villes qui sont elles-même liées par un arc non taxé. Comme les utilisateurs doivent sélectionner deux points : un point d'entrée et un point de sortie, nous représentons les arcs taxés par un graphe complet dont les sommets sont les points d'entrée de l'autoroute.

Nous faisons l'hypothèse que le coût de l'arc taxé $(i, i + 2)$ est égal au coût de l'arc taxé $(i, i + 1)$ plus le coût de l'arc taxé $(i + 1, i + 2)$.

De plus, nous imposons l'utilisation d'au plus un arc taxé par commodité.

Considérons un arc taxé (i, j) , la taxe placée sur cet arc doit être inférieure à la somme des taxes placées sur les sous-chemins qui le compose $(t_{ij} \leq t_{il} + t_{lj} \quad \forall l \in$

$\mathcal{N} \setminus \{i, j\}$). Imaginons que cette inégalité ne soit pas satisfaite et donc que $t_{ij} > t_{il} + t_{lj}$, l'utilisateur pourrait alors sortir de l'autoroute en l et y rentrer juste après (toujours en l), il paierait moins cher. Nous devons imposer cette inégalité pour éviter ce genre de situation.

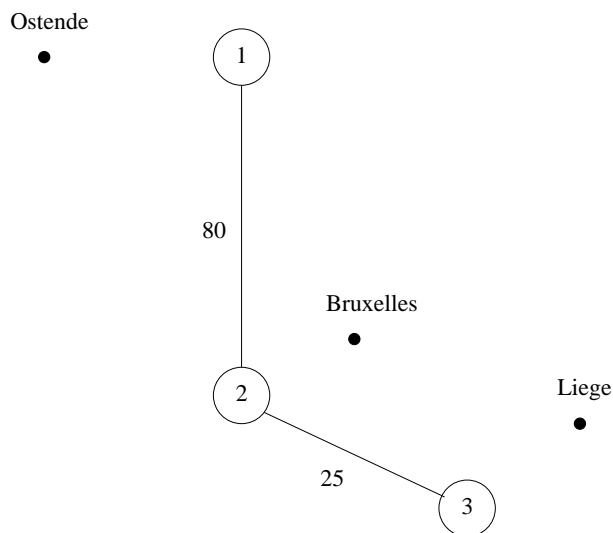


FIG. 5.1 – Exemple introductif

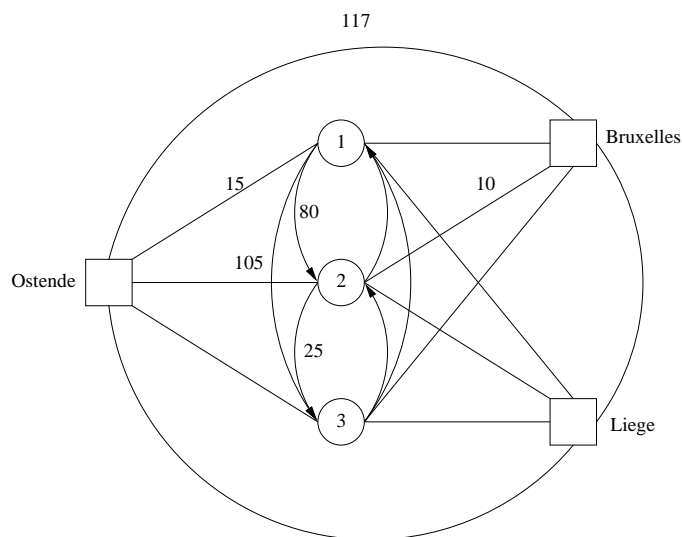


FIG. 5.2 – Exemple de la construction du graphe

5.1.2 Formulation du problème

Les notations que nous allons utiliser sont résumées ci-dessous :
Le problème précédemment évoqué peut être formalisé comme suit :

\mathcal{N}	ensemble des noeuds de l'autoroute
\mathcal{A}	ensemble des arcs taxés
K	ensemble des commodités
$k \in K$	ensemble des utilisateurs qui ont la même origine et la même destination
o_k	origine de la commodité k
d_k	destination de la commodité k
n^k	demande de la commodité k
c_{ij}^k	coût du chemin $o_k \rightarrow i \rightarrow j \rightarrow d_k$
c^k	coût de l'arc non taxé (o_k, d_k)
x_{ij}^k	flot sur l'arc taxé (i, j) pour la commodité k
y^k	flot sur l'arc non taxé (o_k, d_k)
t_{ij}	taxe sur l'arc taxé (i, j)

TAB. 5.1 – Tableau récapitulatif des notations utilisées.

$$\begin{aligned}
 (HTSP) \quad & \max_t \sum_{k \in K} n^k \sum_{(i,j) \in \mathcal{A}} t_{ij} x_{ij}^k \\
 & \text{s.c. } t_{ij} \leq t_{il} + t_{lj} && \forall (i, j) \in \mathcal{A}, \forall l \neq i, j \in \mathcal{N} \\
 & t_{ij} \geq 0 && \forall (i, j) \in \mathcal{A} \\
 & \min_{x,y} \sum_{k \in K} \left(\sum_{(i,j) \in \mathcal{A}} (c_{ij}^k + t_{ij}) x_{ij}^k + c^k y^k \right) \\
 & \text{s.c. } \sum_{(i,j) \in \mathcal{A}} x_{ij}^k + y^k = 1 && \forall k \in K \\
 & x_{ij}^k, y^k \in \{0, 1\} && \forall (i, j) \in \mathcal{A}, \forall k \in K
 \end{aligned}$$

L'objectif du meneur est de maximiser son revenu.

$$t_{ij} \leq t_{il} + t_{lj} \quad \forall (i, j) \in \mathcal{A}, \forall l \neq i, j \in \mathcal{N}$$

Ces contraintes sont des inégalités triangulaires sur les taxes : la taxe pour un arc tarifable (i, j) doit être inférieure ou égale à la taxe du chemin $i \rightarrow l \rightarrow j$ pour chaque noeud $l \in \mathcal{N}$ (cf Figure 5.3).

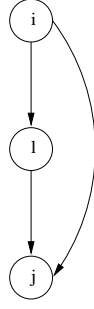


FIG. 5.3 – Exemple d’arcs pour les inégalités triangulaires.

L’objectif du suiveur est de minimiser le coût du chemin qu’il va utiliser. Si les utilisateurs de la commodité k choisissent l’arc taxé (i, j) , ils ont à payer c_{ij}^k qui est le coût du chemin $o_k \rightarrow i \rightarrow j \rightarrow d_k$ majoré de la taxe sur l’arc : t_{ij} . S’ils choisissent l’arc non taxé (o_k, d_k) le coût est c^k .

Les contraintes

$$\sum_{(i,j) \in \mathcal{A}} x_{ij}^k + y^k = 1 \quad \forall k \in K$$

imposent l’utilisation soit de l’arc non taxé reliant o_k à d_k soit d’un arc taxé tel que cet arc ne définit qu’un chemin unique entre l’origine et la destination. Les flots x_{ij}^k et y^k prennent des valeurs positives. Finalement nous supposons que toutes les taxes prennent des valeurs positives.

Nous avons vu que, si les taxes t_{ij} sont fixées, le deuxième niveau est un programme linéaire. Donc nous pouvons le remplacer par ses conditions d’optimalité primal-dual (voir Section 2.2) et nous obtenons le programme suivant :

$$(HTSP1) \max_{x,y,\lambda,t} \sum_{k \in K} n^k \sum_{(i,j) \in \mathcal{A}} t_{ij} x_{ij}^k$$

$$\text{s.c. } t_{ij} \leq t_{il} + t_{lj} \quad \forall (i, j) \in \mathcal{A}, \forall l \in \mathcal{N} \quad (5.1)$$

$$\sum_{i,j \in \mathcal{A}} x_{ij}^k + y^k = 1 \quad \forall k \in K \quad (5.2)$$

$$\lambda^k \leq c_{ij}^k + t_{ij} \quad \forall (i, j) \in \mathcal{A}, \forall k \in K \quad (5.3)$$

$$\lambda^k \leq c^k \quad \forall k \in K \quad (5.4)$$

$$(c_{ij}^k + t_{ij} - \lambda^k) x_{ij}^k = 0 \quad \forall (i, j) \in \mathcal{A}, \forall k \in K \quad (5.5)$$

$$(c^k - \lambda^k) y^k = 0 \quad \forall k \in K \quad (5.6)$$

$$t_{ij} \geq 0 \quad \forall (i, j) \in \mathcal{A}$$

$$x_{ij}^k, y^k \in \{0, 1\} \quad \forall (i, j) \in \mathcal{A}, \forall k \in K.$$

La contrainte (5.5) nous dit que soit le flot sur l'arc tarifable (i, j) est nul soit la contrainte (5.3) est serrée, c'est à dire que $\lambda^k = c_{ij}^k + t_{ij}$. De façon similaire, la contrainte (5.6) nous dit que soit le flot sur l'arc non tarifable est nul soit la contrainte (5.4) est serrée, c'est à dire que $\lambda^k = c^k$.

On peut donc voir que λ^k représente, la longueur du chemin entre o_k et d_k une fois que les taxes sont fixées.

Additionnons les contraintes (5.5) et (5.6). Nous obtenons l'égalité suivante :

$$(c_{ij}^k + t_{ij} - \lambda^k)x_{ij}^k + (c^k - \lambda^k)y^k = 0$$

Ensuite, sommons sur tous les arcs tarifables $(\forall(i, j) \in \mathcal{A})$:

$$\begin{aligned} \sum_{(i,j) \in \mathcal{A}} (c_{ij}^k + t_{ij} - \lambda^k)x_{ij}^k + (c^k - \lambda^k)y^k &= 0 \\ \Downarrow \\ \sum_{(i,j) \in \mathcal{A}} (c_{ij}^k + t_{ij})x_{ij}^k - \lambda^k x_{ij}^k + c^k y^k - \lambda^k y^k &= 0 \\ \Downarrow \\ \sum_{(i,j) \in \mathcal{A}} (c_{ij}^k + t_{ij})x_{ij}^k + c^k y^k - \lambda^k \sum_{(i,j) \in \mathcal{A}} x_{ij}^k + c^k y^k &= 0 \\ \Downarrow \\ \sum_{(i,j) \in \mathcal{A}} (c_{ij}^k + t_{ij})x_{ij}^k + c^k y^k - \lambda^k &= 0 \\ \Downarrow \\ \sum_{(i,j) \in \mathcal{A}} (c_{ij}^k + t_{ij})x_{ij}^k + c^k y^k &= \lambda^k \end{aligned}$$

Cette dernière égalité est équivalente à (5.5) et (5.6), Le (HTSP1) devient alors :

$$\max_{x,y,\lambda,t} \sum_{k \in K} n^k \sum_{(i,j) \in \mathcal{A}} t_{ij} x_{ij}^k \quad (5.7)$$

$$\text{s.c. } t_{ij} \leq t_{il} + t_{lj} \quad \forall(i, j) \in \mathcal{A}, \forall l \in \mathcal{N} \quad (5.7)$$

$$\sum_{i,j \in \mathcal{A}} x_{ij}^k + y^k = 1 \quad \forall k \in K \quad (5.8)$$

$$\lambda^k \leq c_{ij}^k + t_{ij} \quad \forall(i, j) \in \mathcal{A}, \forall k \in K \quad (5.9)$$

$$\lambda^k \leq c^k \quad \forall k \in K \quad (5.10)$$

$$\sum_{(i,j) \in \mathcal{A}} (c_{ij}^k + t_{ij})x_{ij}^k + c^k y^k = \lambda^k \quad \forall k \in K \quad (5.11)$$

$$\begin{aligned} t_{ij} &\geq 0 \quad \forall(i, j) \in \mathcal{A} \\ x_{ij}^k, y^k &\in \{0, 1\} \quad \forall(i, j) \in \mathcal{A}, \forall k \in K. \end{aligned} \quad (5.12)$$

Un problème subsiste encore : la fonction objective et la contrainte (5.11) sont non linéaires. Pour les linéariser, nous allons introduire de nouvelles variables : $t_{ij}^k = t_{ij}x_{ij}^k$. Nous les introduisons pour chaque arc tarifable et pour chaque commodité, comme nous l'avons fait dans la section 3.2. En ajoutant ces nouvelles contraintes nous obtenons le programme suivant :

$$\max_{x,y,\lambda,t} \sum_{k \in K} n^k \sum_{(i,j) \in \mathcal{A}} t_{ij}^k \quad (5.13)$$

$$\text{s.c. } t_{ij} \leq t_{il} + t_{lj} \quad \forall (i,j) \in \mathcal{A}, \forall l \in \mathcal{N} \quad (5.14)$$

$$\sum_{i,j \in \mathcal{A}} x_{ij}^k + y^k = 1 \quad \forall k \in K \quad (5.15)$$

$$\lambda^k \leq c_{ij}^k + t_{ij} \quad \forall (i,j) \in \mathcal{A}, \forall k \in K \quad (5.16)$$

$$\lambda^k \leq c^k \quad \forall k \in K \quad (5.17)$$

$$\sum_{(i,j) \in \mathcal{A}} (c_{ij}^k x_{ij}^k + t_{ij}^k) + c^k y^k = \lambda^k \quad \forall k \in K \quad (5.18)$$

$$t_{ij}^k \leq M_{ij}^k x_{ij}^k \quad \forall (i,j) \in \mathcal{A}, \forall k \in K \quad (5.19)$$

$$t_{ij} - t_{ij}^k \leq N_{ij}(1 - x_{ij}^k) \quad \forall (i,j) \in \mathcal{A}, \forall k \in K \quad (5.20)$$

$$t_{ij}^k \leq t_{ij} \quad \forall (i,j) \in \mathcal{A}, \forall k \in K \quad (5.21)$$

$$x_{ij}^k \in \{0, 1\} \quad \forall (i,j) \in \mathcal{A}, \forall k \in K \quad (5.22)$$

$$y_k > 0 \quad \forall k \in K \quad (5.23)$$

$$t_{ij}^k, t_{ij} \geq 0 \quad \forall (i,j) \in \mathcal{A}, \forall k \in K \quad (5.24)$$

Nous devons maintenant déterminer les valeurs des constantes M_{ij}^k pour renforcer cette formulation, nous allons utiliser d'autres bornes que celles présentées dans le Chapitre 3. Ces bornes sont celles proposées par S. Dewez [8].

Proposition 5.1 *Les constantes*

$$M_{ij}^k = \max\{0, c^k - c_{ij}^k\} \quad \forall k \in K, \forall (i,j) \in \mathcal{A}$$

sont valides pour la formulation (5.13) - (5.24).

Preuve:

Si le voyageur de la commodité k passe par l'arc (i,j) , la contrainte (5.19) devient :

$$t_{ij}^k \leq M_{ij}^k.$$

M_{ij}^k est donc une borne supérieure du revenu engendré par l'arc (i,j) pour la commodité k .

L'arc tarifable (i,j) ne sera utilisé par la commodité k que si le coût du trajet

sur l'arc non tarifable (o_k, d_k) est supérieur ou égal au coût du trajet empruntant l'arc (i, j) qui est soumis à une taxation nulle et qui n'emprunte aucun autre arc tarifable. Donc l'arc (i, j) sera utilisé par la commodité k si $c_{ij}^k + t_{ij}^k \leq c_k$ et nous obtenons la borne supérieure :

$$M_{ij}^k = c^k - c_{ij}^k.$$

□

Afin de compléter notre formulation, nous devons aussi donner une valeur aux N_{ij} pour chaque arc tarifable (i, j) . L'inégalité triangulaire rend cette assignation plus difficile. En effet, lorsque $x_{ij}^k = 0$, la contrainte (5.20) devient $t_{ij} \leq N_{ij}$. N_{ij} est donc une borne supérieure pour la taxe sur l'arc tarifable (i, j) .

Les inégalités triangulaires suivantes concernent les arcs tarifables (i, j) et (i, l) :

$$\begin{aligned} t_{ij} &\leq t_{il} + t_{lj} \\ t_{il} &\leq t_{ij} + t_{jl}, \end{aligned}$$

les inégalité suivantes induisent en utilisant la borne supérieure sur les t_{ij} :

$$\begin{aligned} t_{ij} &\leq t_{il} + t_{lj} \leq N_{il} + N_{lj} \\ t_{ij} &\leq N_{ij} \\ t_{il} &\leq t_{ij} + t_{jl} \leq N_{ij} + N_{jl} \\ t_{il} &\leq N_{il}. \end{aligned}$$

On voit donc que pour déduire la borne supérieure sur t_{ij} nous avons besoin de N_{il} pour t_{il} et vice et versa.

Il n'est donc pas facile de trouver une bonne borne supérieure. La conjecture suivante propose deux bornes supérieures parmi lesquelles on prendra la plus petite.

La première borne supérieure proposée pour t_{ij} est :

$$\max_{l \neq i} \max_{k \in K} M_{il}^k + \max_{l \neq j} \max_{k \in K} M_{lj}^k.$$

Cette borne revient à considérer la somme de la borne supérieure sur t_{il}^k pour tous les arcs tarifables (i, l) qui quittent le noeud i (et ce sur toutes les commodités) et de la borne supérieure sur t_{lj}^k pour tous les arcs tarifables (l, j) qui ont comme extrémité le noeud j (et ce aussi sur toutes les commodités).

La deuxième borne supérieure est la suivante :

$$\max_{(l,m) \in \mathcal{A}} \max_{k \in K} \{c^k - c_{lm}^k, 0\}$$

Cette borne est la même pour tous les arcs tarifables à partir du moment où l'on prend le maximum des M_{lm}^k sur tous les arcs tarifables (l, m) et sur toutes les commodités.

Conjecture 5.1.1 *Les constantes*

$$N_{ij} = \min \left\{ \max_{l \neq i} \max_{k \in K} M_{il}^k + \max_{l \neq j} \max_{k \in K} M_{lj}^k, \max_{(l,m) \in \mathcal{A}} \max_{k \in K} \{c^k - c_{lm}^k, 0\} \right\}$$

$\forall (i, j) \in \mathcal{A}$, sont valides pour HTSP.

5.1.3 Formulation équivalente

Le problème est que les inégalités triangulaires rendent le problème difficile, nous allons proposer une autre formulation comme dans S. Dewez [8].

On voit qu'il y a une inéquation triangulaire pour chaque arc tarifable (i, j) et pour chaque noeud de l'autoroute différent de i et j :

$$t_{ij} \leq t_{il} + t_{lj} \quad \forall (i, j) \in \mathcal{A}, l \in \mathcal{N}, l \neq i, j$$

Cela signifie qu'il y a $|\mathcal{A}|(|\mathcal{N}| - 2)$ inégalités triangulaires. En effet, on a $|\mathcal{A}|$ arcs et pour chaque arc, on a $(|\mathcal{N}| - 2)$ choix de noeuds intermédiaires (les $(|\mathcal{N}| - i$ et $j)$).

Comme nous l'avons supposé, nous avons un graphe complet dont les sommets sont les noeuds de l'autoroute, nous avons donc $|\mathcal{N}|(|\mathcal{N}| - 1)$ arcs tarifables.

Nous avons alors $|\mathcal{N}|(|\mathcal{N}| - 1)(|\mathcal{N}| - 2)$ inégalités triangulaires.

S. Dewez [8] a remarqué que sans ces contraintes, le problème peut être résolu beaucoup plus rapidement mais il se peut que la solution trouvée ne soit pas réalisable pour le problème d'origine vu qu'elle ne respecte pas les contraintes d'inégalité triangulaire.

En prenant un réseau composé de 9 villes, 36 commodités et 10 noeuds taxés et en résolvant le problème avec les inégalités triangulaires, elle obtient un temps de résolution égal à 10 heures. La valeur de la solution était égale à 21007. En résolvant le même problème mais sans les inégalités triangulaires, le temps mis par CPLEX pour trouver une solution était bien inférieur : 4 minutes. La solution était égale à 21233.

L'idée qui est développée dans S. Dewez [8] est donc de remplacer les inégalités triangulaires. Le nombre de contraintes ne va pas diminuer mais le temps de résolution est considérablement réduit. Pour remplacer ces inégalités triangulaires, il faut de nouvelles inégalités.

Proposition 5.2 *En remplaçant les inégalités triangulaires par :*

$$t_{ij} \leq t_{il} + t_{lj} + N_{il}z_{il} + N_{lj}z_{lj} \quad \forall (i, j) \in \mathcal{A} \quad (5.25)$$

$$z_{il} \leq 1 - x_{ij}^k \quad \forall (i, j) \in \mathcal{A}, \forall k \in K \quad (5.26)$$

dans la formulation (5.13) - (5.24) nous obtenons une solution optimale. Cette solution, peut être transformée de façon à satisfaire les inégalités triangulaires tout en maintenant la même solution optimale.

Preuve:

La preuve se base sur le fait qu'en remplaçant les inégalités triangulaires par (5.25) et (5.26), les taxes des arcs qui ne sont pas utilisés n'ont pas à satisfaire les inégalités triangulaires.

On voit donc apparaître trois cas :

1. Il existe deux commodités k_1 et k_2 telles que $x_{il}^{k_1} = 1$ et $x_{lj}^{k_2} = 1$.
Dans ce cas, z_{il} et z_{lj} sont tous les deux égaux à zéro ($z_{il} \leq 1 - x_{ij}^k$) et les inégalités triangulaires sont les mêmes que (5.25) ($t_{ij} \leq t_{il} + t_{lj} + N_{il}z_{il} + N_{lj}z_{lj} \Leftrightarrow t_{ij} \leq t_{il} + t_{lj}$).
2. Il existe une commodité k_1 telle que $x_{il}^{k_1} = 1$ mais il n'y a aucune commodité k_2 telle que $x_{lj}^{k_2} = 1$ (ou inversement).
Dans ce cas, t_{lj} peut être arbitrairement grand (étant donné que l'utilisateur ne passera pas sur cet arc, la taxe sur l'arc (l, j) n'a plus aucune importance).
On a donc $z_{il} = 0$ et $z_{lj} = 1$ ce qui nous mène à :

$$t_{ij} \leq t_{il} + t_{lj} + N_{lj}.$$

Ce qui est équivalent à l'inéquation triangulaire à partir du moment où t_{lj} est très grand (la valeur de N_{lj} sera négligeable à côté de t_{lj}).

3. Finalement, nous aurons le cas où x_{il}^k et x_{lj}^k valent à zéro pour toute commodité k . On a donc t_{il} et t_{lj} qui seront arbitrairement grands et l'inégalité (5.25) est alors maintenue.

Une fois que nous avons la solution optimale pour cette formulation, nous pouvons construire une solution qui satisfait les inégalités triangulaires en résolvant l'inégalité suivante :

$$t_{ij} \leq t_{il} + t_{lj} \quad \forall (i, j) \in \mathcal{A}.$$

Pour résoudre le système, nous créons au départ un graphe contenant tous les noeuds. Nous ajoutons les arcs utilisés par les voyageurs et nous associons à chacun d'eux un poids égal à leur taxe. Ensuite nous ajoutons un arc entre chaque paire de noeuds liés par un chemin. A chaque nouvel arc, nous associons un poids égal à la longueur du chemin entre ses extrémités. Finalement, nous ajoutons les arcs restants, nécessaires à la création du graphe complet, auxquels nous associons le poids $T = \max_{(l,m) \in \mathcal{A}} \max_{k \in K} c^k - c_{lm}^k$ (voir Figure 5.4). Cette construction respecte bien notre hypothèse au niveau du graphe et l'attribution des poids respectent les contraintes. En mettant T aux arcs rajoutés dans la dernière étape, on est sûr qu'ils ne seront jamais utilisés.

□

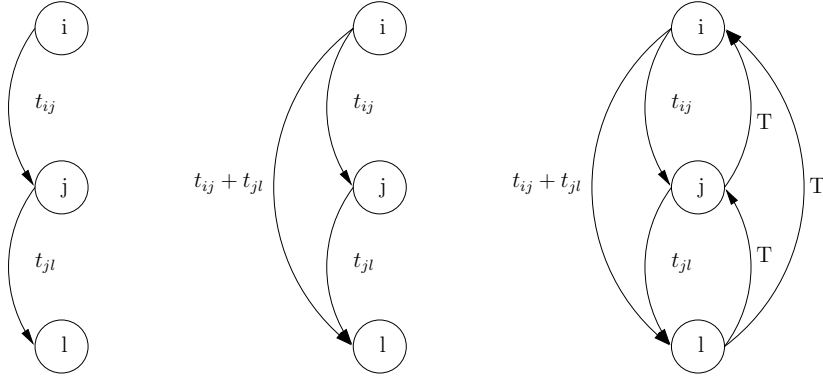


FIG. 5.4 – Etapes de création du graphe.

En ajoutant les nouvelles contraintes (5.25) et (5.26) nous obtenons la formulation suivante :

$$(HTSP2) \max_{x,y,\lambda,t} \sum_{k \in K} n^k \sum_{(i,j) \in \mathcal{A}} t_{ij}^k \quad (5.27)$$

$$\text{s.c. } t_{ij} \leq t_{il} + t_{lj} + N_{il}z_{il} + N_{lj}z_{lj} \quad \forall (i,j) \in \mathcal{A} \quad (5.28)$$

$$z_{il} \leq 1 - x_{ij}^k \quad \forall (i,j) \in \mathcal{A}, \forall k \in K \quad (5.29)$$

$$\sum_{i,j \in \mathcal{A}} x_{ij}^k + y^k = 1 \quad \forall k \in K \quad (5.30)$$

$$\lambda^k \leq c_{ij}^k + t_{ij} \quad \forall (i,j) \in \mathcal{A}, \forall k \in K \quad (5.31)$$

$$\lambda^k \leq c^k \quad \forall k \in K \quad (5.32)$$

$$\sum_{(i,j) \in \mathcal{A}} (c_{ij}^k x_{ij}^k + t_{ij}^k) + c^k y^k = \lambda^k \quad \forall k \in K \quad (5.33)$$

$$t_{ij}^k \leq M_{ij}^k x_{ij}^k \quad \forall (i,j) \in \mathcal{A}, \forall k \in K \quad (5.34)$$

$$t_{ij} - t_{ij}^k \leq N_{ij}(1 - x_{ij}^k) \quad \forall (i,j) \in \mathcal{A}, \forall k \in K \quad (5.35)$$

$$t_{ij}^k \leq t_{ij} \quad \forall (i,j) \in \mathcal{A}, \forall k \in K \quad (5.36)$$

$$x_{ij}^k \in \{0, 1\} \quad \forall (i,j) \in \mathcal{A}, \forall k \in K \quad (5.37)$$

$$y_k \geq 0 \quad \forall k \in K \quad (5.38)$$

$$t_{ij}^k, t_{ij} \geq 0 \quad \forall (i,j) \in \mathcal{A}, \forall k \in K \quad (5.39)$$

Grâce à cette nouvelle formulation, S. Dewez obtient, pour le même problème, la solution optimale de 21007 mais en une heure. La diminution du temps de résolution est donc considérable.

5.2 Un algorithme exact : K-chemin

Dans cette section, nous allons présenter un algorithme exact pour le cas particulier du Toll Setting Problem abordé dans ce chapitre. Cet algorithme est similaire à celui proposé par Marcotte, Didi et Savard [9] pour la cas général du Toll Setting Problem.

Les utilisateurs de chaque commodité k doivent sélectionner l'arc qu'ils vont emprunter pour se rendre de o_k à d_k . Si nous connaissons l'arc choisi par les utilisateurs de chaque commodité k , nous pouvons alors calculer, pour chaque commodité k , le revenu qu'ils engendrent et ceci en calculant le problème *inverse* suivant :

$$\max_{t, t'} \sum_{k \in K} n^k t_{ab(k)}^k \quad (5.40)$$

$$\text{s.c. } t_{ij} \leq t_{il} + t_{lj} \quad \forall (i, j) \in \mathcal{A}, \forall l \in \mathcal{N} \quad (5.41)$$

$$c_{ab(k)}^k + t_{ab(k)}^k \leq c_{ij}^k + t_{ij} \quad \forall (i, j) \in \mathcal{A}, \forall ab(k) \in \mathcal{A}, \forall k \in K \quad (5.42)$$

$$\lambda^k \leq c_{ij}^k + t_{ij} \quad \forall (i, j) \in \mathcal{A}, \forall k \in K \quad (5.43)$$

$$\lambda^k \leq c^k \quad \forall k \in K \quad (5.44)$$

$$t_{ab(k)}^k \leq M_{ab(k)}^k \quad \forall k \in K, \forall ab(k) \in \mathcal{A} \quad (5.45)$$

$$t_{ab(k)} = t'_{ab(k)} \quad \forall k \in K, \forall ab(k) \in \mathcal{A} \quad (5.46)$$

où (a, b) est l'arc choisi par la commodité k , exemple : si $x_{ab(k)}^k = 1$, la commodité k passera par l'arc tarifable (a, b) et si $x_{ab(k)}^k = 0$, la commodité k passera par l'arc non tarifable joignant o_k et d_k . Rappelons que c_{ij}^k est le coût du chemin entre o_k et d_k pour la commodité k et passant par l'arc (i, j) où la taxe est nulle et que c^k est le coût du chemin entre o_k et d_k pour la commodité k et passant par l'arc non tarifable (o_k, d_k) . Nous voyons que dans cette formulation les contraintes (5.43), (5.44). En effet, les λ^k n'ont aucune borne inférieure et n'interviennent pas dans la fonction objective. Nous pouvons donc supprimer ces contraintes.

La contrainte (5.42) impose à elle seule que le chemin pris par la commodité k soit le plus court chemin parmi les chemins disponibles passant par un arc tarifable.

Nous distinguons deux cas :

- $ab(k) = (i, j)$: trivial.
- $ab(k) \neq (i, j)$: on a bien que le chemin passant par l'arc $ab(k)$ pour la commodité k doit être inférieur ou égal à tous les autres chemins passant par des arcs tarifables pour la commodité k .

Intéressons-nous maintenant à l'algorithme.

Nous appelons une solution du niveau inférieur un *K-chemin* et nous le notons :

$$(e^1, e^2, \dots, e^{|K|})$$

où e^k est l'arc choisi par la commodité k . Une borne supérieure sur le revenu associé à ce K -chemin est $\sum_{k \in K} n^k M_{e^k}^k$ (rappelons que $M_{e^k}^k = (c_{e^k}^k - c^k)$). Le premier K -chemin sera la solution nous donnant la plus grande borne supérieure. Le i -ème K -chemin sera la solution donnant la i -ème plus grande borne supérieure.

L'algorithme s'effectue suivant les étapes suivantes. Nous notons S^* la meilleure solution courante et V^* la valeur de cette solution. Nous initialisons la valeur de la solution à 0 et le K -chemin courant (représenté par i) à 1.

Ensuite nous calculons le i -ème K -chemin (nous expliquerons dans la prochaine section comment il faut s'y prendre pour le calculer) et la borne supérieure B_i associé à ce i -ème K -chemin. Si la borne supérieure est inférieure ou égale à la meilleure solution V^* nous nous arrêtons et nous avons comme solution optimale S^* . Dans le cas contraire, nous calculons le revenu V^i du K -chemin en résolvant le problème inverse. Nous notons la solution de ce problème S^i .

Si le revenu V^i est égal à la borne supérieure B_i , S^i est optimale puisque nous maximisons la fonction objective et que les B^i sont strictement décroissants.

Si l'égalité n'est pas vérifiée, nous mettons à jour le meilleur revenu et la meilleure solution courante uniquement si la solution courante est meilleure que la meilleure solution.

Nous incrémentons i de 1 et nous répétons la procédure.

Algorithme exact :

Etape 1 : Initialisation :

- $S^* = \phi$
- $V^* = 0$
- $i = 1$

Etape 2 : Calculons le i ème K -chemin.

Calculons la borne supérieure du revenu :

$$B_i = \sum_{k \in K} n^k M_{ab(k)}^k$$

où (a, b) est l'arc choisi par la commodité k . Si l'arc choisi est l'arc (o_k, d_k) , alors $M_{ab(k)}^k = 0$.

En effet, $M_{ab}^k = c - c_{ab(k)}$ et si $c = c_{ab(k)}$, $M_{ab(k)}^k$ vaut bien 0.

Si $B_i \leq V^*$, on s'arrête avec S^* comme solution optimale.

Etape 3 : Résoudre le problème suivant où (a, b) est l'arc choisi par la com-

modité k et supposons que $x_{ab(k)}^k = 1 \forall k \in K$:

$$\begin{aligned}
& \max_{t, t'} \sum_{k \in K} n^k t_{ab}^k \\
& \text{s.c. } t_{ij} \leq t_{il} + t_{lj} & \forall (i, j) \in \mathcal{A}, \forall l \in \mathcal{N} \\
& c_{ab(k)}^k + t_{ab(k)}^k \leq c_{ij}^k + t_{ij} & \forall (i, j) \in \mathcal{A}, \forall k \in K \\
& t_{ab(k)} = t'_{ab(k)} & \forall k \in K \\
& t_{ab(k)}^k \leq M_{ab(k)}^k & \forall k \in K, \forall ab(k) \in \mathcal{A}
\end{aligned}$$

Si $V_i = B_i$ alors S_i est la solution optimale.

Etape 4 : Si $V^i > V^*$ alors $S^* = S^i$ et $V^* = V^i$.

$i = i + 1$

Aller à l'étape 2.

Nous allons montrer, comme dans S. Dewez [8], que cet algorithme donnera toujours la solution optimale.

Proposition 5.3 *L'algorithme détermine une solution optimale en un nombre fini d'itérations.*

Preuve:

On voit directement que le nombre de K -chemin est fini. Maintenant nous allons montrer que l'algorithme s'arrête avec la solution optimale. Puisque les séquences B^i sont décroissantes, le revenu V^j du K -chemin j , $j = 1, \dots, i - 1$ est inférieur à B^i . Nous notons par V_{opt} la valeur optimale de la solution.

Nous avons alors :

$$V^* \leq V_{opt} \leq \max\{V^*, B^i\}.$$

Le K -chemin S^* correspondant à la valeur V^* est optimal si $B^i \leq V^*$ et le K -chemin S^i est optimal si $V^i = B^i$. Comme la séquence V^* est non décroissante, on obtiendra tôt ou tard la solution optimale.

□

5.2.1 Calcul du i -ème K -chemin

Maintenant nous allons décrire la méthode de calcul pour le i -ème K -chemin. Premièrement nous classons les arcs (i, j) par ordre décroissant de M_{ij}^k pour chaque commodité k et nous les renumérotions tels que :

$$M_1^k \geq M_2^k \geq \dots \geq M_{|\mathcal{A}|}^k$$

où \mathcal{A} est toujours l'ensemble des arcs tarifables. Nous gardons une liste (LIST) des candidats pour le prochain *K-chemin*. Nous initialisons le compteur j (le *K-chemin* courant) à 1.

Le premier *K-chemin* E^1 est composé des arcs ayant la plus grande borne supérieure pour chaque commodité k :

$$E^1 = (a_1^1, a_1^2, \dots, a_1^{|K|})$$

où a_r^k est l'arc ayant la r -ième plus grande borne supérieure pour la commodité k ($M_{a_r^k}^k$). Nous allons aussi garder un vecteur $j(k)$ représentant l'arc choisi par la commodité k pour le *K-chemin* courant. Nous commençons donc par $j(k) = 1$ pour chaque commodité k .

Si $j = i$, où j est l'indice du *K-chemin* courant et i est l'indice du *K-chemin* que l'on veut calculer, on s'arrête avec E^j comme le jème *K-chemin*. Autrement nous calculons les nouveaux candidats, à partir jème *K-chemin*, E^j . Les candidats diffèrent par exactement un seul élément par rapport à E^j . Nous notons par $E^{j,l}$ le candidat où les utilisateurs de la commodité l choisissent le prochain arc ayant la plus grande borne supérieure, c'est à dire l'arc $j(l) + 1$.

Par exemple, les candidats générés à partir du premier *K-chemin* E^1 sont :

$$\begin{aligned} E^{1,1} &= (a_2^1, a_1^2, \dots, a_1^{|K|}) \\ E^{1,2} &= (a_1^1, a_2^2, \dots, a_1^{|K|}) \\ &\vdots \\ E^{1,l} &= (a_1^1, a_1^2, \dots, a_2^l, \dots, a_1^{|K|}) \\ &\vdots \\ E^{1,|K|} &= (a_2^1, a_1^2, \dots, a_2^{|K|}) \end{aligned}$$

Nous calculons la borne supérieure $B^{j,l}$ associée au candidat $E^{j,l}$. La borne supérieure est la somme sur chaque commodité du produit entre la demande (vecteur n) et la borne supérieure de l'arc choisi par chaque commodité :

$$B^{j,l} = \sum_{k \in K} n^k M_{a_{j(l)+1}^k}^k$$

Nous notons par $\text{ind}[j][l][k]$, l'indice de l'arc choisi par la commodité k pour le candidat $E^{j,l}$. Nous ajoutons les nouveaux candidats à la liste (LIST) et nous incrémentons j de 1.

Lors du prochain pas, nous choisissons l'élément de la liste ayant la borne supérieure la plus grande. Nous supprimons cet élément de la liste et nous mettons à jour le vecteur $j(k)$ selon les indices de E^j . Nous itérons tant que $j < i$.

Algorithme du ième K -chemin :

Etape 1 : Classons les M_{ij}^k par ordre décroissant pour chaque commodité k :

$$M_1^k \geq M_2^k \geq \dots \geq M_{|A|}^k$$

Initialisons :

- j à 1
- $LIST = \phi$
- $E^{1,1} = (a_2^1, a_1^2, \dots, a_1^{|K|})$ où a_r^k est l'arc ayant la r -ième borne supérieure la plus grande pour la commodité k
- $j(k)$ à 1 $\forall k \in K$.

Etape 2 : Si $j = i$ on s'arrête.

Etape 3 : Nous calculons les nouveaux candidats :

$$E^{j,l} = (a_1, \dots, a_{|K|}) \forall l \in K$$

où

$$\begin{cases} a_k = a_{j(k)}^k & \text{si } k \neq l \\ a_l = a_{j(l)+1}^l \end{cases}$$

La borne supérieure associée à ce candidat est :

$$B^{j,l} = \sum_{k \in K} n^k M_{a_k}^k$$

Nous notons l'arc sélectionné pour chaque commodité de ce candidat :

$$ind[j][l][k] = \begin{cases} j(k) & \text{si } k \neq l \\ j(k) + 1 & k = l \end{cases}$$

$$LIST = LIST \cup E^{j,l} \text{ et } j = j + 1$$

Etape 4 : Nous choisissons l'élément de LIST ayant la borne supérieure la plus grande :

$$E^j = \arg \max_{E^{j,l} \in LIST} B^{j,l}$$

Nous supprimons cet élément de LIST : $LIST = LIST \setminus E^j$

Etape 5 : Nous mettons à jour les indices $j(k) = ind[j][l][k] \forall k \in K$
Aller à l'étape 2.

Proposition 5.4 *Un K -chemin, contenant deux commodités ayant la même destination et dont les utilisateurs choisissent le même point d'entrée sur l'autoroute mais pas le même point de sortie, peut être supprimé de la liste des candidats.*

Preuve:

Considérons deux commodités k_1 et k_2 telles que $d_{k_1} = d_{k_2} = d$.

Prenons (i, j) l'arc choisi par la première commodité et (i, l) choisi par la deuxième. Les deux commodités utilisent donc des chemins différents pour aller de i à d . Ce qui n'est pas en accord avec la définition du problème car les utilisateurs prendront toujours le chemin le plus court, or ici, ce n'est pas le cas. On peut donc supprimer ce K -chemin des candidats.

□

La Figure 5.5 illustre la proposition.

Proposition 5.5 *Un K -chemin, contenant deux commodités ayant la même origine et dont les utilisateurs choisissent le même point de sortie sur l'autoroute mais pas le même point d'entrée, peut être supprimé de la liste des candidats.*

Preuve:

Elle est similaire à celle de la proposition 5.5

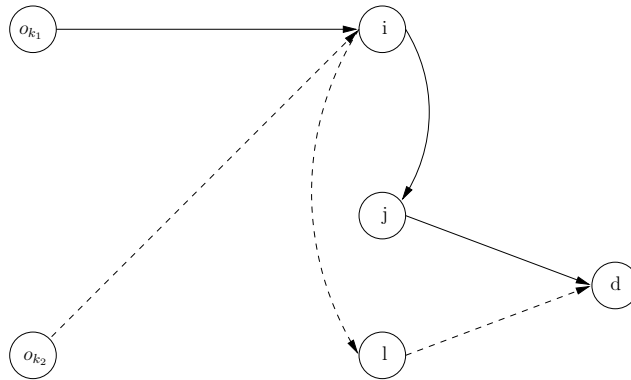


FIG. 5.5 – Exemple d'un cas illustrant la proposition 5.4

Ces propositions présentent malgré tout des désavantages. Le premier est que dans certains cas, aucun candidat n'est généré à partir du premier K -chemin. Un autre est qu'on n'est pas sûr d'atteindre la solution optimale.

En effet, considérons le cas suivant : supposons que le premier K -chemin est tel que trois commodités ont la même destination et que les arcs choisis par les utilisateurs de chaque commodité sont respectivement : (i, j) , (i, l) , (i, m) .

Les candidats générés sont tels que seulement un élément change par rapport au K -chemin père. Il est facile de voir que tous les candidats générés contiendront

un des cas évoqués dans la proposition 5.5. Tous ces *K-chemins* peuvent être supprimés de la liste.

Mais pour itérer dans notre procédure, nous avons besoin de nouveaux candidats dans la liste des candidats.

Le deuxième problème est que l'algorithme peut ne pas trouver la solution optimale car nous supprimons certains *K-chemin*. En effet, cela peut se produire si la solution optimale est celle issue d'une solution *interdite*.

Pour éviter tous ces problèmes, il faudrait trouver une façon différente de générer ces K-chemins.

□

5.3 Heuristiques

Dans cette section nous allons présenter plusieurs heuristiques pour résoudre le problème. Nous allons donner à chaque commodité k le choix entre l'arc non tarifable (o_k, d_k) et un arc tarifable.

L'idée de toutes les heuristiques qui vont être présentées, c'est d'attribuer à chaque commodité l'arc tarifable qui aurait le plus de chance d'être utilisé dans la solution optimale. Il va de soi que nous ne pouvons fixer les taxes à cause des inégalités triangulaires, nous ne pouvons fixer que les flots : pour attribuer l'arc (i, j) à la commodité k , on mettra x_{ij}^k à 1.

Pour obtenir la valeur des taxes associées à chaque arc, nous aurons besoin de résoudre le problème inverse suivant :

$$\max_{x,y,\lambda,t,t'} \sum_{k \in K} n^k t_{ab(k)}'^k \quad (5.47)$$

$$\text{t.q. } t_{ij} \leq t_{il} + t_{lj} + N_{il} z_{il} + N_{lj} z_{lj} \quad \forall (i, j), (i, l), (l, j) \in \mathcal{A}, \forall k \in K \quad (5.48)$$

$$z_{ij} \leq 1 - x_{ij}^k \quad \forall (i, j) \in \mathcal{A}, \forall k \in K \quad (5.49)$$

$$\lambda^k \leq c_{ij}^k + t_{ij} \quad \forall (i, j) \in \mathcal{A}, \forall k \in K \quad (5.50)$$

$$\lambda^k \leq c^k \quad \forall k \in K \quad (5.51)$$

$$\lambda^k = c_{ab(k)}^k x_{ab(k)}^k + t_{ab(k)}'^k + c^k y^k \quad \forall k \in K \quad (5.52)$$

$$t_{ab(k)}'^k \leq M_{ab(k)}^k x_{ab(k)}^k \quad \forall k \in K \quad (5.53)$$

$$t_{ab(k)} - t_{ab(k)}'^k \leq N_{ab(k)} (1 - x_{ab(k)}^k) \quad \forall k \in K \quad (5.54)$$

$$t_{ab(k)}'^k \leq t_{ab(k)} \quad \forall k \in K \quad (5.55)$$

$$x_{ab(k)}^k + y^k = 1 \quad \forall k \in K \quad (5.56)$$

$$x_{ab(k)}^k \in \{0, 1\} \quad \forall k \in K \quad (5.57)$$

$$y^k \geq 0 \quad \forall k \in K \quad (5.58)$$

$$t_{ij}'^k \geq 0 \quad \forall k \in K, \forall (i, j) \in \mathcal{A} \quad (5.59)$$

$$t_{ij} \geq 0 \quad \forall (i, j) \in \mathcal{A}, \quad (5.60)$$

où $ab(k)$ est l'arc tarifable que peut utiliser la commodité k . En effet, dans cette formulation, nous laissons le choix à la commodité k d'utiliser l'arc non tarifable ($y^k = 1$) ou l'arc tarifable sélectionné ($x_{ab(k)}^k = 1$). Ce choix est modélisé par la contrainte (5.56).

Les contraintes (5.48) - (5.49) sont obtenues de la même façon qu'en 5.1.3, les contraintes (5.50) - (5.52) imposent que le chemin utilisé soit le plus court des chemins disponibles pour la commodité k et les contraintes (5.53) - (5.54) sont obtenues de la même façon qu'en 3.2.4.

Finalement, remarquons que $t_{ab(k)}'^k$ n'est plus égal à $t_{ab(k)}$ (comme c'était le cas dans la formulation (5.40) - (5.46)) car, dans cette formulation, il se peut que l'on n'utilise pas l'arc tarifable choisi pour la commodité k .

5.3.1 Heuristique 1

Pour cette première heuristique, nous choisissons l'arc ayant la plus grande borne supérieure. En prenant les arcs ayant la plus grande borne supérieure pour chaque commodité nous nous disons qu'il y a plus de chance d'être proche de la solution optimale qu'en prenant les arcs au hasard ou même ceux ayant la borne supérieure la plus petite.

Si la plus grande borne supérieure pour la commodité k est égale à 0, cela veut dire que la commodité k ne va jamais passer par un arc tarifable, nous mettons donc y^k à 1, dans l'autre cas, l'utilisateur aura le choix entre le chemin passant par l'arc tarifable ou celui qui va directement de l'origine à la destination ($x_{ij}^k + y^k = 1$).

Heuristique 1

- Etape 1
 $k = 1$.
- Etape 2
Prendre pour la commodité k la plus grande borne supérieure M_{ij}^k .
L'arc sélectionné est : $(i, j)^* = \arg \max_{(i,j) \in \mathcal{A}} M_{ij}^k$.
- Etape 3
Si $M_{(i,j)^*}^k \neq 0$ on a la contrainte : $x_{(i,j)^*}^k + y^k = 1$.
Sinon on a la contrainte : $y^k = 1$.
- Etape 4
Si k est égal au nombre de commodités, résoudre le problème inverse (5.47)-(5.60) et s'arrêter.
Sinon, $k = k + 1$ et aller à Etape 2.

5.3.2 Heuristiques 2 et 3

Nous allons choisir les arcs de façon différente. Pour justifier ce choix, nous allons d'abord examiner un petit exemple.

	(1-2)	(1-3)	(2-1)	(2-3)	(3-2)
1	2	20	7	6	9
2	5	3	3	2	2
3	1	10	3	9	6

Dans ce tableau, nous représentons les bornes supérieures (M_{ij}^k) pour chaque arc et ceci pour chaque commodité. En gras, nous avons marqué les arcs choisis

selon l'heuristique 1. On voit clairement que le choix pour la commodité 3 n'est pas ce qui semble être le meilleur. En effet, elle passe par le même arc que la commodité 1 qui, elle, a comme borne supérieure sur l'arc (1-3) 20 alors que la borne supérieure pour la commodité 3 pour ce même arc est 10.

Nous pouvons penser que, dans la solution optimale, les commodités empruntant le même arc ont des bornes supérieures "semblables" sur cet arc.

Cette heuristique essaie de rendre cette condition de similarité lors du choix des arcs. Pour ce faire, nous allons initialement choisir, pour chaque commodité, l'arc ayant la plus grande borne supérieure :

$$a_1^k = \max_{(i,j) \in \mathcal{A}} M_{ij}^k \quad \forall k \in K.$$

où a_1^k est l'arc ayant la plus grande borne supérieure pour la commodité k .

Nous plaçons ces arcs dans une liste : *List*. Ensuite, nous trions ces arcs par valeur décroissante de leur borne supérieure. La liste devient donc une liste classée par ordre décroissant. Lorsque nous prenons un arc placé dans *List*, nous prenons le premier, c'est à dire celui qui a la plus grande borne supérieure parmi les arcs placés dans *List*.

Lors du choix de l'arc (i, j) pour la commodité k , trois cas de figure peuvent se présenter :

1. La borne supérieure sur l'arc est égale à 0.
2. La borne supérieure est supérieure à 0 et aucune autre commodité ne passe déjà par cet arc.
3. La borne supérieure est supérieure à 0 et au moins une autre commodité passe par cet arc.

Dans le premier cas, la commodité k utilisera le chemin non tarifable et nous aurons la contrainte $y^k = 1$.

Dans le deuxième cas, nous aurons la contrainte $x_{(i,j)}^k + y^k = 1$.

Dans le troisième cas, nous devons vérifier que la diminution que va provoquer le choix de cet arc par la commodité k ne sera pas trop grande. Pour ce faire, nous allons comparer M_{ij}^k avec la plus grande borne supérieure déjà placée sur l'arc (i, j) . L'arc (i, j) sera choisi pour la commodité k si :

$$M_{ij}^k \geq c * M_{ij}^*,$$

où c est une constante (par exemple 0,75) et $c * M_{ij}^*$ est la plus grande borne supérieure placée sur (i, j) . Si cette inégalité est vérifiée, l'arc (i, j) sera donc choisi pour la commodité k et nous avons la contrainte $x_{(i,j)}^k + y^k = 1$, si cette inégalité n'est pas vérifiée, on va considérer l'arc suivant et nous plaçons cet arc dans *List*.

Chaque fois qu'un arc est choisi, parmi ceux placés dans *List*, nous le supprimons de la liste. Nous ne rajoutons un arc dans *List* uniquement si le choix de l'arc n'a pas été retenu (échec lors du cas numéro 3). De cette façon, nous ne considérerons plus une commodité pour laquelle un choix d'arc a été retenu.

Si nous revenons à l'exemple ci-dessus, les choix auraient été les suivants :

	(1-2)	(1-3)	(2-1)	(2-3)	(3-2)
1	2	20	7	6	9
2	5	3	3	2	2
3	1	10	3	9	6

En effet nous considérons d'abord la commodité 1 car elle a la plus grande borne supérieure parmi toutes les commodités ($20 > 10 > 5$). Nous choisissons l'arc (1-3) pour cette commodité car aucune autre commodité ne passe par cet arc. Ensuite nous considérons la commodité 3 ($10 > 5$) et comme la commodité 1 passe déjà par l'arc (1-3), nous devons vérifier que la diminution ne soit pas trop "grande" : $20 * 0,75 = 15 > 10$. L'inégalité n'étant pas vérifiée, nous devons considérer l'arc ayant la plus grande borne supérieure inférieure à 10 pour la commodité 3. Cet arc est l'arc (2-3). Nous considérons à nouveau la commodité 3 car elle a la plus grande borne supérieure parmi toutes les commodités pour lesquelles aucun arc n'a encore été choisi ($9 > 5$). Nous choisissons l'arc (2-3) pour la commodité 2 car aucune autre commodité ne passe par cet arc. Finalement il ne reste que la commodité 2, nous choisissons l'arc (1-3) car aucune autre commodité ne passe par cet arc.

Heuristique 2

– Etape 1

Classons les M_{ij}^k par ordre décroissant pour chaque commodité k :

$$M_1^k \geq M_2^k \geq M_3^k \geq \dots \geq M_{|\mathcal{A}|}^k$$

Initialisation :

$indice[k] = 1 \forall k \in K$, ce vecteur permet de sauver l'indice de l'arc ayant la $indice[k]$ -ième borne supérieure la plus grande pour la commodité k .

$E[k] = (o_k, d_k) \forall k \in K$.

$List[k] = a_1^k \forall k \in K$, où a_1^k est l'arc ayant la 1-ième borne supérieure la plus grande pour la commodité k .

$max[i][j] = -1 \forall (i, j) \in \mathcal{A}$.

– Etape 2

Nous prenons l'arc parmi les éléments de *List* ayant la borne supérieure la plus grande :

$$a_{(i,j)}^k = arg \max_{a_i^k \in List} M_{ij}^k$$

Si $a_{(i,j)}^k = 0$ aller à Etape 3.
Sinon aller à Etape 4.

– Etape 3

Si le maximum de List est 0, cela veut dire que tous les arcs qui sont dans List valent 0, toutes les commodités k n'ayant pas encore choisi d'arc passeront donc par l'arc (o_k, d_k) , on vide donc la liste : $List = \phi$
Nous avons la contrainte : $y^k = 1$ pour chaque commodité utilisant (o_k, d_k) .
Aller à Etape 9.

– Etape 4

Nous testons si une commodité passe déjà par l'arc (i, j) .
Si $\max[i][j] = -1$ aller à Etape 5.
Sinon aller à Etape 6.

– Etape 5

Il n'y a aucune commodité qui passe déjà par (i, j) , l'arc choisi pour la commodité k est donc (i, j) .
 $\max[i][j] = M_{ij}^k$
 $E[k] = (i, j)$
 $List = List \setminus a_{(i,j)}^k$
Nous avons la contrainte : $x_{ij}^k + y^k = 1$.
Aller à Etape 9.

– Etape 6

Nous devons tester si la diminution engendrée par le choix de l'arc (i, j) pour la commodité k ne diminue pas trop la taxe maximale potentielle sur l'arc (i, j) .
Si $0,75 * \max[i][j] < M_{ij}^k$ aller à Etape 7.
Sinon aller à Etape 8.

– Etape 7

$E[k] = (i, j)$
 $List = List \setminus a_{(i,j)}^k$
Nous avons la contrainte : $x_{ij}^k + y^k = 1$.
Aller à Etape 9.

– Etape 8

$List = List \setminus a_{(i,j)}^k$
 $\text{indice}[k] = \text{indice}[k] + 1$
 $List = List \cup a_{\text{indice}[k]}^k$
Aller à Etape 9.

– Etape 9

Si $List = \phi$, on résoud le problème inverse (5.47)-(5.60) où l’arc choisi pour la commodité k est $E[k]$ et s’arrêter.

Sinon aller à Etape 2.

Heuristique 3

Pour cette heuristique nous faisons exactement le même raisonnement que dans l’heuristique précédente mais en tenant compte de la demande n^k (nombre d’utilisateurs de la commodité k). Nous prenons donc en compte cette valeur lors du calcul de la diminution :

$$0,75 * max[i][j] < n^k * M_{ij}^k$$

Nous sauvons dans $max[i][j]$ non plus $M_{ij}^{k^*}$ mais $n^{k^*} * M_{ij}^{k^*}$ où k^* est la commodité qui passe par l’arc i, j et qui a la borne supérieure la plus grande parmi toutes celles qui passent par cet arc.

5.3.3 Heuristiques 4 et 5

Nous allons choisir les arcs pour chaque commodité de la même façon que pour l’heuristique 2. La seule différence est que nous allons tenir compte de la perte engendrée lors du choix d’un autre arc.

En effet, pour l’heuristique 2, nous testons si la diminution provoquée par le choix de l’arc (i, j) est inférieure ou non à 75% de la borne supérieure sur cet arc. Si la diminution est supérieure à cette valeur, nous choisissons un autre arc. Mais peut-être que la diminution engendrée par le choix de ce nouvel arc est supérieure à celle que nous obtenions si nous choisissons l’arc (i, j) .

Examinons l’exemple, il représente (en gras) les choix d’arcs effectués après l’application de l’heuristique 2 :

	(1-2)	(1-3)	(2-1)	(2-3)	(3-2)
1	2	20	3	6	8
2	5	4	3	2	2
3	1	14	5	6	5

Nous voyons clairement que le choix de l’arc (2-3) pour la commodité 3 est finalement moins bon que celui que nous aurions effectué en appliquant l’heuristique 1. En effet, la diminution due au choix de l’arc (2-3) ($6-14 = -8$) est supérieure à celle due au choix de l’arc (1-3) ($14-20 = -6$).

Pour éviter de faire ce type de choix, nous comparons donc la diminution entre la borne supérieure de l’arc et la borne supérieure sur cet arc pour la commodité

k et la diminution entre la borne supérieure sur cet arc pour la commodité k et la plus grande borne supérieure inférieure à cette dernière pour la commodité k :

$$\max[i][j] - M_{ij}^k < M_{ij}^k - M_{lm}^k,$$

où M_{lm}^k est la plus grande borne supérieure inférieure à M_{ij}^k pour la commodité k .

Heuristique 4

Toutes les étapes sont les mêmes que celles de l'heuristique 2 sauf l'étape 6.

– Etape 6

Nous devons tester si la diminution engendrée par le choix de l'arc (i, j) pour la commodité k ne diminue pas trop la taxe maximale potentielle sur l'arc (i, j)

Si $0,75 * \max[i][j] < M_{ij}^k$ aller à Etape 7.

Sinon aller à Etape 6Bis.

– Etape 6Bis

Nous devons tester si la diminution engendrée par le choix de l'arc ayant la plus grande borne supérieure inférieure à M_{ij}^k est inférieure ou non à celle engendrée par le choix de (i, j) .

$$a_{lm}^k = a_{\text{indice}[k+1]}^k$$

Si $\max[i][j] - M_{ij}^k > M_{ij}^k - M_{lm}^k$ où a_{lm}^k est l'arc ayant la plus grande borne supérieure inférieure à M_{ij}^k pour la commodité k et M_{lm}^k est la valeur de cette borne supérieure, aller à Etape 8.

Sinon aller à Etape 7.

Si nous considérons à nouveau l'exemple ci-dessus, nous obtenons, après application de l'heuristique 4, le résultat suivant :

	(1-2)	(1-3)	(2-1)	(2-3)	(3-2)
1	2	20	3	6	8
2	5	4	3	2	2
3	1	14	5	6	5

En effet, $14 < 0,75 * 20 = 15$ mais $20 - 14 = 6 < 14 - 6 = 8$ alors nous choisissons quand même l'arc (1-3).

Heuristique 5

Pour cette heuristique nous faisons exactement le même raisonnement que dans l'heuristique précédente mais en tenant compte de la demande n^k (nombre d'utilisateurs de la commodité k). Nous prenons donc en compte cette valeur lors du calcul de la diminution :

$$0,75 * \max[i][j] < n^k * M_{ij}^k$$

mais aussi lors du choix du nouvel arc : $a_{lm}^k = a_{indice[k+1]}^k$

$$max[i][j] - n^k * M_{ij}^k > n^k * (M_{ij}^k - M_{lm}^k).$$

Encore une fois, nous sauvons dans $max[i][j]$ non plus $M_{ij}^{k^*}$ mais $n^{k^*} * M_{ij}^{k^*}$ où k^* est la commodité qui passe par l'arc (i, j) et qui a la borne supérieure la plus grande parmi toutes celles qui passent par cet arc.

5.3.4 Heuristiques 6 et 7

Dans les heuristiques 2,3,4 et 5, nous prenions comme référence la borne supérieure maximale sur l'arc que l'on considérait. On pourrait se demander si prendre la moyenne des bornes supérieures ne serait pas plus représentatif. En effet, considérons l'exemple suivant où les valeurs mises en gras sont celles obtenues après l'application des heuristiques 2 et 4 :

	(1-2)	(1-3)	(2-1)	(2-3)	(3-2)
1	12	20	5	2	7
2	9	15	6	6	6
3	3	15	7	5	5
4	8	14	5	10	1

- Selon l'heuristique 2, nous ne choisissons pas l'arc (1-3) pour la commodité 4 car : $0,75 * 20 > 14$
- Selon l'heuristique 4, nous n'attribuons pas l'arc (1-3) à la commodité 4 car : $0,75 * 20 > 14$ et $20 - 14 > 14 - 10$.

Et pourtant, avec les choix faits ci-dessus, la valeur optimale sera obtenue en mettant une taxe de 15 sur l'arc (1-3), ce qui est assez proche de 14 finalement. Si nous avons comparé le choix de l'arc (1-3) de la commodité 4 avec la moyenne des bornes supérieures sur l'arc (1-3) des commodités qui utilisent déjà l'arc (celles en gras dans la colonne (1-3)) (nous utilisons toujours les mêmes tests que pour les heuristiques 2 et 4 mais en se basant sur la moyenne) nous aurions obtenus :

	(1-2)	(1-3)	(2-1)	(2-3)	(3-2)
1	12	20	5	2	7
2	9	15	6	6	6
3	3	15	7	5	5
4	8	14	5	10	1

En effet, la moyenne des bornes supérieures sur l'arc (1-3) (avant de faire le choix pour la commodité 4) valait : $\frac{20+15+15}{3} = \frac{50}{3}$.

En considérant la moyenne nous avons : $0,75 * \frac{50}{3} = \frac{50}{4} = 12,5$, ce qui est bien inférieur à 14. Nous aurions donc choisi l'arc (1-3) pour la commodité 4.

Heuristique 6

– Etape 1

Classons les M_{ij}^k par ordre décroissant pour chaque commodité k :

$$M_1^k \geq M_2^k \geq M_2^k \geq \dots \geq M_{|\mathcal{A}|}^k$$

Initialisation :

$indice[k] = 1 \forall k \in K$, ce vecteur permet de sauver l'indice de l'arc ayant la $indice[k]$ -ième borne supérieure la plus grande pour la commodité k .

$E[k] = (o_k, d_k) \forall k \in K$.

$List[k] = a_1^k \forall k \in K$, où a_i^k est l'arc ayant la i -ième borne supérieure la plus grande pour la commodité k .

$moyenne[i][j] = -1 \forall (i, j) \in \mathcal{A}$, où $moyenne[i][j]$ contient la somme des bornes supérieures placées sur l'arc (i, j) .

$nbCommodite[i][j] = 0 \forall (i, j) \in \mathcal{A}$, où $nbCommodite[i][j]$ contient le nombre de commodités qui utilisent l'arc (i, j) .

– Etape 2

Nous prenons l'arc parmi les éléments de List ayant la borne supérieure la plus grande :

$$a_{(i,j)}^k = arg \max_{a_i^k \in List} M_{ij}^k$$

Si $a_{(i,j)}^k = 0$ aller à Etape 3.

Sinon aller à Etape 4.

– Etape 3

Si le maximum de List est 0, cela veut dire que tous les arcs qui sont dans List valent 0, toutes les commodités n'ayant pas encore choisi d'arc passeront donc par l'arc (o_k, d_k) , on vide donc la liste : $List = \phi$

Nous avons la contrainte : $y^k = 1$ pour chaque commodité utilisant (o_k, d_k) .

Aller à Etape 9.

– Etape 4

Nous testons si une commodité passe déjà par l'arc (i, j) .

Si $nbCommodite[i][j] = 0$ aller à Etape 5.

Sinon aller à Etape 6.

– Etape 5

Il n'y a aucune commodité qui passe déjà par (i, j) , l'arc choisi pour la commodité k est donc (i, j) .

$moyenne[i][j] = M_{ij}^k$ et $nbCommodite[i][j] = 1$

$E[k] = (i, j)$

$List = List \setminus a_{(i,j)}^k$

Nous avons la contrainte : $x_{ij}^k + y^k = 1$.

Aller à Etape 9.

– Etape 6

Nous devons tester si la diminution engendrée par le choix de l'arc (i, j) pour la commodité k ne diminue pas trop la taxe maximale potentielle sur l'arc (i, j)

Si $0,75 * \frac{\text{moyenne}[i][j]}{\text{nbCommodite}[i][j]} < M_{ij}^k$ aller à Etape 7.

Sinon aller à Etape 6Bis.

– Etape 6Bis

Nous devons tester si la diminution engendrée par le choix de l'arc ayant la plus grande borne supérieure inférieure à M_{ij}^k est inférieure ou non à celle engendrée par le choix de (i, j) .

$$a_{lm}^k = a_{\text{indice}[k+1]}^k$$

Si $\frac{\text{moyenne}[i][j]}{\text{nbCommodite}[i][j]} - M_{ij}^k > M_{ij}^k - M_{lm}^k$ où a_{lm}^k est l'arc ayant la plus grande borne supérieure inférieure à M_{ij}^k pour la commodité k et M_{lm}^k est la valeur de cette borne supérieure, aller à Etape 8.

Sinon aller à Etape 7.

– Etape 7

$$E[k] = (i, j)$$

$$\text{moyenne}[i][j] = \text{moyenne}[i][j] + M_{ij}^k \text{ et } \text{nbCommodite}[i][j] = \text{nbCommodite}[i][j] + 1$$

$$\text{List} = \text{List} \setminus a_{(i,j)}^k$$

Nous avons la contrainte : $x_{ij}^k + y^k = 1$.

Aller à Etape 9.

– Etape 8

$$\text{List} = \text{List} \setminus a_{(i,j)}^k$$

$$\text{indice}[k] = \text{indice}[k] + 1$$

$$\text{List} = \text{List} \cup a_{\text{indice}[k]}^k$$

Aller à Etape 9.

– Etape 9

Si $\text{List} = \phi$, on résout le problème inverse (5.47)-(5.60) où l'arc choisi pour la commodité k est $E[k]$ et s'arrêter.

Sinon aller à Etape 2.

Heuristique 7

Pour cette heuristique nous faisons exactement le même raisonnement que dans

l'heuristique précédente mais en tenant compte de la demande n^k (nombre d'utilisateurs de la commodité k). Nous prenons donc en compte cette valeur lors du calcul de la diminution :

$$0,75 * \frac{\text{moyenne}[i][j]}{\text{nbCommodite}[i][j]} < n^k * M_{ij}^k$$

mais aussi lors du choix du nouvel arc : $a_{lm}^k = a_{\text{indice}[k+1]}^k$

$$\frac{\text{moyenne}[i][j]}{\text{nbCommodite}[i][j]} - n^k * M_{ij}^k > n^k * (M_{ij}^k - M_{lm}^k).$$

Nous ajoutons $n^k * M_{ij}^k$ dans $\text{moyenne}[i][j]$ et non plus M_{ij}^k .

5.3.5 Heuristique 8

Dans les précédentes heuristiques, nous avons toujours utilisé des contraintes booléennes pour les variables de flot (x_{ij}^k et y^k). En plus des inégalités triangulaires, résoudre le problème avec ce type de variables rend la résolution difficile et donc longue.

Dans cette heuristique, nous allons essayer d'utiliser les résultats obtenus avec la relaxation de la méthode exacte pour choisir les arcs que nous allons attribuer à chaque commodité.

Suite à la résolution de la relaxation du problème, nous allons obtenir des x_{ij}^k et y^k non plus entiers mais réels.

La méthode de sélection des arcs est la suivante :

- Si $x_{ij}^k = 0$: $\forall(i, j)$, nous aurons la contrainte $y^k = 1$ et l'arc sélectionné sera l'arc (o_k, d_k) .
- Si $x_{ij}^k > 0$ pour au moins un arc $(i, j) \in \mathcal{A}$, nous allons choisir l'arc tel que le produit $M_{ij}^k * x_{ij}^k$ est maximal pour la commodité k . Nous aurons alors la contrainte $x_{ij}^k + y^k = 1$.

Heuristique 8

- Etape 1.
Résoudre la relaxation de (HTSP2) : (5.27) - (5.39).
- Etape 2.
 $k = 1$.
- Etape 3.

$$(i, j)^* = \arg \max_{i, j} M_{ij}^k * x_{ij}^k$$

Si $x_{(i, j)^*}^k = 0$, ajouter la contrainte : $y^k = 1$ et l'arc choisi pour la commodité k sera (o_k, d_k) .

Sinon ajouter la contrainte : $x_{(i,j)^*}^k + y^k = 1$ et l'arc choisi pour la commodité k sera $(i, j)^*$.

– Etape 4.

Si $k = |K|$ résoudre le problème inverse (5.47)-(5.60) et s'arrêter.

Sinon $k = k + 1$ et aller à Etape 3.

5.3.6 Heuristique 9

Nous adoptons une démarche semblable à celle de l'heuristique précédente mais nous allons un peu changer la méthode du choix des arcs.

En effet, nous ne tenons compte que du produit $x_{ij}^k * M_{ij}^k$ pour déterminer quel arc nous allons choisir. L'idée est de donner la priorité aux x_{ij}^k les plus élevés, c'est à dire que, même si le résultat du produit entre la borne supérieure et le flot n'est pas le plus élevé mais que le flot est supérieur à une borne donnée, nous choisirons cet arc.

Heuristique 9

– Etape 1.

Résoudre la relaxation de (HTSP2) : (5.27) - (5.39).

– Etape 2.

$k = 1$.

– Etape 3.

$$(i, j)^* = \arg \max_{i,j} x_{ij}^k$$

Si $x_{(i,j)^*}^k > 0.75$, ajouter la contrainte $x_{ij}^k + y^k = 1$ et l'arc choisi pour la commodité k sera $(i, j)^*$ et aller à Etape 5.

Sinon aller à Etape 4.

– Etape 4.

$$(k, l)^* = \arg \max_{i,j} M_{ij}^k * x_{ij}^k$$

Si $x_{(k,l)^*}^k = 0$, ajouter la contrainte : $y_k = 1$ et l'arc choisi pour la commodité k sera (o_k, d_k) .

Sinon ajouter la contrainte : $x_{(k,l)^*}^k + y^k = 1$ et l'arc choisi pour la commodité k sera $(k, l)^*$.

– Etape 5.

Si $k = |K|$, résoudre le problème inverse (5.47)-(5.60) et s'arrêter.

Sinon $k = k + 1$ et aller à Etape 3.

5.3.7 Heuristique 10

Nous avons essayé, dans toutes les heuristiques précédentes, d'initialiser le problème afin d'obtenir les meilleurs résultats possibles. Jamais nous n'avons essayé de tenir compte du résultat obtenu après la résolution du problème inverse. En effet, imaginons le cas suivant :

	(1-2)	(1-3)	(2-1)	(2-3)	(3-2)	(3-1)
1	7	20	19	12	5	4
2	5	3	1	7	4	4
3	7	3	4	5	3	4
4	7	3	2	5	3	4
5	3	3	2	7	3	4
6	3	2	1	7	1	3
7	7	3	3	1	1	2

Si nous appliquons l'heuristique 1 nous obtenons :

	(1-2)	(1-3)	(2-1)	(2-3)	(3-2)	(3-1)
1	7	20	19	12	5	4
2	5	3	1	7	4	4
3	7	3	4	5	3	4
4	7	3	2	5	3	4
5	3	3	2	7	3	4
6	3	2	1	7	1	3
7	7	3	3	1	1	2

Avec ce choix et les inégalités triangulaires nous avons : $t_{13} \leq t_{12} + t_{23}$ et donc $t_{13} \leq (7+7)$. Nous ne pourrions mettre qu'une taxe de 14 sur l'arc (1-3) alors que la plus grande borne supérieure sur l'arc (1-3) est 20. Nous pouvons constater que ce choix d'arc n'est pas si intéressant que ça.

Pour éviter ce genre de situation, nous pouvons, après avoir effectué la résolution inverse, vérifier que les taxes placées ne soient pas trop différentes des bornes supérieures. Par exemple, si la taxe sur l'arc (i, j) est inférieure à 75% de la borne supérieure M_{ij}^k placée sur l'arc (i, j) , nous choisissons un autre arc. Nous choisissons celui qui a la plus grande borne supérieure inférieure à M_{ij}^k .

Revenons à l'exemple ci-dessus. Après résolution du problème inverse, la taxe maximale que nous pouvons placer sur l'arc (1-3) est égale à 14 et $14 < 0,75 * 20 = 15$, nous changeons alors notre choix pour la commodité 1. Nous choisissons l'arc

(2-1) car c'est l'arc ayant la plus grande borne supérieure inférieure à 20 pour la commodité 1.

Nous obtenons donc les choix suivants :

	(1-2)	(1-3)	(2-1)	(2-3)	(3-2)	(3-1)
1	7	20	19	12	5	4
2	5	3	1	7	4	4
3	7	3	4	5	3	4
4	7	3	2	5	3	4
5	3	3	2	7	3	4
6	3	2	1	7	1	3
7	7	3	3	1	1	2

Les inégalités triangulaires sont toutes respectées et la solution vaut : $(7+7+7) + 19 + (7+7+7) = 61$.

Une telle heuristique a besoin d'une condition d'arrêt. En effet, lorsque nous changeons le choix des arcs attribués à chaque commodité, nous résolvons à nouveau le problème. Mais, il n'y a aucune raison que le problème évoqué précédemment ne se représente pas avec le nouveau choix d'arcs. Nous devons donc refaire la même opération. Une des conditions d'arrêt possible est que la nouvelle solution obtenue soit moins bonne que la précédente, mais comme il n'y pas spécialement de raison que la nouvelle solution soit meilleure que la précédente, nous pouvons aussi exécuter un nombre fixé de fois la recherche de nouveaux arcs. Nous sauvegardons la meilleure obtenue lors des différentes itérations.

Heuristique 10

– Etape 1.

$$k = 1.$$

$$Sol[k] = \phi \forall k \in K.$$

$$Indice[k] = 1 \forall k \in K.$$

$$cpt = 0.$$

– Etape 2.

Prendre pour la commodité k la plus grande borne supérieure M_{ij}^k .

L'arc sélectionné est : $(i, j)^* = a_i^k$ où a_i^k est l'arc ayant la i -ième borne supérieure la plus grande pour la commodité k .

– Etape 3.

Si $M_{(i,j)^*} \neq 0$, on a la contrainte : $x_{(i,j)^*}^k + y^k = 1$ et $Sol[k] = (i, j)^*$

Sinon on a la contrainte : $y^k = 1$ et $Sol[k] = (o_k, d_k)$.

- Etape 4.
Si $k = |K|$ aller à Etape 5. Sinon, $k = k + 1$ et aller à Etape 2.
- Etape 4Bis.
Si $cpt = 10$, aller à Etape9.
Sinon $cpt = cpt + 1$.
- Etape 5.
Résoudre le problème inverse (5.47)-(5.60). $Taxe[i, j]$ contient les taxes mises sur les arcs $(i, j) \quad \forall (i, j) \in \mathcal{A}$ et $k = 1$.
- Etape 6.
Si $Taxe[Sol(k)] < 0.75 * M_{Sol[k]}^k$, aller Etape 7.
Sinon aller à Etape 8.
- Etape 7.
Choisir l'arc suivant ayant la plus grande borne supérieure inférieure à la borne courante.
 $Indice[k] = Indice[k] + 1$.
 $(k, l)^* = a_{Indice[k]}^k$.
Si $M_{(k,l)^*} \neq 0$ on a la contrainte : $x_{(k,l)^*}^k + y^k = 1$ et $Sol[k] = (k, l)^*$
Sinon on a la contrainte : $y^k = 1$ et $Sol[k] = (o_k, d_k)$.
- Etape8.
Si $k = |K|$ aller à Etape 5. Sinon $k = k + 1$ et aller à Etape6.

Pour mieux comprendre ce que fait l'algorithme, nous allons détailler l'exécution de celui-ci sur l'exemple présenté en début d'heuristique.

Reprenons donc l'exemple suivant :

	(1-2)	(1-3)	(2-1)	(2-3)	(3-2)	(3-1)
1	7	20	19	12	5	4
2	5	3	1	7	4	4
3	7	3	4	5	3	4
4	7	3	2	5	3	4
5	3	3	2	7	3	4
6	3	2	1	7	1	3
7	7	3	3	1	1	2

Considérons l'algorithme à partir de l'étape 1 jusqu'à l'étape 4. Nous allons choisir, pour chaque commodité, les arcs ayant les plus grandes bornes supérieures. En gras dans le tableau ci-dessous.

	(1-2)	(1-3)	(2-1)	(2-3)	(3-2)	(3-1)
1	7	20	19	12	5	4
2	5	3	1	7	4	4
3	7	3	4	5	3	4
4	7	3	2	5	3	4
5	3	3	2	7	3	4
6	3	2	1	7	1	3
7	7	3	3	1	1	2

Tous les $M_{(i,j)^*}$ sont différents de 0, nous avons donc pour chaque commodité $:x_{(i,j)^*}^k + y^k = 1$ et $Sol[k] = (i, j)^*$.

Nous avons alors considéré toutes les commodités, nous résolvons le problème inverse (5.47)-(5.60) (Etape 5).

Considérons les inégalités triangulaires suivantes :

1. $t_{12} \leq t_{13} + t_{32}$
2. $t_{13} \leq t_{12} + t_{23}$
3. $t_{21} \leq t_{23} + t_{31}$

La première et la troisième inégalité sont toujours vérifiées car il n'y a aucune commodité qui passe par l'arc (3-2) ou (3-1). Par contre, si nous voulons placer une taxe de 20 sur l'arc (1-3), la deuxième inégalité n'est pas vérifiée : $t_{13} \leq 7 + 7$. La borne maximale que nous pourrions placer sur l'arc (1-3), obtenue après résolution du problème inverse, est 14.

Le vecteur $Taxe[i, j]$ contient les taxes placées sur les arcs (i, j) après la résolution du problème inverse.

Reprenons l'algorithme où nous l'avons laissé : Etape 6. Considérons la commodité 1, nous voyons que $14 < 0,75 * 20$, nous décidons de modifier notre choix pour la commodité 1, on se rend à l'étape 7. Nous choisissons l'arc suivant pour la commodité 1. Cet arc est l'arc (2-1) et a une borne supérieure égale à 19 pour la commodité 1.

Pour toutes les autres commodités, nous ne devons pas modifier notre choix d'arc. Nous arrivons à l'étape 8 en ayant considéré toutes les commodités, nous résolvons à nouveau le problème inverse.

Considérons les inégalités triangulaires suivantes :

1. $t_{12} \leq t_{13} + t_{32}$
2. $t_{21} \leq t_{23} + t_{32}$
3. $t_{21} \leq t_{23} + t_{31}$

Cette fois-ci toutes les inégalités sont vérifiées car il n'y a aucune commodité qui passe par l'arc (3-2) ou (3-1).

Les taxes correspondent aux bornes supérieures placées sur les arcs, nous ne modifierons pas nos choix d'arcs.

5.3.8 Heuristique 11

Dans toutes les heuristiques déjà présentées, nous ne nous concentrons que sur les bornes supérieures déjà "placées" sur l'arc que nous considérons. Mais le problème est rendu difficile à cause des inégalités triangulaires.

Nous allons donc essayer de prendre en compte, dans nos choix, l'environnement. Ce que nous entendons par environnement, ce sont les arcs situés aux alentours de l'arc que nous considérons. De cette façon nous espérons éviter le cas suivant : nous choisissons un arc car on peut y placer une grande borne supérieure mais nous ne pouvons y placer une taxe proche de sa borne supérieure car les taxes apparaissant dans l'environnement de l'arc sont peu élevées (problème dû aux inégalités triangulaires).

L'environnement de l'arc courant sera constitué au maximum de quatre arcs. Si l'arc courant est (i, j) , l'environnement de l'arc est :

- $(i + 1, j)$
- $(i, i + 1)$
- $(i, i - 1)$
- $(i - 1, j)$

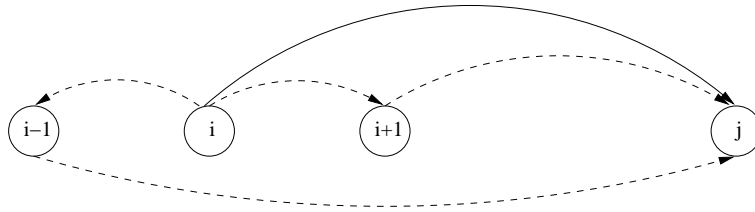


FIG. 5.6 – Environnement de l'arc (i, j) .

Nous devons bien entendu faire attention à ne pas considérer deux fois le même arc.

Nous allons prendre en considération cet environnement lors de nos choix d'arcs. Pour cette heuristique, nous allons nous y prendre comme dans l'heuristique 2 (sauf que l'on considère la moyenne des bornes supérieures placées sur les arcs et non le maximum de ces dernières) en ajoutant une contrainte sur M_{ij}^k : M_{ij}^k ne doit pas être inférieur à 60% de la moyenne des bornes supérieures de l'environnement.

Nous devons donc calculer la moyenne des bornes supérieures de l'environnement. Nous notons \mathcal{E}_{ij} , l'ensemble des arcs dans l'environnement de l'arc (i, j) .

$$moyEnv_{ij} = \frac{1}{|\mathcal{E}_{ij}|} \sum_{(a,b) \in \mathcal{E}_{ij}} moy(a, b),$$

où $moy(ab)$ est la moyenne des bornes supérieures sur l'arc (a, b) :

$$moy(a, b) = \frac{1}{|\mathcal{F}_{ab}|} \sum_{k \in \mathcal{F}_{ab}} M_{ab}^k,$$

où \mathcal{F}_{ab} est l'ensemble des commodités qui ont choisi l'arc (a, b) selon les règles expliquées ci-dessus.

Heuristique 11

– Etape 1.

Classons les M_{ij}^k par ordre décroissant pour chaque commodité k :

$$M_1^k \geq M_2^k \geq M_3^k \geq \dots \geq M_{|\mathcal{A}|}^k$$

Initialisation :

$indice[k] = 1 \forall k \in K$, ce vecteur permet de sauver l'indice de l'arc ayant la $indice[k]$ -ième borne supérieure la plus grande pour la commodité k .

$E[k] = (o_k, d_k) \forall k \in K$.

$List[k] = a_1^k \forall k \in K$, où a_1^k est l'arc ayant la 1-ième borne supérieure la plus grande pour la commodité k .

$mat[i][j] = \phi \forall (i, j) \in \mathcal{A}$, cette matrice contient les commodités qui passent par les arcs (i, j) et aussi les bornes supérieures pour chacune des commodités.

– Etape 2.

Nous prenons l'arc parmi les éléments de List ayant la borne supérieure la plus grande :

$$a_{(i,j)}^k = arg \max_{a_i^k \in List} M_{ij}^k$$

Si $a_{(i,j)}^k = 0$ aller à Etape 3.
Sinon aller à Etape 4.

– Etape 3.

Si le maximum de List est 0, cela veut dire que tous les arcs qui sont dans List valent 0, toutes les commodités n'ayant pas encore choisi d'arc passeront donc par l'arc (o_k, d_k) , on vide donc la liste : $List = \phi$
Nous avons la contrainte : $y^k = 1$ pour chaque commodité utilisant (o_k, d_k) .
Aller à Etape 8.

– Etape 4.

Nous devons tester si la diminution engendrée par le choix de l'arc (i, j) pour la commodité k ne diminue pas trop la moyenne des taxes potentielles sur l'arc (i, j) .
Si $0,75 * moy(mat[i][j]) < M_{ij}^k$ où

$$moy(mat[i][j]) = \frac{1}{|\mathcal{F}_{ij}|} \sum_{l \in \mathcal{F}_{ij}} M_{ij}^l,$$

aller à l'étape 5.

Sinon aller à Etape 7.

– Etape 5.

Nous devons tester si la diminution engendrée par le choix de l'arc (i, j) pour la commodité k ne diminue pas trop la moyenne des taxes potentielles dans l'environnement de l'arc (i, j) .
Si $0,6 * moyEnv(i, j) < M_{ij}^k$ où

$$moyEnv_{ij} = \frac{1}{|\mathcal{E}_{ij}|} \sum_{(a,b) \in \mathcal{E}_{ij}} moy(mat[a][b]),$$

aller à l'étape 6.

Sinon aller à Etape 7.

– Etape 6.

$E[k] = (i, j)$
 $mat[i][j] = mat[i][j] \cup (i, j, k)$
 $List = List \setminus a_{(i,j)}^k$
Nous avons la contrainte : $x_{ij}^k + y^k = 1$.
Aller à Etape 9.

– Etape 7

$List = List \setminus a_{(i,j)}^k$

$indice[k] = indice[k] + 1$
 $List = List \cup a_{indice[k]}^k$
 Aller à Etape 8.

– Etape 8.

Si $List = \phi$, on résout le problème inverse (5.47)-(5.60) où l'arc choisi pour la commodité k est $E[k]$ et s'arrêter.

Sinon aller à Etape 2.

Pour mieux comprendre l'algorithme, regardons l'exemple suivant :

	(1-2)	(1-3)	(2-1)	(2-3)	(3-2)	(3-1)
1	0	0	0	0	12	0
2	0	0	4	0	0	0
3	0	0	0	28	0	0
4	0	9	0	0	0	0
5	6	0	0	0	0	0
6	0	0	0	0	0	12

En appliquant l'heuristique 2 nous aurions choisi les arcs suivants :

	(1-2)	(1-3)	(2-1)	(2-3)	(3-2)	(3-1)
1	0	0	0	0	10	0
2	0	0	4	0	0	0
3	0	0	0	28	0	0
4	0	9	0	0	0	0
5	6	0	0	0	0	0
6	0	0	0	0	0	12

Si nous regardons toutes les inégalités triangulaires :

$$t_{12} \leq t_{13} + t_{32} \Leftrightarrow 6 \leq 9 + 10$$

$$t_{13} \leq t_{12} + t_{23} \Leftrightarrow 9 \leq 6 + 28$$

$$t_{21} \leq t_{23} + t_{31} \Leftrightarrow 4 \leq 28 + 12$$

$$t_{23} \leq t_{21} + t_{13} \Leftrightarrow 28 \leq 4 + 9$$

$$t_{31} \leq t_{32} + t_{21} \Leftrightarrow 12 \leq 10 + 4$$

$$t_{32} \leq t_{31} + t_{12} \Leftrightarrow 10 \leq 12 + 6$$

Nous voyons que l'inégalité triangulaire sur t_{23} impose que la taxe maximal que nous pouvons placer est de 13.

Reprenons l'exemple à son état initial et choisissons les arcs mais en tenant compte de l'environnement :

– Environnement de $(2,3) = (2,1) (1,3)$

- Environnement de (3,2) = (3,1) (1,2)
 - Environnement de (3,1) = (3,2) (2,1)
 - Environnement de (1,2) = (1,3) (3,2)
 - Environnement de (1,3) = (1,3) (3,2)
 - Environnement de (2,1) = (2,3) (3,1)
1. commodité 3 : aucun arc utilisé dans l'environnement de (2-3) → nous choisissons l'arc (2-3).
 2. commodité 6 : aucun arc utilisé dans l'environnement (3-1) → nous choisissons l'arc (3-1).
 3. commodité 1 : aucun arc utilisé dans l'environnement (3-2) → nous choisissons l'arc (3-2).
 4. commodité 4 : l'arc (2-3) est utilisé dans l'environnement de (1-3) : $9 > 0.6 * \frac{28}{2}$ nous choisissons l'arc (1-3).
 5. commodité 5 : les arc (1-3) et (3-2) sont utilisés dans l'environnement de (1-2) : $6 > 0.6 * \frac{10+9}{2}$ nous choisissons l'arc (1-2).
 6. commodité 2 : les arc (2-3) et (3-1) sont utilisés dans l'environnement de (2-1) : $4 < 0.6 * \frac{12+28}{2}$ nous ne choisissons pas l'arc (2-1).
 7. commodité 2 : plus d'arc taxé disponible, nous choisissons l'arc (o_k, d_k) .

Nous obtenons donc les choix suivants :

	(1-2)	(1-3)	(2-1)	(2-3)	(3-2)	(3-1)
1	0	0	0	0	10	0
2	0	0	4	0	0	0
3	0	0	0	28	0	0
4	0	9	0	0	0	0
5	6	0	0	0	0	0
6	0	0	0	0	0	12

Si nous regardons à nouveau l'inégalité triangulaire : $t_{23} \leq t_{21} + t_{13}$, nous n'avons plus le problème précédemment évoqué car il n'y a plus aucune commodité qui passe par l'arc (2-1).

5.3.9 Heuristique 12

Dans toutes les heuristiques précédentes, nous avons utilisé des "magic numbers" pour déterminer si nous choisissons un arc ou non. Les valeurs utilisées n'étaient peut-être pas toujours les plus adaptées au problème. Il aurait été intéressant de pouvoir adapter ces valeurs en fonction du problème.

Afin d'atteindre ce type d'objectif, nous allons utiliser un algorithme de classification non supervisée. Nous allons nous inspirer de l'algorithme de Lumer et Faieta [13] et AntClass, tous deux présentés dans Monmarché [14].

Classification non supervisée

Une méthode de classification non supervisée n'utilise que la dissimilarité entre les éléments d'un ensemble pour partitionner ce dernier en plusieurs classes. Le but de la classification non supervisée est de trouver, parmi un ensemble d'objets, des classes d'objets. Le but de la classification supervisée est, lui, d'utiliser les groupes connus d'objets pour découvrir ce qui les différencie.

Ce qui nous intéresse ici, peut-être assimilé à un problème de partitionnement. Le problème de partitionnement consiste à trouver des partitions parmi N points dans un ensemble à M dimensions telles que les points d'une partition soient plus similaires entre eux qu'avec les points des autres groupes. Le nombre de classes peut ne pas être connu.

Ce que nous voulons faire ressemble bien à un problème de partitionnement car nous voulons que les bornes placées sur un même arc (i, j) soient semblables mais qu'elles soient aussi semblables aux bornes placées sur les arcs adjacents à (i, j) , par contre l'aspect "classes" ne nous intéresse pas.

Partitionnement à l'aide de fourmis

Ce type de méthodes se base sur le comportement de fourmis réelles. En effet, il a été observé [7] que les fourmis étaient capables d'organiser spatialement plusieurs éléments de la fourmilière (les oeufs, les larves et les nymphes).

Le modèle de règles utilisées pour la classification est relativement simple :

- La probabilité qu'une fourmi s'empare d'un élément est d'autant plus grande que l'élément est isolé.
- La probabilité qu'une fourmi dépose l'élément qu'elle transporte est d'autant plus grande que la densité d'éléments du même type dans le voisinage est grande.

Lumer et Faieta [13] ont proposé un algorithme utilisant une mesure de dissimilarité entre les objets. Cette mesure de dissimilarité était exprimée sous forme de

distance euclidienne : $d(x, y) = \left(\sum_{k=1}^M |x_k - y_k|^2 \right)^{\frac{1}{2}}$.

Les objets, comme indiqué ci-dessus, sont de dimension M mais sont représentés dans un espace de dimension moindre, typiquement de dimension 2. Nous pouvons donc voir cet espace comme une grille G . Les fourmis se déplacent sur G et perçoivent une région (l'environnement) R_s de $s \times s$ cases.

Lumer et Faieta ont donné aux fourmis des probabilités de prendre un objet (p_p) ou de le déposer (p_d) :

$$p_p = \left(\frac{k_1}{k_1 + f(o_i)} \right)^2 \quad (5.61)$$

$$p_d = \begin{cases} 2f(o_i) & \text{si } f(o_i) < k_2 s \\ 1 & \text{si } f(o_i) > k_2 s \end{cases} \quad (5.62)$$

où o_i est l'objet considéré, k_1 et k_2 des constantes. La fonction de densité $f(o_i)$ dépend de la position de l'objet sur la grille, de l'objet considéré et de l'entourage, elle est calculée de la manière suivante :

$$f(o_i) = \begin{cases} \frac{1}{s^2} \sum_{o_j \in R_s(r(o_i))} 1 - \frac{d(o_i, o_j)}{\alpha} & \text{si } f > 0 \\ 0 & \text{sinon} \end{cases} \quad (5.63)$$

où $r(o_i)$ est la position de l'objet o_i sur la grille, α est un facteur qui détermine dans quelle mesure la dissimilarité entre deux objets doit être prise en compte et f est la proportion d'éléments perçus dans l'environnement.

$f(o_i)$ est une mesure de similarité moyenne de l'objet o_i avec les objets o_j présents dans son voisinage.

Adaptation de Lumer and Faieta

Ce qui nous intéresse dans l'algorithme de Lumer et Faieta est la gestion du voisinage et non la classification. De plus les probabilités doivent être un peu adaptées.

Voici les modifications que nous allons apporter :

- Lorsque nous regardons la similarité d'un arc (i, j) avec son environnement, nous comparons les moyennes des bornes supérieures placées sur les arcs apparaissant dans l'environnement de (i, j) .
Si suite au test de probabilité de prise d'un élément p_p , nous devons en choisir un, nous prendrons l'élément dont la valeur est la plus éloignée de la moyenne des bornes supérieures placées sur (i, j) .
- Dans le cas des fourmis, un objet isolé sera obligatoirement pris, dans notre cas, si la borne est isolée, c'est à dire qu'il n'y a pas pas d'arcs utilisés par des commodités dans l'environnement de l'arc, nous ne déplacerons pas la borne. La probabilité de prendre doit, dans ce cas, être proche de 0, la densité doit donc être proche de 1. Nous redéfinissons la distance comme suit :

$$d(x, y) = \begin{cases} (|x - y|^2)^{\frac{1}{2}} & \text{si } x \text{ et } y \neq 0 \\ 0 & \text{sinon.} \end{cases}$$

- Les déplacements ne sont plus à faire sur une grille mais d'arc en arc. Si l'arc courant est l'arc (i, j) pour la commodité k , l'arc suivant dans le déplacement sera l'arc pour lequel la borne supérieure pour la commodité k sera la plus grande mais inférieure à celle de l'arc (i, j) .
- Pour les déplacements nous introduisons aussi une probabilité : nous aurons une probabilité de 0,6 de prendre l'arc suivant pour la commodité k , arc qui aura une borne supérieure inférieure à la borne supérieure courante et une probabilité de 0,4 de prendre l'arc précédent, arc qui aura une borne

supérieure supérieure à la borne courante. De cette façon on pourrait choisir un arc qu'on avait abandonné car il se peut que l'environnement de l'arc ait été modifié depuis le changement effectué. De cette façon nous ne "diminuons" pas systématiquement les bornes que nous plaçons.

- l'environnement de l'arc (i, j) est défini de la même façon que pour l'heuristique précédente.

Nous devons aussi modifier un peu le comportement de l'algorithme. En effet, l'algorithme de Lumer et Faieta est utilisé pour de la classification, nous exécutons donc l'algorithme un nombre déterminé de fois pour qu'il puisse mettre en évidence des classes d'objets. Dans notre cas, le problème se présente différemment, nous allons travailler avec un problème initialisé : nous avons déjà placé les bornes supérieures sur les arcs et nous voulons "affiner" ce placement en tenant compte de l'environnement, c'est à dire que nous voulons des "tas" de bornes supérieures semblables, une seule exécution sera nécessaire (ou un nombre très petit).

Les formules pour les probabilités de prendre et déposer restent identiques à (5.61) et (5.62) mais la fonction de densité (similarité) est modifiée :

$$f(i, j) = \begin{cases} \frac{1}{|\mathcal{E}_{ij}|} \sum_{(k,l) \in \mathcal{E}_{ij}} 1 - \frac{d((i,j),(k,l))}{\alpha} & \text{si } f > 0 \\ 1 & \text{sinon} \end{cases} \quad (5.64)$$

où \mathcal{E}_{ij} est l'ensemble d'arcs dans l'environnement de l'arc (i, j) .

Remarquons que la distance doit aussi être adaptée. Si un des arcs présents dans l'environnement de l'arc courant n'est utilisé par aucune commodité, la distance doit être égale à 0 car nous partons toujours du principe qu'une borne isolée a peu de chance d'être "déplacée". En mettant la distance à 0, nous augmenterons la fonction de densité ce qui diminuera la probabilité de "déplacer" une borne.

Remarquons que nous ne le précisons plus, mais les contraintes $x_{ij}^k + y^k = 1$ et $y^k = 1$ sont gérées comme précédemment.

Heuristique 12

- Etape 1.

Classons les M_{ij}^k par ordre décroissant pour chaque commodité k :

$$M_1^k \geq M_2^k \geq M_3^k \geq \dots \geq M_{|\mathcal{A}|}^k$$

Initialisation :

$indice[k] = 1 \forall k \in K$, ce vecteur permet de sauver l'indice de l'arc ayant la $indice[k]$ -ième borne supérieure la plus grande pour la commodité k .

$E[k] = \phi \forall k \in K$.

$mat[i][j] = \phi \forall (i, j) \in \mathcal{A}$, cette matrice contient les commodités qui passent par les arcs $(i, j) \in \mathcal{A}$ et aussi les bornes supérieures pour chacune des

commodités.

com = un simple compteur utilisé dans l'algorithme.

– Etape 2.

Pour chaque commodité k nous choisissons les arcs ayant la plus grande borne supérieure.

$E[k] = a_1^k \forall k \in K$, où a_1^k est l'arc ayant la 1-ième borne supérieure la plus grande pour la commodité k .

$mat[i][j] = mat[i][j] \cup (i, j, k) \forall a_1^k \in \mathcal{A}$ où les sommets d'origine et de destination de a_1^k sont i et j .

$k = 1$.

– Etape 3.

Nous considérons l'arc choisi pour la commodité $k : a_{ij} = E[k]$ où (i, j) sont les sommets d'origine et de destination de l'arc $E[k]$.

Si $a_{ij} \in \mathcal{A}$ aller à Etape 4.

Sinon $com = k$ et aller à Etape 6.

– Etape 4.

Nous générons un nombre aléatoire (prob) entre 0 et 1.

Si $prob < p_p$ où

$$p_p = \left(\frac{k_1}{k_1 + f(i, j)} \right)^2 \quad (5.65)$$

et

$$f(i, j) = \begin{cases} \frac{1}{|\mathcal{E}_{ij}|} \sum_{(k,l) \in \mathcal{E}_{ij}} 1 - \frac{d((i,j),(k,l))}{\alpha} & \text{si } f > 0 \\ 1 & \text{sinon} \end{cases} \quad (5.66)$$

On sélectionne la borne supérieure la plus éloignée de la moyenne des bornes supérieures de l'arc (i, j)

$$a_{ij}^{com} = \arg \max_{com} dist(moy(i, j), M_{ij}^{com})$$

où

$$moy(i, j) = \frac{1}{|\mathcal{F}_{ij}|} \sum_{f \in \mathcal{F}_{ij}} M_{ij}^f,$$

où \mathcal{F}_{ij} est l'ensemble des commodités qui passent par l'arc (i, j) . Aller à Etape 5.

Sinon aller à Etape 10.

- Etape 5.
Nous devons changer de choix pour la commodité com . $mat[i][j] = mat[i][j] \setminus (i, j, com)$.
Aller à Etape 7.
- Etape 6.
Nous générons un nombre aléatoire ($prob$) entre 0 et 1.
Si $prob < 0,3$ et $indice[com] > 0$ nous essaierons de choisir un arc tarifable pour la commodité com , celui qui précède l'arc non tarifable : $a_{indice[com]-1}^k$.
Aller à Etape 8.
Sinon aller à Etape 10.
- Etape 7.
Nous testons si nous allons choisir l'arc suivant ($a_{indice[com]+1}^k$) ou l'arc précédent ($a_{indice[com]-1}^k$).
Nous générons un nombre aléatoire ($prob$) entre 0 et 1.
Si $prob < 0,6$ et $indice[com] = 0$ alors $indice[com] = indice[com] + 1$.
Sinon $indice[com] = indice[com] - 1$.
- Etape 8.
Nous devons tester si l'arc choisi est semblable à l'environnement.
Nous générons un nombre aléatoire ($prob$) entre 0 et 1.
Si $(a, b) \in \mathcal{A}$ et $prob < p_d$, où

$$p_d = \begin{cases} 2f(a, b) & \text{si } f(a, b) < k_2s \\ 1 & \text{si } f(a, b) > k_2s \end{cases}$$
 où (a, b) sont les sommets d'origine et de destination de $a_{indice[com]}^{com}$.
Aller à Etape 9.
Si $(a, b) \in \mathcal{B}$ (arc non tarifable)
 $E[com] = (o_{com}, d_{com})$ et aller à Etape 10.
Sinon aller à Etape 7.
- Etape 9.
 $E[com] = a_{indice[com]}^{com}$ et $mat[a][b] = (a, b, com)$
Aller à Etape 10.
- Etape 10.
Si $k = |K|$, on résout le problème inverse (5.47)-(5.60) où l'arc choisi pour la commodité k est $E[k]$ et s'arrêter.
Sinon $k = k + 1$ aller à Etape 3.

5.3.10 Heuristique 13

Cette heuristique est semblable à l'heuristique précédente étant donné que nous allons encore utiliser une fonction de similarité pour déterminer si nous devons changer nos choix d'arcs ou non. La définition de l'environnement est toujours la même que celle de l'heuristique 10 et nous initialisons le problème de la même façon.

La définition de la distance ne change pas non plus :

$$d(x, y) = \begin{cases} (|x - y|^2)^{\frac{1}{2}} & \text{si } x \text{ et } y \neq 0 \\ 0 & \text{sinon.} \end{cases}$$

Nous redéfinissons la fonction de similarité comme ci-dessous :

$$f(i, j) = \frac{1}{|\mathcal{E}_{ij}|} \sum_{(k, l) \in \mathcal{E}_{ij}} d((i, j), (k, l))$$

où \mathcal{E}_{ij} est l'ensemble des arcs dans l'environnement de l'arc (i, j) .

La définition de la distance est aussi identique à celle de l'heuristique 11 :

$$d((i, j), (k, l)) = |\text{moy}(i, j) - \text{moy}(k, l)|$$

où

$$\text{moy}(i, j) = \frac{1}{|\mathcal{F}_{ij}|} \sum_{f \in \mathcal{F}_{ij}} M_{ij}^f$$

où \mathcal{F}_{ij} est l'ensemble des commodités qui passent par l'arc (i, j) .

La probabilité de prendre, c'est à dire de déplacer l'une des bornes supérieures sur l'arc (i, j) , est égale à :

$$p_p = f(i, j).$$

Nous devons à nouveau décider si le nouveau choix d'arcs est acceptable ou non, c'est à dire, que la borne supérieure sur cet arc pour la commodité k n'est pas trop différente des bornes sur l'arc et dans l'environnement de l'arc.

Nous allons accepter le choix de l'arc (i, j) pour la commodité k dans deux cas :

- Si l'écart entre la borne supérieure que l'on veut placer sur (i, j) et la moyenne des bornes sur (i, j) est inférieur au plus grand écart entre la moyenne des bornes sur (i, j) et les bornes déjà placées (i, j) :

$$d(M_{ij}^k, \text{moy}(i, j)) < \max_{l \in \mathcal{F}_{ij}} d(\text{moy}(i, j), M_{ij}^l)$$

- Si la borne supérieure que l'on veut placer est supérieure à 0,75 fois la moyenne des bornes présentes sur les arcs dans l'environnement de (i, j) et sur (i, j) :

$$M_{ij}^k > 0,75 * \frac{1}{|\mathcal{E}_{ij}| + 1} \left(\sum_{(k, l) \in \mathcal{E}_{ij}} \text{moy}(k, l) + \text{moy}(i, j) \right)$$

A nouveau ces contraintes favorisent le choix d'un arc isolé lorsque nous voulons choisir un nouvel arc pour "déposer" une borne supérieure et ne favorise pas le choix d'un tel arc pour la "prise" d'une borne supérieure.

Heuristique 13

– Etape 1.

Classons les M_{ij}^k par ordre décroissant pour chaque commodité k :

$$M_1^k \geq M_2^k \geq M_3^k \geq \dots \geq M_{|\mathcal{A}|}^k$$

Initialisation :

$indice[k] = 1 \forall k \in K$, ce vecteur permet de sauver l'indice de l'arc ayant la $indice[k]$ -ième borne supérieure la plus grande pour la commodité k .

$E[k] = \phi \forall k \in K$.

$mat[i][j] = \phi \forall (i, j) \in \mathcal{A}$, cette matrice contient les commodités qui passent par les arcs $(i, j) \in \mathcal{A}$ et aussi les bornes supérieures pour chacune des commodités.

– Etape 2.

Pour chaque commodité k nous choisissons les arcs ayant la plus grande borne supérieure.

$E[k] = a_1^k \forall k \in K$, où a_1^k est l'arc ayant la 1-ième borne supérieure la plus grande pour la commodité k .

$mat[i][j] = mat[i][j] \cup (i, j, k) \forall a_1^k \in \mathcal{A}$ où les sommets d'origine et de destination de a_1^k sont i et j .

$k = 1$.

– Etape 3.

Nous considérons l'arc choisi pour la commodité k : $a_{ij} = E[k]$ où (i, j) sont les sommets d'origine et de destination de l'arc $E[k]$.

Si $a_{ij} \in \mathcal{A}$ aller à Etape 4.

Sinon aller à Etape 9.

– Etape 4.

Nous générons un nombre aléatoire (prob) entre 0 et 1.

Si $prob < p_p$ où

$$p_p = \frac{1}{|\mathcal{E}_{ij}|} \sum_{(k,l) \in \mathcal{E}_{ij}} d((i, j), (k, l)).$$

On sélectionne la borne supérieure la plus éloignée de la moyenne des bornes supérieures de l'arc (i, j)

$$a_{ij}^{com} = arg \max_{com} dist(moy(i, j), M_{ij}^{com})$$

où

$$moy(i, j) = \frac{1}{|\mathcal{F}_{ij}|} \sum_{f \in \mathcal{F}_{ij}} M_{ij}^f,$$

où \mathcal{F}_{ij} est l'ensemble des commodités qui passent par l'arc (i, j) . Aller à Etape 5.

Sinon aller à Etape 9.

– Etape 5.

Nous devons changer de choix pour la commodité com .

$$mat[i][j] = mat[i][j] \setminus (i, j, com).$$

– Etape 6.

$$indice[com] = indice[com] + 1$$

Si $(a, b) \in \mathcal{A}$ où (a, b) sont les sommets d'origine et de destination de l'arc $a_{indice[com]}^{com}$ et aller à Etape 7.

Sinon $E[com] = (o_k, d_k)$ et aller à Etape 9.

– Etape 7.

Nous devons tester si l'arc choisi est semblable à l'environnement.

(a, b) sont les sommets d'origine et de destination de l'arc $a_{indice[com]}^{com}$.

Si $d(M_{ab}^{com}, moy(a, b)) < \max_{l \in \mathcal{F}_{ab}} d(moy(a, b), M_{ab}^l)$ aller à Etape 8.

Si $M_{ab}^{com} > 0,75 * \frac{1}{|\mathcal{E}_{ab}|+1} (\sum_{(k,l) \in \mathcal{E}_{ab}} moy(k, l) + moy(a, b))$ aller à Etape 8.

Sinon aller à Etape 6.

– Etape 8.

$$E[com] = a_{indice[com]}^{com} \text{ et } mat[a][b] = (a, b, com)$$

Aller à Etape 9.

– Etape 9.

Si $k = |K|$, on résout le problème inverse (5.47)-(5.60) où l'arc choisi pour la commodité k est $E[k]$ et s'arrêter.

Sinon $k = k + 1$ aller à Etape 3.

5.3.11 Heuristique 14

Dans toutes les heuristiques déjà présentées, nous essayons de choisir au mieux les arcs que les commodités vont utiliser. Pour cette heuristique, nous allons choisir les arcs que les commodités ne devront pas utiliser.

Pour ce faire, nous allons résoudre la relaxation du problème et si dans la solution optimale, l'arc (i, j) n'est utilisé par aucune commodité, nous ajoutons la

contrainte suivante :

$$x_{ij}^k = 0 \quad \forall k \in K,$$

à la formulation (5.27) - (5.39).

Nous essayons donc de trouver une solution parmi les arcs utilisés dans la relaxation.

Heuristique 14

- Etape 1 :
Résoudre la relaxation de (5.27) - (5.39).
- Etape 2 :
Si $x[i][j][k] = 0 \quad \forall k \in K$ dans la solution optimale obtenue à Etape 1, rajouter la contrainte : $x_{ij}^k = 0 \quad \forall k \in K$ à la formulation (5.27) - (5.39).
- Etape 3 :
Résoudre la formulation (5.27) - (5.39) avec les contraintes ajoutées à Etape 2.

5.3.12 Heuristique 15

Cette heuristique n'a pas grand chose à voir avec les heuristiques déjà présentées. Nous allons ici développer une heuristique pour la méthode exacte du K-Chemin : au lieu de garder tous les candidats générés à partir d'un K-Chemin, nous ne garderons que les deux candidats ayant la plus grande borne supérieure.

La seule différence avec la méthode exacte est que lors de la création du i-ème K-Chemin, nous ne gardons dans *List* (étape 3) que les deux candidats ayant les plus grandes bornes supérieures.

5.4 Résultats numériques

Dans nos exemples, nous avons considéré des réseaux de 7, 8 et 9 villes situées hors de l'autoroute et une autoroute constituée de 10 ou 20 noeuds. Notre réseau comporte une commodité entre chaque ville, si n est le nombre de villes, nous avons $\frac{n(n-1)}{2}$ commodités par réseau.

Nous avons donc traité 6 réseaux différents :

- 7 villes et 10 noeuds (21 commodités)
- 7 villes et 20 noeuds (21 commodités)
- 8 villes et 10 noeuds (28 commodités)
- 8 villes et 20 noeuds (28 commodités)
- 9 villes et 10 noeuds (36 commodités)

– 9 villes et 20 noeuds (36 commodités)

Nous construisons le réseau de la même façon que dans S. Dewez [8]. Pour garder une certaine logique, le coût de l'arc $i, i+2$, appartenant à l'autoroute est égal au coût de l'arc $i, i+1$ plus le coût de $i+1, i+2$. Comme expliqué en début de chapitre, tous les noeuds de l'autoroute sont reliés entre eux (graphe complet). Nous devons encore calculer le coût des arcs entre les villes et les points d'entrées de l'autoroute. Pour faire cela, considérons une ville a , nous choisissons aléatoirement le point d'entrée de l'autoroute le plus proche de a ensuite nous calculons aléatoirement le coût de cet arc. Une fois cette opération faite, nous calculons aléatoirement le coût des autres arcs entre cette ville et les points d'entrée de l'autoroute restant. Lors de ce calcul nous devons respecter ces inégalités (supposons que i est le point d'entrée le plus proche de a) :

$$\begin{aligned}c_{a,i} &\leq c_{a,i+1} \leq \dots \leq c_{a,|\mathcal{N}|} \\c_{a,i} &\leq c_{a,i-1} \leq \dots \leq c_{a,1},\end{aligned}$$

où \mathcal{N} est le nombre de noeuds (points d'entrée) de l'autoroute. Nous devons encore calculer le coût de l'arc non tarifable entre deux villes (a, b) , ce coût est égal au minimum des coûts des chemins non tarifables $(a, i) - (i, b)$. Finalement nous générons aléatoirement les demandes pour chaque commodité.

Les tableaux 5.2, 5.3 et 5.4 nous donnent les résultats obtenus sur une moyenne de 10 problèmes. Parmi ces 10 problèmes, nous avons distingué quatre classes de problèmes :

1. Petit écart entre le nombre minimal et maximal d'utilisateurs par commodité et petit écart entre la coût minimal et maximal sur les arcs. La taxe maximale sur les arcs tarifables est égale au coût maximal.
2. Grand écart entre le nombre minimal et maximal d'utilisateurs par commodité et petit écart entre la coût minimal et maximal sur les arcs. La taxe maximale sur les arcs tarifables est égale au coût maximal.
3. Petit écart entre le nombre minimal et maximal d'utilisateurs par commodité et grand écart entre la coût minimal et maximal sur les arcs. La taxe maximale sur les arcs tarifables est égale au coût maximal.
4. Grand écart entre le nombre minimal et maximal d'utilisateurs par commodité et grand écart entre la coût minimal et maximal sur les arcs. La taxe maximale sur les arcs tarifables est égale au coût maximal.

Nous avons fait cette distinction afin de vérifier si les heuristiques étaient sensibles ou non à l'un des paramètres. Pour résoudre les problèmes, nous avons utilisé le solveur Xpress [2],[1] et la bibliothèque BCL [3] pour avoir accès au solveur à partir du code C++. Le code a été exécuté sur un Intel Pentium III (Coppermine) (1004.519MHz) et dans un environnement Linux (Kernel : 2.4.20-64GB-SMP).

Nous avons résolu chaque problème par la méthode exacte (*HTSP2*) où nous avons fixé la borne N à 1000. La solution obtenue avec cette méthode sera notre référence pour chaque problème. Nous pourrions donc calculer un Gap : $\frac{SolHeuris}{SolEx}$ où *SolHeuris* est la solution obtenue avec une heuristique et *SolEx* la solution exacte, (1-Gap) nous donnera l'erreur liée à l'heuristique.

Les tableaux 5.2, 5.3 et 5.4 contiennent les résultats obtenus avec les heuristiques présentées à la Section 5.3 (sauf l'heuristique 15) ainsi que le résultat obtenu par la méthode exacte. Nous voyons qu'avec les heuristiques 2, 4 et 6, nous obtenons des résultats similaires (gap est à 95% environ pour tous les problèmes) et en quelques secondes. Avec l'heuristique 1, nous obtenons des résultats proches des heuristiques 2, 4 et 6 et dans un temps généralement inférieur à ces heuristiques. Nous voyons donc qu'il n'y a pas d'amélioration probante entre les choix effectués dans l'heuristique 1 et ceux des heuristiques 2, 4 et 6. Nous observons certes une amélioration de l'ordre du pourcent pour les heuristiques 2, 4 et 6 par rapport à l'heuristique 1 mais au prix d'une augmentation (légère) du temps de calcul. Nous voyons aussi, que les résultats ne sont nullement perturbés par l'augmentation de villes ou de noeuds.

Les heuristiques 3,5 et 7 fonctionnent respectivement de la même façon que les heuristiques 2, 4 et 6 mais en tenant compte de la demande. Nous voyons que les résultats sont moins bons. Ces résultats peuvent s'expliquer par le fait que nous ne regardons plus uniquement la borne supérieure mais aussi la demande. C'est à dire que, même si la borne est petite mais que la demande est très grande, nous privilégierions le choix de cette borne car elle pourrait engendrer un revenu élevé. Le problème qui peut se poser, c'est que nous choisissons une petite borne et, qu'à cause des inégalités triangulaires, nous devrions diminuer la valeur de plusieurs taxes. En effet, les inégalités triangulaires ne tiennent pas compte de la demande.

Intéressons-nous maintenant aux heuristiques 8 et 9, ces dernières tiennent compte des arcs les plus utilisés lors de la résolution du problème par la relaxation de (*HTSP2*). Les gap obtenus sont de l'ordre de 93%-94% soit légèrement inférieurs aux heuristiques 1, 2, 4 et 6. Le temps de résolution est, lui, bien plus grand et dépend directement du nombre de noeuds dans l'autoroute. Il semblerait donc conclure que tous les arcs les plus utilisés dans la relaxation ne seront pas les arcs utilisés dans (*HTSP2*). Les heuristiques 3, 5, et 7 ainsi que 8 et 9 prennent en compte les demandes pour chaque commodité. Nous remarquons que, pour chacune de ces heuristiques, les résultats sont meilleurs lorsque le nombre de noeuds dans l'autoroute augmente. Dans le cas des heuristiques 8 et 9, les résultats deviennent meilleurs que ceux obtenus avec les heuristiques 1, 2, 4 et 6 lorsque le nombre de noeuds dans l'autoroute augmente.

Revenons aux heuristiques 8 et 9, nous nous attendions à des résultats supérieurs à ceux obtenus avec les 7 premières heuristiques et pourtant ils ne le sont pas vraiment. Nous pouvons donc nous demander si les arcs les plus utilisés (en terme de flot) dans la relaxation de $(HTSP2)$ seront ceux utilisés dans $(HTSP2)$.

Avec l'heuristique 11, nous obtenons un gap souvent entre 92% et 94%. La prise en compte de l'environnement ne joue donc pas un rôle déterminant dans le choix des arcs et diminue même l'efficacité de l'heuristique, étant donné que les résultats sont moins bons que ceux des heuristiques précédentes.

Les heuristiques 12 et 13 nous donnent les plus mauvais résultats. Nous pouvons dire qu'elles ne fonctionnent pas du tout. Nous avons testé 10 fois chacune des deux heuristiques sur chaque problème de façon à avoir une moyenne des résultats. Nous avons remarqué que les résultats pouvaient fortement varier et que jamais le résultat obtenu n'a été supérieur au résultat obtenu suite à l'initialisation (heuristique 1). Ceci porte à croire que les probabilités utilisées pour les heuristiques 12 et 13 ne sont pas adaptées au problème et que, encore une fois, la prise en compte de l'environnement n'apporte aucune amélioration.

Finalement, intéressons-nous à l'heuristique 14. Nous obtenons de très bons résultats (de l'ordre de 98%). Le temps de résolution est beaucoup plus petit que celui de la méthode exacte. Ceci ne peut avoir qu'une seule interprétation : peu d'arcs sont utilisés dans la solution optimale de la relaxation de $(HTSP2)$. Nous vérifions cette interprétation dans le tableau 5.5. Le pourcentage d'arcs utilisés dans la solution optimale de la relaxation de $(HTSP2)$ diminue fortement en fonction du nombre de noeuds présents dans l'autoroute et ce pourcentage augmente naturellement avec le nombre de commodités. La résolution de $(HTSP2)$ sur ce graphe "réduit" est donc beaucoup plus rapide mais reste néanmoins une heuristique car, comme nous le voyons, certains arcs non utilisés dans la solution optimale de la relaxation de $(HTSP2)$ sont utilisés dans la solution optimale de $(HTSP2)$. Néanmoins, les résultats sont très bons.

	10 Noeuds		20 Noeuds	
	Gap	Temps	Gap	Temps
Méthode exacte	100%	2,64	100%	842
Heuristique 1	93%	0,54	94%	1,7
Heuristique 2	95%	0,55	95%	2,1
Heuristique 3	93%	0,56	93%	2,6
Heuristique 4	95%	0,55	95%	2
Heuristique 5	93%	0,56	93%	2,5
Heuristique 6	95%	0,55	95%	2
Heuristique 7	93%	0,55	93%	2,6
Heuristique 8	94%	1,39	95%	10
Heuristique 9	93%	1,42	95%	10
Heuristique 10	92%	4,3	94%	6
Heuristique 11	95%	0,54	95%	2
Heuristique 12	78%	0,52	83%	1,7
Heuristique 13	77%	0,52	85%	1,6
Heuristique 14	98,5%	2,65	98,6%	21,6

TAB. 5.2 – Méthode exacte et heuristiques pour un réseau à 7 villes.

	10 Noeuds		20 Noeuds	
	Gap	Temps	Gap	Temps
Méthode exacte	100%	57	100%	3626
Heuristique 1	94%	0,7	95%	3,5
Heuristique 2	95%	0,7	95%	3,4
Heuristique 3	92%	0,8	94%	3,9
Heuristique 4	95%	0,7	95%	3,4
Heuristique 5	92%	0,9	94%	4,1
Heuristique 6	95%	0,7	95%	4,9
Heuristique 7	92%	1	94%	4,3
Heuristique 8	93%	2	96%	17,8
Heuristique 9	93%	2	96%	19,2
Heuristique 10	94%	4,1	96%	29,3
Heuristique 11	94%	0,7	92%	3,4
Heuristique 12	76%	0,6	80%	3,09
Heuristique 13	73%	0,6	84%	3,1
Heuristique 14	98,6%	10,9	98,5%	166,6

TAB. 5.3 – Méthode exacte et heuristiques pour un réseau à 8 villes.

	10 Noeuds		20 Noeuds	
	Gap	Temps	Gap	Temps
Méthode exacte	100%	48	100%	2185
Heuristique 1	92%	1,35	94%	3,9
Heuristique 2	93%	1,37	94%	4
Heuristique 3	89%	1,37	93%	5
Heuristique 4	94%	1,39	94%	4
Heuristique 5	89%	1,4	93%	5
Heuristique 6	94%	3	94%	7,6
Heuristique 7	89%	3,1	93%	7
Heuristique 8	92%	4,5	94%	23,2
Heuristique 9	92%	6,3	94%	23,9
Heuristique 10	94%	7,7	94%	30
Heuristique 11	90%	1,2	94%	3,9
Heuristique 12	70%	0,9	76%	3,6
Heuristique 13	72%	0,9	79%	3,6
Heuristique 14	97,8%	40,5	98%	207,2

TAB. 5.4 – Méthode exacte et heuristiques pour un réseau à 9 villes.

	10 Noeuds		20 Noeuds	
	Pourcentage	Nombre d'arcs	Pourcentage	Nombre d'arcs
7 villes	12%	11	4%	16
8 villes	17%	15	6%	23
9 villes	19%	17	8%	31

TAB. 5.5 – Pourcentage et nombre d'arcs utilisés en moyenne dans la relaxation de (*HTSP2*).

Les tableaux 5.6 et 5.7 contiennent les résultats et temps obtenus en utilisant l'algorithme exact du K-Chemin ainsi qu'en utilisant l'heuristique 15. Nous avons testé ces deux algorithmes sur 5 problèmes (lignes des tableaux).

Nous remarquons donc que l'heuristique 15 nous donne, en général, la bonne solution mais dans un temps inférieur à celui de la méthode exacte par K-Chemin. Nous pouvons conclure que la solution optimale sera souvent issue des meilleurs candidats liés à chaque K-Chemin. Nous nous sommes contentés de petites instances car, sur de grosses instances, le temps mis par l'algorithme de K-Chemin est très élevé. Le nombre de candidats explose.

Xpress		K-Chemin	Heuristique 15	
Résultat	Temps	Temps	Résultat	Temps
36	0,4	0,4	36	0,4
22	0,4	0,8	22	0,4
60	0,7	0,9	60	0,6
182	0,6	0,7	182	0,67
96	0,66	0,5	96	0,46

TAB. 5.6 – Algorithme du K-chemin et heuristique 15 pour des réseaux à 5 villes et 5 noeuds.

XPress		K-Chemin	Heuristique 15	
Résultat	Temps	Temps	Résultat	Temps
166	1	1,3	156	1,1
140	1	49	140	5
50	0,8	2	50	0,8
30	1,1	1	30	0,9
425	5,6	1952	421	94

TAB. 5.7 – Algorithme du K-chemin et heuristique 15 pour des réseaux à 5 villes et 10 noeuds.

Chapitre 6

Conclusion

Dans le dernier chapitre nous avons étudié un cas particulier du Toll Setting Problem : les arcs tarifables constituent un chemin. La formulation utilisée contient des inégalités triangulaires qui rendent la résolution difficile. La formulation exacte (*HTSP2*) prend beaucoup de temps pour résoudre le problème. Nous avons donc développé des heuristiques utilisant différentes approches pour résoudre le problème. L'approche commune aux sept premières heuristiques est de choisir les arcs empruntés par les utilisateurs en nous servant des bornes supérieures sur les taxes associées aux arcs. Parmi ces sept heuristiques, nous pouvons distinguer deux groupes : le premier ne prend pas en compte, lors du choix des arcs, le nombre d'utilisateurs par commodité (heuristiques 1, 2, 4 et 6) contrairement au second (heuristiques 3, 5 et 7).

Les heuristiques 8 et 9 se basent sur une approche différente : nous tenons compte des arcs les plus utilisés dans la relaxation de (*HTSP2*) pour choisir ceux qu'emprunteront les utilisateurs.

Après la résolution inverse du problème (Section 5.3, formulation (5.47)-(5.60)), l'approche utilisée pour l'heuristique 10 est de tenir compte des taxes placées sur les arcs pour choisir les arcs qu'emprunteront les utilisateurs.

En ce qui concerne les heuristiques 11, 12 et 13, nous avons voulu tenir compte de l'environnement (Section 5.3.8) pour le choix des arcs qu'emprunteront les utilisateurs.

Pour l'heuristique 14, nous ne choisissons pas les arcs que les utilisateurs emprunteront, mais ceux qu'ils n'emprunteront pas. Pour faire ces choix, nous utilisons les arcs empruntés dans la relaxation de (*HTSP2*).

Finalement l'heuristique 15, se base sur l'algorithme exact des K-Chemins (Section 5.2) où nous ne choisissons qu'un nombre réduit de candidats à chaque itération.

Une comparaison détaillée de ces heuristiques figure à la section 5.4. La prise en compte du nombre d'utilisateurs de chaque commodité semble jouer un rôle important lorsque le nombre de noeuds de l'autoroute augmente.

L'heuristique 14 donne de très bons résultats (Gap > 98%) et cela dans un temps

inférieur à la méthode exacte (*HTSP2*) (quelques secondes pour de petites instances et moins de 4 minutes pour un réseau de 9 villes et 20 noeuds dans l'auto-route). Nous avons donc réussi à obtenir des résultats très proches de la solution optimale et dans un temps acceptable.

Finalement, nous développons ci-dessous, deux améliorations possibles pour le Highway Special Case (Chapitre 5).

Nous avons constaté que dans le cas du K-Chemin, en ne considérant que les deux meilleurs candidats à chaque étape (heuristique 15), nous obtenons une solution proche de la solution exacte. Pour cette raison, l'utilisation d'un algorithme génétique semble adéquat. Donnons une idée de la façon dont nous nous y prendrions pour implémenter cet algorithme.

Les individus de la population seraient les K-Chemins (K-Chemins définis de la même façon qu'à la Section 5.2). La population serait constituées de K-Chemins. La fonction d'adaptation traduit la façon dont est adapté l'individu au problème. Dans notre cas, cette fonction pourrait être la borne supérieure des K-Chemins (temps de calcul faible) ou le résultat de la résolution inverse pour chaque K-Chemin (temps de calcul long). Le choix des individus pour les croisements serait fait sur base de leur fonction d'adaptation : au plus la valeur de la fonction d'adaptation des individus est élevée au plus ils ont de chances d'être choisis. Finalement, la mutation au sein d'un K-Chemin se symboliserait par le choix d'un autre arc pour une des commodités du K-Chemin.

Finalement, dans l'algorithme exact du K-Chemin, nous pourrions éviter de gérer les choix des arcs non tarifables lors de la création des K-Chemins. Cela réduirait le nombre de K-Chemins candidats. Pour ce faire, nous devrions changer la formulation donnée à la section 5.2 (5.40)-(5.46) et utiliser plutôt celle de la section 5.3 (5.47)-(5.60). De cette façon, le solveur décidera si les utilisateurs passeront ou non par l'arc choisi dans le K-Chemin. Cette amélioration a un coût : le temps de calcul sera plus long car les utilisateurs ont maintenant le choix entre deux chemins.

Bibliographie

- [1] Dash Associates. *Modeling with Xpress-MP*.
- [2] Dash Associates. *Xpress-MP Getting Started*, 2003.
- [3] Dash Associates. *Xpress-BCL Reference Manual*, 2.6 edition, February 2006.
- [4] W Bialas and M. Karwan. Two-level linear programming. *Management Science*, 30(8) :1004–1020, 1984.
- [5] L. Brotcorne, M. Labbé, P. Marcotte, and G. Savard. Approche bi-niveau and tarification, February 2000.
- [6] L. Brotcorne, M. Labbé, P. Marcotte, and G. Savard. A bilevel model for toll optimization on a multicommodity transportation network, 2001.
- [7] J.L. Deneubourg, S. Goss, N. Franks, A. Sendova-Franks, C. Detrain, and L. Chretien. The dynamics of collective sorting : Robot-like ant and ant-like robot, 1990.
- [8] S. Dewez. *On the Toll Setting Problem*. PhD thesis, Université Libre de Bruxelles, 2004.
- [9] M. Didi, P. Marcotte, and G. Savard. Interne report of gerard.
- [10] M. R. Garey and D. S. Johnson. *Computers and Intractability, A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.
- [11] M. Labbé, P. Marcotte, and G. Savard. A bilevel model of taxation and its application to optimal highway pricing. *Management Science*, 44(12) :1608–1622, 1998. Par 1 of 2.
- [12] M. Labbé, P. Marcotte, and G. Savard. Bilinear bilevel programming, October 1998.
- [13] E. Lumer and B. Faieta. Diversity and adaptation in populations of clustering ants, 1994.
- [14] N. Monmarché. *Algorithmes de fourmis artificielles, applications à la classification and à l'optimisation*. PhD thesis, Université François Rabelais de Tours, 2000.
- [15] S. Roch and P. Marcotte. Design and analysis of an approximation algorithm for stackelberg network pricing. *Les Cahiers du GERARD*, 61, December 2002.

- [16] S. Roch, P. Marcotte, and G. Savard. A bilevel programming approach to optimal price setting. *Les Cahiers du GERARD*, 60, 2000.
- [17] L.C. Thomas. *Games, Theory and Applications*. Dunod, 1984.
- [18] P. Wolper. *Introduction à la calculabilité, cours and exercices corrigés*. Dunod, 2 edition, 2001.

Annexe A

Complexité SAT et 3-SAT

Nous allons prouver la NP-Complétude de SAT (premier problème NP-Complet) et puis 3-SAT comme effectué dans P. Wolper [18] et Garey and Johnson [10].

Définition A.1 Une formule propositionnelle ϕ est satisfaisable ssi il existe une fonction d'interprétation f qui rend vraie ϕ .

Définition A.2 Une formule ϕ est en forme normale conjonctive si c'est une conjonction de disjonctions de littéraux.

Définition A.3 Le problème SAT, consiste à décider la satisfaisabilité d'une formule propositionnelle en forme normale conjonctive.

A.1 SAT

Théorème A.1.1 Le problème SAT est NP-complet

Preuve:

(1) SAT est bien dans NP. En effet, étant donné une fonction d'interprétation f , il existe bien un algorithme polynomial pour décider si f satisfait ϕ

(2) SAT est NP-Comple. On va montrer qu'il existe une transformation polynomiale de tout langage NP vers L_{sat}

- transformation à 2 arguments : un mot w et un langage L .
- on s'appuie sur le fait que chaque langage L est caractérisé par une machine de Turing non déterministe polynomiale (bornée par un polynôme $p(n)$).

Soit un mot w ($|w| = n$) et une machine de Turing non déterministe $M = (Q, \Gamma, \Sigma, \Delta, s, B, F)$ (borné par $p(n)$).

- Q : ensemble fini d'états de la machine de Turing
- Γ : alphabet du ruban
- Σ : est l'alphabet d'entrée (alphabet utilisé par le mot d'entrée), on a $\Sigma \subset \Gamma$
- $s \in Q$: état initial
- $F \subseteq Q$ est l'ensemble des états accepteurs
- $B \in \Gamma \setminus \Sigma$ est le 'symbole blanc', que l'on notera .
- $\Delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$ est la fonction de transition (L et R sont utilisés pour représenter respectivement le déplacement à gauche et à droite de la tête de lecture).

Le schéma de preuve pour montrer le fait que SAT est NP-Comple est le suivant : on va montrer que pour chaque machine M et mot w , on peut construire une formule $\phi_{M,w}$ telle que cette formule est satisfaisable ssi $w \in L(M)$. De plus, on va montrer que la formule construite est en forme normale conjonctive et sa longueur est bornée polynomialement par rapport à $p(n)$.

Idée : un interprétation f satisfait $\phi_{w,M}$ si f "décrit" une exécution acceptée par M sur w .

- une proposition $r_{ij\alpha}$ pour $0 \leq i, j \leq p(n)$ et $\alpha \in \Gamma$. $r_{ij\alpha}$ est vraie ssi lors du pas i de l'exécution, la case j du ruban contient le symbole α .
- une proposition q_{ik} pour $0 \leq i, j \leq p(n)$ et $k \in Q$.
 q_{ik} est vraie ssi lors du pas i de l'exécution de la machine est dans l'état k .
- une proposition p_{ij} pour $0 \leq i, j \leq p(n)$.
 p_{ij} est vraie ssi lors du pas i de l'exécution, la tête de lecture se trouve en position j .
- une proposition c_{ik} pour $0 \leq i, j \leq p(n)$ et $1 \leq k \leq r$.
 c_{ik} est vraie ss le choix effectué lors du pas i de l'exécution est le choix numéro k .

On construit alors la formule $\phi_{w,M}$ comme la conjonction des formules propositionnelles suivantes :

- chaque case du ruban contient exactement un symbole :

$$\bigwedge_{0 \leq i, j \leq p(n)} [(\bigvee_{\alpha \in \Gamma} r_{ij\alpha}) \wedge \bigwedge_{\alpha \neq \alpha' \in \Gamma} (\neg r_{ij\alpha} \vee \neg r_{ij\alpha'})],$$

et $\bigwedge_{0 \leq i, j \leq p(n)} (\bigvee_{\alpha \in \Gamma} r_{ij\alpha})$ signifie que pour un i et un j , la case j au pas i contient un symbole $\alpha \in \Gamma$. S'il n'en contenait pas, cette disjonction serait fausse et la proposition aussi.

$\bigwedge_{\alpha \neq \alpha' \in \Gamma} (\neg r_{ij\alpha} \vee \neg r_{ij\alpha'})$ signifie que la case j au pas i ne contient qu'un et un seul symbole, si ce n'est pas le cas, cette conjonction serait fausse et donc la proposition aussi.

Notons que cette formule a une longueur de $O(p(n)^2)$ car on a $(p(n))$ pas d'exécution et on peut supposer que le nombre de caractères de l'alphabet est inférieur à $p(n)$.

- à chaque moment de l'exécution, la tête de lecture se trouve exactement sur une position du ruban :

$$\bigwedge_{0 \leq i \leq p(n)} \left[\left(\bigvee_{0 \leq j \leq p(n)} p_{ij} \wedge \left(\bigwedge_{0 \leq j' \neq j \leq p(n)} (\neg p_{ij} \vee \neg p'_{ij}) \right) \right) \right]$$

et $\bigvee_{0 \leq j \leq p(n)} p_{ij}$, signifie que la tête de lecture doit être à chaque pas sur une case du ruban. Si ce n'est pas le cas, cette disjonction serait fausse.

$\bigwedge_{0 \leq j' \neq j \leq p(n)} (\neg p_{ij} \vee \neg p'_{ij})$ signifie que la tête de lecture ne peut être que sur une case à la fois pour un même pas.

Notons que cette formule a une longueur bornée par $O(p(n)^3)$, en effet on considère $p(n)$ pas d'exécutions, pour chaque pas, on considère les $p(n)$ cases et ce plusieurs fois.

- Les transitions de la machine sont exprimées à l'aide de formules de la forme suivante :

$$\bigwedge_{\substack{0 \leq i \leq p(n) \\ 0 \leq j \leq p(n) \\ \alpha \in \Gamma}} [(r_{ij\alpha} \wedge \neg p_{ij}) \rightarrow r_{(i+1)j\alpha}].$$

Cette formule peut facilement être mise en forme normale conjonctive ($A \rightarrow B \equiv \neg A \vee B$), elle signifie que si la case j contient le symbole α lors du pas i ($r_{ij\alpha}$) et que la tête de lecture n'est pas sur cette case à ce même pas d'exécution ($\neg p_{ij}$), alors au pas d'exécution $i + 1$, le symbole sur la case j aura toujours la même valeur (α).

Sa longueur est de $O(p(n)^2)$, en effet, pour chaque pas d'exécution, on a le choix entre $p(n)$ cases.

- Autre formule pour formaliser la relation de transition :

$$\bigwedge_{\substack{0 \leq i \leq p(n) \\ 0 \leq j \leq p(n) \\ \alpha \in \Gamma \\ 1 \leq k \leq r}} \left[\begin{array}{l} (q_{ik} \wedge p_{ij} \wedge r_{ij\alpha} \wedge c_{ik}) \rightarrow q_{(i+1)k'} \wedge \\ (q_{ik} \wedge p_{ij} \wedge r_{ij\alpha} \wedge c_{ik}) \rightarrow r_{(i+1)j\alpha'} \wedge \\ (q_{ik} \wedge p_{ij} \wedge r_{ij\alpha} \wedge c_{ik}) \rightarrow p_{(i+1)(j+d)} \end{array} \right]$$

Cette formule peut facilement être mise en forme normale conjonctive ($A \rightarrow B \equiv \neg A \vee B$). Elle signifie que si, lors du pas d'exécution i , la machine est dans l'état k et que la tête de lecture est sur la case j et que la case j contient le symbole α , lors du pas $i + 1$, la machine devra être dans l'état k' , la case j devra contenir un symbole α' et la tête de lecture devra être sur la case $j + d$ où d est la taille du déplacement suite à la transition.

Sa longueur est de $O(p(n)^2)$. En effet, pour chaque pas d'exécution, on a le choix entre $p(n)$ cases.

- La formule suivante exprime que l'état final est atteint :

$$\bigvee_{\substack{0 \leq j \leq p(n) \\ k \in F}} [q_{jk}]$$

Cette formule est de longueur $O(p(n))$.

Donc, la longueur de la formule entière est de $O(p(n)^3)$. Elle peut être construite en temps polynomiale par rapport à $p(n)$. Dès lors, $\phi_{w,M}$ est satisfaisable ssi $w \in L(M)$. On a donc bien une transformation polynomiale de tout langage NP vers SAT.

□

A.2 3SAT

Nous allons maintenant montrer qu'un cas particulier de SAT, appelé 3SAT, est également NP-complet.

Définition A.4 *3SAT : satisfaisabilité des formules propositionnelles en forme normale conjonctive qui ont exactement 3 littéraux par clause*

Théorème A.2.1 *Une fonction d'interprétation $f' : P' \rightarrow \{0, 1\}$ est une extension de $f : P \rightarrow \{0, 1\}$ ssi $P \subset P'$ et $\forall p \in P : f'(p) = f(p)$.*

Preuve:

Il faut montrer que pour toute formule ϕ en forme normale conjonctive, on peut construire une formule ϕ' en forme normale conjonctive avec exactement 3 littéraux par clause et telle que ϕ' est satisfaisable ssi ϕ est satisfaisable.

De plus la formule ϕ' doit être constructible en temps polynomial par rapport à la taille de ϕ .

Pour construire ϕ' , on procède de la façon suivante :

- une clause $\psi \equiv (x_1 \vee x_2)$ contenant deux littéraux est remplacée par :

$$\psi' \equiv (x_1 \vee x_2 \vee y) \wedge (x_1 \vee x_2 \vee \neg y),$$

où $y \in P' \setminus P$ (y est une nouvelle variable). Notons que pour tout f , on a bien f' une extension de $f : f$ est satisfaisable ssi f' l'est aussi.

Et donc, notre transformation préserve bien la satisfaisabilité.

- une clause $\psi \equiv (x_1)$ contenant un seul littéral est remplacée par :

$$\psi' \equiv \begin{aligned} &(x_1 \vee y_1 \vee y_2) \wedge \\ &(x_1 \vee y_1 \vee \neg y_2) \wedge \\ &(x_1 \vee \neg y_1 \vee y_2) \wedge \\ &(x_1 \vee \neg y_1 \vee \neg y_2) \end{aligned}$$

- une clause $\psi \equiv (x_1 \vee x_2 \vee \dots \vee x_i \vee \dots \vee x_l)$ comportant $l \geq 4$ littéraux est remplacée par :

$$\begin{aligned}
& (x_1 \vee x_2 \vee y_1) \wedge (\neg y_1 \vee x_3 \vee y_2) \\
& \wedge (\neg y_2 \vee x_4 \vee y_3) \wedge \dots \\
& \wedge (\neg y_{i-2} \vee x_i \vee y_{i-1}) \wedge \dots \\
& \wedge (\neg y_{l-4} \vee x_{l-2} \vee y_{l-3}) \\
& \wedge (\neg y_{l-3} \vee x_{l-1} \vee x_l)
\end{aligned}$$

A nouveau, on peut se convaincre que la formule obtenue est de taille polynomiale par rapport à la clause de départ et cette clause est satisfaisable par, et seulement par, les extensions des fonctions d'interprétation qui satisfont la clause de départ.

On a donc trouvé une façon d'écrire n'importe quelle formule en forme normale conjonctive sous une formule en forme normale conjonctive avec exactement 3 littéraux.

□