Titre: Title:	Problème de tarification sur un réseau avec demande élastique et contraintes de capacité
Auteur: Author:	Aimé Kamgaing Kuiteing
Date:	2011
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Kamgaing Kuiteing, A. (2011). Problème de tarification sur un réseau avec demande élastique et contraintes de capacité [Thèse de doctorat, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/645/

UNIVERSITÉ DE MONTRÉAL

PROBLÈME DE TARIFICATION SUR UN RÉSEAU AVEC DEMANDE ÉLASTIQUE ET CONTRAINTES DE CAPACITÉ

AIMÉ KAMGAING KUITEING
DÉPARTEMENT DE MATHÉMATIQUES ET GÉNIE INDUSTRIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

THÈSE PRÉSENTÉE EN VUE DE L'OBTENTION
DU DIPLÔME DE PHILOSOPHIÆ DOCTOR
(MATHÉMATIQUES DE L'INGÉNIEUR)
AOÛT 2011

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Cette thèse intitulée :

PROBLÈME DE TARIFICATION SUR UN RÉSEAU AVEC DEMANDE ÉLASTIQUE ET CONTRAINTES DE CAPACITÉ

présentée par : KAMGAING KUITEING, Aimé

en vue de l'obtention du diplôme de : Philosophiæ Doctor

a été dûment acceptée par le jury d'examen constitué de :

M. AUDET, Charles, Ph.D., président.

M. SAVARD, Gilles, Ph.D., membre et directeur de recherche.

M. MARCOTTE, Patrice, Ph.D., membre et codirecteur de recherche.

M. GAMACHE, Michel, Ph.D., membre.

M. RUIZ BARTOLOME, Angel, Doctorat, membre externe.

Je dédie cette thèse à mes parents, avec toute mon affection et tout mon respect. Vous ne m'avez pas seulement donné la vie : vous m'avez transmis le désir passionné de la vivre pleinement. Je ne vous en remercierai jamais assez.

REMERCIEMENTS

Je désire d'abord remercier mes directeurs de thèse Gilles Savard et Patrice Marcotte, pour la confiance qu'ils m'ont accordée en me permettant de réaliser cette thèse sous leur supervision. Après m'avoir proposé un sujet précis et stimulant, ils m'ont laissé une grande liberté de la façon de régler les détails des travaux. Qu'ils trouvent ici tous mes remerciements pour leur présence tant sur le plan professionnel que personnel. J'ai eu un énorme plaisir de travailler avec vous. Un gros merci à vous.

Mes remerciements vont également à M. Audet, M. Gamache, M. Gourdreau et M. Bartolome d'avoir accepté d'être membres du jury de cette thèse. Un merci particulier à M. Audet qui m'a beaucoup aidé grâce à de nombreux échanges sur certains aspects théoriques, et m'a attribué ma première charge de cours à Polytechnique.

Je remercie ensuite mes amis du Gerad pour leur soutien et d'avoir rendu ces dernières années plus agréables et enrichissantes, en particulier Zoumana, Alain, Abdel, Kien, Shadi, Morad, Sharouz, Nicolas, Julio, Quentin, Hicham. Un gros merci à Alain d'avoir accepté de lire cette thèse. Merci également au personnel du Gerad et de Polytechnique pour leur sympathie et leur support inconditionnel. Enfin, mes plus profonds remerciements sont adressés à mon épouse Esther Fany, dont le soutien affectif et moral fut indispensable à la réussite de cette thèse. Je remercie également mon fils Pierre, mon frère Siméon, mes soeurs Joelle, Nadine, Chimène, Nicole, Anna, ma « pauvre » maman Tapithe et mon oncle Jean-Faustin d'avoir cru en moi et de m'avoir encouragé tout le long des années de thèse. En particulier, je témoigne toute ma gratitude à Siméon. Un merci infini à Roddy et Bienvenu pour tout. Enfin, d'autres personnes m'ont encouragé à finir ce travail par des gestes d'amitié dont je suis reconnaissant. Je ne citerai pas de noms ici, pour ne pas en oublier certains.

RÉSUMÉ

Plusieurs extensions du problème de tarification sur un réseau introduit par Labbé *et al.* [46] (modélisé sous la forme d'un programme à deux niveaux) ont été étudiées mais aucun, à notre connaissance, ne considère la demande élastique, qui est le sujet de cette thèse. Plus précisément, ne nous considérons le problème de maximisation des revenus générés par les tarifs imposés sur un sous-ensemble d'arcs d'un réseau de transport multi-produits, en tenant compte que la demande est affectée aux chemins de plus petit coût et dépend du coût total (coût fixe + tarif) de ces chemins, et que certains arcs du réseau ont une capacité finie.

Cette thèse porte sur une étude de modélisation suivie d'une mise en œuvre d'algorithmes et puis d'une étude numérique du problème de tarification sur un réseau avec demande élastique et contraintes de capacité à partir d'une étude approfondie des aspects théoriques.

Dans la première partie, nous étudions le problème de tarification sur un réseau avec demande élastique. Nous proposons, lorsque la demande est linéaire, trois formulations quadratiques en variables mixtes (MIP). Puis, nous montrons qu'il existe des cas où le problème est polynomial. Lorsque la demande est non linéaire, nous développons deux méthodes exactes basées sur une surapproximation des fonctions non linéaires de la fonction objectif d'un programme non linéaire en variables mixtes. Les tests numériques mettent en évidence l'efficacité de l'une des méthodes exactes sur les instances de petite et moyenne taille. Puis, nous développons deux heuristiques basées sur une méthode de région de confiance. Ces heuristiques fournissent des solutions situées en moyenne à 1% de la borne supérieure de la meilleure méthode exacte avec des temps de calcul raisonnable sur des instances de grande taille. L'analyse de sensibilité montre que la difficulté de résolution du problème diminue lorsque l'élasticité de la demande augmente.

Dans la deuxième partie, nous étudions le problème de tarification sur un réseau avec demande élastique où nous incorporons les capacités explicites sur les arcs du réseau. Pour ce faire, nous commençons par prouver que les contraintes de capacité peuvent être déplacées

au premier niveau. Il s'en suit que les coûts des chemins utilisés, pour un produit donné, sont tous égaux. Ainsi, le problème du second niveau se réduit à un problème de plus court chemin par produit comme dans le cas du problème sans contraintes de capacité. La technique utilisée pour reformuler le problème sans contraintes de capacité en un programme à un niveau est utilisée pour obtenir un programme à un niveau. Par la suite, les méthodes développées dans le cas du problème sans contraintes de capacité de la première partie sont adaptées pour résoudre le problème avec contraintes de capacité. Ainsi, lorsque la demande est linéaire, un programme quadratique en variables mixtes est développé. Dans le cas non linéaire, deux méthodes exactes sont développées et les tests numériques montrent l'efficacité de l'une d'elles sur des instances de petite taille. Une heuristique basée sur une méthode de région de confiance est développée et ses solutions sont situées à 2% de la borne supérieure sur les instances de taille moyenne et grande. Nous remarquons que la difficulté de résolution du problème augmente avec les valeurs des capacités sur les arcs. L'analyse de sensibilité montre que l'élasticité de la demande n'a pas d'influence sur la résolution du problème dans le cas linéaire tandis qu'elle augmente lorsque l'élasticité augmente dans le cas non linéaire.

ABSTRACT

Several extensions of the network pricing problem introduced by Labbé *et al.* [46] (modeled as a bilevel program) have been studied but none, to the best of our knowledge, involves elastic demand, which is the topic of this thesis. More specifically, we consider the problem of maximizing the revenue raised from tolls set on a subset of arcs of a multi-commodity transportation network, taking into account that demand is assigned to cheapest paths, and is actually dependent on the total cost (initial cost of carrying the products + toll) of these paths and some arcs of network have finite capacity.

This thesis is concerned with a modeling study followed by an implementation of algorithms and then a numerical study of the network pricing problem with elastic demand and capacity constraints, from the in-depth study of theoretical aspects.

In the first part, we study network pricing problem with elastic demand. In the case of linear demand-cost relationship, we propose three mixed integer (MIP) quadratic formulations. Then, we show that special cases can be solved in polynomial time. In the case of nonlinear demand functions, we develop two exact methods based on an over-estimation of nonlinear functions in the objective function of a non linear mixed integer program. Numerical tests highlight the efficiency of one of the exact methods on small and medium-size instances. Then, we design two heuristics based on a trust-region approach. Heuristics provide solutions whose objective lie within 1% of the upper bound of the best exact method with reasonable Cpu times, even on large instances. Indeed, sensibility analysis show that the difficulty of resolution of the problem decreases with the value of the cost elasticity of demand.

In the second part, we deal with the network pricing problem with elastic demand where we incorporate explicit capacities on arcs of the network. First, we prove that capacity constraints can be moved to the first level. It follows that the path used, for each product, have the same cost. Then, the second level problem reduces to a set of independent shortest-

paths problem as in the case of the problem without capacity constraints. The technique used to remodel the problem without capacity constraints as one level program is used to get a one level program. Thereafter, the methods developed for the problem without capacity constraints in the first part is adapted to solve the problem with capacity constraints. Then, in the case of linear demand, we propose a mixed integer (MIP) quadratic program. In the case of non linear demand, we develop two exact methods and numerical tests show the efficiency of one of the exact methods on small-size instances. A heuristic based on a trust-region approach is also designed and its solutions lie within 2% of the upper bound on medium and large-size instances. We remark that larger the value of capacity on arcs is, the more difficult the resolution of the problem is. Sensibility analysis shows that elasticity has no influence on the resolution of the problem in the linear case while it increases with elasticity in the non linear case.

TABLE DES MATIÈRES

LISTE DES TABLEAUX

LISTE DES FIGURES

LISTE DES SIGLES ET ABRÉVIATIONS

PTR	problème de tarification sur un réseau
DE-CAP	problème de tarification sur un réseau avec demande élastique et contraintes de capacité
DL	problème de tarification sur un réseau avec demande linéaire
DNL	problème de tarification sur un réseau avec demande non linéaire
DE-CAP	problème de tarification sur un réseau avec demande élastique et contraintes de capacités
DNL-CAP	problème de tarification sur un réseau avec demande non linéaire et contraintes de capacités
POI	problème d'optimisation inverse
PDTLP	problème de design et de tarification d'une ligne de produits

INTRODUCTION

Depuis plusieurs années les entreprises ont réalisé que, pour rester compétitives dans un marché hautement concurrentiel, il était nécessaire de raffiner leurs techniques de gestion du revenu, et plus particulièrement la tarification de leurs produits. Ce domaine de recherche en pleine expansion trouve son origine à la fin des années 1970, alors que la tarification et l'optimisation des revenus sont devenues des composantes essentielles de la gestion, des compagnies aériennes d'abord, puis des entreprises offrant des produits périssables et dont les coûts marginaux d'exploitation et d'inventaire sont faibles. Ces caractéristiques se retrouvent dans le domaine des télécommunications, du transport, du spectacle, du tourisme et de l'hôtellerie, entre autres.

Cette thèse traite plus particulièrement d'un problème de tarification sur un réseau physique, tel qu'un réseau autoroutier sujet à des péages, ou encore un réseau de télécommunications (téléphonie, internet) sur lequel des tarifs liés à l'utilisation des lignes de communication sont imposés. Le gestionnaire du réseau cherche à maximiser les revenus provenant des tarifs, tout en tenant compte du comportement des usagers qui minimisent leur coût propre. La programmation à deux niveaux constitue un cadre naturel pour ce problème. Au niveau supérieur, le *meneur* (gestionnaire du réseau) fixe des tarifs sur un sous-ensemble (fixé à priori) des tronçons du réseau et les *suiveurs* (les usagers) minimisent leur coût de transport individuel. Labbé *et al.* [46] ont été les premiers à formuler ce problème sous la forme d'un problème de programmation mathématique à deux niveaux. Par la suite, Roch *et al.* [57] et Grigoriev *et al.* [36] ont montré que ce problème est NP-difficile, et plusieurs études théoriques ou algorithmiques ont été consacrées au problème de base ou à certaines variantes (Dewez [27], Didi *et al.* [29], Cirinei [17], Brotcorne *et al.* [13], Marcotte *et al.* [52], Brotcorne *et al.* [14] et autres). Par contre, l'introduction d'une demande élastique, qui fait l'objet de cette thèse, n'avait encore jamais été l'objet d'une étude systématique.

Le présent travail aborde le problème de tarification sur un réseau avec demande élastique, en y incorporant des capacités explicites sur les arcs du réseau. Notre contribution se situe à trois niveaux : modélisation, algorithmique et numérique. Dans un premier temps, nous exhibons quelques propriétés du problème et présentons des cas polynômiaux. Puis, par l'intermédiaire des reformulations, nous obtenons des reformulations non linéaires impliquant à la fois des variables binaires et des variables continues, pour lesquelles des méthodes de résolution exactes (basées sur des hyperplans de coupe) ou heuristiques (méthode de région de confiance biniveau) sont élaborées et mises en œuvre. Au niveau numérique, nous validons les modèles proposés sur quatre types d'instances, chacun associé à une topologie de réseau particulière. Ensuite, une analyse de sensibilité permet d'évaluer l'impact de l'élasticité de la demande et des contraintes de capacité sur la difficulté de résolution du problème.

Cette thèse, qui comporte quatre chapitres, est structurée de la façon suivante. Le premier chapitre introduit la notation et effectue une revue bibliographique. Le deuxième chapitre est consacré à l'étude du problème avec demande linéaire et de ses propriétés. Nous y développons trois formulations quadratiques mixtes sous contraintes linéaires. La première est basée sur une formulation dans l'espace des arcs et les deux autres sur une formulation dans l'espace des chemins. Au troisième chapitre, nous abordons le cas plus général de la demande non linéaire. Après avoir présenté le modèle utilisé, nous développons deux méthodes exactes basées sur des techniques de surapproximation des fonctions non linéaires qui apparaissent dans le modèle, et deux heuristiques basées sur une méthode de région de confiance. Le quatrième et dernier chapitre est voué à l'étude du problème où nous considérons conjointement l'élasticité de la demande et les contraintes de capacité. Après avoir formulé le problème, nous montrons que les contraintes de capacité normalement associées au second niveau peuvent être déplacées au premier niveau, ce qui permet d'éliminer celles-ci du problème du second niveau et d'adapter les méthodes développées pour le problème sans contraintes de capacité.

CHAPITRE 1

NOTATIONS DE BASE ET REVUE DE LITTÉRATURE

Dans ce chapitre, nous posons les notations utilisées, puis nous présentons les résultats existant dans la littérature du problème de tarification sur un réseau (PTR) et quelques extensions. Nous terminons en mettant l'emphase sur la modélisation de l'élasticité de la demande.

1.1 Notations

Soit un réseau $\mathcal{R}$ modélisé par un graphe orienté $\mathcal{G} = (\mathcal{N},\ \mathcal{A})$ où $\mathcal{N}$ est l'ensemble des nœuds du réseau et $\mathcal{A}$ l'ensemble des arcs du réseau. Chaque arc $a \in \mathcal{A}$ possède un coût unitaire fixe noté c_a. Pour tout arc $a \in \mathcal{A}$, x_a représente le flot total de l'arc a. L'ensemble des arcs $\mathcal{A}$ est partitionné selon qu'un tarif puisse ou non être imposé : $\mathcal{A} = \mathcal{A}_1 \bigcup \mathcal{A}_2$ où $\mathcal{A}_1$ désigne les arcs taxables qui appartiennent à un gestionnaire du réseau et $\mathcal{A}_2$ les arcs non taxables. Le vecteur de tarifs $t = (t_a)_{a \in \mathcal{A}_1}$ est indexé par les éléments de $\mathcal{A}_1$. Les usagers disposent d'un ensemble de $\mathcal{K}$ produits à acheminer en utilisant le réseau $\mathcal{R}$ selon un modèle d'affectation de trafic. À chaque produit $k \in \mathcal{K}$ est associé un couple origine-destination $(o(k), d(k))$ et une fonction de demande d_k. L'ensemble des chemins allant du sommet $o(k)$ au sommet $d(k)$ est noté P_k. Pour tout chemin $p \in P_k$, h_p^k représente le flot du produit k sur le chemin p. Chaque chemin est caractérisé par son coût, égal à la somme des coûts fixes des arcs y figurant et des tarifs qui y sont imposés. Les notations utilisées sont reportées dans le tableau 1.1.

$\mathcal{G}$	Graphe modélisant le réseau
$\mathcal{N}$	Ensemble des sommets du réseau
$\mathcal{A}$	Ensemble des arcs du réseau
$\mathcal{A}_1$	Ensemble des arcs taxables
$\mathcal{A}_2$	Ensemble des arcs non taxables
c_a	Coût fixe unitaire de l'arc a
t_a	Tarif unitaire de l'arc a
x_a	Flot total sur l'arc a
$\mathcal{K}$	Ensemble de produits
$(o(k), d(k))$	Couple origine-destination du produit k
d_k	Fonction de demande du produit k
P_k	Ensemble des chemins reliant les sommets $o(k)$ et $d(k)$
x_a^k	Flot du produit k sur l'arc a
h_p^k	Flot du produit k sur le chemin p
i^-	Ensemble des arcs ayant pour destination le sommet i
i^+	Ensemble des arcs ayant pour origine le sommet i

Tableau 1.1 Notations utilisées

1.2 Problème de tarification sur un réseau (PTR)

Le PTR fait intervenir deux agents. Le premier, gestionnaire du réseau (*meneur*), décide d'une politique de tarification. Le second, usagers (*suiveurs*), utilise le réseau en tenant compte de la politique de tarification imposée par le meneur. Le problème consiste pour le meneur à déterminer la politique de tarification qui maximise son revenu en anticipant la réaction des suiveurs aux tarifs. La programmation mathématique à deux niveaux constitue un outil particulièrement bien adapté pour modéliser le type d'interaction hiérarchique et séquentielle présent dans ce problème.

La première formulation du PTR sous forme d'un problème de programmation mathématique à deux niveaux bilinéaire a été proposée par Labbé *et al.* [46]. Le premier niveau est associé à la maximisation des revenus du meneur générés par l'imposition des tarifs sur un sous-ensemble des arcs. Le second niveau est associé à la modélisation du comportement des usagers qui utilisent le réseau pour acheminer leurs produits. Les usagers font le choix des chemins les moins coûteux en réponse à la politique de tarification du meneur. La formulation mathématique du PTR sous la forme d'un programme mathématique à deux niveaux

est donnée comme suit

$$\max_{t,x'} \sum_{a\in\mathcal{A}_1} t_a \sum_{k\in\mathcal{K}} x_a'^k$$

$$x' \in \arg\min_{x} \sum_{k\in\mathcal{K}} \left(\sum_{a\in\mathcal{A}_1} (c_a + t_a) x_a^k + \sum_{a\in\mathcal{A}_2} c_a x_a^k \right) \tag{1.1}$$

$$\text{s.c} \quad \sum_{a\in i^+} x_a^k - \sum_{a\in i^-} x_a^k = \begin{cases} d^k & \text{si } i = o(k), \\ -d^k & \text{si } i = d(k), \\ 0 & \text{sinon.} \end{cases} \qquad \forall k \in \mathcal{K}, \forall i \in \mathcal{N}, \tag{1.2}$$

$$x_a^k \geq 0 \qquad \forall k \in \mathcal{K}, \forall a \in \mathcal{A}. \tag{1.3}$$

La formulation du PTR nous porte à faire plusieurs remarques. Tout d'abord, afin d'éliminer les solutions triviales procurant un revenu infini au meneur, nous supposons qu'il existe, pour chaque produit, un chemin empruntant aucun arc taxable. Cette hypothèse permet de garantir une borne supérieure sur le revenu réalisé par le meneur.

La seconde remarque surgit lorsque la réponse du suiveur n'est pas unique pour une tarification donnée. Deux approches peuvent être considérées. La première approche, *optimiste*, suppose qu'en situation d'indifférence, les usagers coopèrent avec le meneur pour maximiser les revenus générés, c'est-à-dire que pour deux chemins de coût égal, les usagers utilisent le chemin qui offre le revenu du meneur le plus élevé. L'autre approche, *pessimiste*, considère qu'il n'y a pas coopération entre le meneur et les usagers. Dans ce cas, le meneur doit envisager le pire de la part des usagers. En réduisant la valeur du tarif associé à chaque arc d'une valeur ε suffisamment petite, on peut relâcher cette hypothèse. Dans la suite, nous ne considérons que le cas optimiste.

La dernière remarque porte sur le fait que les tarifs ne sont pas contraints à être positifs. En effet, les tarifs négatifs peuvent être fixés sur certains arcs afin d'engendrer des compensations avec d'autres tarifs lorsque les arcs sont utilisés par plus de deux produits.

Il a été démontré par Labbé *et al.* [46] que la fonction objectif du meneur en fonction des

tarifs est linéaire par morceau et discontinue. Cependant, elle est semi-continue supérieure, ce qui garantit l'existence d'une solution optimale du PTR. Le problème du second niveau (1.1)-(1.3) est séparable par produits $k \in \mathcal{K}$. Les sous-problèmes associés aux produits sont linéaires et correspondent à des problèmes de plus courts chemins.

Il a été montré par Labbé *et al.* [46] et Roch *et al.* [57] que le PTR est fortement NP-difficile. Labbé *et al.* [46] montrent aussi que l'ensemble des instances du PTR ne faisant intervenir qu'un seul arc taxable est de complexité polynômiale. On note également que le problème d'optimisation inverse associée au PTR, qui consiste à déterminer les tarifs optimaux compatibles avec les chemins utilisés par les usagers, est polynômial.

1.2.1 Procédure de traitement du réseau

Bouthou *et al.* [11] ont proposé une procédure de simplification de la topologie du réseau. Cette procédure a été renforcée par Dewez [27]. Elle est composée d'une phase de transformation topologique du réseau et d'une phase d'élimination d'arcs. La phase de transformation construit un réseau auxiliaire pour chaque produit où, entre toute paire d'arcs taxables, l'ensemble des parcours non taxables est remplacé par un seul arc dont le coût correspond à celui du plus court chemin. La phase d'élimination élimine l'ensemble des chemins *dominés*. Un chemin est dominé s'il n'existe aucune tarification sous laquelle il est le plus court. Le problème est ensuite résolu sur la base des chemins restants.

En pratique, ce prétraitement réduit le nombre d'arcs potentiellement utilisables de chaque produit et par conséquent le nombre de variables dans la formulation. Il en résulte une réduction du temps de calcul grâce à la résolution des sous-problèmes souvent de plus petite taille mais aucune amélioration de la qualité de la relaxation linéaire.

1.2.2 Formulation en nombres entiers (MIP)

Les méthodes de résolution qui ont été développées font en général intervenir une formulation équivalente sur un seul niveau. Comme problèmes de plus courts chemins, chacun des

sous-problèmes du second niveau (un par produit) est linéaire. Le théorème de dualité forte s'applique et le second niveau peut être remplacé par ses conditions d'optimalité primales-duales. Les conditions d'optimalité incluent des contraintes de complémentarité qui sont par nature combinatoires et difficiles à traiter. Suite à l'introduction de variables binaires et quelques manipulations, une formulation équivalente linéaire en nombres entiers est exhibée. Cette reformulation a été utilisée par Labbé *et al.* [46] pour définir le programme linéaire mixte par arcs suivant :

FORM:ARC

$$\max_{t,x,\lambda} \sum_{k\in\mathcal{K}} d_k \sum_{a\in\mathcal{A}_1} t_a^k \tag{1.4}$$

$$\text{s.c} \quad \sum_{a\in i^+} x_a^k - \sum_{a\in i^-} x_a^k = \begin{cases} 1 & \text{si } i = o(k) \\ -1 & \text{si } i = d(k) \\ 0 & \text{sinon} \end{cases} \qquad \forall i \in \mathcal{N},\ \forall k \in \mathcal{K} \tag{1.5}$$

$$\lambda_j^k - \lambda_i^k \leq c_a + t_a \qquad \forall a = (i,j) \in \mathcal{A}_1,\ k \in \mathcal{K} \tag{1.6}$$

$$\lambda_j^k - \lambda_i^k \leq c_a \qquad \forall a = (i,j) \in \mathcal{A}_2,\ k \in \mathcal{K} \tag{1.7}$$

$$\sum_{a\in\mathcal{A}_1\cup\mathcal{A}_2} c_a x_a^k + \sum_{a\in\mathcal{A}_1} t_a^k = \lambda_{d(k)}^k - \lambda_{o(k)}^k \qquad \forall k \in \mathcal{K} \tag{1.8}$$

$$-M_a^k x_a^k \leq t_a^k \leq M_a^k x_a^k \qquad \forall a \in \mathcal{A}_1,\ \forall k \in \mathcal{K} \tag{1.9}$$

$$-N_a(1 - x_a^k) \leq t_a^k - t_a \leq N_a(1 - x_a^k) \qquad \forall a \in \mathcal{A}_1,\ \forall k \in \mathcal{K} \tag{1.10}$$

$$x_a^k \in \{0,1\} \qquad \forall a \in \mathcal{A}_1, \forall a \in \mathcal{K} \tag{1.11}$$

où λ_i^k est la variable duale du problème du second niveau associée à la contrainte de conservation de flot (1.2) au noeud i pour le produit k, la variable t_a^k représente le terme non linéaire $t_a x_a^k$ et est égal à t_a si l'arc a porte du flot et 0 sinon, et où M_a^k, N_a sont des constantes suffisamment grandes identiques à celles définies par Dewez [27].

La fonction objectif (1.4) maximise le revenu du meneur. Les contraintes (1.5) et (1.11)

(respectivement (1.6) et (1.7)) sont les contraintes d'admissibilités primale (respectivement duale) du problème du second niveau. Les contraintes (1.8) imposent l'égalité entre les objectifs primal et dual du problème du second niveau. Les contraintes (1.9) et (1.10) imposent le tarif t_a^k à zéro lorsque $x_a^k = 0$ et à t_a lorsque $x_a^k = 1$.

Dans le but de proposer un programme linéaire mixte par chemins, Didi *et al.* [29] ont utilisé cette reformulation afin d'obtenir le programme suivant :

FORM:CH

$$\max_{t,h,T,U} \sum_{k\in\mathcal{K}} d_k T_k \tag{1.12}$$

$$\text{s.c} \quad T_k \leq \sum_{a\in p\cap\mathcal{A}_1} t_a + M_k(1-h_p^k) \qquad \forall p\in P_k,\ \forall k\in\mathcal{K} \tag{1.13}$$

$$\sum_{a\in p} c_a + \sum_{a\in p\cap\mathcal{A}_1} t_a - M_p^k(1-h_p^k) \leq U_k \qquad \forall p\in P_k,\ \forall k\in\mathcal{K} \tag{1.14}$$

$$U_k \leq \sum_{a\in p} c_a + \sum_{a\in p\cap\mathcal{A}_1} t_a \qquad \forall p\in P_k,\ \forall k\in\mathcal{K} \tag{1.15}$$

$$U_k = T_k + \sum_{p\in P_k} h_p^k \sum_{a\in p} c_a \qquad \forall k\in\mathcal{K} \tag{1.16}$$

$$\sum_{p\in P_k} h_p^k = 1 \qquad \forall k\in\mathcal{K} \tag{1.17}$$

$$h_p^k \in \{0,1\} \qquad \forall p\in P_k,\ \forall k\in\mathcal{K}. \tag{1.18}$$

où U_k désigne le coût maximal des chemins utilisés. Puisqu'un seul chemin est utilisé pour chaque produit, U_k représente le coût du chemin utilisé, T_k le revenu généré par unité de flot du produit k et M_k et M_p^k des constantes suffisamment grandes identiques à celles définies par Dewez [27]. La fonction objectif (1.12) maximise le revenu du meneur. Les contraintes (1.14) et (1.15) assurent l'utilisation du chemin le plus court par les usagers pour chaque produit. Les contraintes (1.13) et (1.16) permettent le respect de la définition de la variable T_k. Les contraintes (1.17) assurent qu'un seul chemin est utilisé pour chaque produit.

On retrouve également cette reformulation dans Bouhtou *et al.* [11], Brotcorne *et al.*

[13] et [12], Dewez [27], Cirinei [17] et Heilporn [41]. Des outils spécialisés tels que CPLEX permettent de résoudre le problème à l'optimalité pour des instances de petite taille.

1.2.3 Résolution via des méthodes exactes

Parmi les méthodes exactes proposées, on peut faire référence à celles développées par Dewez [27] et Didi *et al.* [29] et Cirinei [17].

Dewez [27] exploite la formulation par arcs (FORM:ARC) et propose une méthode exacte de type plans coupants. Pour ce faire, elle améliore la valeur des constantes (N_a et M_a^k) en calculant des bornes supérieures sur les tarifs. Puis, elle développe de nouvelles inégalités valides liant deux à plusieurs produits. Les expériences numériques montrent que l'ajustement des constantes permet de diviser par deux la valeur du saut d'intégralité à la racine de l'arbre (bien qu'il demeure encore élevé) alors que les inégalités valides réduisent le nombre de nœuds explorés ainsi que le temps de calcul.

Didi *et al.* [29] ont également proposé une méthode exacte basée sur une formulation par chemins du problème du second niveau. Elle est basée sur une énumération intelligente des solutions du problème du second niveau. L'idée est de générer à chaque itération un vecteur chemin appelé multi-chemin où le $k^{\text{ème}}$ élément du vecteur est le chemin utilisé par les usagers pour acheminer le produit k. Puis, une solution réalisable du PTR est obtenue en déterminant les tarifs optimaux compatibles avec les chemins du vecteur multi-chemin, et une borne supérieure est également calculée pour ce vecteur multi-chemin. Cette méthode exploite la monotonie de la borne supérieure. La procédure s'arrête lorsque pour un vecteur multi-chemin donné, la borne inférieure locale est plus grande ou égale à la borne supérieure globale. Les résultats numériques montrent que cette méthode exacte est très efficace sur des instances de petite taille, mais échoue sur des instances de taille moyenne et grande car l'augmentation du nombre de produits implique un très grand nombre de multi-chemins. En plus, d'une itération à une autre, les multi-chemins sont presque identiques et la borne supérieure décroît très lentement.

Cirinei [17] propose une méthode exacte basée sur le concept de multi-chemins proposé par Didi *et al.* [29]. Il améliore la génération des k-plus courts chemins afin de réduire le nombre de multi-chemins non réalisables. Ensuite, il introduit une nouvelle structure de données afin d'éliminer les redondances lors de la génération des multi-chemins. Puis, il exploite cette structure pour le calcul de nouvelles bornes supérieures sur le revenu d'un multi-chemin. Les résultats numériques mettent en évidence l'efficacité de cette méthode sur les instances où les tarifs sont réels et s'avère moins efficace lorsque les tarifs sont positifs.

1.2.4 Résolution via des méthodes approchées

Les méthodes référencées ci-dessus ne sont pas assez efficaces pour résoudre les instances de grande taille du PTR. Il est donc nécessaire d'utiliser des méthodes approchées.

Brotcorne *et al.* [13] ont proposé deux heuristiques. La première est basée sur une recherche séquentielle par arcs exploitant la procédure de résolution exacte proposée par Labbé *et al.* [46] pour le PTR ayant un seul arc taxable. La deuxième est une approche primale-duale basée sur la reformulation du PTR sous forme d'un problème bilinéaire sur un niveau (en remplaçant le problème du second niveau par ses conditions d'optimalité) et sur l'utilisation d'une pénalité quadratique maintenant les tarifs identiques pour chaque arc taxable quel qu'en soit le produit. Cette formulation non linéaire est résolue par l'algorithme de Frank-Wolfe. Les expériences numériques montrent que l'heuristique primale-duale est plus efficace que la recherche séquentielle.

Didi *et al.* [29] ont proposé une heuristique appelée BLOSH qui exploite le fait que la formulation par chemins qu'ils proposent est efficace sur les instances de taille moyenne. L'idée est de diminuer le nombre de variables binaires de cette formulation en fixant pour un sous-ensemble de produits les chemins utilisés au second niveau. Les expériences numériques montrent que les instances de taille moyenne sont résolues par BLOSH à l'optimalité et ceci en un temps raisonnable.

Cirinei [17] a proposé une métaheuristique basée sur une recherche avec tabou. Cette

recherche exploite le voisinage de la méthode locale développée au préalable. La méthode de recherche locale repose sur l'exploration des solutions du problème du second niveau évaluées en termes de revenu du meneur par la résolution d'un problème d'optimisation inverse. Il propose une méthode basée sur la génération de colonnes pour résoudre de manière plus rapide le problème d'optimisation inverse. Cette métaheuristique fournit des solutions à moins de 1% des solutions optimales en des temps raisonnables. Notons que les solutions optimales sont trouvées à partir d'une des formulations en nombres entiers présentées ci-dessus.

Pour terminer cette section, nous évoquons l'utilisation des méthodes de décomposition (benders ou génération de colonnes) pour la résolution du PTR. À partir des formulations FORM:ARC et FORM:CH, nous constatons que le problème est séparable par produits lorsque l'on ne tient pas compte des contraintes impliquant les tarifs. On peut donc utiliser les méthodes de décomposition pour la résolution du PTR. Cependant, ces méthodes de décomposition ont été implémentées par Heilporn [41] et les résultats obtenus ont été mauvais comparés à ceux des méthodes présentées ci-dessus.

1.2.5 Extensions

Brotcorne *et al.* [14] ont étudié un cas particulier du PTR où les contraintes de capacité ont été ajoutées au problème du second niveau. Sous certaines hypothèses, ils montrent que ce problème est équivalent au problème obtenu en déplaçant les contraintes de capacité du second niveau vers le premier niveau. Puis, ils développent une méthode approchée basée sur la relaxation lagrangienne pour résoudre le problème.

Marcotte *et al.* [52] ont étudié le PTR avec segmentation continue de la demande. Les usagers sont répartis selon la valeur monétaire qu'ils accordent au temps. Ils caractérisent à l'aide d'un gradient la variation du trafic sur chaque arc par rapport à une variation des tarifs imposés. Il en découle un algorithme de montée où le point de départ est obtenu en résolvant de façon exacte le problème discrétisé.

Fortin [31] a étudié le PTR avec congestion et segmentation continue de la demande.

L'approche de résolution est la même que celle proposée par Marcotte *et al.* [52]. Il propose une méthode de montée qui utilise comme point de départ la solution optimale du problème discrétisé.

Poirier [56] a étudié le PTR avec congestion et segmentation discrète de la demande où le choix des arcs taxables est déterminé par le modèle (problème de tarification autour d'un centre-ville). La congestion sur chaque arc du réseau est exprimée par une fonction délai qui croît avec le flot sur cet arc.o Dans un premier temps, il discrétise la fonction délai afin d'obtenir un programme linéaire mixte (MIP). Puis, il développe un algorithme itératif qui utilise le MIP comme sous-problème. Sur un petit exemple, cette méthode converge vers la solution exacte, mais ce n'est pas le cas pour tous les problèmes résolus, où elle converge au moins vers un optimum local.

Pour la programmation mathématique à deux niveaux non linéaire, Colson *et al.* [19] proposent une méthode approchée basée sur la méthode de région de confiance. Le modèle de la méthode de région de confiance est un programme bi-niveau linéaire-quadratique où toutes les fonctions du problème sont linéarisées autour de la solution courante, à l'exception de la fonction objectif du second niveau qui est remplacée par une approximation quadratique autour de la solution courante. Puis, le problème du second niveau du modèle est remplacé par les conditions de Kuhn-Tucker afin d'obtenir un programme à un seul niveau. Enfin, les contraintes de complémentarité sont réécrites sous une forme linéaire en introduisant les variables binaires et des grandes constantes. Les résultats numériques réalisés sur les petites instances ont tendance à confirmer l'efficacité de la méthode.

1.3 Modélisation de l'élasticité de la demande d'un bien

Le but de cette partie est de définir le concept d'élasticité de la demande et de présenter un modèle pour le problème étudié.

L'élasticité de la demande d'un bien est un concept économique qui mesure le degré de sensibilité de la demande du bien aux variations de prix (« élasticité-prix »), c'est-à-dire la

réaction des consommateurs aux variations du prix du bien. L'élasticité de la demande d'un bien se définit de la manière suivante

$$\text{élasticité de la demande} = \frac{\text{changement en pourcentage de la quantité demandée du bien}}{\text{changement en pourcentage du prix du bien}}$$

Formellement, elle se définit par

$$E_d = \lim_{\Delta p \longrightarrow 0} \frac{\Delta d/d}{\Delta p/p} = \frac{\partial d}{\partial p} \times \frac{p}{d}$$

où E_d est l'élasticité de la demande, $d(p)$ la fonction de demande, p le prix et $\frac{\partial d}{\partial p}$ la dérivée de la demande par rapport au prix.

Les biens qui existent se classent en deux catégories selon le signe de l'élasticité de leur demande. Nous avons ainsi les biens typiques et les biens atypiques.

1.3.1 Les biens typiques

Un bien est typique lorsque l'élasticité de sa demande est négative ($E_d < 0$) c'est-à-dire que la demande du consommateur diminue quand le prix du bien augmente. La décroissance de la demande avec le prix s'interprète par le fait que le consommateur préfère payer moins cher. De plus, pour respecter son budget, il doit diminuer la quantité de sa consommation lorsque le prix du bien augmente. La quasi-totalité des biens sont typiques. Par définition, la demande d'un bien est élastique si la valeur absolue de son élasticité est supérieure à 1, sinon elle est inélastique. Dans la suite, lorsque nous parlerons de l'élasticité d'un bien typique, nous considérerons plutôt sa valeur absolue.

Les courbes de la demande représentées à la figure 1.1 ci-dessous sont tirées de Kant [45].

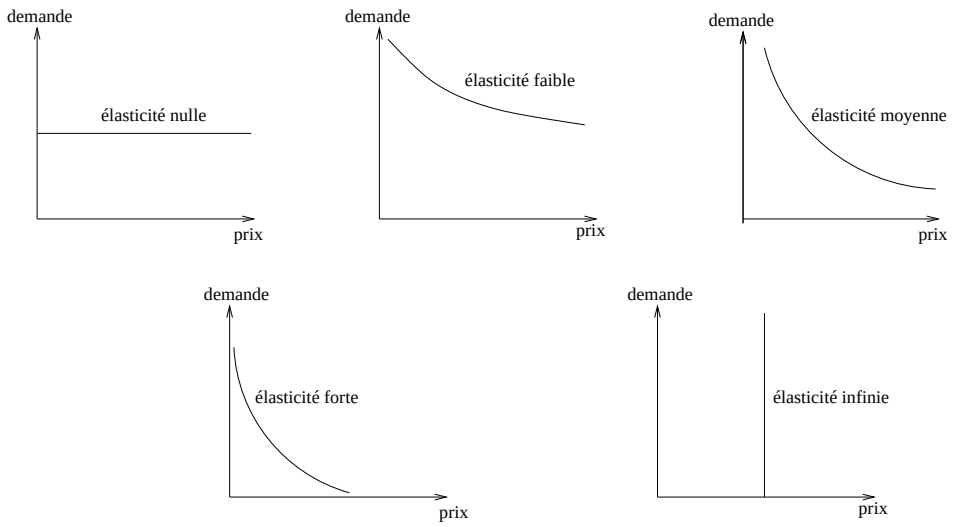


Figure 1.1 Évolution de la demande en fonction du prix

Quand l'élasticité de la demande d'un bien est nulle (demande fixe), la courbe de sa demande est horizontale, le consommateur (fanatique) est prêt à acheter quelque soit le prix du bien (sous réserve de ne pas dépasser son budget), comme pour un billet de la finale du championnat du monde de son sport préféré. Quand l'élasticité de la demande d'un bien est infinie, la courbe de sa demande est verticale : il y a tellement d'offre, qui tire les prix du bien vers le bas, que si un vendeur augmente un peu ses prix, sa demande chute à zéro car les consommateurs iront chez la compétition.

Lorsque la demande d'un bien est peu élastique par rapport aux prix, une hausse du prix du bien n'entraîne pas de grande baisse de la demande, comme pour l'essence (pour ceux qui ont une voiture) et le pain (il faut bien manger).

1.3.2 Les biens atypiques

Un bien est atypique lorsque l'élasticité de sa demande est positive ($E_d > 0$) c'est-à-dire que la demande du consommateur augmente avec le prix du bien. Plus le prix est élevé, plus la demande augmente, ce qui semble paradoxal. On retrouve dans cette catégorie les biens de luxe et certains consommateurs qui souhaitent en effet se démarquer des autres par les biens qu'ils possèdent. Ces demandeurs veulent donc des prix élevés, synonymes d'une

qualité supérieure ou l'appartenance à une certaine classe sociale. C'est l'effet de snobisme ou d'ostentation. Ainsi, le constructeur d'automobile Rolls-Royce a significativement augmenté ses prix à la fin des années 1960 pour se démarquer de la concurrence et a néanmoins augmenté ses revenus.

1.3.3 Modélisation de la demande

Les biens proposés dans le domaine du transport sont typiques car la demande des biens diminue lorsque le prix augmente. Dans ce domaine, la demande élastique mesure le degré de sensibilité de la demande en produits en fonction des coûts (fixes + tarifs imposées) des arcs (ou chemins) empruntés pour acheminer les produits de leur origine vers leur destination. Plus précisément, la demande d'un produit est une fonction non négative, continue et décroissante du coût des arcs utilisés pour acheminer ce produit. Dans ce cas, l'hypothèse faite sur le PTR qu'il existe toujours un chemin n'empruntant aucun arc taxable pour chaque produit peut être négligée.

La modélisation du comportement de la demande est définie à partir du modèle utilisé pour le problème du second niveau. Une fois que le meneur a décidé d'une politique tarifaire, les usagers doivent choisir les chemins pour acheminer leurs produits. Le problème correspondant est une affectation de trafic.

Affectation de trafic

La méthode d'affectation la plus classique et probablement la plus utilisée est basée sur la notion d'*équilibre de trafic*. Wardrop [65] a développé le principe d'équilibre-usager qui s'énonce comme suit : à l'équilibre, le coût de transport de chaque usager est minimal. Supposons que cette approche est utilisée pour l'affectation du trafic. En l'absence des contraintes de capacité, les usagers utilisent les plus courts chemins pour acheminer leurs produits. Nous avons donc un problème de plus courts chemins. Dans ce cas, la demande d'un produit donné est une fonction qui dépend du coût du plus court chemin dudit produit. En présence des

contraintes de capacité, plusieurs modèles ont été considérés mais le plus répandu suppose que l'on associe des variables duales aux contraintes de capacité. La variable duale associée à un arc a de capacité finie est alors interprétée comme un coût correspondant au délai d'attente pour accéder à cet arc. Ainsi, si l'arc a n'est pas saturé, la variable duale associée à sa contrainte de capacité est nulle tandis qu'elle est positive ou nulle si l'arc est saturé. Dans ce cas, la fonction de demande d'un produit donné est une fonction qui dépend du coût du chemin utilisé le plus long pour acheminer ledit produit.

Une approche d'affectation du trafic avec contraintes de capacité différente de l'approche classique et plus complexe a été introduite par Marcotte *et al.* [49]. L'originalité du modèle proposé repose sur l'hypothèse que les usagers utilisent plutôt des *stratégies* que des chemins. Une stratégie dicte, en chacun des nœuds du réseau, un ordre de préférences parmi les successeurs. Celle-ci doit proposer un arc de déviation lorsque le premier choix correspond à un arc de capacité finie. Ainsi, contrairement à un chemin, une stratégie est toujours réalisable. Ils définissent ensuite le coût d'une stratégie à l'aide des probabilités d'accès aux arcs comme l'espérance du coût du chemin utilisé. Ils définissent également une extension du principe d'équilibre-usager au modèle stratégique.

Dans cette thèse, nous utilisons l'approche classique en supposant que le problème du second niveau est une affectation de trafic où chaque usager minimise son coût de transport (équilibre-usager). En présence des contraintes de capacité, nous montrons au chapitre 4 que les contraintes de capacité peuvent être déplacées du second niveau vers le premier niveau. Il en résulte que les variables duales associées aux contraintes de capacité sont nulles et donc les chemins utilisés pour acheminer un produit donné ont le même coût.

CHAPITRE 2

PROBLÈME DE TARIFICATION SUR UN RÉSEAU AVEC DEMANDE LINÉAIRE

Dans ce chapitre, nous étudions le problème de tarification sur un réseau avec demande linéaire (DL). Nous supposons que la fonction de demande du produit k est $d_k(U_k) = a_k - b_k U_k$ (voir figure 2.1)

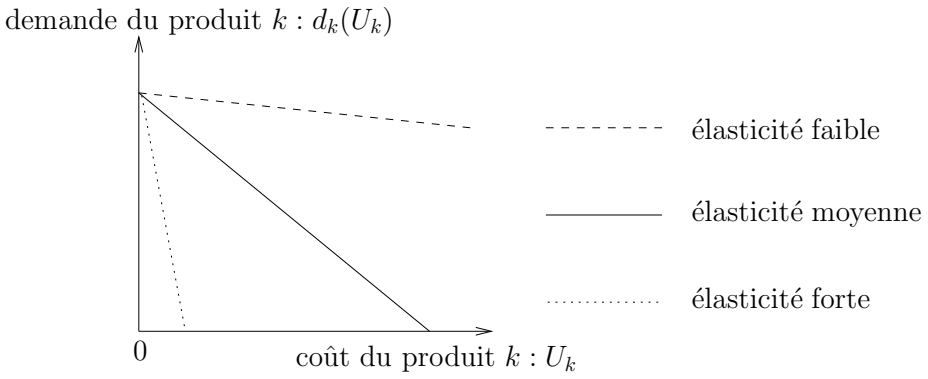


Figure 2.1 Demande linéaire du produit $k : d_k(U_k) = a_k - b_k U_k$

où a_k, b_k sont des constantes positives et U_k le coût du chemin utilisé pour acheminer une unité du produit k.

Nous exploitons les formulations proposées par Labbé *et al.* [46] et Didi *et al.* [29] pour résoudre le PTR afin de développer trois programmes quadratiques mixtes sous contraintes linéaires (MIP) pour résoudre le DL. Quelques propriétés du DL sont énoncées et des cas polynômiaux sont présentés.

Ce chapitre s'organise comme suit. À la section 2.1 un exemple du DL est donné sur un réseau de petite taille. Nous proposons trois formulations quadratiques mixtes sous contraintes linéaires pour résoudre le DL à la section 2.2. Quelques propriétés du DL sont données aux sections 2.3 et 2.4. À la section 2.5, nous présentons les cas polynômiaux du DL. Pour en

finir, les expériences numériques sont présentées à la section 2.6.

2.1 Exemple du DL

Considérons le réseau décrit à la figure 2.2 comprenant deux produits.

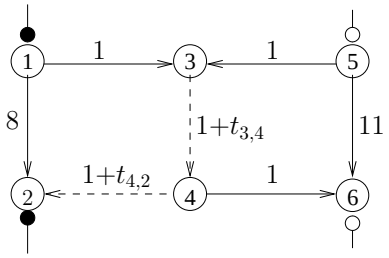


Figure 2.2 Exemple de réseau du DL

Les informations sur les produits sont données au tableau 4.1 suivant :

	o - d	demande	chemins	coût
produit 1	$1 \longrightarrow 2$	$d_1(U_1) = 25 - U_1$	$1 \to 3 \to 4 \to 2$ $1 \to 2$	$3 + t_{3,4} + t_{4,2}$ 8
produit 2	$5 \longrightarrow 6$	$d_2(U_2) = 50 - 2U_2$	$5 \to 3 \to 4 \to 6$ $5 \to 6$	$3 + t_{3,4}$ 11

Tableau 2.1 Données de l'exemple 2.1

où U_1 (resp. U_2) représente le coût unitaire du chemin utilisé pour acheminer le produit 1 (resp. produit 2). Le but est de fixer les tarifs sur les arcs taxables (en pointillés) (3,4) et (4,2) afin de maximiser le revenu du meneur.

En imposant les tarifs positifs, la solution optimale est obtenue pour les valeurs $t_{3,4} = 5$ et $t_{4,2} = 0$ et est donnée par :

	chemin utilisé	coût	revenu généré
produit 1	$1 \to 3 \to 4 \to 2$	$U_1 = 8$	$(25 - 8) \times (5 + 0) = 85$
produit 2	$5 \to 3 \to 4 \to 6$	$U_2 = 8$	$(50 - 2 \times 8) \times 5 = 170$
		revenu total	$85 + 170 = 255$

Tableau 2.2 Solution optimale avec les tarifs positifs

Nous remarquons que les chemins $1 \to 3 \to 4 \to 2$ et $1 \to 2$ du produit 1 ont le même coût 8. Sachant qu'il y a une coopération entre le meneur et les suiveurs, les suiveurs utiliseront le chemin $1 \to 3 \to 4 \to 2$ car il génère plus de revenu pour le meneur que le chemin $1 \to 2$.

En supposant qu'il n'y a pas de restriction sur les tarifs, la solution optimale est obtenue pour les valeurs $t_{3,4} = 8$ et $t_{4,2} = -3$ et est donnée par :

	Chemin utilisé	coût	revenu généré
produit 1	$1 \to 3 \to 4 \to 2$	$U_1 = 8$	$(25 - 8) \times (8 - 3) = 85$
produit 2	$5 \to 3 \to 4 \to 6$	$U_2 = 11$	$(50 - 2 \times 11) \times 8 = 224$
		revenu total	$85 + 224 = 305$

Tableau 2.3 Solution optimale avec les tarifs positifs et négatifs

Le tarif négatif sur l'arc $(4, 2)$ représente une subvention aux usagers qui utilisent cet arc.

2.2 Formulations quadratiques mixtes sous contraintes linéaires du DL

2.2.1 Formulation par arcs

Afin de proposer une formulation quadratique mixte sous contraintes linéaires par arcs, nous exploitons la formulation par arcs FORM:ARC proposée par Labbé *et al.* [46] pour le PTR (voir section 1.2.2) . À partir de FORM:ARC, nous introduisons la variable T_k qui est le revenu généré par une unité de flot du produit k. T_k est égale à $\sum_{a \in \mathcal{A}_1} t_a^k$ où t_a^k a été utilisé dans FORM:ARC pour rendre linéaire le terme $t_a x_a^k$ et représente le revenu généré par unité

de flot de l'arc a du produit k. Ainsi, le coût U_k du plus court chemin se réécrit de la manière suivante :

$$U_k = \sum_{a\mathcal{A}_1 \cup \mathcal{A}_2} c_a x_a^k + \sum_{a \in \mathcal{A}_1} t_a x_a^k = \sum_{a\mathcal{A}_1 \cup \mathcal{A}_2} c_a x_a^k + \sum_{a \in \mathcal{A}_1} t_a^k = \sum_{a\mathcal{A}_1 \cup \mathcal{A}_2} c_a x_a^k + T_k.$$

En remplaçant, dans la fonction objectif, la demande d_k par sa valeur $a_k - b_k U_k$, le terme associé au produit k se réécrit comme suit :

$$\begin{aligned} d_k \sum_{a \in \mathcal{A}_1} t_a^k &= (a_k - b_k U_k) T_k \\ &= a_k T_k - b_k \left(\sum_{a\mathcal{A}_1 \cup \mathcal{A}_2} c_a x_a^k + T_k \right) T_k \\ &= a_k T_k - b_k \sum_{a\mathcal{A}_1 \cup \mathcal{A}_2} c_a T_k x_a^k - b_k T_k^2. \end{aligned}$$

Pour rendre linéaire le terme bilinéaire $T_k x_a^k$, nous utilisons la linéarisation standard qui introduit une nouvelle variable $T_k^a = T_k x_a^k$. Puis, nous ajoutons au modèle l'ensemble de contraintes linéaires ci-dessous qui permet le respect de la définition de la variable T_k^a.

$$0 \leq T_k^a \leq M_k x_a^k \qquad \forall a \in \mathcal{A}_1 \cup \mathcal{A}_2,\ \forall k \in \mathcal{K} \qquad (2.1)$$

$$-M_k(1 - x_a^k) \leq T_k^a - T_k \leq M_k(1 - x_a^k) \qquad \forall a \in \mathcal{A}_1 \cup \mathcal{A}_2,\ \forall k \in \mathcal{K} \qquad (2.2)$$

où M_k est une constante suffisamment grande. Les contraintes (2.1) permettent de fixer T_k^a à 0 lorsque $x_a^k = 0$. Autrement dit, le revenu du produit k est nul pour cet arc lorsque son flot est nul. Les contraintes (2.2) permettent de fixer T_k^a à T_k lorsque $x_a^k = 1$, c'est-à-dire que le revenu du produit k est T_k pour cet arc lorsque son flot n'est pas nul. Nous obtenons la

formulation par arcs suivante

DL:ARC

$$\max_{t,x,\lambda,T_k} \sum_{k\in\mathcal{K}} \left(a_k T_k - b_k \sum_{a\in\mathcal{A}_1\cup\mathcal{A}_2} c_a T_k^a - b_k T_k^2 \right)$$

$$\text{s.c} \quad \sum_{a\in i^+} x_a^k - \sum_{a\in i^-} x_a^k = \begin{cases} 1 & \text{si } i = o(k) \\ -1 & \text{si } i = d(k) \\ 0 & \text{sinon} \end{cases} \qquad \forall i \in \mathcal{N},\ \forall k \in \mathcal{K}$$

$$\lambda_j^k - \lambda_i^k \leq c_a + t_a \qquad \forall a = (i,j) \in \mathcal{A}_1,\ k \in \mathcal{K}$$

$$\lambda_j^k - \lambda_i^k \leq c_a \qquad \forall a = (i,j) \in \mathcal{A}_2,\ k \in \mathcal{K}$$

$$\sum_{a\in\mathcal{A}_1\cup\mathcal{A}_2} c_a x_a^k + \sum_{a\in\mathcal{A}_1} t_a^k = \lambda_{d(k)}^k - \lambda_{o(k)}^k \qquad \forall k \in \mathcal{K}$$

$$T_k = \sum_{a\in\mathcal{A}_1} t_a^k \qquad \forall k \in \mathcal{K}$$

$$-M_a^k x_a^k \leq t_a^k \leq M_a^k x_a^k \qquad \forall a \in \mathcal{A}_1,\ \forall k \in \mathcal{K}$$

$$-N_a(1 - x_a^k) \leq t_a^k - t_a \leq N_a(1 - x_a^k) \qquad \forall a \in \mathcal{A}_1,\ \forall k \in \mathcal{K}$$

$$0 \leq T_k^a \leq M^k x_a^k \qquad \forall a \in \mathcal{A}_1 \cup \mathcal{A}_2,\ \forall k \in \mathcal{K}$$

$$-M_k(1 - x_a^k) \leq T_k^a - T_k \leq M_k(1 - x_a^k) \qquad \forall a \in \mathcal{A}_1 \cup \mathcal{A}_2,\ \forall k \in \mathcal{K}$$

$$x_a^k \in \{0,1\} \qquad \forall a \in \mathcal{A}_1, \forall k \in \mathcal{K}$$

où M_a^k, N_a et M_k sont des grandes constantes identiques à celles définies par Dewez [27]. DL:ARC est un problème de maximisation d'une fonction objectif quadratique concave sous contraintes linéaires mixtes (MIP).

2.2.2 Formulations par chemins

À partir de la formulation par chemins FORM:CH proposée par Didi *et al.* [29] (voir section 1.2.2), nous remplaçons la demande d_k du produit k par sa valeur $a_k - b_k U_k$. La

fonction objectif de ce produit se réécrit alors comme suit :

$$\begin{aligned} d_k T_k &= (a_k - b_k U_k) T_k \\ &= a_k T_k - b_k T_k \left(T_k + \sum_{p \in P_k} h_p^k \sum_{a \in p} c_a \right) \\ &= a_k T_k - b_k T_k^2 - \sum_{p \in P_k} T_k h_p^k \sum_{a \in p} c_a. \end{aligned}$$

On note la présence du terme bilinéaire $T_k h_p^k$. Afin de le rendre linéaire, nous utilisons encore une fois la technique de linéarisation standard qui introduit une nouvelle variable $T_k^p = T_k h_p^k$. Puis, nous ajoutons au modèle l'ensemble des contraintes linéaires ci-dessous qui permet le respect de la définition de la variable T_k^p, c'est-à-dire T_k^p est égal à T_k si le chemin p est utilisé et 0 sinon.

$$\begin{aligned} 0 \leq T_k^p \leq M_p^k h_p^k \qquad & \forall p \in P_k,\ \forall k \in \mathcal{K} \\ -M_p^k (1 - h_p^k) \leq T_k^p - T_k \leq M_p^k (1 - h_p^k) \qquad & \forall p \in P_k,\ \forall k \in \mathcal{K}. \end{aligned}$$

Nous obtenons ainsi la première formulation quadratique mixte sous contraintes linéaires par chemins suivante

DL:CH1

$$\max_{t,h,T,U} \sum_{k\in\mathcal{K}} \left(a_k T_k - b_k T_k^2 - b_k \sum_{p\in P_k} T_k^p \sum_{a\in p} c_a \right)$$

$$\begin{aligned}
\text{s.c} \quad & T_k \le \sum_{a\in p\cap\mathcal{A}_1} t_a + M_p^k(1-h_p^k) && \forall p\in P_k,\ \forall k\in\mathcal{K} \\
& \sum_{a\in p} c_a + \sum_{a\in p\cap\mathcal{A}_1} t_a - M_p^k(1-h_p^k) \le U_k && \forall p\in P_k,\ \forall k\in\mathcal{K} \\
& U_k \le \sum_{a\in p} c_a + \sum_{a\in p\cap\mathcal{A}_1} t_a && \forall p\in P_k,\ \forall k\in\mathcal{K} \\
& U_k = T_k + \sum_{p\in P_k} h_p^k \sum_{a\in p} c_a && \forall k\in\mathcal{K} \\
& \sum_{p\in P_k} h_p^k = 1 && \forall k\in\mathcal{K} \\
& -M_k(1-h_p^k) \le T_k^p - T_k \le M_k(1-h_p^k) && \forall p\in P_k,\ \forall k\in\mathcal{K} \\
& 0 \le T_k^p \le M_k h_p^k && \forall p\in P_k,\ \forall k\in\mathcal{K} \\
& h_p^k \in \{0,1\} && \forall p\in P_k,\ \forall k\in\mathcal{K}.
\end{aligned}$$

où M_p^k et M_k sont des grandes constantes identiques à celles définies par Dewez [27]. DL:CH1 est un problème de maximisation d'une fonction objectif quadratique concave sous contraintes linéaires mixtes (MIP).

Dans le but d'obtenir une formulation qui contient moins de contraintes impliquant les variables binaires que DL:CH1, la linéarisation du terme $T_k h_p^k$ est faite de manière différente de la linéarisation standard. En fait, nous conservons la variable T_k^p qui remplace le terme non linéaire $T_k h_p^k$ mais nous exploitons le fait que pour chaque produit, un seul chemin est utilisé à l'optimalité. Notons par p_k^* le chemin utilisé pour acheminer le produit k à l'optimalité, on a $h_{p_k^*}^k = 1$ et $h_p^k = 0$ pour tout chemin $p \in P_k \backslash \{p_k^*\}$, ce qui implique que $T_k^{p_k^*} = T_k$ et $T_k^p = 0$ pour tout chemin $p \in P_k \backslash \{p_k^*\}$. Pour exprimer cela, nous ajoutons au modèle les contraintes

suivantes

$$0 \leq T_k^p \leq M_p^k h_p^k \qquad \forall p \in P_k,\ \forall k \in \mathcal{K} \tag{2.3}$$

$$T_k - \sum_{p \in P_k} T_k^p = 0 \qquad \forall k \in \mathcal{K}. \tag{2.4}$$

Les contraintes (2.3) permettent de fixer T_k^p à 0 si le chemin p n'est pas utilisé. Les contraintes (2.4) permettent de fixer la variable $T_k^{p_k^*}$ associé au seul chemin utilisé p_k^* à T_k. Nous obtenons ainsi la deuxième formulation par chemins suivante

DL:CH2

$$\max_{t,h,T,U} \sum_{k \in \mathcal{K}} \left(a_k T_k - b_k T_k^2 - b_k \sum_{p \in P_k} T_k^p \sum_{a \in p} c_a \right)$$

$$\begin{aligned}
\text{s.c} \quad & T_k \leq \sum_{a \in p \cap \mathcal{A}_1} t_a + M_p^k (1 - h_p^k) && \forall p \in P_k,\ \forall k \in \mathcal{K} \\
& \sum_{a \in p} c_a + \sum_{a \in p \cap \mathcal{A}_1} t_a - M_p^k (1 - h_p^k) \leq U_k && \forall p \in P_k,\ \forall k \in \mathcal{K} \\
& U_k \leq \sum_{a \in p} c_a + \sum_{a \in p \cap \mathcal{A}_1} t_a && \forall p \in P_k,\ \forall k \in \mathcal{K} \\
& U_k = T_k + \sum_{p \in P_k} h_p^k \sum_{a \in p} c_a && \forall k \in \mathcal{K} \\
& \sum_{p \in P_k} h_p^k = 1 && \forall k \in \mathcal{K} \\
& T_k - \sum_{p \in P_k} T_k^p = 0 && \forall k \in \mathcal{K} \\
& 0 \leq T_k^p \leq M_p^k h_p^k && \forall p \in P_k,\ \forall k \in \mathcal{K} \\
& h_p^k \in \{0, 1\} && \forall p \in P_k,\ \forall k \in \mathcal{K}.
\end{aligned}$$

où M_p^k est une grande constante identique à celle définie par Dewez [27].

De même que la première formulation par chemins, DL:CH2 est un problème de maximisation d'une fonction objectif quadratique concave sous contraintes linéaires mixtes (MIP).

Cependant, DL:CH2 contient $2\sum_{k\in\mathcal{K}}|P_k| - |\mathcal{K}|$ contraintes impliquant les variables binaires de moins que DL:CH1. Ces contraintes proviennent de la linéarisation des termes bilinéaires du modèle.

2.3 Fonction objectif du meneur

Labbé et *al.* [46] ont montré que la fonction objectif du PTR en fonction des tarifs est linéaire par morceaux, discontinue et semi-continue supérieurement. Dans le même ordre d'idée, on montre que :

Proposition 2.3.1 *La fonction objectif du meneur du DL est quadratique concave par morceaux, discontinue et semi-continue supérieurement.*

Preuve Considérons la formulation par chemins DL:CH2. Supposons que pour chaque produit k, l'on connaisse le chemin p_k^* utilisé. La fonction objectif du meneur devient

$$\sum_{k\in\mathcal{K}}\left(a_kT_k - b_kT_k^2 - b_kT_k\sum_{a\in p_k^*}c_a\right) = \sum_{k\in\mathcal{K}}\left(\left(a_k - b_k\sum_{a\in p_k^*}c_a\right)T_k - b_kT_k^2\right).$$

Cette fonction quadratique est continue et concave car $b_k \geq 0$.

Pour la suite, nous notons le vecteur de tarif $t = (t_1, t_2, \ldots, t_j, \ldots, t_{|\mathcal{A}_1|})$. On dira que $t \leq t^i$ si $t_j \leq t_j^i$ pour tout $j = 1, \ldots, |\mathcal{A}_1|$ et que $t \in [t^i, t^{i+1}]$ si $t_j^i \leq t_j \leq t_j^{i+1}$ pour tout $j = 1, \ldots, |\mathcal{A}_1|$ où t^i désigne le vecteur de tarifs correspondant à un optimum local.

Pour un vecteur de tarifs t donné, la solution optimale du problème du second niveau est atteinte à un sommet du polyèdre défini par les contraintes du problème du second niveau. Le sommet $h(t^0) = h^0$ reste optimal tant que t ne dépasse pas un vecteur seuil t^1. Donc h^0 reste optimal pour le problème du second niveau pour t appartenant à $[t^0, t^1]$. La fonction objectif du meneur est concave sur $[t^0, t^1]$ tel que présenté ci-dessus car le vecteur des chemins utilisés est connu a priori. Lorsque t augmente au-delà de t^1, le sommet h^0 n'est plus optimal pour le problème du second niveau. Soit h^1 le nouveau sommet optimal. h^1 reste optimal pour les

valeurs de t appartenant à $]t^1, t^2]$. Le reste du processus se construit de manière similaire. Ainsi, si t est plus grand qu'un certain vecteur de tarifs t^n, alors le sommet h^n est composé des chemins de la compétition de chaque produit. Dès lors, la valeur de la fonction objectif du meneur chute à 0. Nous concluons que la fonction objectif du meneur est concave et continue sur $[t^i,\ t^{i+1}]$ pour tout $i = 0, \ldots, n$.

Il reste à montrer que la fonction objectif du meneur est discontinue et semi-continue supérieurement. Soient les trois vecteurs de tarifs t^{i-1}, t^i et t^{i+1} tels que définis précédemment. Le sommet h^{i-1} (resp. h^i) est optimal pour le problème du second niveau pour les valeurs de t appartenant à $[t^{i-1},\ t^i]$ (resp. $[t^i,\ t^{i+1}]$). Pour $t \in [t^{i-1},\ t^i]$ (resp. $t \in [t^i,\ t^{i+1})$, la fonction objectif du meneur est $F(t, h^{i-1})$ (resp. $F(t, h^i)$), ce qui montre que F est discontinue aux points $t = t^i$. Cependant, vu qu'il y a coopération entre le meneur et les suiveurs, la fonction objectif F au point t^i est définie par $\max\{F(t^i, h^{i-1}), F(t^i, h^i)\}$. Ainsi aux points de discontinuité t^i, pour tout t proche de t^i, soit le revenu généré en imposant les tarifs du vecteur t est proche de celui de t^i soit il est inférieur, ce qui montre que la fonction objectif est semi-continue supérieure.

□

Cette propriété est illustrée sur le réseau de la figure 2.3 ayant un arc taxable et deux produits.

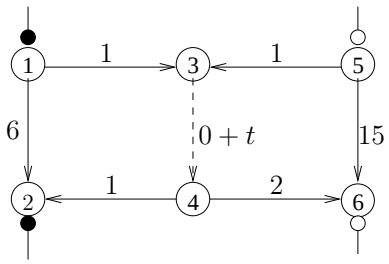


Figure 2.3 Réseau avec un arc taxable et deux produits

Les informations sur les produits sont données au tableau 2.4.

	o - d	demande	chemins	coût
produit 1	$1 \longrightarrow 2$	$d_1(U_1) = 20 - 3U_1$	$1 \to 3 \to 4 \to 2$ $1 \to 2$	$2 + t$ 6
produit 2	$5 \longrightarrow 6$	$d_2(U_2) = 23 - U_2$	$5 \to 3 \to 4 \to 6$ $5 \to 6$	$3 + t$ 15

Tableau 2.4 Données du réseau de la figure 2.3

Le but est de fixer le tarif sur l'arc taxable $(3, 4)$ afin de maximiser le revenu du meneur.

Nous remarquons que si le tarif de l'arc $(3, 4)$ est inférieur ou égal à 4, les chemins utilisés sont $1 \to 3 \to 4 \to 2$ et $5 \to 3 \to 4 \to 6$, et la totalité du flot emprunte l'arc $(3, 4)$. Dès que le tarif sur l'arc $(3, 4)$ devient strictement supérieur à 4, cet arc n'est plus attractif pour le produit 1 qui emprunte le chemin $1 \to 2$, mais reste attractif pour le produit 2 s'il est compris entre 4 et 12. Finalement, dès que le tarif sur cet arc devient strictement supérieur à 12, cet arc n'est plus attractif pour les deux produits. La fonction objectif du meneur est donc définie par :

$$\begin{cases} (20 - 3 \times (2 + t))t + (23 - 1 \times (3 + t))t = 34t - 4t^2 & \text{si } \ t \in [0, 4] \\ (23 - 1 \times (3 + t))t = 20t - t^2 & \text{si } \ t \in]4, 12] \\ 0 & \text{si } \ t \in]12, +\infty[\end{cases}$$

et est représentée à la figure 2.4 ci-dessous. Les courbes en gras représentent la fonction objectif du meneur.

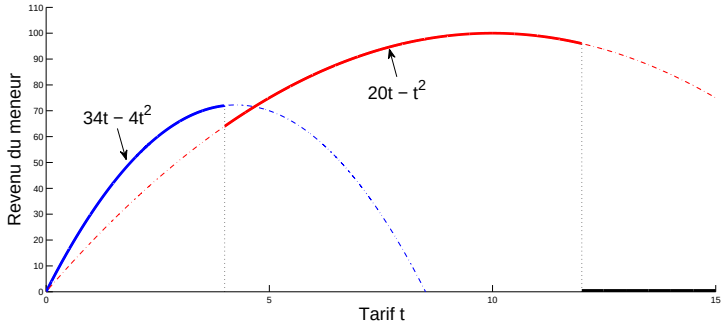


Figure 2.4 Valeur de la fonction objectif en fonction du tarif

Cette fonction est concave par morceaux, discontinue et semi-continue supérieurement. La solution optimale est obtenue pour la valeur $t = 10$ et le revenu du meneur est 100.

2.4 Borne supérieure sur le revenu du meneur

Sans perte de généralité, considérons la deuxième formulation par chemins DL:CH2. Soit $\gamma^k(\infty)$ le coût du plus court chemin allant de $o(k)$ à $d(k)$ lorsque les tarifs du meneur sont fixés à l'infini. La valeur de $\gamma^k(\infty)$ représente le coût du chemin de la compétition du produit k. Soit $\gamma^k(0)$ le coût du plus court chemin allant de $o(k)$ à $d(k)$ lorsque les tarifs du meneur sont fixés à 0.

Proposition 2.4.1 *Une borne supérieure (BS) sur le revenu du meneur est donnée par*

$$BS = \sum_{k\in\mathcal{K}} \left(a_k T_k^* - b_k \left(T_k^*\right)^2 - b_k T_k^* \sum_{a\in p_0^k} c_a \right)$$

$$où \quad T_k^* = \min \left\{ \gamma^k(\infty) - \gamma^k(0),\ \frac{1}{2b_k} \left(a_k - b_k \sum_{a\in p_0^k} c_a \right) \right\} \quad \forall k \in \mathcal{K}.$$

Preuve La fonction objectif du meneur associée au produit k avant la linéarisation est donnée par :

$$a_k T_k - b_k T_k^2 - b_k \sum_{p \in P_k} T_k h_p^k \sum_{a \in p} c_a.$$

Soit p_0^k le plus court chemin du produit k obtenu en fixant tous les tarifs à 0. Le coût fixe du chemin p_0^k est le plus petit coût fixe parmi les coûts des chemins du produit k, donc

$$a_k T_k - b_k T_k^2 - b_k \sum_{p \in P_k} T_k h_p^k \sum_{a \in p} c_a \ \leq \ a_k T_k - b_k T_k^2 - b_k T_k \sum_{a \in p_0^k} c_a$$

La fonction à droite de l'inégalité est concave et atteint sa valeur maximale lorsque T_k est égal à $\dfrac{1}{2b_k}\left(a_k - b_k \sum_{a \in p_0^k} c_a\right)$. Puisque $\gamma^k(\infty) - \gamma^k(0)$ est la borne supérieure sur T_k qui est le revenu généré par unité de flot du produit k, la borne supérieure sur le revenu généré par le produit k notée BS_k est donnée par

$$\mathrm{BS}_k = a_k T_k^* - b_k \left(T_k^*\right)^2 - b_k T_k^* \sum_{a \in p_0^k} c_a \quad \text{où} \quad T_k^* = \min\left\{\gamma^k(\infty) - \gamma^k(0),\ \frac{1}{2b_k}\left(a_k - b_k \sum_{a \in p_0^k} c_a\right)\right\}$$

Finalement, en sommant ces bornes supérieures sur l'ensemble des produits, nous obtenons la borne supérieure BS.

Notons que si pour un produit k, il n'existe aucun chemin possédant seulement les arcs non taxables, alors $T_k^* = \dfrac{1}{2b_k}\left(a_k - b_k \sum_{a \in p_0^k} c_a\right)$.

□

Cette borne supérieure (BS) n'est pas toujours atteinte et dans certains cas, peut être de mauvaise qualité.

2.5 Quelques cas polynômiaux du DL

Bien que le DL soit NP-difficile, il existe certaines instances qui appartiennent à la classe des problèmes polynômiaux.

2.5.1 Le DL où tous les arcs du réseau sont taxables

On considère le DL où tous les arcs du réseau appartiennent au meneur ($\mathcal{A} = \mathcal{A}_1$ et $\mathcal{A}_2 = \emptyset$). Dans ce cas, le meneur est le seul à proposer des arcs aux usagers pour acheminer leurs produits. On parle alors de *monopole*.

Avant de présenter les propositions, notons par n le nombre total d'arcs distincts des chemins p_0^k, $k \in \mathcal{K}$.

Proposition 2.5.1 *Le DL où tous les arcs du réseau sont taxables se résout en $\mathcal{O}(|\mathcal{K}|^2 n)$ si le système linéaire suivant*

$$\sum_{a \in p_0^k} t_a \;=\; \min \left\{ \gamma^k(\infty) - \gamma^k(0), \frac{1}{2b_k} \left(a_k - b_k \sum_{a \in p_0^k} c_a \right) \right\} \qquad \forall k \in \mathcal{K} \qquad (2.5)$$

possède au moins une solution

Preuve Soit la solution obtenue en faisant passer le flot sur les chemins p_0^k du produit k, puis en imposant les tarifs sur les arcs de la façon à satisfaire le système (2.5) et

$$t_a \;=\; +\infty \qquad \forall a \notin \bigcup_{k \in \mathcal{K}} \left\{ a \mid a \in p_0^k \right\} \qquad (2.6)$$

Le système linéaire (2.5) est un système qui possède $|\mathcal{K}|$ équations à n inconnues. L'affectation (2.5) permet à la fonction objectif du meneur d'atteindre sa valeur maximale car le revenu généré par chaque produit est maximal (la borne supérieure du meneur est atteinte, voir section 2.4) et l'affectation (2.6) garantit que les chemins p_0^k sont les plus courts. On sait que pour tout chemin $p_0^k, p_0^{k'}$ avec $k \neq k'$, il existe toujours un arc a tel que $a \in p_0^k$ et

$a \notin p_0^{k'}$ ou réciproquement car les paires origine-destination sont distinctes et tous les arcs du réseau sont taxables. Donc les lignes de la matrice du système linéaire (2.5) sont deux à deux linéairement indépendantes. Par contre, si $n \geq |\mathcal{K}|$, le système linéaire (2.5) n'est pas toujours linéairement indépendant (de plein rang). Cependant s'il est de plein rang, il possède au moins une solution car les tarifs sont positifs et négatifs. Dans le cas où $n < |\mathcal{K}|$, le système linéaire (2.5) n'est pas linéairement indépendants mais peut posséder une solution. En somme, si le système linéaire (2.5) possède une solution, le temps de résolution du DL est égal au temps de résolution du système linéaire (2.5) qui est $\mathcal{O}(|\mathcal{K}|^2 n)$.

□

Lorsque $|\mathcal{K}| \leq 2$, le système (2.5) est linéairement indépendant et le problème se résout en $\mathcal{O}(n)$. Cependant, lorsque $|\mathcal{K}| \geq 3$ et $n \geq |\mathcal{K}|$ ce système n'est pas toujours linéairement indépendant. Nous présentons à la figure 2.5.1 ci-dessous la proportion des systèmes linéaires (2.5) de rang plein, le rang du système linéaire (2.5) et le nombre d'arcs distincts n en fonction du nombre de produits, pour une moyenne de 10 instances des quatre familles de réseaux Didi, Delaunay, Voronoï et Grille. Ces familles de réseaux sont présentés à la section 2.6.1.

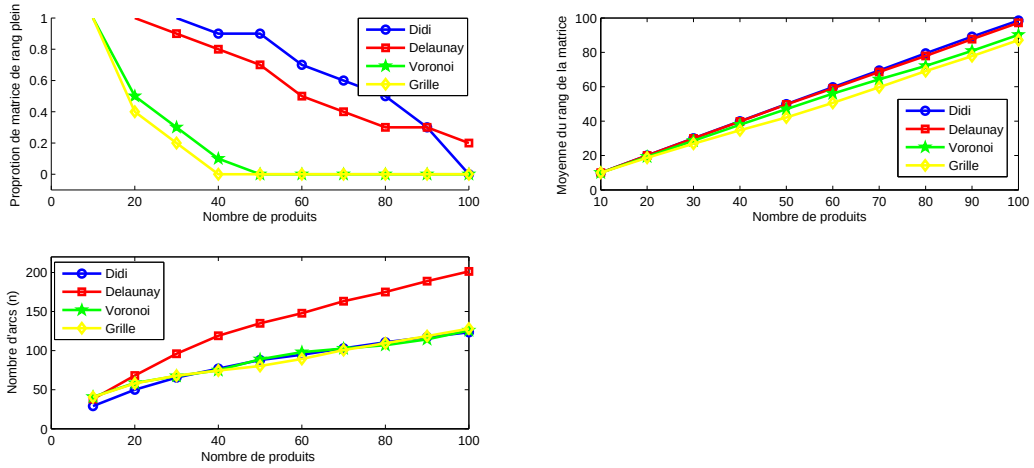


Figure 2.5 proportion des systèmes (2.5) de plein rang, le rang du système (2.5) et le nombre d'arcs distincts n en fonction du nombre de produits $|\mathcal{K}|$

Nous remarquons que la proportion des systèmes linéaires de rang plein décroît avec le nombre

de produits et que le rang du système linéaire est en moyenne égal à $|\mathcal{K}| - 1$ ou $|\mathcal{K}| - 2$. Nous déduisons donc que le système linéaire (2.5) n'admet pas toujours de solution. Nous conjecturons que ce problème est NP-difficile lorsque le nombre de produits est supérieur à 3 ($|\mathcal{K}| \geq 3$).

En restreignant les tarifs sur les arcs à ne prendre que des valeurs positives, on montre que :

Proposition 2.5.2 *Le DL monoproduit où tous les arcs du réseau sont taxables et les tarifs positifs se résout en* $\mathcal{O}(n)$.

Preuve L'idée de la preuve est similaire à celle utilisée pour la proposition 2.5.1. La solution optimale est obtenue en imposant les tarifs sur les arcs de la façon suivante

$$\sum_{a \in p_0} t_a = \min \left\{ \gamma(\infty) - \gamma(0), \frac{1}{2b} \left(a - b \sum_{a \in p_0} c_a \right) \right\}$$

$$t_a \; = \; +\infty \qquad\qquad \forall \; a \notin p_0$$

Le revenu généré est maximal avec cette affectation car la borne supérieure sur le revenu du meneur proposée à la section 2.4 est atteinte.

□

Dans le cas de deux produits, ce résultat n'est plus valide. L'exemple suivant montre qu'à l'optimalité les chemins p_0^k ne sont pas forcément utilisés. Pour l'illustrer, considérons le réseau de la figure 2.6 où tous les arcs du réseau sont taxables.

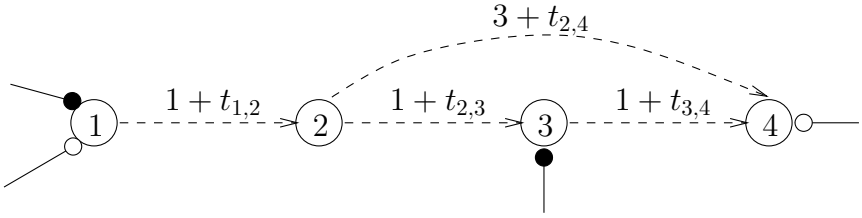


Figure 2.6 Réseau où tous les arcs sont taxables

Ce réseau est soumis à la demande de deux produits. Les informations sur les produits sont

données au tableau 2.5.

	o - d	demande	chemins	coût
produit 1	$1 \longrightarrow 3$	$d_1(U_1) = 22 - U_1$	$1 \to 2 \to 3$	$2 + t_{1,2} + t_{2,3}$
produit 2	$1 \longrightarrow 4$	$d_2(U_2) = 26 - 2U_2$	$1 \to 2 \to 3 \to 4$ $1 \to 2 \to 4$	$3 + t_{1,2} + t_{2,3} + t_{3,4}$ $4 + t_{1,2} + t_{2,4}$

Tableau 2.5 Données du réseau de la figure 2.6

Le but est de fixer les tarifs positifs sur les arcs du réseau afin de maximiser le revenu du meneur.

Les chemins p_0^1 et p_0^2 sont définis par $p_0^1 = 1 \to 2 \to 3$ et $p_0^2 = 1 \to 2 \to 3 \to 4$. Si ces chemins sont utilisés, le revenu généré est $400/3 = 133.33$ et est obtenu en fixant $t_{1,2} = 20/3$, $t_{2,3} = t_{3,4} = 0$ et $t_{2,4} = +\infty$. Cette solution n'est pas optimale car le chemin $1 \to 2 \to 3$ du produit 1 est un sous-chemin du chemin $1 \to 2 \to 3 \to 4$ du produit 2, ce qui empêche le meneur de générer un revenu maximal pour chacun des produits car le tarif optimal pour le chemin p_0^1 est 10 et 5 pour le chemin p_0^2. Cependant, la solution optimale est obtenue en utilisant les chemins $1 \to 2 \to 3$ pour le produit 1 et $1 \to 2 \to 4$ pour le produit 2. Le revenu généré est $281/2 = 140.5$ et est obtenu en fixant $t_{1,2} = 0$, $t_{2,3} = 10$, $t_{2,4} = 9/2$ et $t_{3,4} = +\infty$. Par contre, si les tarifs étaient réels, la solution optimale serait obtenue en utilisant les chemins $1 \to 2 \to 3$ pour le produit 1 et $1 \to 2 \to 3 \to 4$ pour le produit 2. Les tarifs optimaux seraient obtenus en fixant $t_{1,2} = 10$, $t_{2,3} = 0$, $t_{3,4} = -5$ et $t_{2,4} = +\infty$ pour un revenu de 150 (le problème est facile car $|\mathcal{K}| = 2$ et $n = 3$, voir proposition 2.5.1).

2.5.2 DL avec un seul arc taxable

On montre que le DL avec un seul arc taxable est polynômial. Ce résultat est une extension de celui présenté par Labbé *et al.* [46] pour le PTR.

Un seul produit

Le réseau se restreint à deux chemins allant de o à d. Le premier chemin p_0 est le plus court chemin obtenu en fixant le tarif de l'arc taxable à 0 et le second chemin est le plus court chemin obtenu en interdisant l'utilisation de l'arc taxable (chemin de la compétition). Soit t le tarif de l'arc taxable, $M = \gamma(\infty) - \gamma(0)$ est la borne supérieure sur t. Le tarif t est non négatif car le réseau possède un seul produit.

Tant que le tarif t de l'arc taxable reste inférieur ou égal à M, le chemin p_0 est utilisé. De ce fait, la solution optimale est obtenue en résolvant le problème suivant :

$$\max_{t} \quad \left(a - b\sum_{a\in p_0} c_a\right) t - bt^2 \tag{2.7}$$

$$0 \leq \ t \leq M \tag{2.8}$$

La fonction objectif (2.7) maximise le revenu qui correspond à celui généré lorsque le chemin p_0 est utilisé. La contrainte (2.8) impose que le chemin p_0 soit utilisé.

Le problème ci-dessus est un problème de maximisation d'une fonction quadratique concave avec une contrainte de borne. Il se résout facilement et la solution analytique est donnée par :

$$\begin{cases} t = 0 & \text{si } a - b\sum_{a\in p_0} c_a \leq 0 \\ t = \dfrac{1}{2b}\left(a - b\sum_{a\in p_0} c_a\right) & \text{sinon si } \dfrac{1}{2b}\left(a - b\sum_{a\in p_0} c_a\right) \leq M \\ t = M & \text{sinon.} \end{cases}$$

$\mathcal{K}$ produits

L'idée utilisée pour développer la méthode de résolution est de faire passer le maximum de produits par l'arc taxable afin de générer un revenu maximal. En fait, pour chaque produit k, le réseau se restreint à deux chemins. Le chemin p_0^k et celui de la compétition. Le chemin p_0^k est utilisé lorsque le tarif t est inférieur ou égal à $M_k = \gamma^k(\infty) - \gamma^k(0)$. Supposons que les

produits sont classés par ordre croissant de leurs bornes supérieures sur t, c'est-à-dire

$$M_1 \leq M_2 \leq M_3 \leq \ldots \leq M_{\mathcal{K}}.$$

De façon similaire au cas monoproduit, on propose l'algorithme suivant pour la résolution du problème.

Étape 1 $Z^* = -\infty$ et $t^* = +\infty$

Étape 2 Pour $k = 1, \ldots, \mathcal{K}$

$$\text{résoudre Z} = \begin{cases} \max\limits_{t} & \sum\limits_{j \geq k}^{\mathcal{K}} \left(a_j - b_j \sum\limits_{a \in p_0^j} c_a \right) t - b_j t^2 \\ & 0 \leq t \leq M_k \end{cases}$$

si $Z > Z^*$ alors $Z^* = Z$ et $t^* = arg \max\limits_{t} Z$

Étape 3 Retourner les valeurs de t^* et Z^*.

À l'itération k de l'étape 2, on maximise la somme des revenus générés par les produits tels que le chemin utilisé contient l'arc taxable. Plus précisément, lorsque $t \leq M_k$, les chemins p_0^j pour $j = k, \ldots, \mathcal{K}$ sont utilisés car $M_k \leq M_j$. Cet algorithme, qui effectue $|\mathcal{K}|$ itérations et résout à chaque itération un problème linéaire, est de complexité polynômiale.

2.5.3 Le problème d'optimisation inverse

Le problème d'optimisation inverse (POI) consiste à déterminer les tarifs maximisant le revenu du meneur, compatibles avec les chemins utilisés par les usagers. Nous montrons que dans le cas multi-produit, le POI se réduit à la maximisation d'une fonction objectif quadratique concave sous contraintes linéaires et dans le cas monoproduit, il se réduit à la recherche d'un plus court chemin sur un réseau.

Considérons la formulation par arcs DL:ARC et supposons que la solution du problème du second niveau est connue. Puisque les flots sont fixés, les contraintes primales du problème

du second niveau sont satisfaites et les contraintes qui imposent l'égalité des fonctions objectif primale et duale associées au problème du second niveau peuvent être simplifiées. Nous obtenons alors la formulation du POI suivante

$$\max_{t,\lambda,T} \sum_{k\in\mathcal{K}} \left(a_k T_k - b_k T_k \sum_{a\in\mathcal{A}|x_a^k=1} c_a - b_k T_k^2 \right) \tag{2.9}$$

$$\text{s.c} \quad \lambda_j^k - \lambda_i^k \leq c_{ij} + t_{ij} \qquad \forall (i,j) \in \mathcal{A}_1 | x_{ij}^k = 0 \tag{2.10}$$

$$\lambda_j^k - \lambda_i^k \leq c_{ij} \qquad \forall (i,j) \in \mathcal{A}_2 | x_{ij}^k = 0 \tag{2.11}$$

$$\lambda_j^k - \lambda_i^k = c_{ij} + t_{ij} \qquad \forall (i,j) \in \mathcal{A}_1 | x_{ij}^k = 1 \tag{2.12}$$

$$\lambda_j^k - \lambda_i^k = c_{ij} \qquad \forall (i,j) \in \mathcal{A}_2 | x_{ij}^k = 1 \tag{2.13}$$

$$T_k = \sum_{(i,j)\in\mathcal{A}_1|x_{ij}^k=1} t_{ij} \qquad \forall k \in \mathcal{K}. \tag{2.14}$$

La fonction objectif (2.9) maximise le revenu du meneur. Les contraintes (2.10) et (2.11) sont les contraintes d'admissibilité duale du problème du second niveau pour les arcs qui ne portent pas du flot. Les contraintes (2.12) et (2.13) sont les contraintes d'admissibilité duale du problème du second niveau pour les arcs qui portent du flot. Ces contraintes sont mises à l'égalité afin de respecter la complémentarité entre la variable de flot et la contrainte duale associée à cette variable. Les contraintes (2.14) donnent le revenu généré par unité de flot pour chaque produit.

Nous remplaçons dans les contraintes (2.14) les variables t_{ij} par $\lambda_j^k - \lambda_i^k - c_{ij}$ pour tout

arc $(i,j) \in \mathcal{A}_1$ et pour tout produit k tel que $x_{ij}^k = 1$. Nous obtenons alors

$$\begin{aligned}
T_k &= \sum_{(i,j)\in\mathcal{A}_1|x_{ij}^k=1} (\lambda_j^k - \lambda_i^k - c_{ij}) && \forall k \in \mathcal{K} \\
&= \sum_{(i,j)\in\mathcal{A}|x_{ij}^k=1} (\lambda_j^k - \lambda_i^k) - \sum_{(i,j)\in\mathcal{A}_2|x_{ij}^k=1} (\lambda_j^k - \lambda_i^k) - \sum_{(i,j)\in\mathcal{A}_1|x_{ij}^k=1} c_{ij} && \forall k \in \mathcal{K} \\
&= \sum_{(i,j)\in\mathcal{A}|x_{ij}^k=1} (\lambda_j^k - \lambda_i^k) - \sum_{(i,j)\in\mathcal{A}_2|x_{ij}^k=1} c_{ij} - \sum_{(i,j)\in\mathcal{A}_1|x_{ij}^k=1} c_{ij} && \forall k \in \mathcal{K} \\
&= \sum_{(i,j)\in\mathcal{A}|x_{ij}^k=1} (\lambda_j^k - \lambda_i^k) - \sum_{(i,j)\in\mathcal{A}|x_{ij}^k=1} c_{ij} && \forall k \in \mathcal{K} \\
&= \lambda_{d(k)} - \lambda_{o(k)} - \sum_{(i,j)\in\mathcal{A}|x_{ij}^k=1} c_{ij} && \forall k \in \mathcal{K}.
\end{aligned}$$

$\lambda_{d(k)} - \lambda_{o(k)}$ représente le coût total du chemin utilisé pour le produit k, et T_k la différence entre ce coût total et le coût fixe du chemin utilisé pour le produit k. Nous remplaçons les variables T_k par $\lambda_{d(k)} - \lambda_{o(k)} - \sum_{(i,j)\in\mathcal{A}|x_{ij}^k=1} c_{ij}$ dans la fonction objectif (2.9) du produit k.

$$\begin{aligned}
&\left(a_k - b_k \sum_{a\in\mathcal{A}|x_a^k=1} c_a\right) T_k - b_k T_k^2 \\
&= \left(a_k - b_k \sum_{a\in\mathcal{A}|x_a^k=1} c_a\right) \left(\lambda_{d(k)} - \lambda_{o(k)} - \sum_{(i,j)\in\mathcal{A}|x_a^k=1} c_a\right) - b_k \left(\lambda_{d(k)} - \lambda_{o(k)} - \sum_{(i,j)\in\mathcal{A}|x_a^k=1} c_a\right)^2 \\
&= \left(a_k - b_k \sum_{a\in\mathcal{A}|x_a^k=1} c_a\right) (\lambda_{d(k)} - \lambda_{o(k)}) - \left(a_k - b_k \sum_{a\in\mathcal{A}|x_a^k=1} c_a\right) \sum_{a\in\mathcal{A}|x_a^k=1} c_a - b_k(\lambda_{d(k)} - \lambda_{o(k)})^2 \\
&\qquad + 2b_k(\lambda_{d(k)} - \lambda_{o(k)}) \sum_{a\in\mathcal{A}|x_a^k=1} c_a + b_k \left(\sum_{a\in\mathcal{A}|x_a^k=1} c_a\right)^2 \\
&= -b_k(\lambda_{d(k)}^k - \lambda_{o(k)}^k)^2 + \left(a_k + b_k \sum_{a\in\mathcal{A}|x_a^k=1} c_a\right) (\lambda_{d(k)}^k - \lambda_{o(k)}^k) - a_k \sum_{a\in\mathcal{A}|x_a^k=1} c_a
\end{aligned}$$

La formulation du POI se réduit à :

OPTINV

$$\max_{t,\lambda} \sum_{k\in\mathcal{K}} \left(-b_k(\lambda^k_{d(k)} - \lambda^k_{o(k)})^2 + \left(a_k + b_k \sum_{a\in\mathcal{A}|x^k_a=1} c_a \right) (\lambda^k_{d(k)} - \lambda^k_{o(k)}) - a_k \sum_{a\in\mathcal{A}|x^k_a=1} c_a \right)$$

sous contraintes (2.10) à (2.13).

Nous obtenons donc un problème de maximisation d'une fonction objectif quadratique concave sous contraintes linéaires. Ce problème se résout de manière similaire d'un programme linéaire. Ainsi, OPTINV appartient à la classe des problèmes polynômiaux car .

Le POI monoproduit

Dans le cas d'un seul produit, la formulation du OPTINV associée est

$$\max_{t,\lambda} -b(\lambda_d - \lambda_o)^2 + (a + b \sum_{a\in\mathcal{A}|x_a=1} c_a)(\lambda_d - \lambda_o) - a \sum_{a\in\mathcal{A}|x_a=1} c_a \tag{2.15}$$

$$\lambda_j - \lambda_i \leq c_{ij} + t_{ij} \qquad \forall (i,j) \in \mathcal{A}_1 | x_{ij} = 0 \tag{2.16}$$

$$\lambda_j - \lambda_i \leq c_{ij} \qquad \forall (i,j) \in \mathcal{A}_2 | x_{ij} = 0 \tag{2.17}$$

$$\lambda_j - \lambda_i = c_{ij} + t_{ij} \qquad \forall (i,j) \in \mathcal{A}_1 | x_{ij} = 1 \tag{2.18}$$

$$\lambda_j - \lambda_i = c_{ij} \qquad \forall (i,j) \in \mathcal{A}_2 | x_{ij} = 1 \tag{2.19}$$

En remplaçant t_{ij} par $\lambda_j - \lambda_i - c_{ij}$ pour tout arc $(i,j) \in \mathcal{A}_1$ tel que $x_{ij} = 1$ et en posant $t_{ij} = +\infty$ pour tout arc $(i,j) \in \mathcal{A}$ tel que $x_{ij} = 0$, les contraintes (2.16) et (2.18) sont

satisfaites et peuvent être éliminées. La formulation du POI monoproduit se réduit à

$$\max_{\lambda} -b(\lambda_d - \lambda_o)^2 + (a + b \sum_{a \in \mathcal{A}|x_a=1} c_a)(\lambda_d - \lambda_o) - a \sum_{a \in \mathcal{A}|x_a=1} c_a \tag{2.20}$$

$$\begin{aligned} \text{s.c} \quad & \lambda_j - \lambda_i \leq c_{ij} && \forall (i,j) \in \mathcal{A}_2 \\ & \lambda_i - \lambda_j \leq -c_{ij} && \forall (i,j) \in \mathcal{A}_2 | x_{ij} = 1. \end{aligned}$$

La fonction objectif (2.20) est quadratique et concave.

Sans perte de généralité, posons $\lambda_o = 0$. On a alors que λ_i représente le coût du plus court chemin de l'origine o à i. La fonction objectif (2.20) atteint sa valeur maximale en $\lambda_d - \lambda_o = \frac{1}{2b}\left(a + b \sum_{a \in \mathcal{A}|x_a=1} c_a\right)$, et est croissante en λ_d sur l'intervalle $\left]-\infty,\ \frac{1}{2b}\left(a + b \sum_{a \in \mathcal{A}|x_a=1} c_a\right)\right]$. Nous déduisons que la solution optimale vérifie implicitement l'inégalité

$$\lambda_d \leq \frac{1}{2b}\left(a + b \sum_{a \in \mathcal{A}|x_a=1} c_a\right).$$

La formulation du POI présentée ci-dessus est donc équivalente à la formulation suivante.

OPTINV1

$$\max_{\lambda} \lambda_d - \lambda_o$$

$$\begin{aligned} \text{s.c} \quad & \lambda_j - \lambda_i \leq c_{ij} && \forall (i,j) \in \mathcal{A}_2 \\ & \lambda_i - \lambda_j \leq -c_{ij} && \forall (i,j) \in \mathcal{A}_2 | x_{ij} = 1 \\ & \lambda_d - \lambda_o \leq \frac{1}{2b}\left(a + b \sum_{a \in \mathcal{A}|x_a=1} c_a\right) \\ & \lambda_o = 0. \end{aligned}$$

Tel que nous le précisons dans la proposition suivante, la résolution de OPTINV1 consiste à rechercher un plus court chemin de o à d sur un réseau dûment défini. Ce résultat est

également une extension de celui présenté par Labbé *et al.* [46] pour le PTR. La particularité de ce résultat, par rapport à celui présenté par Labbé *et al.* [46], est l'ajout de l'arc (o, d) de coût $\frac{1}{2b}\left(a + b\sum_{a\in\mathcal{A}|x_a=1} c_a\right)$. Cet ajout est dû à la contrainte suivante

$$\lambda_d - \lambda_o \leq \frac{1}{2b}\left(a + b\sum_{a\in\mathcal{A}|x_a=1} c_a\right)$$

qui permet aux tarifs des arcs utilisés de ne pas excéder la valeur qui génère le revenu maximal.

Proposition 2.5.3 *Soient $\mathcal{G} = (N, \mathcal{A} = \mathcal{A}_1 \cup \mathcal{A}_2, c)$ un réseau orienté et p^* un chemin élémentaire de o à d. Définissons le graphe $\mathcal{G}'$ où $\mathcal{A}' = \mathcal{A}_2 \cup \{(i,j)|(j,i) \in \mathcal{A}_2 \cap p^*\} \cup \{(o,d)\}$. Le coût des arcs $(i,j) \in \mathcal{A}'$ est défini par $c'_{ij} = c_{ij}$ si $(i,j) \in \mathcal{A}_2$, $c'_{ij} = -c_{ij}$ si $(j,i) \in \mathcal{A}_2 \cap p^*$ et $c'_{od} = \frac{1}{2b}\left(a + b\sum_{a\in\mathcal{A}|x_a=1} c_a\right)$. Alors les tarifs optimaux sont donnés par $t_{ij} = \lambda_j - \lambda_i - c_{ij}$ où pour tout sommet s, λ_s est le coût du plus court chemin de o à s dans le graphe G'. S'il n'existe pas de plus court chemin de o à d dans G' alors le problème d'optimisation inverse monoproduit n'admet pas de solution.*

Cette proposition est illustrée sur l'exemple de la figure 2.7 (a) composé d'un produit d'origine 1 et de destination 5, d'une demande $d(U) = 28 - 2U$ et pour laquelle la solution du problème du second niveau est le chemin $1 \to 2 \to 3 \to 5$. Le réseau $\mathcal{G}'$ résultant est représenté à la figure 2.7 (b) ci-dessous.

Les valeurs des variables duales λ correspondantes aux plus courts chemins de 1 vers les autres sommets dans $\mathcal{G}'$ sont : $\lambda_1 = 0$, $\lambda_2 = 2$, $\lambda_3 = 7$, $\lambda_4 = +\infty$ et $\lambda_5 = 11$. Nous obtenons ainsi les tarifs $t_{2,4} = +\infty$ et $t_{2,3} = 3$. En fait, le plus court chemin de 1 à 5 est l'arc $(1,5)$. Cet arc $(1,5)$ de coût 11 qui a été défini dans le graphe $\mathcal{G}'$ permet de contrôler les tarifs de telle sorte qu'ils n'excèdent pas la valeur qui génère un revenu maximal pour le meneur. Si cet arc avait été ignoré alors les tarifs sur les arcs auraient été $t_{2,4} = +\infty$ et $t_{2,3} = 4$ pour un revenu de 16 au lieu de 18 lorsqu'il est pris en compte.

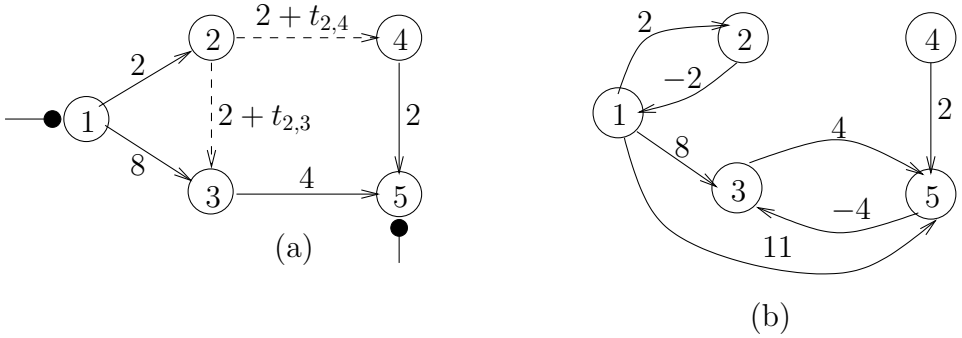


Figure 2.7 Exemple de transformation du réseau pour la résolution du POI monoproduit

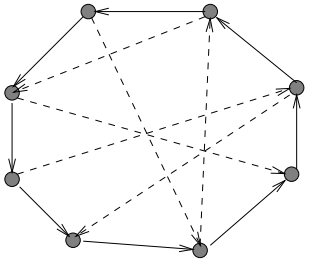
2.6 Expérimentations

Les expérimentations effectuées permettent d'évaluer de façon quantitative l'efficacité des formulations proposées. Nous présentons d'abord les paramètres de modélisation ayant servi aux tests numériques (section 2.6.1). À la section 2.6.2 nous présentons le plan d'expérience. Finalement, la section 2.6.3 est consacrée aux résultats des tests numériques.

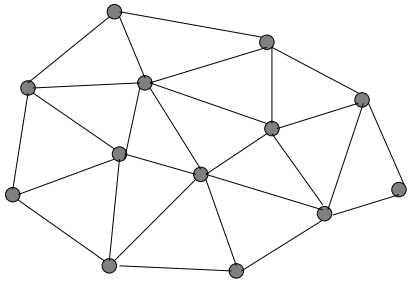
2.6.1 Paramètres de modélisation

Pour réaliser les diverses évaluations désirées, nous supposons connu un graphe $\mathcal{G} = (\mathcal{N}, \mathcal{A})$. À partir de ce graphe, une instance du problème est obtenue en générant un ensemble $\mathcal{A}_1$ d'arcs taxables, un ensemble $\mathcal{K}$ de produits et un ensemble $d_k(U_k)_{k\in\mathcal{K}}$ de fonctions de demande associées aux produits. Les différentes stratégies utilisées pour générer le graphe, l'ensemble d'arcs taxables et l'ensemble de produits sont les stratégies utilisées par Cirinei [17] que nous présentons ci-dessous. Nous avons généré les fonctions de demande associées aux produits en nous inspirant des procédures utilisées dans la littérature (Hearn *et al.* [40], Yang *et al.* [70] et Lei *et al.* [48] et autres).

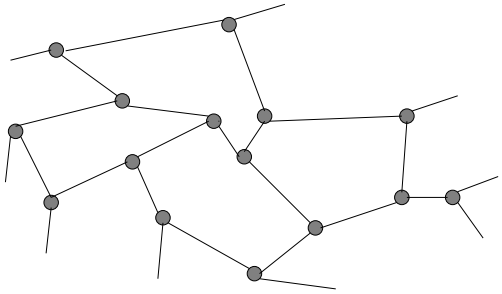
Les instances sont regroupées en familles où chaque famille est associée à une structure particulière du graphe $\mathcal{G}$. Les différentes familles d'instances utilisées sont :



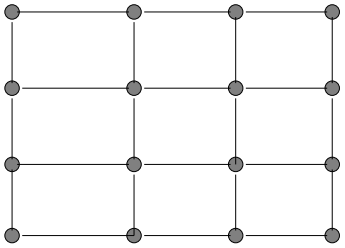
1) Structure de Didi, Marcotte et Savard (DMS)



2) Structure de Delaunay



3) Structure de Voronoï



4) Structure de Grille (4×4)

Figure 2.8 Structure de graphes des différentes familles d'instances

Réseaux DMS, Didi *et al.* [29] : Ils sont basés sur des graphes $\mathcal{G}$ de 60 nœuds et 200 arcs. Ils sont construits en créant un cycle reliant chaque nœud du graphe afin d'assurer la connexité du réseau. Puis, des arcs sont ajoutés aléatoirement au réseau. La figure 2.8 (1) présente un exemple où les arcs ajoutés sont les arcs en pointillés qui se trouvent à l'intérieur du cercle.

Réseaux Delaunay : Ils sont basés sur la triangulation de Delaunay. Le choix de cette structure est motivé par le fait qu'elle est très souvent utilisée pour modéliser les réseaux de télécommunications (voir Zuyev *et al.* [72]). À partir de la triangulation d'un ensemble de 60 points répartis uniformément, le graphe est obtenu en considérant les arêtes des triangles comme des arcs à double sens. La moyenne du nombre d'arcs est 330. Un exemple est présenté à la figure 2.8 (2).

Réseaux Voronoï : Ils sont basés sur les diagrammes de Voronoï. La particularité de ces graphes est que tous les nœuds sont de degré 3. En regardant une carte routière, nous constatons que les points d'intersections des grands axes routiers sont souvent de degré 3.

Cette observation a motivé le choix de ces graphes. Le diagramme de Voronoï est le dual d'une triangulation de Delaunay. Plus précisément, les nœuds de ces diagrammes sont les centres des cercles circonscrits des triangles appartenant à la triangulation de Delaunay et les arcs à double sens sont obtenus en reliant les centres de cercles circonscrits de 2 triangles adjacents. Ces graphes sont générés de telle sorte qu'ils possèdent en moyenne 60 nœuds. Pour cela, le nombre d'arcs est en moyenne 150. Le diagramme de Voronoï associé à la triangulation de la figure 2.8 (2) est présenté à la figure 2.8 (3).

Grilles : Ils correspondent à de grilles de dimensions (12×5) comportant 60 nœuds et 206 arcs, du même type que celles représentées à la figure 2.8 (4). La ligne reliant deux nœuds sur la figure est équivalente à la représentation des arcs à double sens. De tels réseaux ont été utilisés par Brotcorne *et al.* [13].

Sélection de produits

La procédure de sélection des produits que nous utilisons a été proposée par Cirinei [17]. La stratégie qu'il propose est intermédiaire entre les stratégies proposées par Didi *et al.* [29] ; Grigoriev *et al.* [36] et Brotcorne *et al.* [13]. Celle-ci suppose que les coordonnées de chaque nœud du graphe sont connues. Dans la première phase, on détermine les nœuds définissant l'enveloppe convexe de l'ensemble des nœuds $\mathcal{N}$. À partir de ces nœuds, on choisit 25% des produits en sélectionnant la paire de nœuds (i, j) qui maximise la distance qui les sépare et on crée le nouveau produit ayant pour origine i et de destination j. Ensuite, le choix des 75% des produits repose sur un choix totalement aléatoire de l'origine et de la destination. Ce choix aléatoire a été proposé par Didi *et al.* [29] et Grigoriev *et al.* [36].

Sélection des arcs taxables

La procédure utilisée a été proposée par Brotcorne *et al.* [13]. Son principe est de choisir des arcs susceptibles d'être utilisés par plusieurs produits. Pour cela, la fréquence d'apparition des arcs dans les plus courts chemins de chaque produit est calculée. Plus la fréquence d'un

arc est élevée, plus son attractivité est grande par rapport aux produits. Puis, on impose que 66% des arcs taxables soient choisis parmi les arcs ayant une fréquence d'apparition élevée et que les 34% restants soient sélectionnés suivant la stratégie proposée par Didi *et al.* [29] et Grigoriev *et al.* [36] qui est une sélection aléatoire des arcs taxables. Finalement, le coût des arcs taxables est divisé par 2 par rapport aux couts initiaux afin de les rendre plus attractifs et renforcer l'interaction entre les produits.

Construction des fonctions de demande

La fonction de demande du produit k a la forme $d_k(U_k) = a_k - b_k U_k$ où b_k est choisi uniformément dans l'intervalle $[1, 5]$. Le coefficient a_k est choisi de façon aléatoire dans l'intervalle $[50 \times (b_k + 1) \times v,\ 50 \times (b_k + 2) \times v]$ où v est une valeur choisie entre la moyenne du coût des chemins du produit k et le coût du plus court chemin ne contenant aucun arc taxable (chemin de la compétition) du même produit. L'analyse de sensibilité sur l'élasticité de la demande est faite en variant la pente de la demande du produit k sur l'intervalle $[0, 100b_k]$. Le coefficient multiplicatif 50 des bornes de intervalle des valeurs de a_k permet d'avoir une demande non négative lorsque la pente s'approche de $100b_k$.

Génération des chemins

Pour générer l'ensemble des chemins, un prétraitement du réseau par produit développé par Bouhtou *et al.* [11] et renforcé par Dewez [27] a été effectué. Ce prétraitement utilise les règles de dominance sur les coûts des chemins pour éliminer les arcs inutiles de chaque produit. Il s'ensuit une réduction du nombre de chemins par produit. Étant donné que le nombre de chemins par produit peut être très élevé, nous générons par produit les 500 premiers plus courts chemins de coût inférieur au coût du chemin de la compétition. Une fois le problème résolu dans l'espace des chemins, nous revenons dans l'espace des arcs afin de vérifier que les chemins utilisés à l'optimalité sont bel et bien de plus petits coûts. Le code informatique qui génère les chemins a été développé par Cirinei [17] et utilise l'algorithme de Yen [71] pour

générer les k-plus courts chemins. Le tableau 2.6 ci-dessous présente la moyenne du total des chemins générés pour les formulations par chemins.

		tarifs positifs			tarifs non contraignants		
		%t			%t		
Type de réseau	#OD	10	15	20	10	15	20
Didi	20	70.10	91.50	140.00	118.30	131.80	389.70
	40	135.84	185.72	279.20	196.16	288.04	708.08
	60	204.12	280.56	427.32	283.74	472.80	1059.18
	80	270.08	380.72	574.16	368.56	733.52	1452.48
	100	329.70	468.00	725.20	439.50	885.60	1847.50
Delaunay	10	35.40	71.30	136.20	43.70	163.20	380.40
	20	96.70	186.80	322.00	140.60	414.70	837.00
	30	164.70	359.31	673.50	234.03	836.70	1761.30
	40	227.80	508.76	915.60	322.76	1232.40	2476.80
	50	292.90	647.10	1154.00	425.40	1594.00	3114.00
	60	365.16	836.52	1427.40	542.94	2078.40	3967.80
Voronoï	10	53.20	171.00	215.30	131.30	1074.40	1344.30
	20	106.80	317.00	406.90	238.60	2002.20	2715.40
	30	165.39	454.50	605.79	391.83	2781.60	4009.80
	40	216.64	588.92	807.60	489.20	3596.40	5312.00
	50	275.80	724.90	994.00	627.80	4399.00	6416.50
	60	333.90	841.80	1179.60	737.46	5079.60	7544.40
Grille	10	76.40	114.80	293.40	159.20	369.60	985.30
	20	164.80	295.40	652.40	322.00	933.20	2089.40
	30	248.22	567.90	1070.79	462.60	1725.60	3393.30
	40	337.84	799.20	1419.28	623.80	2343.20	4327.20
	50	411.60	960.00	1659.10	752.60	2738.50	5048.50
	60	472.56	1031.70	1811.52	843.48	2895.00	5440.20

Tableau 2.6 Nombre total de chemins générés pour les formulations par chemins pour les quatres familles de réseaux

Le nombre de chemins où les tarifs sont positifs est inférieur au nombre de chemins où les tarifs sont non contraignants car plusieurs propriétés permettant d'éliminer les chemins dominés lorsque les tarifs sont positifs ne sont pas applicables lorsque les tarifs sont non contraignants.

2.6.2 Plan d'expérience

Les expériences numériques sont réalisées sur les quatre familles d'instances présentées ci-dessus en considérant dans un premier temps les tarifs positifs, puis réels. Le nombre de produits varie entre 10 et 60 et la proportion d'arcs taxables est de 10%, 15% et 20%. Afin d'étudier l'effet de l'élasticité de la demande sur le DL, une analyse de sensibilité est faite sur

la pente de la fonction de demande. Nous multiplions la pente de chaque fonction de demande par 100, 1 et puis 0 dans le but d'avoir des fonctions de demande à élasticité forte, faible et le cas standard. Nous avons décidé de choisir trois valeurs pour la pente de la demande car le but principale des tests est d'évaluer l'efficacité des formulations proposées et non de faire une étude approfondie de l'impact de l'élasticité sur la résolution du problème.

Pour l'exécution des programmes quadratiques mixtes sous contraintes linéaires, nous avons utilisé un ordinateur muni d'un système d'exploitation GNU/LINUX openSUSE 11.0 possèdant un processeur Intel Core 2 cadencé à 2.13 GHz. Les programmes sont codés en C++ et font appel aux procédures de la librairie « ILOG CPLEX Callable Library » pour résoudre les programmes quadratiques mixtes sous contraintes linéaires en utilisant CPLEX 10.1.1. Nous gardons les paramètres par défaut de CPLEX, sauf le temps d'exécution maximal qui est fixé à 5 heures (18000 secondes) et le MIP gap à $1e$-20. Les résultats numériques ci-dessous présentent la moyenne sur 10 instances.

2.6.3 Résultats numériques

Nous commençons par présenter les résultats numériques obtenus lorsque les tarifs sont positifs. Pour ce cas, une analyse de sensibilité de l'élasticité de la demande est effectuée afin de mesurer son impact sur la résolution du DL. Pour ce faire, nous considérons le DL avec des fonctions de demande à élasticité forte, puis faible. Ensuite, la formulation DL:CH2 est utilisée pour résoudre le DL avec des fonctions de demande à élasticité nulle (demande inélastique) car nous avons remarqué qu'avec une élasticité non nulle cette formulation est la meilleure.

La plupart des résultats numériques sont présentés sous la forme suivante.

Instance		DL:ARC				DL:CH1				DL:CH2			
#OD	%t	Cpu	Gap	#it	Nopt	Cpu	Gap	#it	Nopt	Cpu	Gap	#it	Nopt

Tableau 2.7 Format de tableau pour la présentation des résultats du DL

On y présente la formulation par arcs (DL:ARC), la première formulation par chemins

(DL:CH1), la deuxième formulation par chemins (DL:CH2), le nombre de produits (#OD), le pourcentage d'arcs taxables (%t), le temps de calcul en secondes (Cpu), la qualité de la solution (Gap) définie par l'écart (%) entre la borne supérieure (Z_{sup}) et la meilleure solution trouvée (Z_{inf}) de la façon suivante : $((Z_{sup} - Z_{inf})/Z_{inf}) \times 100$, le nombre d'itérations (#it) et le nombre d'instances n'ayant pas pu être résolues à l'optimalité avant le temps d'exécution maximal (Nopt). Les résultats numériques de la résolution du DL par les trois formulations développées sont reportés aux tableaux 2.8, 2.9, 2.10 et 2.11.

Élasticité forte $(100 \times b_k)$

Instance		DL:ARC				DL:CH1				DL:CH2			
#OD	%t	Cpu	Gap	#it	Nopt	Cpu	Gap	#it	Nopt	Cpu	Gap	#it	Nopt
20	10	448.41	0.00	3.76e+06	0	16.29	0.00	2.35e+05	0	0.08	0.00	6.54e+02	0
20	15	6523.60	9.41	4.80e+07	3	827.42	0.00	8.61e+06	0	0.11	0.00	8.35e+02	0
20	20	15004.87	252.30	9.06e+07	8	11423.44	98.24	1.13e+08	6	0.20	0.00	1.21e+03	0
40	10					4600.52	4.73	2.67e+07	2	1.07	0.00	8.86e+03	0
40	15					16314.00	181.66	1.54e+08	9	13.66	0.00	1.05e+05	0
40	20					18054.62	347.31	1.08e+08	10	78.90	0.00	4.89e+05	0
60	10									28.62	0.00	2.21e+05	0
60	15									198.27	0.00	1.02e+06	0
60	20									3205.79	0.00	1.23e+07	0
80	10									2328.81	0.00	9.35e+06	1
80	15									7742.98	2.06	3.29e+07	2
80	20									15257.66	3.73	5.65e+07	8
100	10									6104.80	2.28	1.92e+07	3
100	15									12296.44	11.67	2.34e+07	6
100	20									17319.16	15.19	2.28e+07	9

Élasticité faible $(1 \times b_k)$

Instance		DL:ARC				DL:CH1				DL:CH2			
#OD	%t	Cpu	Gap	#it	Nopt	Cpu	Gap	#it	Nopt	Cpu	Gap	#it	Nopt
20	10	0.24	0.00	1.97e+03	0	0.09	0.00	7.68e+02	0	0.06	0.00	5.37e+02	0
20	15	0.61	0.00	3.90e+03	0	0.20	0.00	1.28e+03	0	0.11	0.00	8.50e+02	0
20	20	2.96	0.00	8.31e+03	0	0.37	0.00	2.00e+03	0	0.19	0.00	1.32e+03	0
40	10	3.43	0.00	2.19e+04	0	1.16	0.00	7.80e+03	0	0.69	0.00	6.33e+03	0
40	15	7.60	0.00	2.86e+04	0	1.48	0.00	6.83e+03	0	0.83	0.00	5.72e+03	0
40	20	53.19	0.00	1.62e+05	0	6.70	0.00	3.04e+04	0	4.82	0.00	3.68e+04	0
60	10	102.23	0.00	6.74e+05	0	4.29	0.00	3.33e+04	0	5.12	0.00	3.03e+04	0
60	15	109.41	0.00	4.57e+05	0	15.01	0.00	1.11e+05	0	19.42	0.00	1.26e+05	0
60	20	2827.43	0.00	7.34e+06	0	391.03	0.00	2.09e+06	0	340.08	0.00	1.64e+06	0
80	10	1945.96	0.00	1.16e+07	1	57.31	0.00	3.95e+05	0	93.01	0.00	5.06e+05	0
80	15	3976.82	0.72	1.11e+07	2	1890.82	0.04	1.15e+07	1	613.75	0.00	2.91e+06	0
80	20	12035.95	3.75	2.42e+07	6	6746.97	0.26	3.17e+07	2	7847.14	0.42	3.33e+07	3
100	10	2195.38	0.00	4.18e+06	1	380.90	0.00	1.95e+06	0	681.47	0.00	3.69e+06	0
100	15	5640.92	1.84	1.02e+07	2	2373.94	0.39	1.32e+07	1	2473.75	0.42	1.48e+07	1
100	20	16253.23	7.66	1.73e+07	9	14547.11	3.08	6.12e+07	6	12090.63	3.01	6.28e+07	6

Élasticité nulle $(0 \times b_k)$

Instance		DL:ARC				DL:CH1				DL:CH2			
#OD	%t	Cpu	Gap	#it	Nopt	Cpu	Gap	#it	Nopt	Cpu	Gap	#it	Nopt
20	10									0.02	0.00	1.23e+02	0
20	15									0.05	0.00	2.32e+02	0
20	20									0.09	0.00	2.58e+02	0
30	10									0.05	0.00	2.18e+02	0
30	15									0.10	0.00	3.66e+02	0
30	20									0.23	0.00	8.85e+02	0
40	10									0.17	0.00	1.53e+03	0
40	15									0.26	0.00	1.24e+03	0
40	20									0.89	0.00	5.49e+03	0
60	10									0.60	0.00	5.27e+03	0
60	15									2.30	0.00	2.25e+04	0
60	20									23.32	0.00	2.16e+05	0
80	10									5.43	0.00	5.57e+04	0
80	15									60.37	0.00	5.15e+05	0
80	20									657.88	0.00	4.37e+06	0
100	10									11.55	0.00	1.06e+05	0
100	15									1392.18	0.00	9.15e+06	0
100	20									7914.38	0.66	3.85e+07	2

Tableau 2.8 Réseaux **DMS** : résultats obtenus avec tarifs positifs et demande linéaire

Élasticité forte $(100 \times b_k)$

Instance		DL:ARC				DL:CH1				DL:CH2			
#OD	%t	Cpu	Gap	#it	Nopt	Cpu	Gap	#it	Nopt	Cpu	Gap	#it	Nopt
10	10	5.35	0.00	6.52e+04	0	1.02	0.00	1.71e+04	0	0.02	0.00	1.45e+02	0
10	15	4235.73	4.21	2.33e+07	2	677.07	0.00	6.39e+06	0	0.03	0.00	1.91e+02	0
10	20	16705.07	1406.54	1.04e+08	8	5777.22	985.96	4.47e+07	2	0.07	0.00	2.27e+02	0
20	10					4717.68	54.31	4.57e+07	2	0.12	0.00	8.32e+02	0
20	15					14499.95	948.47	1.26e+08	8	0.26	0.00	1.03e+03	0
20	20					18099.19	1969.88	1.09e+08	10	1.08	0.00	3.34e+03	0
30	10									7.33	0.00	5.78e+04	0
30	15									12.05	0.00	5.25e+04	0
30	20									99.70	0.00	2.39e+05	0
40	10									61.65	0.00	3.94e+05	0
40	15									813.36	0.00	2.24e+06	0
40	20									6394.36	0.70	1.06e+07	3
50	10									2253.68	0.64	1.11e+07	1
50	15									2569.80	0.34	5.02e+06	1
50	20									13648.51	2.55	1.86e+07	7
60	10									8874.88	0.81	2.97e+07	3
60	15									14315.71	5.41	9.85e+06	6
60	20									18076.28	10.92	5.09e+06	10

Élasticité faible $(1 \times b_k)$

Instance		DL:ARC				DL:CH1				DL:CH2			
#OD	%t	Cpu	Gap	#it	Nopt	Cpu	Gap	#it	Nopt	Cpu	Gap	#it	Nopt
10	10	0.05	0.00	7.24e+02	0	0.03	0.00	2.93e+02	0	0.02	0.00	1.88e+02	0
10	15	0.17	0.00	2.23e+03	0	0.05	0.00	5.66e+02	0	0.03	0.00	2.35e+02	0
10	20	0.58	0.00	4.46e+03	0	0.10	0.00	9.76e+02	0	0.05	0.00	2.84e+02	0
20	10	0.53	0.00	3.38e+03	0	0.12	0.00	9.31e+02	0	0.08	0.00	6.28e+02	0
20	15	1.41	0.00	6.45e+03	0	0.34	0.00	1.69e+03	0	0.16	0.00	7.91e+02	0
20	20	5.55	0.00	1.63e+04	0	1.24	0.00	4.13e+03	0	0.69	0.00	2.35e+03	0
30	10	1.96	0.00	8.53e+03	0	0.40	0.00	2.10e+03	0	0.20	0.00	1.26e+03	0
30	15	39.24	0.00	1.18e+05	0	7.93	0.00	2.59e+04	0	4.09	0.00	2.28e+04	0
30	20	468.93	0.00	1.18e+06	0	30.77	0.00	7.00e+04	0	22.16	0.00	6.62e+04	0
40	10	17.47	0.00	7.02e+04	0	2.63	0.00	1.52e+04	0	2.75	0.00	2.41e+04	0
40	15	125.55	0.00	2.78e+05	0	33.37	0.00	1.01e+05	0	19.75	0.00	1.02e+05	0
40	20	1486.54	0.00	2.06e+06	0	401.08	0.00	7.91e+05	0	282.51	0.00	8.26e+05	0
50	10	33.47	0.00	1.20e+05	0	5.62	0.00	2.80e+04	0	3.15	0.00	2.39e+04	0
50	15	290.75	0.00	6.26e+05	0	121.47	0.00	3.05e+05	0	78.69	0.00	3.26e+05	0
50	20	3544.13	0.25	4.70e+06	1	2007.91	0.09	3.35e+06	1	1971.28	0.05	4.88e+06	1
60	10	575.10	0.00	1.32e+06	0	62.34	0.00	3.77e+05	0	18.29	0.00	1.39e+05	0
60	15	7080.56	0.24	8.45e+06	2	2067.02	0.16	4.29e+06	1	2104.40	0.08	6.61e+06	1
60	20	12177.40	1.55	1.00e+07	4	6846.58	0.33	1.19e+07	3	5785.66	0.16	1.43e+07	2

Élasticité nulle $(0 \times b_k)$

Instance		DL:ARC				DL:CH1				DL:CH2			
#OD	%t	Cpu	Gap	#it	Nopt	Cpu	Gap	#it	Nopt	Cpu	Gap	#it	Nopt
10	10									0.01	0.00	3.77e+01	0
10	15									0.01	0.00	4.84e+01	0
10	20									0.03	0.00	7.61e+01	0
20	10									0.04	0.00	1.47e+02	0
20	15									0.08	0.00	1.76e+02	0
20	20									0.31	0.00	6.35e+02	0
30	10									0.09	0.00	3.04e+02	0
30	15									1.15	0.00	6.21e+03	0
30	20									6.78	0.00	1.19e+04	0
40	10									0.61	0.00	4.73e+03	0
40	15									10.71	0.00	4.88e+04	0
40	20									87.96	0.00	2.07e+05	0
50	10									0.77	0.00	4.90e+03	0
50	15									10.85	0.00	5.04e+04	0
50	20									1013.30	0.00	2.46e+06	0
60	10									5.19	0.00	3.29e+04	0
60	15									1876.68	0.02	3.95e+06	1
60	20									3163.25	0.12	4.73e+06	1

Tableau 2.9 Réseaux **Delaunay** : résultats obtenus avec tarifs positifs et demande linéaire

Élasticité forte $(100 \times b_k)$

Instance		DL:ARC				DL:CH1				DL:CH2			
#OD	%t	Cpu	Gap	#it	Nopt	Cpu	Gap	#it	Nopt	Cpu	Gap	#it	Nopt
10	10	117.33	0.00	1.01e+06	0	9.58	0.00	1.14e+05	0	0.10	0.00	6.95e+02	0
10	15	13189.59	175.74	8.12e+07	6	5365.70	19.93	2.43e+07	2	0.82	0.00	2.55e+03	0
10	20	15256.75	346.69	7.27e+07	8	9764.99	97.79	3.77e+07	5	3.77	0.00	1.56e+04	0
20	10					1443.45	0.00	8.38e+06	0	0.92	0.00	6.35e+03	0
20	15					16704.44	212.28	7.50e+07	9	271.76	0.00	6.33e+05	0
20	20					16289.45	310.85	4.51e+07	9	2030.95	0.00	4.39e+06	0
30	10					7055.79	62.54	4.22e+07	3	32.24	0.00	1.81e+05	0
30	15					18055.32	393.45	7.27e+07	10	5945.49	0.26	1.22e+07	2
30	20					18046.89	395.11	3.52e+07	10	10162.22	4.12	1.79e+07	5
40	10									55.52	0.00	2.88e+05	0
40	15									9671.55	1.98	1.75e+07	4
40	20									15693.91	5.57	2.39e+07	7
50	10									520.05	0.00	2.84e+06	0
50	15									14882.33	8.21	3.50e+07	8
50	20									18062.33	11.97	2.89e+07	10
60	10									6027.55	7.38	9.49e+06	2
60	15									18065.13	26.29	3.06e+06	10
60	20									18080.82	29.86	9.40e+05	10

Élasticité faible $(1 \times b_k)$

Instance		DL:ARC				DL:CH1				DL:CH2			
#OD	%t	Cpu	Gap	#it	Nopt	Cpu	Gap	#it	Nopt	Cpu	Gap	#it	Nopt
10	10	0.77	0.00	3.83e+03	0	0.12	0.00	9.96e+02	0	0.10	0.00	6.50e+02	0
10	15	11.42	0.00	4.34e+04	0	1.69	0.00	9.58e+03	0	1.44	0.00	1.08e+04	0
10	20	72.77	0.00	2.64e+05	0	3.49	0.00	1.04e+04	0	2.22	0.00	9.11e+03	0
20	10	5.11	0.00	2.44e+04	0	1.21	0.00	1.04e+04	0	0.52	0.00	6.25e+03	0
20	15	1343.97	0.00	4.80e+06	0	30.04	0.00	1.40e+05	0	29.60	0.00	2.44e+05	0
20	20	4815.32	0.94	1.11e+07	2	244.21	0.00	6.01e+05	0	182.40	0.00	8.19e+05	0
30	10	37.32	0.00	1.57e+05	0	9.76	0.00	6.66e+04	0	3.51	0.00	4.15e+04	0
30	15	5722.49	0.68	1.33e+07	2	2690.36	0.30	5.66e+06	1	1484.78	0.00	7.36e+06	0
30	20	10652.71	2.29	1.51e+07	5	1957.69	0.00	4.45e+06	0	3357.52	0.08	1.23e+07	1
40	10	93.53	0.00	4.06e+05	0	22.63	0.00	1.85e+05	0	23.63	0.00	2.07e+05	0
40	15	13846.55	2.48	3.33e+07	5	4417.11	1.24	1.22e+07	2	4076.51	0.85	1.48e+07	2
40	20	16376.51	6.40	2.37e+07	9	7399.04	1.78	1.75e+07	3	6947.90	1.28	2.15e+07	3
50	10	1343.05	0.00	5.24e+06	0	82.64	0.00	5.51e+05	0	89.29	0.00	7.01e+05	0
50	15	17339.53	39.98	3.27e+07	9	10274.06	3.37	3.39e+07	5	9511.06	2.71	4.37e+07	5
50	20	17887.68	27.49	1.85e+07	9	15159.44	7.12	3.01e+07	8	15427.59	7.08	3.95e+07	8
60	10	2785.72	0.53	9.91e+06	1	549.53	0.00	2.88e+06	0	410.96	0.00	2.20e+06	0
60	15	18010.50	96.49	2.59e+07	10	14949.50	7.97	2.76e+07	8	15832.44	5.16	4.65e+07	8
60	20	18009.61	320.66	1.56e+07	10	18048.52	52.21	1.27e+06	10	18047.73	11.13	3.13e+07	10

Élasticité nulle $(0 \times b_k)$

Instance		DL:ARC				DL:CH1				DL:CH2			
#OD	%t	Cpu	Gap	#it	Nopt	Cpu	Gap	#it	Nopt	Cpu	Gap	#it	Nopt
10	10									0.04	0.00	1.87e+02	0
10	15									0.32	0.00	1.34e+03	0
10	20									0.59	0.00	1.77e+03	0
20	10									0.18	0.00	1.29e+03	0
20	15									5.85	0.00	4.08e+04	0
20	20									61.10	0.00	1.99e+05	0
30	10									1.39	0.00	1.31e+04	0
30	15									172.79	0.00	6.14e+05	0
30	20									1139.76	0.00	2.95e+06	0
40	10									3.56	0.00	2.76e+04	0
40	15									3859.87	0.29	1.14e+07	2
40	20									5643.53	0.96	1.19e+07	3
50	10									14.94	0.00	1.14e+05	0
50	15									6471.32	1.89	1.77e+07	3
50	20									10789.97	4.58	2.72e+07	5
60	10									167.68	0.00	5.37e+05	0
60	15									10184.26	3.19	2.32e+07	5
60	20									16607.25	8.01	3.09e+07	9

Tableau 2.10 Réseaux **Voronoi** : résultats obtenus avec tarifs positifs et demande linéaire

Élasticité forte $(100 \times b_k)$

Instance		DL:ARC				DL:CH1				DL:CH2			
#OD	%t	Cpu	Gap	#it	Nopt	Cpu	Gap	#it	Nopt	Cpu	Gap	#it	Nopt
10	10	1594.52	0.00	1.45e+07	0	195.71	0.00	1.65e+06	0	0.11	0.00	4.31e+02	0
10	15	7994.64	370.13	5.88e+07	4	2845.84	0.00	1.85e+07	0	0.14	0.00	4.10e+02	0
10	20	18029.86	7268.18	9.85e+07	10	14135.28	3902.52	5.04e+07	7	4.02	0.00	8.11e+03	0
20	10					9323.31	1852.54	5.76e+07	5	9.36	0.00	3.89e+04	0
20	15					14521.14	9141.50	6.98e+07	8	21.43	0.00	5.93e+04	0
20	20					17990.05	3745.42	4.44e+07	9	857.70	0.00	9.81e+05	0
30	10									1855.59	5.51	4.44e+06	1
30	15									2415.78	0.71	2.57e+06	1
30	20									11021.73	6.54	9.22e+06	6
40	10									1973.96	57.82	2.59e+06	1
40	15									4431.54	9.33	3.88e+06	2
40	20									11349.62	34.50	5.66e+06	6
50	10									2096.99	34.70	4.23e+06	1
50	15									10244.79	267.87	1.60e+07	5
50	20									12865.00	34.20	8.14e+06	6
60	10									7520.63	116.83	8.88e+06	4
60	15									14124.76	613.81	6.89e+06	7
60	20									16416.66	115.78	1.64e+06	9

Élasticité faible $(1 \times b_k)$

Instance		DL:ARC				DL:CH1				DL:CH2			
#OD	%t	Cpu	Gap	#it	Nopt	Cpu	Gap	#it	Nopt	Cpu	Gap	#it	Nopt
10	10	0.88	0.00	6.40e+03	0	0.18	0.00	9.87e+02	0	0.08	0.00	4.54e+02	0
10	15	1.09	0.00	6.09e+03	0	0.23	0.00	1.45e+03	0	0.11	0.00	7.48e+02	0
10	20	4.45	0.00	1.76e+04	0	1.25	0.00	2.78e+03	0	0.69	0.00	2.14e+03	0
20	10	11.54	0.00	6.39e+04	0	1.38	0.00	7.41e+03	0	0.85	0.00	7.56e+03	0
20	15	898.70	0.00	4.37e+06	0	41.80	0.00	1.06e+05	0	12.20	0.00	6.49e+04	0
20	20	2961.43	1.01	7.67e+06	1	1857.87	0.55	1.13e+06	1	1830.17	0.71	2.21e+06	1
30	10	534.95	0.00	3.02e+06	0	20.03	0.00	8.28e+04	0	16.62	0.00	1.19e+05	0
30	15	9645.63	4.01	2.67e+07	5	2463.98	0.70	2.37e+06	1	2049.69	0.89	2.48e+06	1
30	20	12868.66	7.40	2.40e+07	6	9100.00	2.63	8.31e+06	5	8227.49	3.28	1.49e+07	4
40	10	2667.65	0.13	1.10e+07	1	78.64	0.00	3.38e+05	0	44.27	0.00	2.90e+05	0
40	15	13890.68	9.17	3.33e+07	7	9099.72	2.79	1.30e+07	5	9117.61	3.08	2.11e+07	5
40	20	16662.97	12.50	2.07e+07	9	9533.66	5.68	6.03e+06	5	9375.56	4.96	9.13e+06	5
50	10	6684.31	1.41	2.12e+07	2	2077.20	0.37	5.96e+06	1	1116.92	0.00	4.64e+06	0
50	15	12860.13	11.88	2.61e+07	7	9346.38	4.74	1.42e+07	5	9139.57	4.92	1.57e+07	5
50	20	18027.06	17.47	1.54e+07	10	12875.13	7.57	1.19e+07	6	10822.40	7.16	1.14e+07	5
60	10	7634.67	3.60	2.02e+07	4	3996.72	0.81	1.11e+07	2	3965.79	0.58	1.60e+07	2
60	15	14168.50	42.21	2.11e+07	7	10054.19	7.56	7.78e+06	5	9467.56	6.57	1.74e+07	5
60	20	18017.69	78.75	1.39e+07	10	15032.53	14.94	3.02e+06	8	12211.80	9.06	1.48e+07	6

Élasticité nulle $(0 \times b_k)$

Instance		DL:ARC				DL:CH1				DL:CH2			
#OD	%t	Cpu	Gap	#it	Nopt	Cpu	Gap	#it	Nopt	Cpu	Gap	#it	Nopt
10	10									0.05	0.00	1.30e+02	0
10	15									0.06	0.00	1.83e+02	0
10	20									0.46	0.00	2.48e+02	0
20	10									0.34	0.00	1.47e+03	0
20	15									3.07	0.00	1.13e+04	0
20	20									1429.00	0.00	9.33e+05	0
30	10									4.19	0.00	2.12e+04	0
30	15									1883.13	0.20	1.03e+06	1
30	20									6208.21	1.73	3.60e+06	3
40	10									8.98	0.00	6.41e+04	0
40	15									5762.39	1.18	8.62e+06	2
40	20									9092.36	4.59	2.89e+06	5
50	10									832.08	0.00	2.87e+06	0
50	15									8396.43	3.32	1.11e+07	4
50	20									10397.13	6.00	5.28e+06	5
60	10									3026.08	0.10	9.61e+06	1
60	15									9045.13	5.22	1.40e+07	5
60	20									11071.50	7.87	6.42e+06	6

Tableau 2.11 Réseaux **Grille** : résultats obtenus avec tarifs positifs et demande linéaire

Lorsque l'*élasticité est forte*, nous constatons que les bornes supérieures des trois formulations sont de mauvaise qualité. En fait avec une élasticité forte, une faible variation du tarif entraîne une forte variation de la demande, ce qui implique une différence élevée entre la demande du problème relaxé et la demande de la solution trouvée. La borne supérieure est d'autant plus mauvaise lorsque le nombre de contraintes provenant de la linéarisation des termes bilinéaires est élevé. C'est pour cette raison que la formulation par chemins DL:CH2 est la meilleure des trois formulations proposées. Elle est suivie de la formulation par chemins (DL:CH1) qui est meilleure que la formulation par arcs (DL:ARC). En fait, DL:CH1 contient plus de contraintes impliquant les variables binaires que DL:CH2. La différence de contraintes provenant de la linéarisation des termes bilinéaires entre les deux formulations par chemins est d'environ deux fois le nombre total de chemins. La formulation par arcs DL:ARC quant à elle possède des contraintes dont plus de la moitié impliquent des variables binaires et des grandes constantes. Dans la quasi-majorité des cas, le nombre de contraintes de la formulation par arcs est plus élevé que celui de la formulation par chemins. Les résultats numériques confirment l'efficacité de DL:CH2 par rapport aux deux autres formulations. Par exemple, en considérant les instances du réseau de Voronoï et Grille, nous remarquons que l'on est capable de résoudre avec DL:CH2 les instances de 20 produits en moins de 35 minutes. Par contre, on n'est pas capable de résoudre plus de la moitié des instances de 10 produits avec DL:CH1 et DL:ARC dans les délais du temps d'exécution maximal. De plus, lorsque la taille des instances augmente, l'erreur relative (gap) entre la meilleure solution trouvée et la borne supérieure des formulations DL:CH1 et DL:ARC croît très rapidement.

Dans le but de montrer la mauvaise qualité des bornes supérieures des formulations DL:ARC et DL:CH1 par rapport à DL:CH2, nous présentons dans le tableau 2.12 ci-dessous, l'écart relatif (en %) entre la pire borne supérieure des trois formulations obtenue à la racine de l'arbre de branchement et la borne supérieure de chacune des formulations. Cet écart relatif est représenté par la colonne ***Er***. Pour chaque formulation, nous présentons également le nombre de fois où la pire borne supérieure lui correspond. Ce nombre est représenté par la

colonne ***Nb***.

	Instance		DL:ARC		DL:CH1		DL:CH2	
	#OD	%t	Er	Nb	Er	Nb	Er	Nb
DMS	10	10	2.08	5	4.76	5	2765.87	0
	10	15	2.20	5	3.69	5	8345.90	0
	10	20	1.39	3	4.50	7	1126.19	0
	20	10	1.55	4	7.25	6	1059.66	0
	20	15	0.99	6	4.26	4	1669.47	0
	20	20	1.43	5	3.22	5	1015.94	0
	30	10	1.10	5	4.58	5	1029.36	0
	30	15	0.83	7	4.45	3	1386.90	0
	30	20	1.14	5	3.11	5	830.62	0
Delaunay	10	10	10.30	2	2.19	8	10957.48	0
	10	15	10.42	0	0.00	7	5968.83	0
	10	20	6.26	1	0.24	9	12522.17	0
	20	10	6.14	4	2.94	6	12514.85	0
	20	15	8.02	1	0.34	8	5998.16	0
	20	20	6.58	0	0.00	10	4532.91	0
	30	10	4.20	4	1.56	6	5647.52	0
	30	15	6.94	1	0.08	9	4370.72	0
	30	20	5.72	0	0.00	10	3405.06	0
Voronoi	10	10	6.15	3	1.58	7	11288.05	0
	10	15	4.60	2	0.09	9	1896.60	0
	10	20	3.32	1	0.05	9	1157.65	0
	20	10	4.28	2	0.45	8	2039.51	0
	20	15	3.81	2	0.16	8	1061.12	0
	20	20	2.66	1	0.06	9	558.30	0
	30	10	3.88	2	0.48	8	1060.85	0
	30	15	3.40	2	0.13	8	690.36	0
	30	20	2.41	2	0.08	8	492.76	0
Grille	10	10	15.70	0	0.00	10	14439.63	0
	10	15	14.26	0	0.00	9	6654.19	0
	10	20	10.94	0	0.00	10	12406.32	0
	20	10	15.06	1	0.08	9	19467.28	0
	20	15	15.10	0	0.00	10	8327.21	0
	20	20	11.33	0	0.00	10	5745.43	0
	30	10	13.95	1	0.07	9	8291.55	0
	30	15	13.59	0	0.00	10	6342.39	0
	30	20	9.94	0	0.00	10	4473.16	0

Tableau 2.12 Comparaison des bornes supérieures obtenues à la racine de l'arbre de branchement de CPLEX des formulations DL:ARC, DL:CH1 et DL:CH2 sur les instances de 10, 20 et 30 produits avec l'élasticité forte

Nous constatons que les bornes supérieures à la racine de l'arbre de branchement de la formulation DL:CH2 sont en moyenne à 5000% de la pire borne supérieure des trois formulations. Mieux encore, les bornes supérieures de DL:CH2 sont toujours inférieures aux bornes supérieures des deux autres formulations. Dans la plupart des cas, la pire borne supérieure à la racine de l'arbre de branchement provient de la formulation DL:CH1. Cependant, la borne supérieure de DL:CH1 décroît plus rapidement que celle de DL:ARC car fixer une variable binaire associée à un chemin dans l'arbre de branchement de DL:CH1 équivaut à fixer, dans celui de DL:ARC, un sous-ensemble de variables binaires correspondant aux arcs appartenant à ce chemin. Nous présentons, aux figures 2.9 et 2.10 ci-dessous, l'évolution des bornes supérieures et inférieures sur le revenu du meneur en fonction du temps de calcul des formulations

DL:ARC, DL:CH1 et DL:CH2. Notons que la courbe Borne inf (resp. sup) fait référence à l'évolution de la borne inférieure (resp. supérieure).

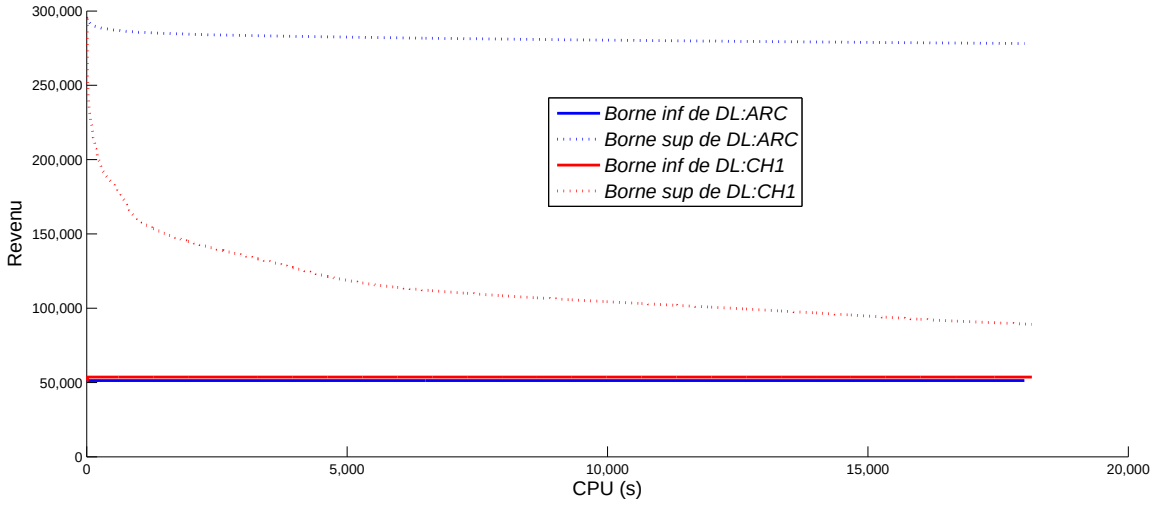


Figure 2.9 Évolution des bornes supérieure et inférieure sur le revenu du meneur en fonction du temps de calcul des formulations DL:ARC et DL:CH1 sur un réseau de **Voronoï** de 10 produits et 20% d'arcs taxables avec l'élasticité forte

La figure 2.9 montre qu'à la racine de l'arbre de branchement, les bornes supérieures des formulations DL:ARC et DL:CH1 ont presque la même valeur. Par ailleurs, la borne supérieure de DL:CH1 décroît rapidement tandis que celle de DL:ARC décroît très lentement comme le montre la figure 2.9 où le gap de DL:CH1 passe de 452% à 66% alors que celui de DL:ARC passe de 476% à 441%. Nous remarquons que les formulations trouvent rapident une bonne solution entière et les bornes supérieures peinent à décroître avec le temps pour fermer le gap. Lorsque la taille des instances augmente, le gap de DL:CH1 devient également très mauvais car sa borne supérieure à la racine de l'arbre est de mauvaise qualité, et elle décroît très lentement comme le présente la figure 2.10 ci-dessous où le gap de DL:CH1 passe de 3175% à 978% tandis que celui de DL:CH2 de 94% à 19%.

Pour des instances de petite taille, nous remarquons que les formulations DL:ARC et DL:CH1 trouvent généralement une solution de bonne qualité mais la borne supérieure est très mau-

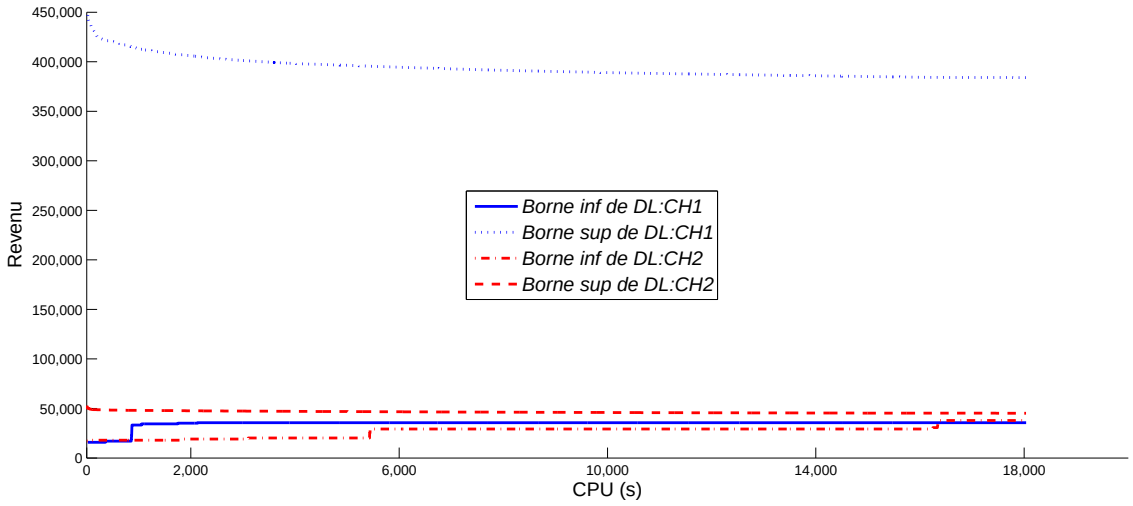


Figure 2.10 Évolution des bornes supérieure et inférieure sur le revenu du meneur en fonction du temps de calcul des formulations DL:CH1 et DL:CH2 sur un réseau de **Voronoï** de 30 produits et 20% d'arcs taxables avec l'élasticité forte

vaise. Le tableau 2.13 ci-dessous confirme cette remarque en comparant le gap réel et le « gap obtenu » de DL:ARC et DL:CH1 pour les instances du réseau de Delaunay, Voronoï et Grille avec 10 produits où le gap réel est l'écart relatif entre la solution trouvée et la solution optimale tandis que le « gap obtenu » est le gap fourni par la formulation. La solution optimale est obtenue avec la formulation DL:CH2.

Instance		Delaunay				Voronoi				Grille			
		Gap-ARC		Gap-CH1		Gap-ARC		Gap-CH1		Gap-ARC		Gap-CH1	
#OD	%t	(1)	(2)	(1)	(2)	(1)	(2)	(1)	(2)	(1)	(2)	(1)	(2)
10	10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
10	15	4.21	0.00	0.00	0.00	175.74	0.00	19.93	0.00	370.13	0.00	0.00	0.00
10	20	1406.54	0.00	985.96	0.00	346.69	0.60	97.79	0.00	7268.18	15.13	3902.52	2.57

(1) = gap « btenu » par les formulations DL:ARC ou DL:CH1 selon le cas
(2) = gap réel

Tableau 2.13 Comparaison entre le gap réel et le gap « obtenu » des formulations DL:ARC et DL:CH1 sur les instances de 10 produits

Pour les instances avec 10% et 15% d'arcs taxables, DL:ARC et DL:CH1 obtiennent la solution optimale mais la mauvaise qualité de la borne supérieure empêche de prouver l'optimalité de cette solution. Pour les instances avec 20% d'arcs taxables, bien qu'ils ne trouvent pas

toujours la solution optimale, le gap obtenu est de très mauvaise qualité comparé au gap réel. En considérant les instances de Grille avec 10 produits et 20% d'arcs taxables, on passe d'un « gap obtenu » par DL:ARC de 7268.18% à un gap réel de 15.13% tandis qu'avec DL:CH1, on passe d'un « gap obtenu » de 3902.52% à un gap réel de 2.57%.

Lorsque l'*élasticité est faible*, nous remarquons qu'à la racine de l'arbre de branchement, les bornes supérieures des trois formulations sont quasi-égales. Cette remarque provient du fait qu'avec une élasticité faible, une forte variation des tarifs sur les arcs taxables entraîne une très faible variation de la demande, ce qui implique un écart relativement faible aux nœuds de l'arbre de branchement entre la demande du problème relaxé et celle correspondante à la solution trouvée. Le tableau 2.14 ci-dessous présente l'écart relatif (en %) entre la pire borne supérieure des trois formulations développées obtenue à la racine de l'arbre de branchement et la borne supérieure de chacune des formulations.

	Instance		DL:ARC		DL:CH1		DL:CH2	
	#OD	%t	Er	Nb	Er	Nb	Er	Nb
DMS	10	10	4.63	0	0	10	0.04	1
	10	15	4.75	0	0	10	0.02	0
	10	20	4.88	0	0	10	0.03	0
	40	10	6.01	0	0	10	0.04	0
	40	15	8.37	0	0	10	0.06	0
	40	20	7.25	0	0	10	0.05	0
	60	10	7.07	0	0	10	0.05	0
	60	15	8.33	0	0	10	0.06	0
	60	20	7.68	0	0	10	0.06	0
Delaunay	10	10	4.07	0	0	10	0.01	2
	10	15	1.98	0	0	10	0.01	0
	10	20	2.10	0	0	10	0.01	0
	40	10	4.66	0	0	10	0.03	0
	40	15	4.19	0	0	10	0.03	0
	40	20	3.47	0	0	10	0.02	0
	60	10	5.81	0	0	10	0.04	0
	60	15	4.85	0	0	10	0.03	0
	60	20	3.76	0	0	10	0.02	0
Voronoi	10	10	5.62	0	0	10	0.03	0
	10	15	6.00	0	0	10	0.01	0
	10	20	5.38	0	0	10	0.02	0
	40	10	8.37	0	0	10	0.05	0
	40	15	6.91	0	0	10	0.02	0
	40	20	7.49	0	0	10	0.02	0
	60	10	8.98	0	0	10	0.05	0
	60	15	5.96	0	0	10	0.02	0
	60	20	7.15	0	0	10	0.02	0
Grille	10	10	4.43	0	0	10	0.02	1
	10	15	2.59	0	0	10	0.02	0
	10	20	3.16	0	0	10	0.01	0
	40	10	5.47	0	0	10	0.05	0
	40	15	4.40	0	0	10	0.02	0
	40	20	4.43	0	0	10	0.01	0
	60	10	5.73	0	0	10	0.05	0
	60	15	5.00	0	0	10	0.03	0
	60	20	4.24	0	0	10	0.01	0

Tableau 2.14 Comparaison des bornes supérieures obtenues à la racine de l'arbre de branchement de CPLEX des formulations DL:ARC, DL:CH1 et DL:CH2 sur les instances de 10, 40 et 60 produits avec l'élasticité faible

Nous constatons que les bornes supérieures à la racine de l'arbre de branchement des formulations par chemins (DL:CH1 et DL:CH2) sont presque égales. La formulation par arcs (DL:ARC) a toujours la borne supérieure de plus petite valeur, et est en moyenne à 5% des bornes supérieures des formulations par chemins. Cependant sa borne supérieure décroît plus lentement que les bornes supérieures des formulations par chemins. De plus, les bornes inférieures de DL:CH1 et DL:CH2 croissent plus rapidement que la borne inférieure de DL:ARC (voir figure 2.11).

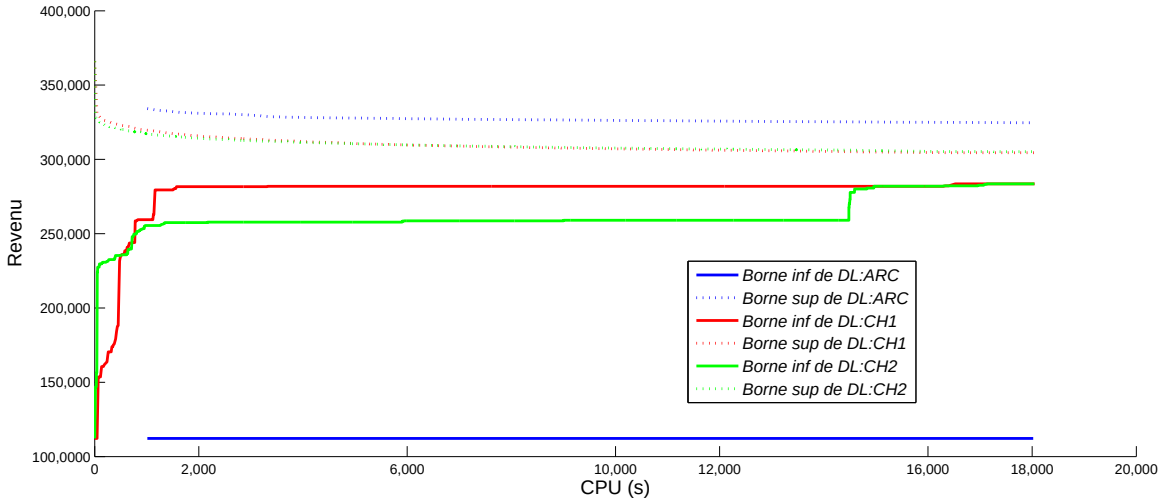


Figure 2.11 Évolution des bornes supérieure et inférieure sur le revenu du meneur en fonction du temps de calcul des formulations DL:ARC, DL:CH1 et DL:CH2 sur une instance de **Voronoï** de 40 produits avec 20% d'arcs taxables et l'élasticité faible

Pour les instances de petites tailles, les trois formulations obtiennent une solution optimale avec un temps d'exécution de moins de 60 secondes. Pour les instances de taille moyenne, les formulations par chemins sont à peu près équivalentes car leurs temps d'exécution sont du même ordre et la différence entre leurs gaps est inférieure à 0.7%. Pour les instances de grande taille ($\geq$ 60 produits pour les instances du réseau de Delaunay, Voronoï et Grille), la formulation DL:CH2 est meilleure que DL:CH1 car le gap obtenu est en moyenne inférieur à celui de DL:CH1. À titre d'illustration, en considérant les instances du réseau Voronoï avec 60

produits et 20% d'arcs taxables, aucune des trois formulations n'a pu résoudre une instance à l'optimalité, cependant le gap obtenu avec DL:CH2 est de 11%, celui avec DL:CH1 est de 52% et celui avec DL:ARC est de 320%.

Nous remarquons que pour les instances de petite taille, bien que le DL avec l'élasticité forte, faible et nulle se résout à l'optimalité, le temps d'exécution du DL est plus élevé avec l'élasticité forte, suivi de l'élasticité faible et enfin l'élasticité nulle. À titre d'illustration, en considérant les instances de Voronoï avec 20 produits et 20% d'arcs taxables, la résolution du DL avec élasticité forte, faible et nulle a un temps d'exécution respectif de 34 minutes, 3 minutes et 1 minute. Pour les instances de grande taille, nous présentons dans le tableau 2.15 ci-dessous les résultats obtenus avec DL:CH2 en considérant les instances du réseau de Delaunay, Voronoï et Grille (resp. DMS) avec 60 (resp. 100) produits.

		élasticité forte $(100 \times b_k)$				**élasticité faible** $(1 \times b_k)$				**élasticité nulle** $(0 \times b_k)$			
Type de réseau	%t	Cpu	Gap	Gap max	Nopt	Cpu	Gap	Gap max	Nopt	Cpu	Gap	Gap max	Nopt
DMS	10	6104.80	2.28	13.22	3	681.47	0.00	0.00	0	11.55	0.00	0.00	0
	15	12296.44	11.67	26.10	6	2473.75	0.42	4.20	1	1392.18	0.00	0.00	0
	20	17319.16	15.19	28.47	9	12090.63	3.01	6.82	6	7914.38	0.66	3.61	2
Delaunay	10	8874.88	0.81	5.08	3	18.29	0.00	0.00	0	5.19	0.00	0.00	0
	15	14315.71	5.41	21.87	6	2104.40	0.08	0.84	1	1876.68	0.02	0.23	1
	20	18076.28	10.92	27.01	10	5785.66	0.16	1.17	2	3163.25	0.12	1.23	1
Voronoi	10	6027.55	7.38	57.95	2	410.96	0.00	0.00	0	167.68	0.00	0.00	0
	15	18065.13	26.29	51.91	10	15832.44	5.16	18.28	8	10184.26	3.19	15.59	5
	20	18080.82	29.86	47.22	10	18047.73	11.13	28.07	10	16607.25	8.01	28.14	9
Grille	10	7520.63	116.83	1024.32	4	3965.79	0.58	4.17	2	3026.08	0.10	0.99	1
	15	14124.76	613.81	2927.78	7	9467.56	6.57	25.78	5	9045.13	5.22	27.41	5
	20	16416.66	115.78	318.24	9	12211.80	9.06	26.78	6	11071.50	7.87	26.00	6

Tableau 2.15 Résultats obtenus avec la formulation par chemins DL:CH2 en considérant les instances du réseau de Delaunay, Voronoï et Grille (resp. DMS) avec 60 (resp. 100) produits

Nous remarquons qu'il y a une très grande différence entre le gap obtenu avec l'élasticité forte et celui obtenu avec l'élasticité faible ou nulle. La même remarque est faite sur le nombre d'instances du DL n'ayant pas pu être résolues à l'optimalité. Pour ces instances, il y a une faible différence entre les résultats obtenus avec l'élasticité faible et ceux obtenus avec l'élasticité nulle car le temps d'exécution, le gap, le gap maximal et le nombre d'instances n'ayant pas pu être résolues à l'optimalité sont quasi-égaux.

Une autre façon d'expliquer l'augmentation de la difficulté du DL avec l'élasticité de la demande se trouve dans la configuration de la solution optimale du DL. Soit $t = (t_1, t_2, \ldots, t_{|\mathcal{A}_1|})$

un vecteur de tarifs. Pour $t^0 = (0, \ldots, 0)$, on a $h(t^0) = h^0$ qui est le vecteur constitué des chemins utilisés par chaque produit. Le sommet $h(t^0) = h^0$ reste optimal pour le problème du second niveau tant que t ne dépasse pas un vecteur seuil t^1. Ainsi le sommet h^i reste optimal pour tout $t \in [t^{i-1}, t^i]$ où t^{i-1} et t^i sont des vecteurs seuils. Lorsque l'élasticité est nulle, nous remarquons que les optima locaux sont atteints aux vecteurs seuils t^i. Nous observons presque le même scénario lorsque l'élasticité est faible. Cependant lorsque l'élasticité augmente, les optima locaux ne sont plus forcément atteints aux vecteurs seuils t^i car la fonction objectif du meneur n'augmente pas toujours avec les tarifs, ce qui entraîne lors de la résolution une erreur relative plus élevé car il est plus facile de déterminer un optimum local correspondant à un vecteur seuil t^i qu'un optimum local différent de t^i.

Dans l'exemple ci-dessous, nous illustrons que lorsque l'élasticité est forte, les optima locaux ne correspondent pas forcément aux vecteurs seuils des tarifs. Soit le réseau de la figure 2.12 ayant un arc taxable et deux produits.

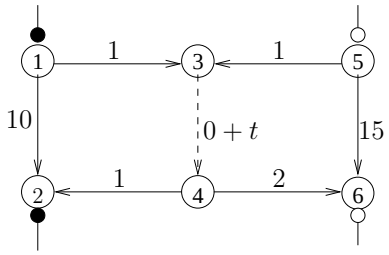


Figure 2.12 Réseau avec un arc taxable et deux produits

Les informations sur les produits sont données au tableau 2.16.

	o - d	demande	chemins	coût
produit 1	$1 \longrightarrow 2$	$d_1(U_1) = 80 - bU_1$	$1 \to 3 \to 4 \to 2$ $1 \to 2$	$2 + t$ 10
produit 2	$5 \longrightarrow 6$	$d_2(U_2) = 2200 - 2bU_2$	$5 \to 3 \to 4 \to 6$ $5 \to 6$	$3 + t$ 15

Tableau 2.16 Données du réseau de la figure 2.12

Le but est de fixer le tarif sur l'arc taxable $(3, 4)$ afin de maximiser le revenu du meneur en faisant varier b.

L'arc $(3, 4)$ est attractif pour les deux produits si $t \leq 8$ tandis qu'il est attractif seulement pour le produit 2 si $8 \leq t \leq 12$ et n'est attractif pour aucun produit si $t > 12$. La fonction objectif du meneur est donc définie par :

$$\begin{cases} (3000 - 8b - 3bt)t & \text{si } \ 0 \leq t \leq 8 \\ (2200 - 6b - 2bt)t & \text{si } \ 8 < t \leq 12 \\ 0 & \text{si } \ 12 < t \end{cases}$$

et est représentée à la figure 2.13 ci-dessous pour les valeurs de $b = 0, 20, 40, 60, 80$ et 100.

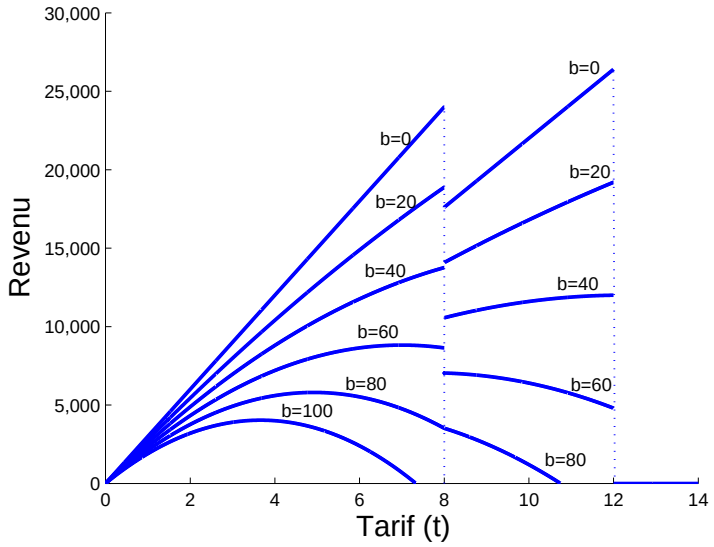


Figure 2.13 Revenu du meneur en fonction du tarif

Les tarifs seuils dans ce cas sont $t = 8$ et $t = 12$. Nous présentons dans le tableau 2.17 le nombre d'optima locaux et les tarifs correspondant en fonction de b.
Pour $b = 60$, 80 et 100, le tarif optimal est différent des tarifs seuils (8 et 12).

Pour terminer, nous présentons les résultats obtenus lorsque les tarifs sont positifs ou négatifs. Pour ce faire, nous utilisons la formulation DL:CH2 pour résoudre le DL car nous avons vu qu'elle est meilleure que les deux autres formulations (DL:ARC et DL:CH1). Puis,

b	Optima locaux	tarif optimal
0	2 optima locaux en $t = 8$ et $t = 12$	$t = 12$
20	2 optima locaux en $t = 8$ et $t = 12$	$t = 12$
40	2 optima locaux en $t = 8$ et $t = 12$	$t = 8$
60	1 optima local en $t = 7$	$t = 7$
80	1 optimum local en $t = 59/12$	$t = 59/12$
100	1 optimum local en $t = 11/3$	$t = 11/3$

Tableau 2.17 Optima locaux en fonction de la pente de la demande

une analyse de la sensibilité sur l'élasticité des fonctions de demande est effectuée en considérant les fonctions de demande à élasticité forte, faible et nulle. Ces résultats sont reportés dans les tableaux 2.18, 2.19, 2.20 et 2.21.

Instance		**élasticité forte** ($100 \times b_k$)				**élasticité faible** ($1 \times b_k$)				**élasticité nulle** ($0 \times b_k$)			
#OD	%t	Cpu	Gap	#it	Nopt	Cpu	Gap	#it	Nopt	Cpu	Gap	#it	Nopt
20	10	0.14	0.00	5.04e+02	0	0.11	0.00	6.38e+02	0	0.06	0.00	1.23e+02	0
20	15	0.25	0.00	1.39e+03	0	0.15	0.00	1.09e+03	0	0.09	0.00	2.49e+02	0
20	20	1.09	0.00	1.72e+03	0	1.48	0.00	2.19e+03	0	1.25	0.00	1.34e+03	0
40	10	5.26	0.00	4.15e+04	0	2.59	0.00	9.41e+03	0	0.57	0.00	2.06e+03	0
40	15	18.27	0.00	8.42e+04	0	2.11	0.00	1.01e+04	0	0.89	0.00	2.47e+03	0
40	20	181.62	0.00	5.75e+05	0	56.98	0.00	1.31e+05	0	20.14	0.00	3.15e+04	0
60	10	157.76	0.00	6.81e+05	0	9.93	0.00	4.11e+04	0	1.18	0.00	4.87e+03	0
60	15	4899.37	0.84	1.40e+07	1	85.45	0.00	4.27e+05	0	20.96	0.00	1.04e+05	0
60	20	8095.00	1.00	1.53e+07	3	2163.95	0.00	5.09e+06	0	920.00	0.00	1.50e+06	0
80	10	4260.39	3.05	1.92e+07	2	305.58	0.00	8.20e+05	0	38.45	0.00	1.66e+05	0
80	15	10448.27	5.67	2.55e+07	5	2310.53	0.00	5.44e+06	0	469.34	0.00	2.13e+06	0
80	20	16243.67	8.54	2.75e+07	9	10938.29	3.24	1.80e+07	5	8239.27	1.68	1.57e+07	4
100	10	6036.47	2.63	2.62e+07	3	1837.48	0.00	6.04e+06	1	729.22	0.00	1.95e+06	0
100	15	15438.13	10.87	4.29e+07	8	4480.85	0.92	1.33e+07	2	2730.25	0.44	7.40e+06	1
100	20	16775.46	18.18	2.07e+07	9	16233.59	6.86	3.29e+07	9	14316.84	4.13	2.40e+07	7

Tableau 2.18 Réseaux **DMS** : résultats obtenus avec demande linéaire

Instance		**élasticité forte** ($100 \times b_k$)				**élasticité faible** ($1 \times b_k$)				**élasticité nulle** ($0 \times b_k$)			
#OD	%t	Cpu	Gap	#it	Nopt	Cpu	Gap	#it	Nopt	Cpu	Gap	#it	Nopt
10	10	0.02	0.00	1.63e+02	0	0.03	0.00	1.80e+02	0	0.01	0.00	4.89e+01	0
10	15	0.07	0.00	2.09e+02	0	0.07	0.00	3.79e+02	0	0.04	0.00	9.69e+01	0
10	20	0.35	0.00	5.49e+02	0	0.23	0.00	7.00e+02	0	0.14	0.00	1.94e+02	0
20	10	0.13	0.00	6.60e+02	0	0.12	0.00	5.51e+02	0	0.06	0.00	1.61e+02	0
20	15	1.40	0.00	1.52e+03	0	0.86	0.00	1.43e+03	0	0.42	0.00	3.53e+02	0
20	20	3.26	0.00	3.82e+03	0	3.34	0.00	3.12e+03	0	1.99	0.00	7.55e+02	0
30	10	2.64	0.00	1.98e+04	0	0.60	0.00	2.64e+03	0	0.18	0.00	4.08e+02	0
30	15	25.57	0.00	6.65e+04	0	11.43	0.00	1.58e+04	0	4.64	0.00	4.03e+03	0
30	20	29.92	0.00	2.95e+04	0	140.78	0.00	1.41e+05	0	53.66	0.00	4.03e+04	0
40	10	214.52	0.00	9.12e+05	0	9.49	0.00	5.22e+04	0	2.33	0.00	1.15e+04	0
40	15	1851.10	0.01	2.68e+06	1	220.03	0.00	2.49e+05	0	40.30	0.00	3.81e+04	0
40	20	2367.38	0.00	2.77e+06	0	1117.96	0.00	8.88e+05	0	618.79	0.00	3.70e+05	0
50	10	146.05	0.00	6.84e+05	0	13.78	0.00	5.97e+04	0	3.11	0.00	1.29e+04	0
50	15	239.52	0.00	6.21e+05	0	1091.37	0.00	1.20e+06	0	289.65	0.00	2.62e+05	0
50	20	4235.55	0.05	7.82e+06	2	4156.63	0.34	2.37e+06	2	3654.91	0.15	1.96e+06	1
60	10	7366.28	2.95	3.57e+07	3	88.60	0.00	3.76e+05	0	14.46	0.00	8.90e+04	0
60	15	11568.01	3.16	1.32e+07	6	3406.67	0.30	2.64e+06	1	2957.58	0.14	1.94e+06	1
60	20	16672.21	3.81	1.32e+07	9	9997.19	1.07	5.90e+06	5	6746.83	0.57	3.55e+06	2

Tableau 2.19 Réseau de **Delaunay** : résultats obtenus avec demande linéaire

Instance		**élasticité forte** ($100 \times b_k$)				**élasticité faible** ($1 \times b_k$)				**élasticité nulle** ($0 \times b_k$)			
#OD	%t	Cpu	Gap	#it	Nopt	Cpu	Gap	#it	Nopt	Cpu	Gap	#it	Nopt
10	10	0.26	0.00	6.71e+02	0	0.15	0.00	5.07e+02	0	0.12	0.00	1.43e+02	0
10	15	10.87	0.00	5.00e+03	0	13.96	0.00	9.73e+03	0	9.60	0.00	2.14e+03	0
10	20	15.46	0.00	1.06e+04	0	161.94	0.00	2.64e+05	0	85.33	0.00	5.08e+04	0
20	10	2.82	0.00	1.03e+04	0	0.78	0.00	2.14e+03	0	0.66	0.00	5.13e+02	0
20	15	781.33	0.00	2.28e+05	0	244.53	0.00	1.54e+05	0	137.70	0.00	3.71e+04	0
20	20	2191.19	0.43	9.27e+05	1	2219.54	0.00	1.52e+06	0	2316.92	0.12	5.42e+05	1
30	10	116.22	0.00	3.71e+05	0	25.28	0.00	8.35e+04	0	17.01	0.00	1.96e+04	0
30	15	9135.54	0.48	2.51e+06	3	5377.21	0.88	1.63e+06	2	2583.51	0.61	2.67e+05	1
30	20	11068.36	2.85	2.98e+06	4	11756.31	2.23	2.93e+06	6	8558.30	1.61	1.09e+06	4
40	10	1226.35	0.00	2.31e+06	0	215.07	0.00	5.30e+05	0	38.23	0.00	6.16e+04	0
40	15	15754.10	4.66	4.37e+06	8	12047.98	4.03	5.06e+06	6	9827.01	2.47	1.70e+06	5
40	20	16482.95	8.00	2.68e+06	9	16314.46	6.96	2.34e+06	9	16103.77	8.84	1.27e+06	8
50	10	5689.58	1.94	1.03e+07	1	729.48	0.00	1.99e+06	0	195.33	0.00	3.04e+05	0
50	15	18099.71	12.68	3.57e+06	10	16199.09	9.87	5.04e+06	8	15916.98	7.25	1.73e+06	8
50	20	18081.83	18.24	2.34e+06	10	16321.52	25.92	1.16e+06	9	16358.50	30.63	8.49e+05	9
60	10	8251.90	9.27	1.83e+07	4	3728.07	0.75	5.86e+06	2	1114.73	0.00	1.14e+06	0
60	15	18070.38	25.59	2.62e+06	10	18064.84	19.62	3.27e+06	10	18021.42	9.76	2.29e+06	10
60	20	18085.57	25.88	1.43e+06	10	16521.06	36.17	1.09e+06	9	16289.42	26.67	4.02e+05	9

Tableau 2.20 Réseaux **Voronoi** : résultats obtenus avec demande linéaire

Instance		**élasticité forte** ($100 \times b_k$)				**élasticité faible** ($1 \times b_k$)				**élasticité nulle** ($0 \times b_k$)			
#OD	%t	Cpu	Gap	#it	Nopt	Cpu	Gap	#it	Nopt	Cpu	Gap	#it	Nopt
10	10	0.40	0.00	7.24e+02	0	0.25	0.00	4.94e+02	0	0.21	0.00	1.30e+02	0
10	15	1.01	0.00	5.46e+02	0	0.96	0.00	1.05e+03	0	0.70	0.00	2.39e+02	0
10	20	7.77	0.00	5.96e+03	0	7.71	0.00	3.08e+03	0	4.17	0.00	5.25e+02	0
20	10	11.84	0.00	2.43e+04	0	6.35	0.00	2.05e+04	0	2.73	0.00	3.60e+03	0
20	15	132.83	0.00	1.85e+05	0	164.71	0.00	1.93e+05	0	64.18	0.00	2.58e+04	0
20	20	1895.80	0.00	8.41e+05	0	2112.10	0.66	2.86e+05	1	1968.86	0.28	3.03e+05	1
30	10	428.21	0.00	8.48e+05	0	52.92	0.00	1.63e+05	0	25.87	0.00	4.68e+04	0
30	15	3026.39	2.71	2.42e+06	1	4234.51	1.10	2.09e+06	2	2576.15	0.97	2.69e+05	1
30	20	11758.28	4.44	2.74e+06	6	9158.90	3.28	9.90e+05	5	8795.03	1.91	8.47e+05	4
40	10	3658.80	39.82	3.92e+06	2	703.70	0.00	1.29e+06	0	1824.16	0.17	1.35e+06	1
40	15	10587.19	30.90	4.85e+06	5	9673.40	5.03	4.87e+06	5	7734.47	2.50	1.39e+06	3
40	20	15166.65	40.66	2.97e+06	8	11344.52	7.41	1.75e+06	6	10437.80	4.58	5.94e+05	5
50	10	4948.59	43.55	5.65e+06	2	3942.50	0.82	7.54e+06	2	3577.38	0.40	3.61e+06	1
50	15	12894.75	38.23	5.50e+06	7	11276.46	6.70	6.81e+06	6	9525.30	5.90	1.71e+06	5
50	20	16329.96	40.20	1.71e+06	9	13883.15	9.41	3.01e+06	7	11482.36	6.53	6.75e+05	6
60	10	5890.58	31.10	4.22e+06	3	5840.19	0.87	8.19e+06	3	4009.45	0.85	3.09e+06	2
60	15	14874.38	91.41	9.92e+06	8	11290.74	9.65	4.83e+06	6	9847.88	8.66	1.89e+06	5
60	20	16313.32	74.32	1.60e+06	9	15261.57	12.96	2.97e+06	8	12167.28	8.27	1.38e+06	5

Tableau 2.21 Réseaux **Grille** : résultats obtenus avec demande linéaire

Nous remarquons qu'en général, les instances où les tarifs ne sont pas contraignants sont plus difficiles à résoudre que celles où les tarifs sont positifs. En effet, accepter un tarif négatif implique que les produits qui n'étaient pas intéressants lorsque les tarifs étaient positifs peuvent devenir intéressants, et par conséquent il y a plusieurs solutions à explorer d'où une plus grande complexité combinatoire du problème.

De manière générale, le gap moyen et le nombre moyen d'instances n'ayant pas pu être résolues à l'optimalité avec tarifs réels sont supérieurs à ceux avec tarifs positifs. Cependant, lorsque l'élasticité est forte et la taille des instances grande, le gap moyen obtenu avec les tarifs réels est inférieur à celui obtenu avec les tarifs positifs. En d'autres termes, la borne supérieure fournie lorsque les tarifs sont réels est meilleure que celle fournie lorsque les tarifs sont positifs. À titre d'illustration, en considérant les instances de Grille avec 60 produits, le gap moyen obtenu avec les tarifs réels est 66% tandis que celui obtenu avec les tarifs positifs est 283%. Nous remarquons également qu'avec des tarifs réels, la difficulté du DL augmente également avec l'élasticité de la demande.

2.7 Conclusion

Les formulations proposées dans ce chapitre sont des extensions des formulations proposées par Labbé *et al.* [46] et Didi *et al.* [29] pour résoudre le PTR. Parmi ces formulations, la deuxième formulation par chemins (DL:CH2) est la meilleure car elle fournit des solutions de bonne qualité et reste stable lorsque l'élasticité varie, contrairement aux deux autres formulations qui sont très mauvaises lorsque l'élasticité est forte.

Nous avons utilisé la formulation DL:CH2 pour faire une analyse de sensibilité du DL lorsque l'élasticité varie. Cette analyse nous montre que la difficulté du DL augmente avec l'élasticité de la demande (la pente de la demande).

Nous avons enfin montré qu'il existe plusieurs cas où le DL est polynômial. Il s'agit du DL avec un seul arc taxable, du POI associé au DL et enfin du DL où tous les arcs sont taxables, les tarifs réels et le nombre de produits inférieur ou égal à deux.

CHAPITRE 3

PROBLÈME DE TARIFICATION SUR UN RÉSEAU AVEC DEMANDE NON LINÉAIRE

Le problème de tarification sur un réseau avec demande non linéaire (DNL) que nous étudions dans ce chapitre est une extension du problème étudié au chapitre précédent. Nous commençons par présenter le modèle mathématique utilisé pour résoudre le DNL (section 3.1). La borne supérieure sur le revenu du meneur est donnée à la section 3.2. À la section 3.3, nous présentons les cas polynômiaux du DNL. Puis, nous proposons une méthode exacte pour résoudre le DNL (section 3.4). À la section 3.5, nous résolvons le problème d'optimisation inverse (POI) associé au DNL et nous l'exploitons pour améliorer la méthode exacte. À la section 3.6 nous développons deux heuristiques basées sur une méthode de région de confiance. Finalement, les expériences numériques sont présentées à la section 3.7.

3.1 Formulation du problème

Au chapitre 2, nous avons pour une demande linéaire vu que la formulation par chemins donne des meilleurs résultats que la formulation par arcs. Nous utilisons donc cette formulation pour développer les méthodes de résolution du DNL. Nous considérons que la demande du produit k, $d_k(U_k)$, qui dépend de l'utilité U_k du produit k, est une fonction non linéaire, non négative et strictement décroissante (voir figure 3.1) où l'utilité U_k est le coût du plus court chemin utilisé pour acheminer le produit k.

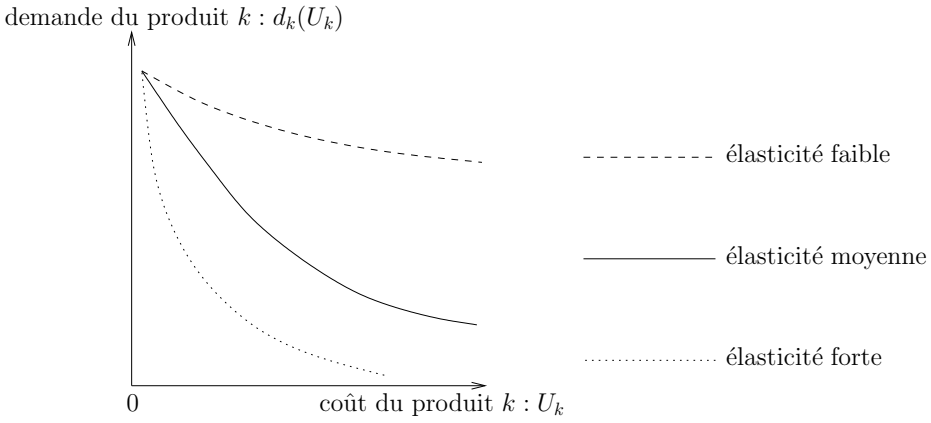


Figure 3.1 Demande non linéaire du produit k

À partir de la formulation par chemins FORM:CH (voir section 1.2.2), nous remplaçons pour tout produit k la demande inélastique d_k par la demande élastique $d_k(U_k)$. Le revenu généré par le produit k est donné par la fonction $d_k(U_k)T_k$ où T_k est le revenu généré par une unité de flot du produit k. On sait que :

$$U_k = T_k + \sum_{p \in P_k} h_p^k \sum_{a \in p} c_a \implies T_k = U_k - \sum_{p \in P_k} h_p^k \sum_{a \in p} c_a$$

$$\text{d'où} \quad d_k(U_k)T_k = d_k(U_k)U_k - d_k(U_k) \sum_{p \in P_k} h_p^k \sum_{a \in p} c_a.$$

En remplaçant $d_k(U_k)T_k$ par l'expression équivalente trouvée ci-dessus, nous obtenons la formulation suivante

DNL:CH

$$\max_{t,h,T,U} \sum_{k\in\mathcal{K}} \left(d_k(U_k)U_k - d_k(U_k) \sum_{p\in P_k} h_p^k \sum_{a\in p} c_a \right)$$

$$\begin{aligned}
\text{s.c} \quad & T_k \leq \sum_{a\in p\cap\mathcal{A}_1} t_a + M_p^k(1-h_p^k) && \forall p\in P_k,\ \forall k\in\mathcal{K} \\
& \sum_{a\in p} c_a + \sum_{a\in p\cap\mathcal{A}_1} t_a - M_p^k(1-h_p^k) \leq U_k && \forall p\in P_k,\ \forall k\in\mathcal{K} \\
& U_k \leq \sum_{a\in p} c_a + \sum_{a\in p\cap\mathcal{A}_1} t_a && \forall p\in P_k,\ \forall k\in\mathcal{K} \\
& U_k = T_k + \sum_{p\in P_k} h_p^k \sum_{a\in p} c_a && \forall k\in\mathcal{K} \\
& \sum_{p\in P_k} h_p^k = 1 && \forall k\in\mathcal{K} \\
& h_p^k \in \{0,1\} && \forall p\in P_k,\ \forall k\in\mathcal{K}.
\end{aligned}$$

DNL:CH est un problème de maximisation d'une fonction objectif non linéaire sous contraintes linéaires et variables mixtes.

Pour modéliser le comportement de la demande par rapport aux variations du prix, les chercheurs ont souvent utilisé deux classes de fonctions de demande : les fonctions de demande à élasticité constante et non constante (pour la définition du concept d'élasticité, voir la section 1.3).

Les fonctions de demande à élasticité constante β prennent la forme $d(U) = \alpha U^{-\beta}$ où α, β sont des constantes positives avec $\beta > 1$. Cette classe de fonctions a été proposée par l'économiste Moore [54]. Quant aux fonctions de demande à élasticité non constante, nous utilisons les fonctions de la forme $d(U) = \alpha \exp(-\beta U)$ ayant une élasticité βU où α et β sont des constantes positives. Cette forme a été proposée par Babonneau *et al.* [5], Hearn *et al.* [40], Yang *et al.* [70] et bien d'autres.

D'après ce qui précède, la fonction $d(U)U$ est pseudo-concave (croissante puis décroissante) ou concave (croissante) ou tout simplement convexe (décroissante) pour tout U positif comme le montre la figure 3.2.

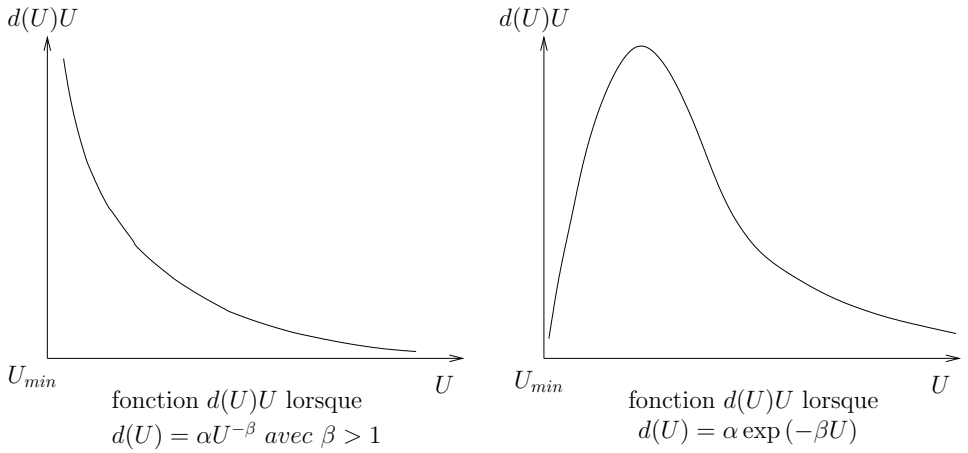


Figure 3.2 Forme de la fonction $d(U)U$

Lorsque $d(U) = \alpha \exp(-\beta U)$, la fonction $d(U)U$ est pseudo-concave et on montre qu'elle est concave pour les valeurs de U appartenant à $\left]-\infty, \frac{1}{\beta}\right]$ puis convexe pour les valeurs de U appartenant à $\left[\frac{1}{\beta}, +\infty\right[$. La méthode exacte du DNL proposée à la section 3.4 suivante est basée sur la surapproximation (resp. sous-approximation) de la fonction $d(U)U$ (resp. $d(U)$) de la formulation DNL:CH.

3.2 Borne supérieure sur le revenu du meneur

La borne supérieure est calculée de la même manière que celle du DL présenté à la section 2.4, c'est-à-dire en supposant que le chemin utilisé pour le produit k est p_0^k, et en calculant le revenu généré correspondant à ce chemin p_0^k.

Proposition 3.2.1 *La borne supérieure (BS) sur le revenu du meneur est donnée par*

$$BS = \sum_{k\in\mathcal{K}} d_k\left(\sum_{a\in p_0^k} c_a + T_k^*\right) T_k^*$$

$$où \quad T_k^* = \begin{cases} \min\left\{\gamma^k(\infty) - \gamma^k(0),\ \sum_{a\in p_0^k} c_a/\left(\beta_k - 1\right)\right\} & si\ d(U_k) = \alpha_k U_k^{-\beta} \\ \min\left\{\gamma^k(\infty) - \gamma^k(0),\ 1/\beta_k\right\} & si\ d(U_k) = \alpha_k \exp\left(-\beta U_k\right). \end{cases}$$

Preuve La fonction objectif du meneur associée au produit k est donnée par :

$$d_k(U_k)T_k.$$

Le coût fixe du chemin p_0^k est le plus petit coût fixe parmi les coûts des chemins du produit k, donc

$$d_k(U_k)T_k \leq d_k\left(\sum_{a\in p_0^k} c_a + T_k\right) T_k$$

car d_k est strictement décroissante. La fonction $d_k\left(\sum_{a\in p_0^k} c_a + T_k\right) T_k$ est pseudo-concave (croissante puis décroissante) et donc elle atteint sa valeur maximale pour la valeur de T_k qui annule la dérivée de $d_k\left(\sum_{a\in p_0^k} c_a + T_k\right) T_k$. Cette valeur de T_k est $\sum_{a\in p_0^k} c_a/\left(\beta_k - 1\right)$ (resp. $1/\beta_k$) lorsque la fonction de demande est $d(U_k) = \alpha_k U_k^{-\beta}$ (resp. $d(U_k) = \alpha_k \exp\left(-\beta U_k\right)$). Puisque $\gamma^k(\infty) - \gamma^k(0)$ est la borne supérieure sur T_k, la borne supérieure sur le revenu généré par le

produit k notée BS_k est donnée par

$$\mathrm{BS}_k = \; d_k \left(\sum_{a \in p_0^k} c_a + T_k^* \right) T_k^*$$

$$\text{où} \;\; T_k^* = \begin{cases} \min \left\{ \gamma^k(\infty) - \gamma^k(0), \; \sum_{a \in p_0^k} c_a / \left(\beta_k - 1\right) \right\} & \text{si } d(U_k) = \alpha_k U_k^{-\beta} \\ \min \left\{ \gamma^k(\infty) - \gamma^k(0), \; 1/\beta_k \right\} & \text{si } d(U_k) = \alpha_k \exp\left(-\beta U_k\right) \end{cases}$$

Finalement, en sommant ces bornes supérieures sur l'ensemble des produits, nous obtenons la borne supérieure BS.

Notons également que si pour un produit k, il n'existe aucun chemin possédant seulement les arcs non taxables, alors

$$\text{où} \;\; T_k^* = \begin{cases} \sum_{a \in p_0^k} c_a / \left(\beta_k - 1\right) & \text{si } d(U_k) = \alpha_k U_k^{-\beta} \\ 1/\beta_k & \text{si } d(U_k) = \alpha_k \exp\left(-\beta U_k\right) \end{cases}$$

□

3.3 Quelques cas polynômiaux du DNL

Nous présentons quelques instances du DNL qui se résolvent en temps polynômial.

3.3.1 Le DNL où tous les arcs du réseau sont taxables

Notons par n le nombre total d'arcs distincts des chemins p_0^k, $k \in \mathcal{K}$. Tel que présenté pour le DL, on montre que :

Proposition 3.3.1 *Le DNL où tous les arcs du réseau sont taxables se résout en* $\mathcal{O}(|\mathcal{K}|^2 n)$

si le système linéaire suivant

$$\sum_{a \in p_0^k} t_a \;=\; T_k^* \qquad\qquad \textit{pour tout } k \in \mathcal{K} \tag{3.1}$$

possède au moins une solution où T_k^* est défini à la section 3.2 ci-dessus.

Preuve La preuve est similaire à celle de la proposition 2.5.1.

□

Telque présenté dans le cas DL, lorsque $|\mathcal{K}| \leq 2$, le système (3.1) est linéairement indépendant et le problème se résout en $\mathcal{O}(n)$. Cependant lorsque $|\mathcal{K}| \geq 3$ ce système ne possède pas toujours une solution.

En restreignant les tarifs sur les arcs à ne prendre que des valeurs positives, on montre comme dans le cas DL que :

Corollaire 3.3.2 *Le DNL monoproduit où tous les arcs du réseau sont taxables et les tarifs positifs se résout en $\mathcal{O}(n)$.*

Preuve Il suffit de passer le flot sur le chemin p_0^1 pour obtenir la solution optimale en imposant les tarifs sur les arcs de la façon suivante :

$$\sum_{a \in p_0^1} t_a = T_1^*$$

$$t_a \;=\; +\infty \qquad\qquad \forall\ a \notin p_0^1$$

Le revenu généré est maximal avec cette affectation car la borne supérieure sur le revenu du meneur proposée à la section 3.2 est atteinte.

□

3.3.2 Le DNL avec un seul arc taxable

On a montré que le DL avec un seul arc taxable est polynômial. Pour le DNL, on montre qu'il est polynômial pour le cas monoproduit. À partir du corollaire 3.3.2, on déduit que

la solution optimale du problème monoproduit est obtenue en imposant à l'arc taxable t la valeur T_1^*.

Dans le cas multi-produits, on a montré pour le DL que la solution optimale est obtenue en résolvant $|\mathcal{K}|$ sous-problèmes de maximisation d'une somme de fonctions concaves sous contraintes linéaires où les fonctions concaves sont les fonctions du revenu généré par les produits lorsque l'arc taxable est utilisé. Ces $|\mathcal{K}|$ sous-problèmes sont de complexité polynômiale. Pour le DNL, on montre également que le problème se ramène à la résolution de $|\mathcal{K}|$ sous-problèmes de maximisation d'une somme de fonctions pseudo-concaves sous contraintes linéaires. Cependant ces $|\mathcal{K}|$ sous-problèmes ne sont pas de complexité polynômiale car la somme de fonctions pseudo-concaves n'est pas pseudo-concave. Les fonctions pseudo-concaves sont les fonctions du revenu généré par les produits lorsque le chemin utilisé contient l'arc taxable.

3.4 Méthode exacte

La méthode exacte possède une phase d'initialisation et une phase itérative. À la phase d'initialisation, à partir de la formulation DNL:CH, nous utilisons une technique de discrétisation pour surapproximer (resp. sous-approximer) la fonction non linéaire $d_k(U_k)U_k$ (resp. $d_k(U_k)$) pour tout produit k, afin d'obtenir un programme linéaire en variables mixtes (MIP). À la phase itérative, ce MIP est résolu et à partir de la solution obtenue nous ajoutons des coupes au MIP dans le but d'améliorer la surapproximation (resp. sous-approximation) de $d_k(U_k)U_k$ (resp. $d_k(U_k)$). La méthode résultante est appelée EXACT-1. Notons que ces coupes sont inspirées de celles développées par Audet [2] et Audet *et al.* [3].

3.4.1 Phase d'initialisation

La phase d'initialisation procède de la manière suivante. À partir de la formulation DNL:CH, nous remplaçons les fonctions non linéaires $d_k(U_k)$ et $d_k(U_k)U_k$ par les variables Y_k et W_k respectivement. Par la suite, nous ajoutons à DNL:CH un ensemble de contraintes

linéaires en variables mixtes qui lient les variables U_k et Y_k puis U_k et W_k afin de mieux surapproximer (resp. sous-approximer) la fonction non linéaire $d_k(U_k)U_k$ (resp. $d_k(U_k)$). Ces contraintes sont ajoutées de manière à borner supérieurement les fonctions $d_k(U_k)$ et $d_k(U_k)U_k$ dans la fonction objectif. Il faut noter que $d_k(U_k)$ apparaît dans l'objectif avec un coefficient négatif c'est pour cela que nous la sous-approximons.

De manière formelle, la convexité stricte de la fonction $d_k(U_k)$ permet d'obtenir une sous-approximation de $d_k(U_k)$ en ajoutant à la formulation DNL:CH les contraintes linéaires suivantes

$$Y_k \geq a_k^l + b_k^l U_k \quad \forall\ l = 1, \ldots, L_k \tag{3.2}$$

où a_k^l et b_k^l sont les coefficients de la tangente à la fonction $d_k(U_k)$ au point U_k^l, et U_k^l un point de la discrétisation de l'intervalle $[\gamma^k(0),\ \gamma^k(\infty)]$ en L_k points. L'intervalle $[\gamma^k(0),\ \gamma^k(\infty)]$ est l'intervalle des valeurs de U_k. Le domaine réalisable du couple (U_k, Y_k) est le polyèdre constitué des tangentes à fonction $d_k(U_k)$ aux points U_k^l pour tout $l = 1, \ldots, L_k$ (voir figure 3.3 ci-dessous).

La fonction $d_k(U_k)$ apparaît dans la fonction objectif avec un coefficient négatif et puisque l'on maximise la fonction objectif, à l'optimalité le couple (U_k, Y_k) se trouve sur la frontière du polyèdre qui définit son domaine réalisable.

En remplaçant $d_k(U_k)$ par la variable Y_k, l'expression $d_k(U_k) \sum_{p \in P_k} h_p^k \sum_{a \in p} c_a$ de la fonction objectif se réécrit $Y_k \sum_{p \in P_k} h_p^k \sum_{a \in p} c_a = \sum_{p \in P_k} Y_k h_p^k \sum_{a \in p} c_a$. On note la présence du terme bilinéaire $Y_k h_p^k$. Pour le rendre linéaire, nous introduisons la variable Y_k^p qui remplace $Y_k h_p^k$, puis nous ajoutons au modèle les contraintes linéaires suivantes qui assurent l'égalité $Y_k^p = Y_k h_p^k$ (voir

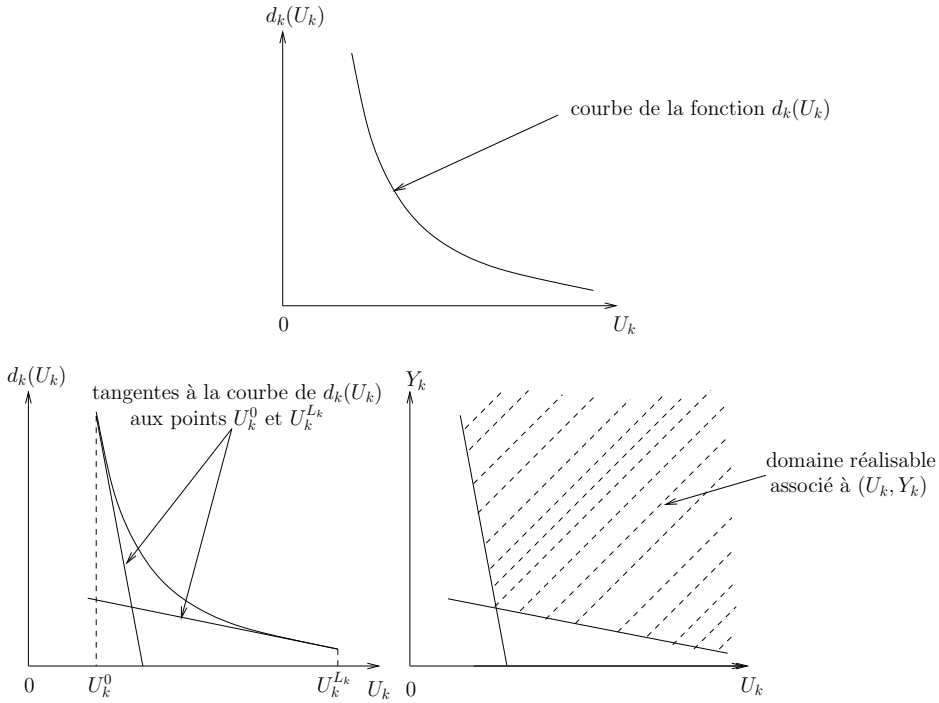


Figure 3.3 Sous-approximation de la fonction $d_k(U_k)$

section 2.2.2)

$$0 \ \leq \ Y_k^p \leq Dh_p^k \qquad \forall p \in P_k \tag{3.3}$$

$$Y_k - \sum_{p \in P_k} Y_k^p = 0 \tag{3.4}$$

où D est une constante suffisamment grande.

Pour surapproximer $d_k(U_k)U_k$, trois cas se posent (voir figure 3.2). Si $d_k(U_k)U_k$ est pseudo-concave, pour la surapproximer, nous supposons qu'elle est concave sur l'intervalle $[\gamma^k(0),\ \tilde{\gamma}^k]$ et convexe sur l'intervalle $[\tilde{\gamma}^k,\ \gamma^k(\infty)]$ où $\tilde{\gamma}^k$ est compris entre $\gamma^k(0)$ et $\gamma^k(\infty)$. La surapproximation de la partie concave se fait par l'ajout à DNL:CH des contraintes linéaires suivantes

$$W_k \leq a_k^i + b_k^i U_k + \overline{M}_k s_k \qquad \forall\ i = 1, \ldots, I_k \tag{3.5}$$

$$U_k \leq \tilde{\gamma}^k(1 - s_k) + M s_k \tag{3.6}$$

$$s_k \in \{0, 1\} \tag{3.7}$$

où $M = \gamma^k(\infty)$, $\overline{M}_k$ une constante suffisamment grande égale à $d_k(\gamma^k(0))\gamma^k(0)$ si l'élasticité est constante et $d_k(\overline{U}_k)\overline{U}_k$ si l'élasticité est non constante où $\overline{U}_k$ est la valeur qui annule la dérivée de $d_k(U_k)U_k$, s_k la variable binaire égale à 0 si on se trouve sur la partie concave et 1 sinon, a_k^i et b_k^i les coefficients de la tangente à la fonction $d_k(U_k)U_k$ au point U_k^i et U_k^i un point de la discrétisation de l'intervalle $[\gamma^k(0),\ \tilde{\gamma}_k]$ en I_k points (voir figure 3.4).

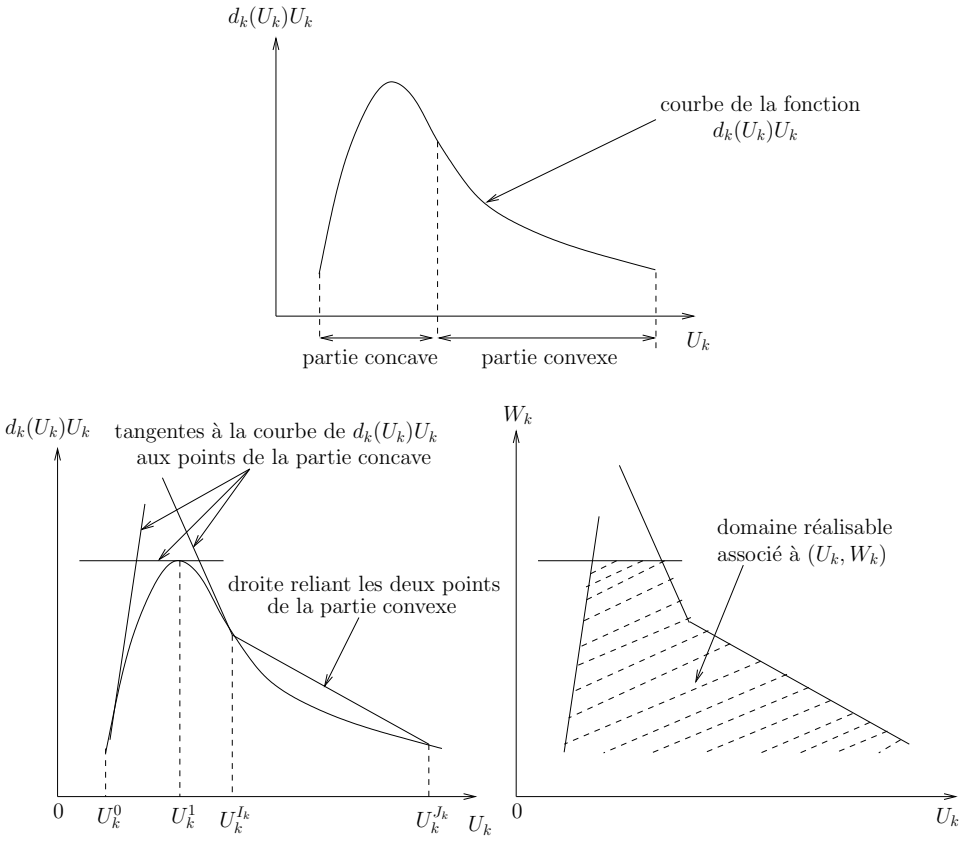


Figure 3.4 Surapproximation de la fonction $d_k(U_k)U_k$

Les contraintes (3.5) sont valides lorsque $s_k = 0$ c'est-à-dire que l'on se trouve dans la partie concave, et deviennent redondantes lorsque $s_k = 1$ car la contrainte devient $U_k \leq \gamma^k(\infty)$ et est toujours valide. La contrainte (3.6) impose que U_k appartienne à la partie concave si $s_k = 0$. Le domaine réalisable du couple (U_k, W_k) sur cet intervalle est le polyèdre constitué des tangentes à la fonction $d_k(U_k)U_k$ aux points U_k^i pour tout $i = 1, \ldots, I_k$. À l'optimalité, le couple (U_k, W_k) se trouve sur la frontière du polyèdre qui définit son domaine réalisable.

Pour surapproximer la fonction $d_k(U_k)U_k$ sur la partie convexe, nous ajoutons à DNL:CH

un ensemble de contraintes linéaires par morceaux, ce qui entraine l'introduction des variables binaires. L'intervalle $[\tilde{\gamma}^k,\ \gamma^k(\infty)]$ est divisé en plusieurs sous-intervalles $[U_k^j,\ U_k^{j+1}]$ pour $j = 1, \ldots, J_k - 1$ et à chaque sous-intervalle on associe une variable binaire qui indique si U_k appartient à cet sous-intervalle ou pas. Soit z_k^j la variable binaire qui est égale à 1 si U_k appartient au sous-intervalle $[U_k^j,\ U_k^{j+1}]$ et à 0 sinon. La contrainte linéaire correspondante au sous-intervalle $[U_k^j,\ U_k^{j+1}]$ est la droite passant par les points $\left(U_k^j, d_k(U_k^j)U_k^j\right)$ et $\left(U_k^{j+1}, d_k(U_k^{j+1})U_k^{j+1}\right)$. Les contraintes ajoutées à DNL:CH pour surapproximer $d_k(U_k)U_k$ sur la partie convexe se présentent comme suit

$$W_k \leq a_k^j + b_k^j U_k + \overline{M}_k(1 - z_k^j) \qquad \forall\ j = 1, \ldots, J_k - 1 \tag{3.8}$$

$$U_k \geq U_k^j z_k^j \qquad \forall\ j = 1, \ldots, J_k - 1 \tag{3.9}$$

$$U_k \leq U_k^{j+1} z_k^j + \overline{N}_k^j(1 - z_k^j) \qquad \forall\ j = 1, \ldots, J_k - 1 \tag{3.10}$$

$$z_k^j\ \in\ \{0, 1\} \qquad \forall\ j = 1, \ldots, J_k - 1 \tag{3.11}$$

$$\sum_{j=1}^{J_k - 1} z_k^j = s_k \tag{3.12}$$

où $\overline{N}_k^j = \gamma^k(\infty) - U_k^{j+1}$, a_k^j et b_k^j les coefficients de la droite passant par les points $\left(U_k^j, d_k(U_k^j)U_k^j\right)$ et $\left(U_k^{j+1}, d_k(U_k^{j+1})U_k^{j+1}\right)$ et J_k le nombre de points de la discrétisation de l'intervalle $[\tilde{\gamma}_k,\ \gamma^k(\infty)]$.

Les contraintes (3.8) assurent que α_k surapproxime $d_k(U_k)U_k$ sur chaque sous-intervalle de la partie convexe. Les contraintes (3.8) sont redondantes lorsque U_k appartient à la partie concave car $s_k = 0$. Les contraintes (3.9) à (3.12) permettent le respect de la définition de la variable z_k^j. Le domaine réalisable du couple (U_k, W_k) est défini par le polyèdre constitué des contraintes linéaires par morceaux sur chaque sous-intervalle $[U_k^j,\ U_k^{j+1}]$ pour tout $j = 1, \ldots, J_k - 1$ (voir figure 3.4 ci-dessus). Puisque l'on maximise la fonction objectif, à l'optimalité le couple (U_k, W_k) se trouve sur la frontière du polyèdre qui définit son domaine réalisable.

Si $d_k(U_k)U_k$ est concave (resp. convexe), la technique de surapproximation est la même que celle utilisée pour surapproximer $d_k(U_k)U_k$ lorsqu'elle est pseudo-concave car il suffit de

considérer qu'il n'existe pas de partie convexe (resp. concave) et que la partie concave (resp. convexe) est l'intervalle des valeurs de U_k.

La surapproximation (resp. sous-approximation) de la fonction $d_k(U_k)U_k$ (resp. $d_k(U_k)$)

pour tout produit k nous mène alors au modèle MIP suivant :

DNL:App

$$\max_{t,h,T,U,W,Y,z,s} \sum_{k\in\mathcal{K}} \left(W_k - \sum_{p\in P_k} Y_k^p \sum_{a\in p} c_a \right)$$

$$\begin{aligned}
\text{s.c} \quad & T_k \le \sum_{a\in p\cap\mathcal{A}_1} t_a + M_p^k(1-h_p^k) && \forall p\in P_k,\ \forall k\in\mathcal{K}\\
& \sum_{a\in p} c_a + \sum_{a\in p\cap\mathcal{A}_1} t_a - M_p^k(1-h_p^k) \le U_k && \forall p\in P_k,\ \forall k\in\mathcal{K}\\
& U_k \le \sum_{a\in p} c_a + \sum_{a\in p\cap\mathcal{A}_1} t_a && \forall p\in P_k,\ \forall k\in\mathcal{K}\\
& U_k = T_k + \sum_{p\in P_k} h_p^k \sum_{a\in p} c_a && \forall k\in\mathcal{K}\\
& \sum_{p\in P_k} h_p^k = 1 && \forall k\in\mathcal{K}\\
& Y_k \ge a_k^l + b_k^l U_k && \forall l=1,\dots,L_k,\ \forall k\in\mathcal{K}\\
& 0 \ \le\ Y_k^p \le D h_p^k && \forall p\in P_k,\ \forall k\in\mathcal{K}\\
& Y_k - \sum_{p\in P_k} Y_k^p = 0 && \forall k\in\mathcal{K}\\
& W_k \le a_k^i + b_k^i U_k + M s_k && \forall\ i=1,\dots,I_k,\ \forall k\in\mathcal{K}\\
& U_k \le \tilde{\gamma}^k(1-s_k) + M s_k && \forall k\in\mathcal{K}\\
& W_k \le a_k^j + b_k^j U_k + M(1-z_k^j) + N(1-s_k) && \forall l=1,\dots,J_k-1,\ \forall k\in\mathcal{K}\\
& U_k \ge U_k^j z_k^j && \forall l=1,\dots,J_k-1,\ \forall k\in\mathcal{K}\\
& U_k \le U_k^{j+1} z_k^j + M(1-z_k^j) && \forall l=1,\dots,J_k-1,\ \forall k\in\mathcal{K}\\
& \sum_{j=1}^{J_k-1} z_k^j = s_k && \forall k\ \in\mathcal{K}\\
& h_p^k \in \{0,1\} && \forall p\in P_k,\ \forall k\in\mathcal{K}\\
& s_k\ \in\ \{0,1\} && \forall k\in\mathcal{K}\\
& z_k^j\ \in\ \{0,1\} && \forall\ j=1,\dots,J_k-1,\ \forall k\in\mathcal{K}.
\end{aligned}$$

DNL:App est un programme linéaire mixte (MIP) qui approxime DNL:CH. La valeur de la fonction objectif du MIP est une borne supérieure du DNL car W_k (resp. Y_k) est une surapproximation de la fonction $d_k(U_k)U_k$ (resp. $d_k(U_k)$). De plus, à partir d'une solution réalisable du MIP, on peut déduire une solution réalisable du DNL. Soit $(t^*, h^*, T^*, U^*, W^*, Y^*, z^*, s^*)$ une solution réalisable du MIP. La solution (t^*, h^*, T^*, U^*) est réalisable pour DNL:CH car toutes ses contraintes sont présentes dans le MIP. La valeur objectif de la solution réalisable (t^*, h^*, T^*, U^*) est une borne inférieure du DNL.

3.4.2 Phase itérative

La phase itérative résout à chaque itération le modèle DNL:App puis ajoute des contraintes à ce modèle afin d'améliorer la surapproximation (resp. sous-approximation) de $d_k(U_k)U_k$ (resp. $d_k(U_k)$) pour tout produit k. Soit $(t^*, h^*, T^*, U^*, z^*, W^*, Y^*, s^*)$ la solution obtenue à partir du modèle DNL:App. Si la surapproximation (resp. sous-approximation) de $d_k(U_k)U_k$ (resp. $d_k(U_k)$) obtenue à partir de cette solution n'est pas satisfaisante, c'est-à-dire l'un des écarts $|d_k(U_k^*) - Y_k^*|$ et $|d_k(U_k^*)U_k^* - W_k^*|$ est supérieur à un seuil donné ε pour au moins un produit k, alors un ensemble de contraintes est ajouté au modèle DNL:App et ce dernier est résolu de nouveau. Le but de l'ajout de ces contraintes est d'éliminer la solution courante tout en améliorant les approximations des fonctions $d_k(U_k)$ et $d_k(U_k)U_k$.

Ainsi, si $|d_k(U_k^*) - Y_k^*| > \varepsilon$ alors la contrainte linéaire suivante est ajoutée au modèle DNL:App

$$Y_k \geq a_k + b_k U_k \tag{3.13}$$

où a_k et b_k sont les coefficients de la tangente à la fonction $d_k(U_k)$ au point U_k^* (voir figure 3.5).

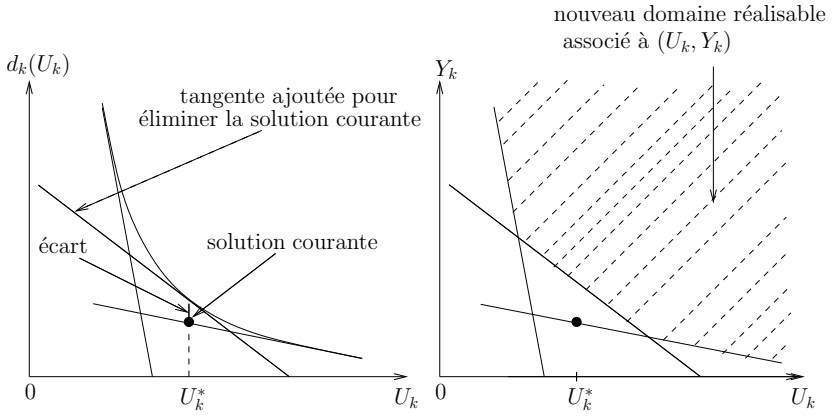


Figure 3.5 Amélioration de la sous-approximation de la fonction $d_k(U_k)$

Si $|d_k(U_k^*)U_k^* - W_k^*| > \varepsilon$, deux cas se présentent :

Cas 1 : U_k^* appartient à la partie concave, alors la contrainte linéaire suivante

$$W_k \leq a_k + b_k U_k + M s_k \tag{3.14}$$

est ajoutée à DNL:App où a_k et b_k sont les coefficients de la tangente à la fonction $d_k(U_k)U_k$ au point U_k^* et s_k la variable binaire égale 0 si U_k appartient à la partie concave et 1 sinon (voir figure 3.6).

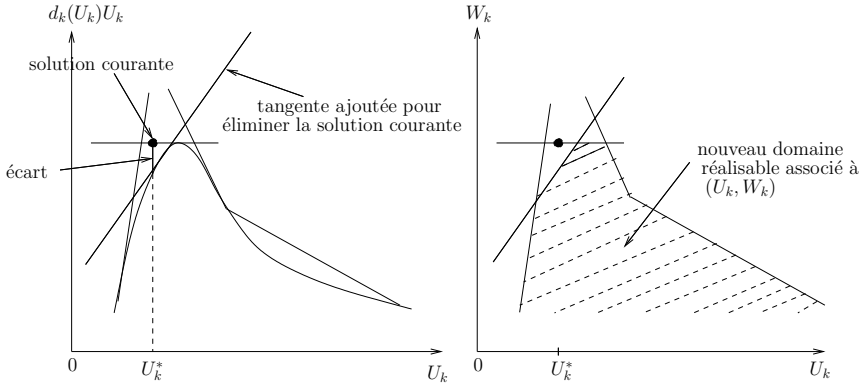


Figure 3.6 Amélioration de la surapproximation de la fonction $d_k(U_k)U_k$ sur la partie concave

Cas 2 : U_k^* appartient à la partie convexe et supposons que U_k^* appartient au sous-intervalle $[U_k^j, U_k^{j+1}]$. Pour éliminer la solution courante, l'intervalle $[U_k^j, U_k^{j+1}]$ est divisé en deux sous-

intervalles $[U_k^j, U_k^*]$ et $[U_k^*, U_k^{j+1}]$. On associe une variable binaire z_k^1 (resp. z_k^2) au sous-intervalle $[U_k^j, U_k^*]$ (resp. $[U_k^*, U_k^{j+1}]$). Cette variable binaire indique si U_k appartient à cet sous-intervalle ou pas. À chaque sous-intervalle, la contrainte linéaire définie sur ce sous-intervalle est donnée par la droite qui passe par les deux points ayant pour abscisses les bornes du sous-intervalle et pour ordonnées les valeurs de la fonction $d_k(U_k)U_k$ à ces bornes (voir figure 3.7).

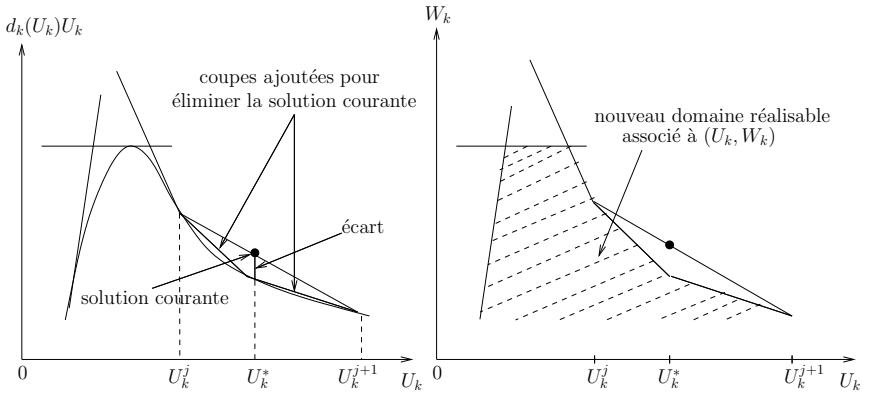


Figure 3.7 Amélioration de la surapproximation de la fonction $d_k(U_k)U_k$ sur la partie convexe

Les contraintes linéaires suivantes

$$W_k \leq a_k^1 + b_k^1 U_k + M(1 - z_k^1) + N(1 - s_k) \tag{3.15}$$

$$U_k \geq U_k^j z_k^1 \tag{3.16}$$

$$U_k \leq U_k^* z_k^1 + M(1 - z_k^1) \tag{3.17}$$

$$W_k \leq a_k^2 + b_k^2 U_k + M(1 - z_k^2) + N(1 - s_k) \tag{3.18}$$

$$U_k \geq U_k^* z_k^2 \tag{3.19}$$

$$U_k \leq U_k^{j+1} z_k^2 + M(1 - z_k^2) \tag{3.20}$$

$$z_k^1, z_k^2 \ \in \ \{0, 1\} \tag{3.21}$$

sont ajoutées à DNL:App où M, N sont des constantes suffisamment grandes, a_k^1 et b_k^1 (resp. a_k^2 et b_k^2) les coefficients de la droite passant par les points $\left(U_k^j, d_k(U_k^j)U_k^j\right)$ et $(U_k^*, d_k(U_k^*)U_k^*)$

(resp. $(U_k^*, d_k(U_k^*)U_k^*)$ et $\left(U_k^{j+1}, d_k(U_k^{j+1})U_k^{j+1}\right)$) (voir contraintes (3.15) et (3.18)). Les contraintes (3.16) et (3.17) (resp. (3.19) et (3.20)) permettent de fixer z_k^1 (resp. z_k^2) à 1 si U_k appartient à l'intervalle $[U_k^j, U_k^*]$ (resp. $[U_k^*, U_k^{j+1}]$) et 0 sinon.

La sous-approximation de la fonction $d_k(U_k)$ ne nécessite pas l'ajout des variables binaires, ce qui n'augmente donc pas de manière significative la difficulté du problème. Par contre, la surapproximation de la partie convexe de la fonction $d_k(U_k)U_k$ nécessite l'ajout des variables binaires sur cette partie, ce qui augmente l'aspect combinatoire du problème car en plus du choix de l'ensemble de chemins qui génère un revenu optimal, il faut détecter dans la partie convexe le sous-intervalle où appartient l'utilité U_k qui correspond au revenu optimal. Nous proposons une autre méthode exacte appelée EXACT-2 qui consiste à réécrire la fonction $d_k(U_k)U_k$ de telle sorte que la nouvelle fonction à surapproximer ait une partie convexe plus étroite afin de diminuer la difficulté associée à cette surapproximation. Pour ce faire, nous introduisons la variable Γ_k et nous imposons

$$U_k = \Gamma_k + \sum_{a \in p_0^k} c_a \tag{3.22}$$

où p_0^k est le chemin de plus petit coût fixe du produit k. En fait, Γ_k correspondant au revenu généré par une unité de flot du chemin p_0^k. Puis, nous réécrivons la fonction objectif du produit

k comme suit :

$$
\begin{aligned}
& d_k(U_k)U_k - d_k(U_k) \sum_{p \in P_k} \sum_{a \in p} c_a h_p^k \\
& = d_k(U_k) \left(\Gamma_k + \sum_{a \in p_0^k} c_a \right) - d_k(U_k) \sum_{p \in P_k} h_p^k \sum_{a \in p} c_a \\
& = d_k(U_k)\Gamma_k + d_k(U_k) \sum_{a \in p_0^k} c_a - d_k(U_k) \sum_{p \in P_k} h_p^k \sum_{a \in p} c_a \\
& = d_k \left(\Gamma_k + \sum_{a \in p_0^k} c_a \right) \Gamma_k + d_k(U_k) \left[\sum_{p \in P_k} h_p^k \sum_{a \in p_0^k} c_a - \sum_{p \in P_k} h_p^k \sum_{a \in p} c_a \right] \qquad \text{car} \quad \sum_{p \in P_k} h_p^k = 1 \\
& = d_k \left(\Gamma_k + \sum_{a \in p_0^k} c_a \right) \Gamma_k - d_k(U_k) \sum_{p \in P_k} h_p^k \left(\sum_{a \in p} c_a - \sum_{a \in p_0^k} c_a \right). \qquad (3.23)
\end{aligned}
$$

La deuxième méthode exacte, EXACT-2, est basée sur le programme mathématique obtenu en remplaçant dans DNL:CH la fonction objectif du produit k par (3.23) et en ajoutant la contrainte (3.22). Pour la méthode EXACT-2, la procédure consiste à surapproximer les fonctions $-d_k(U_k)$ et $d_k \left(\Gamma_k + \sum_{a \in p_0^k} c_a \right) \Gamma_k$. Les surapproximations sont identiques à celles présentées ci-dessus. Notons que le coefficient de $d_k(U_k)$ dans (3.23) est négatif car p_0^k étant le chemin de plus petit coût fixe du produit k, $\sum_{a \in p} c_a - \sum_{a \in p_0^k} c_a \geq 0$ pour tout $p \in P_k$.

Les longueurs de l'intervalle des valeurs de U_k et Γ_k sont égales, la partie convexe de $d_k(U_k)U_k$ est plus large que celle de $d_k \left(\Gamma_k + \sum_{a \in p_0^k} c_a \right) \Gamma_k$. De plus, $d_k \left(\Gamma_k + \sum_{a \in p_0^k} c_a \right) \Gamma_k$ borne inférieurement $d_k(U_k)U_k$. Il en résulte que la surapproximation de $d_k \left(\Gamma_k + \sum_{a \in p_0^k} c_a \right) \Gamma_k$ augmente moins l'aspect combinatoire du problème comparée à celle de $d_k(U_k)U_k$ pour une discrétisation identique de l'intervalle des valeurs de U_k et Γ_k. Il s'ensuit que la méthode EXACT-2 aura un temps d'exécution inférieur à celui de la méthode EXACT-1.

3.4.3 Algorithme exact

Trois paramètres d'entrée sont nécessaires : la tolérance ε sur l'approximation des fonctions $d_k(U_k)$ et $d_k(U_k)U_k$ (resp. $d_k\left(\Gamma_k + \sum_{a \in p_0^k} c_a\right)\Gamma_k$) de la méthode EXACT-1 (resp. EXACT-2), le nombre de points L_k de la discrétisation de l'intervalle des valeurs de U_k pour la sous-approximation de la fonction $d_k(U_k)$ et le nombre de points I_k (resp. J_k) de la discrétisation de l'intervalle des valeurs de U_k sur la partie concave (resp. convexe) pour la surapproximation de la fonction $d_k(U_k)U_k$ ou $d_k\left(\Gamma_k + \sum_{a \in p_0^k} c_a\right)\Gamma_k$. L'algorithme $\varepsilon-$exact se présente comme suit :

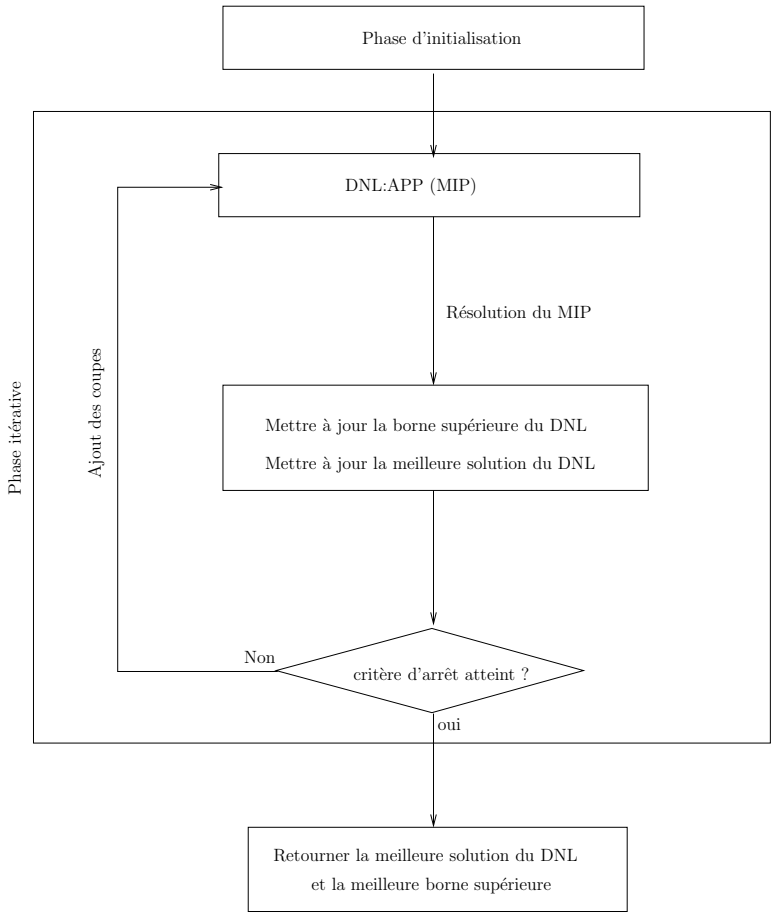


Figure 3.8 Algorithme exact

3.5 Optimisation inverse et amélioration de la méthode exacte

Dans la méthode exacte, après la résolution du MIP, DNL:App, nous déduisons une solution réalisable du DNL. Cependant, les tarifs de cette solution ne génèrent pas toujours le meilleur revenu correspondant aux chemins utilisés. Pour remédier à cette situation, nous supposons connu les chemins utilisés pour acheminer les produits et nous déterminons les tarifs sur les arcs qui maximisent le revenu. Ce problème est connu sous le nom de problème d'optimisation inverse (POI).

3.5.1 Optimisation inverse

Soit p_k^* le chemin utilisé par les usagers pour le produit k c'est-à-dire que $h^k_{p_k^*} = 1$ et $h^k_p = 0$ pour tout $p \in P_k \backslash \{p_k^*\}$. L'utilité U_k du produit k est donnée par $U_k = T_k + \sum_{a \in p_k^*} c_a$ et le revenu généré par ce produit k s'écrit alors

$$d_k(U_k)T_k = d_k\left(\sum_{a \in p_k^*} c_a + T_k\right) T_k.$$

En considérant la formulation par chemins DNL:CH (voir section 3.1) et en supposant que la solution du problème du suiveur est connue, nous obtenons la formulation du POI suivante

OPTINV

$$\max_{t,T,U} \sum_{k \in \mathcal{K}} d_k\left(\sum_{a \in p_k^*} c_a + T_k\right) T_k \tag{3.24}$$

$$\text{s.c} \quad T_k = \sum_{a \in p_k^* \cap \mathcal{A}_1} t_a \qquad \forall k \in \mathcal{K} \tag{3.25}$$

$$U_k \leq \sum_{a \in p} c_a + \sum_{a \in p \cap \mathcal{A}_1} t_a \qquad \forall p \in P_k,\ \forall k \in \mathcal{K} \tag{3.26}$$

$$U_k = T_k + \sum_{a \in p_k^*} c_a \qquad \forall k \in \mathcal{K}. \tag{3.27}$$

Les contraintes (3.25) assurent que le revenu généré par unité de flot provient du tarif des arcs taxables des chemins utilisés. Les contraintes (3.26) imposent que les chemins utilisés soient les plus courts. Les contraintes (3.27) assurent que les utilités correspondent aux coûts des chemins utilisés. La fonction objectif (3.24) est une somme de fonctions pseudo-concaves car chaque fonction $d_k \left(\sum_{a \in p_k^*} c_a + T_k \right) T_k$ qui dépend de T_k est pseudo-concave. Le problème OPTINV est donc la maximisation d'une somme de fonctions pseudo-concaves sous contraintes linéaires.

Pour résoudre OPTINV, la technique utilisée est la même que celle développée pour les méthodes exactes du DNL, c'est-à-dire que nous débutons par une phase d'initialisation qui consiste à surapproximer les fonctions $d_k \left(\sum_{a \in p_k^*} c_a + T_k \right) T_k$ du modèle OPTINV afin d'obtenir un programme linéaire mixte (MIP), suivie d'une phase itérative qui à chaque itération résout ce MIP, et à partir de la solution obtenue ajoute des coupes au MIP dans le but d'éliminer la solution courante et améliorer la surapproximation. La surapproximation de la fonction $d_k \left(\sum_{a \in p_k^*} c_a + T_k \right) T_k$ se fait de la même manière que celle de la fonction $d_k(U_k)U_k$ présentée à la section 3.4.

Notons que le temps de résolution du POI est négligeable comparé à celui du DNL car la seule difficulté liée à la résolution du OPTINV est la surapproximation des fonctions pseudo-concaves. Quant à la résolution du DNL:CH, en plus de cette surapproximation, il faut déterminer les chemins utilisés pour acheminer les produits, ce qui augmente l'aspect combinatoire du problème.

3.5.2 Amélioration de la méthode exacte

Supposons qu'à une itération donnée, l'ensemble des chemins utilisés par la solution optimale du modèle surapproximé DNL:App est $\{p_k^*,\ k \in \mathcal{K}\}$. Nous résolvons le POI où l'ensemble des chemins utilisés est $\{p_k^*,\ k \in \mathcal{K}\}$. Si la solution obtenue a un revenu strictement inférieur au revenu de la meilleure solution trouvée du DNL, alors nous interdisons toute solution dont

l'ensemble des chemins utilisés est $\{p_k^*,\ k \in \mathcal{K}\}$ en ajoutant à DNL:App la coupe suivante

$$\sum_{k \in \mathcal{K}} h^k_{p_k^*} \ \leq \ |\mathcal{K}| - 1. \tag{3.28}$$

Cette coupe évite l'utilisation simultanée des chemins $\cup_{k \in \mathcal{K}} \{p_k^*\}$ dont le revenu généré est inférieur à celui du meilleur revenu courant. Elle réduit également l'ensemble réalisable des variables de flots tout en encourageant l'exploration d'autres solutions réalisable dans le but de trouver des solutions de meilleur revenu. Nous remarquons dans les tests numériques que cette coupe est peu ajoutée à DNL:App car la solution réalisable trouvée à l'itération courante génère un revenu supérieur ou égal au meilleur revenu obtenu aux itérations précédentes. Cependant, lorsqu'elle est utilisée, elle fait décroître de manière considérable la borne supérieure.

De plus, si le revenu de la solution trouvée avec le POI est supérieur au revenu de la meilleure solution trouvée du DNL, alors nous retenons cette solution comme la nouvelle meilleure solution du DNL.

3.6 Heuristiques

Les heuristiques proposées sont basées sur une méthode de région de confiance. Une méthode de région de confiance est un procédé itératif, qui au contraire d'une recherche linéaire qui explore la fonction objectif dans une direction donnée, explore toutes les directions autorisées à partir de la solution courante en limitant la taille du pas. En limitant la taille du pas, on se retrouve à explorer une région bien précise (dite de « confiance »). La fonction ainsi explorée est un *modèle* de la fonction objectif. La méthode opère plutôt sur le modèle et valide ensuite le résultat sur la fonction objectif. Le principe des méthodes de région de confiance est de remplacer le problème d'optimisation initial par un modèle plus simple. Pour plus d'informations sur les méthodes de région de confiance, nous renvoyons le lecteur à Conn *et al.* [20].

L'heuristique procède comme suit : à partir d'une solution courante donnée, nous construisons le modèle, puis nous le résolvons ; si la solution obtenue est bonne, c'est-à-dire si la fonction objectif du modèle décrit correctement le comportement de la fonction objectif de DNL:CH, la région de confiance peut être augmentée afin de déterminer des solutions plus performantes : l'étape est dans ce cas un succès. La nouvelle solution courante du DNL est déduite de la solution fournie par le modèle. Dans le cas contraire c'est-à-dire quand le modèle ne prédit pas convenablement le comportement de la fonction objectif de DNL:CH, on dit que l'étape a échoué, la région de confiance est restreinte, et la solution courante reste inchangée. Le processus est répété jusqu'à la satisfaction d'un critère d'arrêt. La meilleure solution du DNL est retenue au cours du processus.

Le problème d'optimisation initial, dans ce cas, est DNL:CH où la fonction objectif est non linéaire. Le modèle est obtenu en approximant la fonction objectif autour d'une région de confiance. La première approximation se fait en linéarisant les fonctions de demande autour de la solution courante, ce qui nous mène à l'heuristique 1. La deuxième approximation se fait en linéarisant la fonction objectif autour de la solution courante, ce qui nous mène à l'heuristique 2.

3.6.1 Définition du modèle de l'heuristique 1

Soit $(\overline{t}, \overline{h}, \overline{U}, \overline{T})$ une solution courante de DNL:CH, en linéarisant la fonction demande $d_k(U_k)$ de chaque produit k autour de $\overline{U}_k$, nous obtenons une approximation de la fonction demande $d_k(U_k) \approx a_k - b_k U_k$ où a_k et b_k sont les coefficients de la tangente à la fonction $d_k(U_k)$ au point $\overline{U}_k$. Dans un premier temps, nous remplaçons dans la fonction objectif de DNL:CH, chaque fonction $d_k(U_k)$ par $a_k - b_k U_k$. Puis, nous ajoutons la contrainte $\overline{t} - \Delta t \leq t \leq \overline{t} + \Delta t$ où Δt est le rayon de la région de confiance. Cette contrainte permet aux tarifs des arcs taxables de rester proches de ceux de la solution $(\overline{t}, \overline{h}, \overline{U}, \overline{T})$ afin d'obtenir une approximation acceptable de la fonction objectif réelle. Le modèle obtenu est donc un DL autour d'une région de confiance. Nous avons développé des programmes quadratiques mixtes

sous contraintes linéaires pour la résolution du DL au chapitre 2. Parmi ces programmes, nous utilisons DL:CH2 pour résoudre ce modèle car son temps de calcul et sa qualité de solution sont meilleurs que ceux des deux autres MIPs.

Le modèle de la méthode de région de confiance de l'heuristique 1 se formule alors comme suit :

QUAD

$$\max_{t,h,T,U} \sum_{k\in\mathcal{K}} \left(a_k T_k - b_k T_k^2 - b_k \sum_{p\in P_k} T_k^p \sum_{a\in p} c_a \right) \tag{3.29}$$

$$\text{s.c} \quad \sum_{a\in p} c_a + \sum_{a\in p\cap\mathcal{A}_1} t_a - M_p^k(1-h_p^k) \le U_k \qquad \forall p\in P_k,\ \forall k\in\mathcal{K} \tag{3.30}$$

$$U_k \le \sum_{a\in p} c_a + \sum_{a\in p\cap\mathcal{A}_1} t_a \qquad \forall p\in P_k,\ \forall k\in\mathcal{K} \tag{3.31}$$

$$U_k = T_k + \sum_{p\in P_k} h_p^k \sum_{a\in p} c_a \qquad \forall k\in\mathcal{K} \tag{3.32}$$

$$\sum_{p\in P_k} h_p^k = 1 \qquad \forall k\in\mathcal{K} \tag{3.33}$$

$$T_k - \sum_{p\in P_k} T_k^p = 0 \qquad \forall\mathcal{K} \tag{3.34}$$

$$0 \le T_k^p \le M_p^k h_p^k \qquad \forall p\in P_k,\ \forall k\in\mathcal{K} \tag{3.35}$$

$$h_p^k \in \{0,1\} \qquad \forall p\in P_k,\ \forall k\in\mathcal{K} \tag{3.36}$$

$$\bar{t}_a - \Delta t_a \le t_a \le \bar{t}_a + \Delta t_a \qquad \forall a\in\mathcal{A}_1. \tag{3.37}$$

QUAD est la formulation DL:CH2 présentée à la section 2.2.2 où l'on ajoute la contrainte (3.37) pour délimiter la taille du pas du vecteur de tarifs.

3.6.2 Définition du modèle de l'heuristique 2

Soit $(\overline{t},\overline{h},\overline{U},\overline{T})$ une solution courante de DNL:CH et $d_k = d_k(U_k)$. La linéarisation de la fonction objectif du produit k autour de $(\overline{U}_k,\overline{T}_k)$ est $d_k(U_k)T_k \approx d_k(\overline{U}_k)\overline{T}_k + d_k(\overline{U}_k)(T_k - \overline{T}_k) + \overline{T}_k(d_k - d_k(\overline{U}_k)) = d_k(\overline{U}_k)T_k + \overline{T}_k d_k - d_k(\overline{U}_k)\overline{T}_k$. Nous remplaçons la fonction objectif

de chaque produit k de DNL:CH par l'approximation $d_k(\overline{U}_k)T_k + \overline{T}_k d_k - d_k(\overline{U}_k)\overline{T}_k$, puis nous ajoutons la contrainte $\bar{t} - \Delta t \leq t \leq \bar{t} + \Delta t$ où Δt est le rayon de la région de confiance. Puisque la variable d_k n'intervient pas dans DNL:CH, pour éviter qu'elle ne prenne une valeur infinie, nous imposons la contrainte $d_k \leq a_k - b_k U_k$ où a_k et b_k sont les coefficients de la tangente à la fonction $d_k(U_k)$ au point $\overline{U}_k$. Ainsi, d_k demeure plus proche de la demande réelle d'autant plus qu'elle apparaît dans la fonction objectif avec un coefficient positif.

Le modèle de la méthode de région de confiance pour l'heuristique 2 se formule alors comme suit :

LIN

$$\max_{t,h,T,U,d} \sum_{k \in \mathcal{K}} \left(d_k(\overline{U}_k)T_k + \overline{T}_k d_k - d_k(\overline{U}_k)\overline{T}_k \right) \tag{3.38}$$

$$\text{s.c} \quad \sum_{a \in p} c_a + \sum_{a \in p \cap \mathcal{A}_1} t_a - M_p^k (1 - h_p^k) \leq U_k \qquad \forall p \in P_k,\ \forall k \in \mathcal{K} \tag{3.39}$$

$$U_k \leq \sum_{a \in p} c_a + \sum_{a \in p \cap \mathcal{A}_1} t_a \qquad \forall p \in P_k,\ \forall k \in \mathcal{K} \tag{3.40}$$

$$U_k = T_k + \sum_{p \in P_k} h_p^k \sum_{a \in p} c_a \qquad \forall k \in \mathcal{K} \tag{3.41}$$

$$\sum_{p \in P_k} h_p^k = 1 \qquad \forall k \in \mathcal{K} \tag{3.42}$$

$$h_p^k \in \{0, 1\} \qquad \forall p \in P_k,\ \forall k \in \mathcal{K} \tag{3.43}$$

$$\bar{t}_a - \Delta t_a \leq t_a \leq \bar{t}_a + \Delta t_a \qquad \forall a \in \mathcal{A}_1 \tag{3.44}$$

$$d_k \leq a_k - b_k U_k \qquad \forall k \in \mathcal{K}. \tag{3.45}$$

LIN est un programme linéaire en variables mixtes (MIP). Les contraintes (3.39) à (3.43) sont identiques à celles de DNL:CH. Les contraintes (3.44) permettent de délimiter la région de confiance afin que l'objectif du LIN soit plus proche de l'objectif réel. Les contraintes (3.45) assurent que les variables d_k ne prennent pas une valeur infinie et soit proche de la demande réelle.

LIN contient $\sum_{k \in \mathcal{K}} |P_k| - |\mathcal{K}|$ variables continues et $\sum_{p \in P_k} |P_k|$ contraintes de moins que QUAD.

3.6.3 Algorithme

Les paramètres d'entrée nécessaires pour l'algorithme sont : une solution de départ du DNL, un rayon de la région de confiance et deux constantes qui permettent de vérifier si le modèle constitue une bonne approximation locale du DNL ou pas, et trois autres constantes qui contrôlent le taux de variation du rayon de la région de confiance. Les solutions de départ sont obtenues en générant de manière aléatoire les tarifs sur les arcs, puis en résolvant le problème du second niveau pour déterminer les plus courts chemins compatibles aux tarifs générés.

HEURISTIQUE

Étape 0 : *Initialisation*

Soit (t^0, h^0, T^0, U^0) la solution de départ du DNL.

Soit Δ_0 le rayon de la région de confiance.

Soient les constantes $\gamma_1 = 0.6$, $\gamma_2 = 0.9$ et $\gamma_3 = 1.6$.

Soient les constantes $\eta_1 = 0.05$ et $\eta_2 = 0.9$

Soit $l = 0$ le compteur d'itérations.

Étape 1 : *Construction du modèle*

Définir le modèle en linéarisant la fonction objectif de DNL:CH autour de la solution (t^l, h^l, T^l, U^l) en utilisant l'une des deux techniques présentées aux sections 3.6.1 et 3.6.2.

Étape 2 : *Résolution du modèle*

Résoudre le modèle (QUAD pour l'heuristique 1 et LIN pour l'heuristique 2). Soit (t^c, h^c, T^c, U^c) la solution obtenue.

Étape 3 : *Évaluation du critère de performance et mise à jour de la solution courante*

Soit f (resp. f_s) la fonction objectif du DNL (resp. modèle). Le quotient ρ qui calcule le critère de performance est donné par

$$\rho = \frac{f(t^l, h^l) - f(t^c, h^c)}{f(t^l, h^l) - f_s(t^c, h^c)}$$

Si $\rho \geq \eta_1$ alors le modèle est adéquat et $(t^{l+1}, h^{l+1}, T^{l+1}, U^{l+1}) = (t^c, h^c, T^c, U^c)$.
Sinon le modèle n'est pas adéquat et $(t^{l+1}, h^{l+1}, T^{l+1}, U^{l+1}) = (t^l, h^l, T^l, U^l)$.
L'exception est faite si la solution (t^c, h^c, T^c, U^c) fournit le meilleur revenu obtenu jusqu'à présent. Dans ce cas, $(t^{l+1}, h^{l+1}, T^{l+1}, U^{l+1}) = (t^c, h^c, T^c, U^c)$.

Étape 4 : *Mise à jour du rayon de la région de confiance*

$$\Delta_{l+1} = \begin{cases} \gamma_1 \Delta_l & \text{si } \rho < \eta_1 \\ \gamma_2 \Delta_l & \text{si } \eta_1 \leq \rho \leq \eta_2 \\ \gamma_3 \Delta_l & \text{si } \rho > \eta_2 \end{cases}$$

Étape 5 : *Critère d'arrêt*

* $\|t^{l+1} - t^l\| < \varepsilon$ après une itération réussie. Dans ce cas, l'amélioration est trop faible et on se trouve dans un voisinage très étroit d'un optimum local.
* Le nombre d'itérations maximal est atteint ($l + 1 > l_{max} = 200$).
* Le temps d'exécution maximal est atteint.

Si aucune de ces conditions n'est vérifiée alors poser $l = l + 1$ et aller à l'étape 1.

3.7 Expérimentation

Les expériences numériques effectuées permettent d'évaluer la performance des méthodes exactes et heuristiques.

Nous avons défini deux classes de fonctions de demande : $d_k(U_k) = \alpha_k U_k^{-\beta_k}$ et $d_k(U_k) = \alpha_k \exp\left(-\frac{\beta_k}{C^k_{moy}} U_k\right)$ où C^k_{moy} est la moyenne des coûts fixes des chemins du produit k. Le coefficient α_k de la fonction de demande $d_k(U_k) = \alpha_k U_k^{-\beta_k}$ $\left(\text{resp. } d_k(U_k) = \alpha_k \exp\left(-\frac{\beta_k}{C^k_{moy}} U_k\right)\right)$ du produit k est défini par $\alpha_k = \left\lceil \frac{\xi_k \left(1.5 C^k_{moy}\right)^{1.5}}{10^5} \right\rceil$ $\left(\text{resp. } \alpha_k = \left\lceil \frac{\xi_k \exp(1.5\beta_k)}{10^5} \right\rceil\right)$ où ξ_k une valeur choisie uniformément dans l'intervalle $\left[\exp\left(8\left\{\frac{1}{2} + \frac{3}{2}\frac{\gamma_k(\infty)}{C^k_{moy}}\right\}\right), \exp\left(8.5\left\{\frac{1}{2} + \frac{3}{2}\frac{\gamma_k(\infty)}{C^k_{moy}}\right\}\right)\right]$. Ces choix sont effectués de manière à rendre la majorité des produits intéressants.

Les expériences numériques sont effectuées sur les quatre familles de réseaux présentées à la section 2.6.1. Nous considérons les tarifs positifs et nous varions le nombre de produits entre 5 et 60. La proportion d'arcs taxables est fixée à 10%, 15% et 20%. La procédure 'IloPiecewiseLinear' de la librairie "ILOG CPLEX Callable Library" est utilisée pour définir les fonctions linéaires par morceaux. Nous fixons les valeurs de β_k à 2 et 5 afin de varier l'élasticité de la demande. Les résultats numériques sont obtenus en fixant ε à 10^{-2}. Les intervalles des valeurs de U_k et Γ_k sont discrétisés en 20 points pour la surapproximation et la sous-approximation des fonctions non linéaires à la phase d'initialisation des méthodes exactes. Pour les heuristiques, nous utilisons 100 solutions de départ, nous fixons à 200 le nombre d'itérations maximal et à 5 heures (18000 secondes) le temps d'exécution maximal. Le choix des valeurs des constantes de l'étape 0 des heuristiques est inspiré de Colson *et al.* [19].

Les résultats numériques présentent la moyenne sur 10 instances et sont présentés sous la forme suivante.

		EXACT-1						EXACT-2					Heuristique				
#OD	%t	Cpu	Gap	BGap	#it	NbCoupe	Nopt	Cpu	Gap	#it	NbCoupe	Nopt	Cpu	Cpu*	Gap	Nbpoint	Nopt

La colonne Gap de EXACT-1 correspond à l'erreur relative (en %) fournie par cette méthode tandis que BGap est l'erreur relative de la solution fournie par EXACT-1 en utilisant plutôt

la borne supérieure de EXACT-2 car elle est de meilleure qualité. La colonne Cpu* représente le temps requis par l'heuristique pour trouver la meilleure solution.

Résultats numériques obtenus avec l'élasticité constante Lorsque l'élasticité est constante, la fonction $d_k(U_k)U_k$ est convexe sur l'intervalle des valeurs de U_k tandis que la fonction $d_k\left(\sum_{a\in p_k^0} c_a + \Gamma_k\right)\Gamma_k$ est pseudo-concave sur l'intervalle des valeurs de Γ_k.

Les résultats numériques sont reportés aux tableaux 3.1, 3.2, 3.3, 3.4.

Élasticité constante : $\beta_k = 2$ pour tout k

		EXACT-1						EXACT-2					Heuristique-1					Heuristique-2				
#OD	%t	Cpu	Gap	BGap	#it	NbCoupe	Nopt	Cpu	Gap	#it	NbCoupe	Nopt	Cpu	Cpu*	Gap	Nbpoint	Nopt	Cpu	Cpu*	Gap	Nbpoint	Nopt
5	10	0.38	0.15	0.01	5.10	0.00	0	0.01	0.01	1.00	0.00	0	29.37	1.27	0.01	100.00	0	47.64	1.64	0.01	100.00	0
5	15	0.77	0.07	0.01	5.10	0.10	0	0.02	0.01	1.10	0.00	0	26.11	0.77	0.01	100.00	0	62.73	1.30	0.01	100.00	0
5	20	2.75	0.13	0.01	7.20	0.00	0	0.04	0.01	1.20	0.00	0	55.09	2.61	0.01	100.00	0	144.82	14.24	0.01	100.00	0
10	10	12.56	0.11	0.01	6.80	0.20	0	0.04	0.01	1.10	0.00	0	52.28	0.69	0.01	100.00	0	55.10	5.46	0.01	100.00	0
10	15	44.35	0.12	0.01	7.60	0.40	0	0.05	0.01	1.00	0.00	0	73.61	4.98	0.01	100.00	0	106.19	4.98	0.01	100.00	0
10	20	54.97	0.10	0.01	9.20	0.10	0	0.07	0.01	1.20	0.00	0	106.49	3.18	0.01	100.00	0	164.25	19.61	0.01	100.00	0
15	10	208.01	0.14	0.01	8.00	0.20	0	0.09	0.01	1.10	0.00	0	105.71	6.74	0.01	100.00	0	137.25	4.89	0.02	100.00	0
15	15	3270.81	0.16	0.01	7.78	0.33	1	0.12	0.01	1.20	0.00	0	138.51	3.18	0.01	100.00	0	121.13	18.18	0.01	100.00	0
15	20	4619.88	0.12	0.02	9.50	0.70	2	0.15	0.02	1.10	0.00	0	203.93	5.18	0.02	100.00	0	209.14	45.77	0.02	100.00	0
20	10	2521.05	0.12	0.02	9.67	0.33	1	0.12	0.02	1.00	0.00	0	150.79	2.46	0.02	100.00	0	135.16	21.62	0.04	100.00	0
20	15	4272.44	0.26	0.01	8.67	0.33	1	0.21	0.02	1.30	0.00	0	211.46	8.06	0.01	100.00	0	166.52	23.03	0.01	100.00	0
20	20	8888.55	0.33	0.02	7.60	0.30	4	0.26	0.02	1.10	0.00	0	325.25	11.92	0.02	100.00	0	244.34	61.66	0.02	100.00	0
30	10	8200.15	0.59	0.02	8.33	0.33	4	0.50	0.02	1.44	0.00	0	452.90	10.19	0.02	100.00	0	272.73	29.57	0.02	100.00	0
30	15	11789.25	2.12	0.03	5.50	0.70	6	0.52	0.02	1.30	0.00	0	438.14	28.58	0.02	100.00	0	234.94	50.64	0.02	100.00	0
30	20	15232.17	10.33	0.52	4.20	0.40	8	1.33	0.02	1.40	0.00	0	829.34	99.00	0.03	100.00	0	298.35	41.23	0.03	100.00	0
40	10							0.98	0.02	1.50	0.12	0	972.33	106.58	0.02	100.00	0	162.45	24.15	0.02	100.00	0
40	15							2.25	0.02	1.50	0.00	0	1113.72	70.59	0.01	100.00	0	277.37	45.15	0.02	100.00	0
40	20							4.15	0.02	1.40	0.00	0	2230.60	168.39	0.02	100.00	0	469.44	118.45	0.02	100.00	0
50	10							4.39	0.02	1.62	0.00	0	1859.35	263.48	0.02	100.00	0	197.76	41.35	0.02	100.00	0
50	15							11.08	0.02	1.40	0.00	0	2849.59	505.66	0.02	100.00	0	407.08	76.19	0.02	100.00	0
50	20							34.00	0.02	1.60	0.00	0	6406.24	666.41	0.02	97.30	2	825.77	283.36	0.03	100.00	0
60	10							13.64	0.02	1.62	0.12	0	2703.00	450.24	0.02	100.00	0	239.35	38.53	0.02	100.00	0
60	15							79.19	0.02	1.60	0.00	0	6154.19	864.87	0.02	97.40	1	618.76	62.92	0.04	100.00	0
60	20							88.22	0.02	2.00	0.11	0	10860.56	1207.60	0.01	90.56	3	2314.47	643.56	0.03	100.00	0

Élasticité constante : $\beta_k = 5$ pour tout k

		EXACT-1						EXACT-2					Heuristique-1					Heuristique-2				
#OD	%t	Cpu	Gap	BGap	#it	NbCoupe	Nopt	Cpu	Gap	#it	NbCoupe	Nopt	Cpu	Cpu*	Gap	Nbpoint	Nopt	Cpu	Cpu*	Gap	Nbpoint	Nopt
5	10	13.79	0.12	0.02	17.60	0.30	0	0.03	0.02	1.20	0.00	0	16.48	0.15	0.01	100.00	0	80.81	3.62	0.02	100.00	0
5	15	20.74	0.15	0.02	17.80	0.40	0	0.03	0.02	1.10	0.00	0	19.74	0.17	0.02	100.00	0	132.29	8.46	0.02	100.00	0
5	20	79.26	0.08	0.01	18.80	0.30	0	0.05	0.01	1.30	0.00	0	32.72	0.24	0.01	100.00	0	104.82	4.38	0.01	100.00	0
10	10	7336.23	0.13	0.02	18.90	0.90	3	0.07	0.02	1.20	0.00	0	40.91	0.28	0.02	100.00	0	129.87	24.03	0.02	100.00	0
10	15	12895.36	0.56	0.02	16.80	0.70	6	0.08	0.02	1.20	0.00	0	51.94	0.35	0.01	100.00	0	133.45	27.69	0.01	100.00	0
10	20	15094.00	0.42	0.01	16.90	1.20	7	0.09	0.01	1.30	0.00	0	62.46	0.39	0.01	100.00	0	140.23	44.84	0.11	100.00	0
15	10	18032.84	2.55	0.02	12.10	2.30	10	0.10	0.02	1.20	0.00	0	71.97	0.94	0.02	100.00	0	126.21	10.79	0.02	100.00	0
15	15	18034.24	10.09	0.02	8.40	1.70	10	0.12	0.02	1.20	0.00	0	92.60	0.58	0.02	100.00	0	142.26	25.22	0.02	100.00	0
15	20	18027.34	11.18	0.01	7.90	0.80	10	0.13	0.01	1.30	0.00	0	114.71	0.61	0.01	100.00	0	160.55	45.49	0.01	100.00	0
20	10	18035.24	26.12	0.02	7.12	2.12	8	0.17	0.02	1.33	0.00	0	106.95	1.63	0.01	100.00	0	117.82	35.91	0.01	100.00	0
20	15	18052.74	47.97	0.02	4.50	1.00	10	0.23	0.01	1.40	0.00	0	152.88	0.87	0.01	100.00	0	144.02	17.61	0.01	100.00	0
20	20	18043.41	66.70	0.09	3.90	0.20	10	0.21	0.02	1.30	0.00	0	192.93	2.04	0.02	100.00	0	184.93	52.50	0.02	100.00	0
30	10							0.31	0.01	1.44	0.11	0	208.18	1.34	0.01	100.00	0	121.22	37.94	0.01	100.00	0
30	15							0.51	0.01	1.40	0.00	0	414.05	51.52	0.01	100.00	0	143.73	23.59	0.01	100.00	0
30	20							0.60	0.01	1.40	0.10	0	579.01	4.61	0.01	100.00	0	185.20	56.77	0.01	100.00	0
40	10							0.51	0.01	1.38	0.00	0	357.27	2.01	0.01	100.00	0	158.82	45.48	0.01	100.00	0
40	15							0.89	0.02	1.30	0.10	0	893.52	87.26	0.01	100.00	0	173.65	63.38	0.01	100.00	0
40	20							1.38	0.02	1.40	0.20	0	1834.97	30.12	0.01	100.00	0	210.28	81.32	0.01	100.00	0
50	10							0.67	0.01	1.25	0.00	0	657.25	5.79	0.01	100.00	0	154.11	63.38	0.01	100.00	0
50	15							1.45	0.02	1.10	0.10	0	3724.82	142.01	0.01	93.20	1	202.48	65.16	0.01	100.00	0
50	20							2.53	0.02	1.40	0.30	0	6749.98	74.47	0.01	84.70	2	273.14	111.37	0.01	100.00	0
60	10							1.60	0.01	1.25	0.00	0	743.35	66.78	0.01	100.00	0	79.18	36.40	0.02	100.00	0
60	15							4.06	0.02	1.20	0.10	0	4345.20	244.73	0.01	100.00	0	255.35	83.31	0.02	100.00	0
60	20							16.54	0.02	1.33	0.22	0	8644.77	18.63	0.01	87.67	3	398.89	97.11	0.01	100.00	0

Tableau 3.1 Réseaux **DMS** : résultats obtenus avec demande non linéaire (élasticité constante)

Élasticité constante : $\beta_k = 2$ pour tout k

		EXACT-1						EXACT-2					Heuristique-1					Heuristique-2				
#OD	%t	Cpu	Gap	BGap	#it	NbCoupe	Nopt	Cpu	Gap	#it	NbCoupe	Nopt	Cpu	Cpu*	Gap	Nbpoint	Nopt	Cpu	Cpu*	Gap	Nbpoint	Nopt
5	10	0.12	0.05	0.00	2.30	0.00	0	0.03	0.00	1.00	0.00	0	21.11	0.34	0.00	100.00	0	13.41	0.14	0.00	100.00	0
5	15	0.10	0.02	0.00	1.90	0.00	0	0.04	0.00	1.00	0.00	0	16.53	0.19	0.00	100.00	0	6.74	2.62	0.00	100.00	0
5	20	0.10	0.04	0.00	1.50	0.00	0	0.06	0.00	1.00	0.00	0	17.73	0.50	0.00	100.00	0	6.01	0.13	0.00	100.00	0
10	10	0.45	0.07	0.00	3.70	0.00	0	0.06	0.00	1.00	0.00	0	31.61	0.97	0.00	100.00	0	31.05	0.55	0.00	100.00	0
10	15	1.24	0.08	0.00	4.10	0.10	0	0.08	0.00	1.00	0.00	0	46.00	5.88	0.00	100.00	0	46.35	9.85	0.00	100.00	0
10	20	1.61	0.06	0.01	3.40	0.10	0	0.13	0.01	1.00	0.00	0	50.50	17.03	0.01	100.00	0	41.03	0.57	0.01	100.00	0
15	10	3.48	0.10	0.00	4.10	0.00	0	0.10	0.00	1.00	0.00	0	41.07	2.97	0.00	100.00	0	58.43	6.37	0.00	100.00	0
15	15	39.58	0.12	0.00	4.50	0.00	0	0.17	0.00	1.10	0.00	0	69.61	11.36	0.00	100.00	0	127.32	20.22	0.00	100.00	0
15	20	33.42	0.13	0.01	4.90	0.00	0	0.41	0.01	1.00	0.00	0	116.73	44.33	0.01	100.00	0	198.70	94.02	0.01	100.00	0
20	10	63.19	0.13	0.01	4.60	0.10	0	0.20	0.01	1.00	0.00	0	60.66	12.90	0.01	100.00	0	79.95	20.39	0.01	100.00	0
20	15	616.36	0.14	0.01	5.60	0.10	0	0.39	0.01	1.10	0.00	0	115.96	25.52	0.01	100.00	0	157.70	59.33	0.01	100.00	0
20	20	4119.27	0.12	0.01	6.70	0.30	1	0.88	0.01	1.10	0.00	0	235.09	61.21	0.01	100.00	0	884.42	497.64	0.01	100.00	0
30	10	5801.38	0.21	0.03	5.30	0.00	2	0.83	0.01	1.00	0.00	0	116.76	50.86	0.02	100.00	0	352.29	49.11	0.01	100.00	0
30	15	13113.22	0.98	0.01	3.20	0.10	7	1.72	0.01	1.10	0.00	0	321.73	98.96	0.01	100.00	0	1510.11	135.49	0.01	100.00	0
30	20	18042.24	8.82	0.43	2.00	0.00	10	11.86	0.01	1.00	0.00	0	757.38	170.77	0.02	100.00	0	807.08	290.03	0.01	100.00	0
40	10							1.61	0.02	1.00	0.00	0	196.18	71.42	0.12	100.00	0	521.22	162.88	0.02	100.00	0
40	15							3.97	0.01	1.10	0.00	0	618.64	139.08	0.01	100.00	0	786.00	403.92	0.01	100.00	0
40	20							194.58	0.02	1.20	0.00	0	1425.05	597.46	0.02	100.00	0	1601.48	785.64	0.02	100.00	0
50	10							5.76	0.02	1.10	0.00	0	351.87	201.27	0.12	100.00	0	395.37	113.93	0.02	100.00	0
50	15							11.18	0.02	1.10	0.00	0	1310.84	572.23	0.02	100.00	0	4790.17	1344.74	0.02	96.10	1
50	20							251.33	0.02	1.10	0.00	0	5185.12	2921.90	0.08	91.30	1	4133.02	433.44	0.02	100.00	0
60	10							23.86	0.02	1.10	0.00	0	885.65	259.71	0.07	100.00	0	3355.94	489.25	0.03	100.00	0
60	15							156.33	0.02	1.30	0.00	0	5246.01	1712.39	0.02	93.10	2	6279.02	4193.25	0.11	80.10	2
60	20							4223.88	0.16	1.20	0.00	2	8627.35	5952.74	0.40	75.60	3	8881.02	4853.96	0.28	80.50	3

Élasticité constante : $\beta_k = 5$ pour tout k

		EXACT-1						EXACT-2					Heuristique-1					Heuristique-2				
#OD	%t	Cpu	Gap	BGap	#it	NbCoupe	Nopt	Cpu	Gap	#it	NbCoupe	Nopt	Cpu	Cpu*	Gap	Nbpoint	Nopt	Cpu	Cpu*	Gap	Nbpoint	Nopt
5	10	1.71	0.13	0.01	12.20	0.10	0	0.02	0.01	1.10	0.00	0	37.42	0.80	0.01	100.00	0	50.01	12.62	0.01	100.00	0
5	15	4.25	0.22	0.02	12.90	0.40	0	0.03	0.02	1.00	0.00	0	44.96	1.50	0.02	100.00	0	60.75	5.94	0.02	100.00	0
5	20	51.96	0.21	0.02	14.10	0.10	0	0.05	0.02	1.00	0.00	0	48.96	0.49	0.02	100.00	0	75.98	10.77	0.02	100.00	0
10	10	422.16	0.15	0.01	14.70	0.50	0	0.05	0.01	1.00	0.00	0	54.57	5.10	0.01	100.00	0	73.95	27.95	0.01	100.00	0
10	15	7943.96	0.34	0.01	13.00	1.30	3	0.07	0.01	1.20	0.00	0	58.83	3.41	0.01	100.00	0	82.96	21.73	0.01	100.00	0
10	20	14650.40	0.85	0.01	12.80	0.30	7	0.09	0.01	1.00	0.00	0	67.23	0.63	0.01	100.00	0	117.32	24.08	0.01	100.00	0
15	10	12184.50	1.29	0.02	11.10	0.50	5	0.07	0.02	1.00	0.00	0	65.15	5.89	0.02	100.00	0	91.03	24.67	0.02	100.00	0
15	15	16389.67	8.91	0.02	5.90	1.30	9	0.13	0.02	1.20	0.00	0	79.90	11.51	0.02	100.00	0	115.55	62.68	0.02	100.00	0
15	20	18035.27	18.20	0.04	3.90	0.20	10	0.18	0.02	1.00	0.00	0	100.16	6.36	0.02	100.00	0	166.63	54.69	0.02	100.00	0
20	10	18021.77	10.73	0.09	5.11	0.56	9	0.15	0.02	1.10	0.00	0	81.65	11.35	0.02	100.00	0	114.47	33.00	0.02	100.00	0
20	15	18041.55	34.39	0.06	2.86	0.43	7	0.21	0.02	1.10	0.00	0	104.16	7.01	0.02	100.00	0	153.96	41.94	0.02	100.00	0
20	20	18055.79	111.42	0.13	1.78	0.11	9	0.30	0.02	1.00	0.00	0	143.31	6.11	0.02	100.00	0	213.67	43.24	0.03	100.00	0
30	10							0.42	0.02	1.20	0.00	0	123.79	22.16	0.02	100.00	0	154.60	63.01	0.03	100.00	0
30	15							0.81	0.02	1.20	0.00	0	189.81	17.55	0.02	100.00	0	239.76	107.66	0.03	100.00	0
30	20							1.64	0.02	1.10	0.00	0	442.13	30.53	0.02	100.00	0	411.55	158.36	0.02	100.00	0
40	10							0.81	0.02	1.20	0.00	0	170.25	36.70	0.02	100.00	0	194.62	81.32	0.02	100.00	0
40	15							2.20	0.03	1.20	0.00	0	322.00	21.66	0.03	100.00	0	323.84	113.09	0.04	100.00	0
40	20							3.60	0.02	1.20	0.00	0	838.73	43.82	0.02	100.00	0	565.98	166.61	0.05	100.00	0
50	10							4.87	0.02	1.50	0.00	0	254.61	57.26	0.02	100.00	0	235.31	99.33	0.03	100.00	0
50	15							7.13	0.02	1.40	0.00	0	469.09	47.43	0.02	100.00	0	388.04	116.42	0.05	100.00	0
50	20							9.11	0.02	1.10	0.00	0	2996.82	86.87	0.02	99.80	1	719.92	236.42	0.05	100.00	0
60	10							5.14	0.02	1.40	0.00	0	427.31	102.32	0.02	100.00	0	478.30	257.95	0.07	100.00	0
60	15							44.47	0.03	1.50	0.00	0	940.80	227.31	0.03	100.00	0	696.15	304.64	0.04	100.00	0
60	20							22.95	0.02	1.40	0.00	0	3587.58	56.15	0.02	100.00	0	1154.57	447.13	0.04	100.00	0

Tableau 3.2 Réseaux **Delaunay** : résultats obtenus avec demande non linéaire (élasticité constante)

Élasticité constante : $\beta_k = 2$ pour tout k

		EXACT-1						EXACT-2					Heuristique-1					Heuristique-2				
#OD	%t	Cpu	Gap	BGap	#it	NbCoupe	Nopt	Cpu	Gap	#it	NbCoupe	Nopt	Cpu	Cpu*	Gap	Nbpoint	Nopt	Cpu	Cpu*	Gap	Nbpoint	Nopt
5	10	0.15	0.11	0.01	2.40	0.10	0	0.04	0.01	1.00	0.00	0	17.57	0.19	0.01	100.00	0	8.34	0.10	0.01	100.00	0
5	15	6.48	0.16	0.02	5.40	0.10	0	0.09	0.02	1.00	0.00	0	59.67	2.37	0.02	100.00	0	57.76	1.60	0.02	100.00	0
5	20	8.61	0.12	0.02	6.40	0.00	0	0.11	0.02	1.00	0.00	0	68.42	1.13	0.04	100.00	0	60.69	6.84	0.02	100.00	0
10	10	9.28	0.12	0.01	5.30	0.00	0	0.12	0.01	1.10	0.00	0	48.33	1.49	0.01	100.00	0	45.42	1.14	0.01	100.00	0
10	15	892.39	0.14	0.03	7.60	0.50	0	0.41	0.03	1.00	0.00	0	136.23	44.15	0.03	100.00	0	176.89	87.19	0.03	100.00	0
10	20	5775.89	0.13	0.02	8.20	0.90	3	0.67	0.02	1.10	0.00	0	145.86	39.08	0.02	100.00	0	962.37	450.27	0.02	100.00	0
15	10	126.82	0.11	0.01	6.50	0.10	0	0.29	0.01	1.40	0.00	0	64.97	3.96	0.01	100.00	0	52.25	21.29	0.01	100.00	0
15	15	12278.36	1.16	0.02	7.70	1.30	6	1.34	0.01	1.20	0.00	0	224.74	67.65	0.05	100.00	0	258.17	127.45	0.01	100.00	0
15	20	16040.93	11.65	0.44	5.67	0.67	8	2.58	0.01	1.40	0.00	0	282.88	106.65	0.18	100.00	0	283.66	147.39	0.04	100.00	0
20	10	3238.98	0.12	0.02	6.50	0.30	1	0.43	0.02	1.20	0.00	0	90.84	15.08	0.02	100.00	0	59.53	26.40	0.02	100.00	0
20	15	15390.11	27.23	0.13	5.30	0.60	8	2.29	0.03	1.10	0.00	0	315.87	121.79	0.08	100.00	0	306.02	169.27	0.09	100.00	0
20	20	16258.80	20.82	0.55	4.40	0.60	9	6.14	0.02	1.30	0.00	0	389.31	119.35	0.02	100.00	0	393.15	231.70	0.02	100.00	0
30	10							1.43	0.02	1.30	0.00	0	153.15	36.44	0.05	100.00	0	127.82	28.65	0.02	100.00	0
30	15							128.80	0.02	1.60	0.20	0	462.78	145.78	0.03	100.00	0	447.46	125.49	0.02	100.00	0
30	20							24.83	0.02	1.40	0.00	0	576.10	286.41	0.02	100.00	0	655.96	378.10	0.03	100.00	0
40	10							3.63	0.02	1.70	0.00	0	227.44	92.82	0.02	100.00	0	569.86	76.94	0.08	100.00	0
40	15							2134.67	0.02	1.70	0.30	0	710.55	374.43	0.01	100.00	0	650.91	246.97	0.01	100.00	0
40	20							367.21	0.02	1.50	0.00	0	934.00	568.75	0.02	100.00	0	1327.88	546.50	0.02	100.00	0
50	10							8.90	0.02	1.60	0.00	0	315.33	117.29	0.08	100.00	0	790.17	129.08	0.04	100.00	0
50	15							4926.99	0.08	1.40	0.00	1	1356.44	744.32	0.09	100.00	0	988.64	489.31	0.08	100.00	0
50	20							1931.42	0.07	1.50	0.00	1	1715.03	444.25	0.07	100.00	0	1117.40	576.49	0.08	100.00	0
60	10							76.94	0.01	1.90	0.10	0	638.95	94.27	0.07	100.00	0	510.74	169.19	0.07	100.00	0
60	15							10294.02	0.12	1.70	0.10	3	2606.29	1272.81	0.12	100.00	0	1798.21	577.59	0.12	100.00	0
60	20							5112.01	0.07	1.50	0.00	1	2996.06	1061.41	0.10	100.00	0	2345.53	580.36	0.07	100.00	0

Élasticité constante : $\beta_k = 5$ pour tout k

		EXACT-1						EXACT-2					Heuristique-1					Heuristique-2				
#OD	%t	Cpu	Gap	BGap	#it	NbCoupe	Nopt	Cpu	Gap	#it	NbCoupe	Nopt	Cpu	Cpu*	Gap	Nbpoint	Nopt	Cpu	Cpu*	Gap	Nbpoint	Nopt
5	10	22.60	0.12	0.01	11.00	0.50	0	0.03	0.01	1.30	0.00	0	35.49	0.37	0.01	100.00	0	36.31	2.50	0.01	100.00	0
5	15	123.65	0.10	0.01	17.30	0.40	0	0.08	0.01	1.30	0.00	0	53.45	0.55	0.01	100.00	0	76.41	12.74	0.01	100.00	0
5	20	219.08	0.06	0.01	18.90	0.50	0	0.11	0.01	1.20	0.00	0	59.60	5.45	0.01	100.00	0	91.53	13.82	0.01	100.00	0
10	10	9159.79	0.61	0.02	12.90	0.90	5	0.09	0.02	1.30	0.00	0	55.48	3.63	0.02	100.00	0	68.99	15.79	0.02	100.00	0
10	15	17278.46	2.36	0.02	12.10	2.10	9	0.16	0.02	1.10	0.00	0	86.02	9.82	0.02	100.00	0	133.49	49.17	0.02	100.00	0
10	20	14729.32	14.58	0.02	12.00	2.20	8	0.23	0.02	1.20	0.00	0	97.24	4.94	0.02	100.00	0	149.93	40.22	0.02	100.00	0
15	10	18037.27	24.56	0.02	8.56	2.00	9	0.25	0.02	1.50	0.00	0	76.23	15.09	0.02	100.00	0	94.62	23.38	0.02	100.00	0
15	15	18029.09	52.22	0.06	6.70	1.50	10	0.45	0.02	1.30	0.00	0	115.11	5.63	0.02	100.00	0	165.37	62.78	0.02	100.00	0
15	20	18043.72	209.53	0.01	4.80	1.20	10	0.44	0.01	1.10	0.00	0	137.00	6.17	0.01	100.00	0	202.02	87.53	0.01	100.00	0
20	10	16055.63	159.94	0.70	4.22	0.89	8	0.47	0.01	1.50	0.00	0	92.02	9.05	0.01	100.00	0	116.60	23.41	0.01	100.00	0
20	15	18043.58	254.33	0.19	3.40	0.60	10	0.60	0.02	1.10	0.00	0	136.62	4.18	0.02	100.00	0	196.99	73.92	0.02	100.00	0
20	20	18039.71	289.85	0.01	2.90	0.50	10	0.75	0.01	1.10	0.00	0	171.19	6.32	0.01	100.00	0	239.28	101.12	0.01	100.00	0
30	10							0.82	0.01	1.30	0.00	0	110.89	12.11	0.01	100.00	0	131.81	31.78	0.01	100.00	0
30	15							1.35	0.02	1.10	0.00	0	203.61	25.61	0.02	100.00	0	270.55	165.84	0.02	100.00	0
30	20							2.91	0.01	1.00	0.00	0	267.38	41.78	0.01	100.00	0	341.17	119.48	0.01	100.00	0
40	10							1.24	0.01	1.22	0.00	0	133.44	46.35	0.01	100.00	0	159.98	36.69	0.01	100.00	0
40	15							3.44	0.02	1.30	0.00	0	283.56	30.96	0.02	100.00	0	334.11	205.45	0.02	100.00	0
40	20							192.30	0.02	1.10	0.10	0	460.35	103.83	0.01	100.00	0	424.39	191.06	0.01	100.00	0
50	10							2.89	0.01	1.44	0.00	0	132.03	32.95	0.01	100.00	0	164.92	84.51	0.01	100.00	0
50	15							5.83	0.02	1.30	0.00	0	373.85	13.10	0.02	100.00	0	328.22	149.74	0.02	100.00	0
50	20							1994.15	0.03	1.10	0.00	1	617.48	89.91	0.02	100.00	0	414.16	205.16	0.03	100.00	0
60	10							8.14	0.01	1.56	0.00	0	270.29	65.92	0.01	100.00	0	394.91	138.06	0.01	100.00	0
60	15							57.48	0.02	1.30	0.00	0	480.09	73.05	0.02	100.00	0	648.31	147.71	0.02	100.00	0
60	20							1818.08	0.05	1.00	0.00	1	2591.01	968.67	0.04	97.40	1	812.98	446.66	0.04	100.00	0

Tableau 3.3 Réseaux **Voronoï** : résultats obtenus avec demande non linéaire (élasticité constante)

Élasticité constante : $\beta_k = 2$ pour tout k

		EXACT-1						EXACT-2					Heuristique-1					Heuristique-2				
#OD	%t	Cpu	Gap	BGap	#it	NbCoupe	Nopt	Cpu	Gap	#it	NbCoupe	Nopt	Cpu	Cpu*	Gap	Nbpoint	Nopt	Cpu	Cpu*	Gap	Nbpoint	Nopt
5	10	0.57	0.07	0.01	4.10	0.00	0	0.03	0.01	1.00	0.00	0	24.26	0.46	0.01	100.00	0	16.28	0.23	0.01	100.00	0
5	15	0.31	0.07	0.00	2.90	0.00	0	0.03	0.00	1.00	0.00	0	18.12	0.53	0.00	100.00	0	11.25	0.23	0.00	100.00	0
5	20	0.70	0.15	0.01	3.60	0.10	0	0.09	0.01	1.00	0.00	0	47.92	3.77	0.01	100.00	0	43.14	1.85	0.01	100.00	0
10	10	1804.09	0.13	0.01	4.30	0.10	1	0.15	0.01	1.10	0.00	0	69.32	1.28	0.01	100.00	0	42.01	7.33	0.01	100.00	0
10	15	5.08	0.13	0.01	2.90	0.00	0	0.16	0.01	1.00	0.00	0	49.61	3.06	0.01	100.00	0	30.83	0.30	0.01	100.00	0
10	20	50.61	0.12	0.01	2.90	0.30	0	0.62	0.01	1.00	0.00	0	176.08	21.16	0.01	100.00	0	248.59	4.87	0.01	100.00	0
15	10	3597.32	0.38	0.01	3.00	0.00	1	0.38	0.01	1.20	0.00	0	109.93	12.46	0.01	100.00	0	44.92	7.24	0.01	100.00	0
15	15	2089.01	0.26	0.01	3.10	0.00	1	0.80	0.01	1.00	0.00	0	166.19	12.53	0.01	100.00	0	133.85	3.64	0.01	100.00	0
15	20	4703.20	0.70	0.51	2.50	0.10	2	21.40	0.01	1.22	0.00	0	1049.98	249.79	0.01	100.00	0	946.05	119.88	0.01	100.00	0
20	10	5438.81	0.83	0.07	3.10	0.10	2	1.03	0.01	1.30	0.00	0	189.59	16.24	0.01	100.00	0	1709.05	16.36	0.01	100.00	0
20	15	9961.05	1.42	0.27	3.00	0.00	5	9.40	0.02	1.10	0.00	0	755.46	158.23	0.02	100.00	0	1650.62	51.94	0.02	100.00	0
20	20	16923.73	2.06	0.20	2.30	0.00	9	20.68	0.02	1.10	0.10	0	3105.40	608.12	0.02	100.00	0	5463.10	76.06	0.02	94.90	1
30	10							8.63	0.02	1.30	0.10	0	720.35	58.36	0.02	100.00	0	2079.41	10.40	0.02	100.00	0
30	15							1925.09	0.12	1.10	0.00	1	5240.64	1699.47	0.12	90.00	2	9455.91	2194.24	0.30	68.50	5
30	20							3221.91	0.06	1.33	0.11	1	14141.72	1828.52	0.06	75.22	6	13699.33	2402.74	0.06	67.67	5
40	10							26.36	0.02	1.30	0.00	0	2998.20	75.38	0.02	100.00	0	1240.39	106.42	0.02	100.00	0
40	15							2513.65	0.50	1.20	0.00	1	8866.29	2420.29	0.50	76.90	4	12960.83	2167.11	0.51	54.90	7
40	20							9218.17	0.43	1.30	0.20	4	12901.23	2813.17	9.37	61.60	6	14875.46	6529.75	1.34	32.10	8
50	10							587.33	0.02	1.40	0.00	0	3638.73	187.81	0.02	95.90	1	6361.89	2004.89	0.02	81.70	3
50	15							5461.12	0.72	1.00	0.00	2	10195.19	1804.39	0.79	71.10	5	11598.20	3763.94	0.75	59.00	6
50	20							11247.16	1.92	1.20	0.10	6	16889.02	6321.62	2.34	39.70	9	16366.29	10639.03	4.05	20.71	6
60	10							2160.24	0.05	1.20	0.00	1	7506.69	498.68	0.05	91.00	2	4624.95	161.39	0.05	91.30	1
60	15							7291.98	1.16	1.10	0.00	4	11271.94	1881.07	1.22	64.40	5	15115.26	3466.99	1.17	43.20	7
60	20							10799.20	1.94	1.20	0.10	5	17283.63	9417.01	2.04	36.80	9	17380.85	11583.49	4.43	30.57	6

Élasticité constante : $\beta_k = 5$ pour tout k

		EXACT-1						EXACT-2					Heuristique-1					Heuristique-2				
#OD	%t	Cpu	Gap	BGap	#it	NbCoupe	Nopt	Cpu	Gap	#it	NbCoupe	Nopt	Cpu	Cpu*	Gap	Nbpoint	Nopt	Cpu	Cpu*	Gap	Nbpoint	Nopt
5	10	8.81	0.17	0.02	11.30	0.20	0	0.03	0.02	1.00	0.00	0	43.23	0.42	0.02	100.00	0	150.87	17.06	0.02	100.00	0
5	15	33.88	0.18	0.01	11.40	0.30	0	0.04	0.01	1.10	0.00	0	45.38	2.66	0.01	100.00	0	203.15	18.50	0.01	100.00	0
5	20	405.51	0.15	0.01	15.40	0.70	0	0.14	0.01	1.30	0.00	0	77.22	2.25	0.01	100.00	0	239.55	3.52	0.01	100.00	0
10	10	6702.81	0.40	0.02	10.60	0.70	3	0.11	0.02	1.10	0.00	0	77.36	4.35	0.02	100.00	0	205.25	34.37	0.02	100.00	0
10	15	6808.36	0.40	0.01	11.10	0.60	3	0.13	0.01	1.30	0.00	0	91.72	4.72	0.01	100.00	0	279.57	45.90	0.01	100.00	0
10	20	16757.74	3.08	0.03	9.50	1.40	9	0.40	0.03	1.10	0.00	0	246.00	25.13	0.03	100.00	0	407.69	70.03	0.03	100.00	0
15	10	10574.93	4.11	0.23	7.70	0.70	5	0.33	0.01	1.30	0.00	0	129.68	6.30	0.01	100.00	0	224.75	69.93	0.01	100.00	0
15	15	13561.59	8.29	0.14	4.70	0.30	6	0.39	0.01	1.30	0.00	0	191.74	5.63	0.01	100.00	0	367.10	18.63	0.01	100.00	0
15	20	18025.79	40.25	0.20	2.80	0.50	10	1.42	0.02	1.30	0.00	0	842.41	76.84	0.05	100.00	0	593.10	185.79	0.02	100.00	0
20	10	12719.56	27.54	1.02	4.60	0.70	7	1.19	0.02	1.20	0.00	0	219.39	20.44	0.02	100.00	0	301.46	70.13	0.02	100.00	0
20	15	16357.73	30.37	0.56	4.00	0.00	8	0.83	0.02	1.30	0.00	0	458.59	47.02	0.02	100.00	0	1268.43	234.29	0.02	100.00	0
20	20	18032.64	113.01	0.71	1.40	0.20	10	3.60	0.02	1.40	0.00	0	1856.79	103.40	0.02	100.00	0	995.28	264.43	0.02	100.00	0
30	10							4.34	0.02	1.40	0.00	0	563.93	24.36	0.02	100.00	0	538.00	128.84	0.03	100.00	0
30	15							6.27	0.01	1.50	0.00	0	2703.97	24.63	0.01	100.00	0	1812.20	478.19	0.02	100.00	0
30	20							9.99	0.02	1.20	0.00	0	8348.38	293.05	0.02	98.90	2	1941.25	569.42	0.02	100.00	0
40	10							36.13	0.02	1.40	0.10	0	1119.78	131.81	0.02	100.00	0	1111.80	450.36	0.02	100.00	0
40	15							10.35	0.02	1.50	0.00	0	5166.30	239.38	0.02	97.00	1	2542.19	428.79	0.02	100.00	0
40	20							23.29	0.02	1.40	0.00	0	11755.06	92.61	0.02	85.70	5	4006.77	1014.38	0.02	100.00	0
50	10							17.58	0.02	1.50	0.00	0	2261.71	219.67	0.02	100.00	0	1479.02	317.44	0.02	100.00	0
50	15							21.21	0.02	1.50	0.00	0	7797.47	859.49	0.02	92.70	2	4118.06	1294.39	0.02	100.00	0
50	20							50.57	0.02	1.50	0.00	0	14839.19	818.34	0.02	65.90	7	7025.12	2666.49	0.03	93.00	2
60	10							65.05	0.02	1.50	0.00	0	4989.12	363.14	0.02	98.20	1	2269.62	243.02	0.02	100.00	1
60	15							41.89	0.02	1.30	0.00	0	9584.71	269.96	0.02	89.30	3	2725.54	601.95	0.02	100.00	0
60	20							106.80	0.01	1.60	0.10	0	14973.62	779.87	0.01	62.22	6	6671.08	1596.54	0.01	97.90	1

Tableau 3.4 Réseaux **Grille** : résultats obtenus avec demande non linéaire (élasticité constante)

Nous commençons par faire une justification du choix du nombre de points utilisés pour la discrétisation des intervalles des valeurs de U_k et Γ_k lors des surapproximations (resp. sous-approximation) des fonctions non linéaires $d_k(U_k)U_k$ et $d_k\left(\Gamma_k + \sum_{a \in p_0} c_a\right)\Gamma_k$ (resp. $d_k(U_k)$). Nous avons testé les méthodes exactes sur quelques instances en utilisant 10, 20 et 30 points de discrétisation. Nous avons constaté qu'avec 10 points, bien qu'à chaque itération le MIP provenant des approximations possédait moins de variables binaires, le temps de calcul était le plus élevé, ceci dû au nombre élevé d'itérations. Avec 30 points, le nombre d'itérations était presque similaire à celui de 20 points cependant son temps de calcul était plus élevé qu'avec 20 points. Nous avons donc décidé de présenter les tests numériques seulement avec 20 points de discrétisation sur les intervalles des valeurs de U_k et Γ_k.

La méthode EXACT-2 est plus efficace que la méthode EXACT-1 et ce sur les quatre réseaux utilisés pour les tests numériques. La méthode EXACT-1 trouve généralement une solution de bonne qualité mais sa borne supérieure est très mauvaise comparée à celle fournie par EXACT-2 (voir colonne Gap et Bgap de la méthode EXACT-1). À titre d'exemple, en considérant $\beta_k = 5$ et les instances de 20 produits et moins du réseau Voronoï (resp. Grille), le temps moyen d'exécution, l'erreur relative moyenne et le pourcentage d'instances ayant atteint le temps maximal avec EXACT-1 est 3 heures 25 minutes, 84.02% et 65.83% (resp. 2 heures 47 minutes, 19% et 50.83%) contre 1 seconde, 0.01% et 0% (resp. 1 seconde, 0.01% et 0%) avec EXACT-2. Notons que pour ces instances, le temps d'exécution de EXACT-1 est 4863 (resp. 8) fois celui de EXACT-2 lorsque $\beta_k = 2$.

Avec la méthode EXACT-1, la difficulté du DNL augmente avec l'élasticité de la demande. En fait, la surapproximation de $d_k(U_k)U_k$ est plus difficile lorsque l'élasticité est forte car l'ensemble des droites par morceaux ajoutées pour surapproximer $d_k(U_k)U_k$ ont des pentes plus fortes que celles de l'ensemble des droites par morceaux obtenues avec l'élasticité faible. À titre d'illustration, en considérant le réseau Voronoï avec 20 produits et moins, le temps d'exécution moyen, l'erreur relative moyenne et le nombre d'itérations moyen de la méthode EXACT-1 avec $\beta_k = 2$ est 1 heure 37 minutes, 5.15% et 5.94 contre 3 heures 25 minutes,

84.02% et 9.56 avec $\beta_k = 5$.

Avec EXACT-2, la difficulté du DNL augmente lorsque l'élasticité diminue. En fait, la méthode EXACT-2 trouve rapidement une borne inférieure de bonne qualité au DNL. Lorsque l'élasticité est forte, il y a une très grande différence entre les bornes supérieures des nœuds de l'arbre de branchement. Cependant, la borne supérieure de la plupart des nœuds de l'arbre de branchement est inférieure ou égale à la meilleure borne inférieure trouvée, ce qui permet d'élaguer ces nœuds et donc d'accélérer la résolution du DNL. Par contre lorsque l'élasticité est faible, les bornes supérieures des nœuds de l'arbre de branchement ont presque la même valeur, et cette valeur est supérieure à la valeur de la meilleure borne inférieure trouvée, ce qui ne permet pas d'élaguer rapidement les nœuds de l'arbre de branchement et donc ralentit la résolution du DNL. À titre d'exemple, en considérant la Grille avec 60 produits, le temps d'exécution moyen, l'erreur relative moyenne et le pourcentage moyen d'instances ayant atteint le temps maximal obtenus lorsque $\beta_k = 2$ est 2 heures, 1.05% et 33.33% contre 1 minute, 0.01% et 0% lorsque $\beta_k = 5$. Pour ces instances, le temps d'exécution obtenu avec $\beta_k = 2$ est 95 fois celui obtenu avec $\beta_k = 5$. La méthode EXACT-2 est très efficace lorsque l'élasticité est forte ($\beta_k = 5$) comme en témoignent les résultats obtenus avec 60 produits de Voronoï (resp. Grille) où le temps moyen d'exécution et l'erreur relative moyenne sont 10 minutes et 0.02% (resp. 2 minutes et 0.02%). Elle est également efficace sur les instances de 40 produits et moins lorsque l'élasticité est faible. Cependant pour les instances de plus de 40 produits, bien que le temps moyen d'exécution soit élevé, l'erreur relative moyenne demeure assez bonne. Pour ces instances, nous avons fait recours aux heuristiques.

D'après les résultats obtenus des deux heuristiques, nous constatons que le temps de calcul de l'heuristique 2 est légèrement inférieur à celui de l'heuristique 1. La qualité des solutions fournies par les deux heuristiques est presque la même. Cependant, l'heuristique 1 trouve rapidement une meilleure solution que l'heuristique 2. Dès lors, nous utiliserons l'heuristique 1 pour faire une comparaison avec la méthode EXACT-2.

Les heuristiques développées fournissent des solutions de bonne qualité avec des temps

de calcul raisonnables comparés à EXACT-2 lorsque l'élasticité est faible ($\beta_k = 2$). De plus, elles trouvent rapidement des solutions de bonne qualité. À titre d'exemple, en considérant les instances de Voronoi (resp. Grille) de 60 produits et $\beta_k = 2$, le temps d'exécution moyen et l'erreur relative moyenne est 1 heure 26 minutes et 1.33% (resp. 2 heures et 1.05%) pour EXACT-2, 35 minutes et 0.096% (resp. 3 heures 20 minutes et 1.10%) pour l'heuristique 1 et 26 minutes et 0.086% (resp. 3 heures 26 minutes et 1.88%) pour l'heuristique 2. Cependant pour ces instances de Grille, le temps moyen où l'heuristique trouve la meilleure solution est 1 heure pour l'heuristique 1 et 1 heure 24 minutes pour l'heuristique 2. Le rapport du temps moyen de EXACT-2 et du temps moyen auquel l'heuristique trouve la meilleure solution est 2.66 pour l'heuristique 1 et 1.5 pour l'heuristique 2. Les figures 3.9, 3.11 et 3.10 montrent l'évolution de EXACT-2 et de l'heuristique 1.

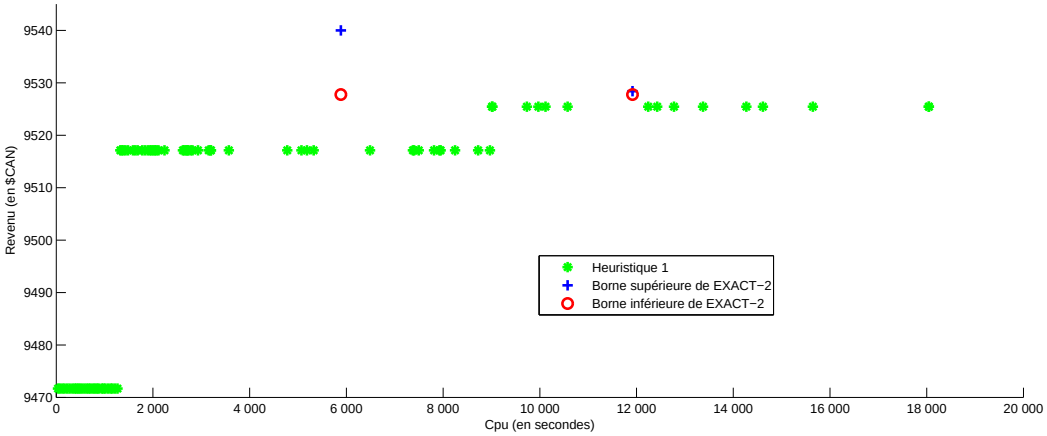


Figure 3.9 Évolution de la méthode EXACT-2 et l'heuristique 1 en fonction du temps d'exécution sur une instance de **Grille** ayant 60 produits, 20% d'arcs taxables et $\beta_k = 2$

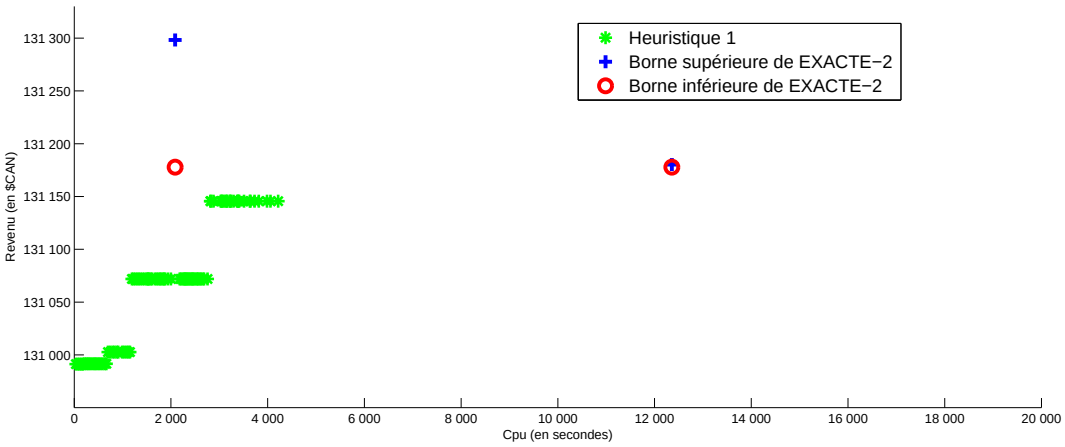


Figure 3.10 Évolution de la méthode EXACT-2 et l'heuristique en fonction du temps d'exécution sur une instance de **Voronoï** ayant 60 produits, 20% d'arcs taxables et $\beta_k = 2$

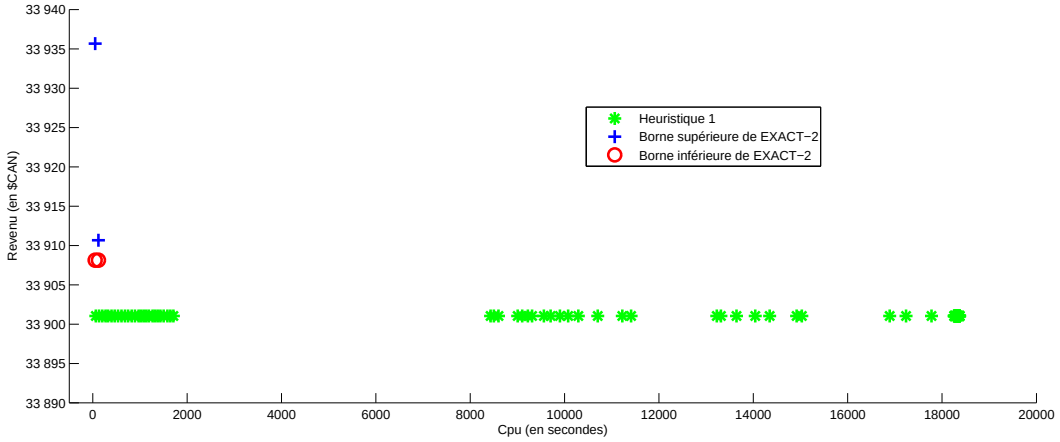


Figure 3.11 Évolution de la méthode EXACT-2 et l'heuristique en fonction du temps d'exécution sur une instance de **Grille** ayant 60 produits, 20% d'arcs taxables et $\beta_k = 5$

Le temps de calcul de l'heuristique 2 est en général inférieur à celui de l'heuristique 1 mais l'heuristique 1 trouve rapidement une meilleure solution comparée à l'heuristique 2. Les erreurs relatives des deux heuristiques sont quasi-égales et sont situées en moyenne à 1% de la borne supérieure de EXACT-2 lorsque $\beta_k = 2$. Notons également que le temps de calcul des heuristiques diminue lorsque l'élasticité augmente.

Résultats numériques obtenus avec l'élasticité non constante Lorsque l'élasticité est non constante, les fonctions $d_k(U_k)U_k$ et $d_k\left(\sum_{a\in p_k^0} c_a + \Gamma_k\right)\Gamma_k$ sont pseudo-concaves sur l'intervalle de leurs variables respectives. Cependant, la partie convexe de la fonction $d_k(U_k)U_k$ demeure plus large que celle de la fonction $d_k\left(\sum_{a\in p_k^0} c_a + \Gamma_k\right)\Gamma_k$. Les méthodes développées sont testées sur les réseaux Voronoï et Grille car leurs instances sont plus difficiles que celles de Didi et Delaunay. Les résultats numériques sont reportés aux tableaux 3.5, 3.6 ci-dessous ne contiennent que les méthodes exactes et l'heuristique 1. En fait, nous avons vu dans le cas de l'élasticité constante que les deux heuristiques fournissent à peu près les solutions de même qualité et que l'heuristique 1 trouve rapidement la meilleure solution comparée à l'heuristique 2.

Élasticité non constante : $\beta_k = 2$ pour tout k

		EXACT-1						EXACT-2					Heuristique-1				
#OD	%t	Cpu	Gap	BGap	#it	NbCoupe	Nopt	Cpu	Gap	#it	NbCoupe	Nopt	Cpu	Cpu*	Gap	Nbpoint	Nopt
5	10	0.05	0.01	0.01	1.00	0.00	0	0.04	0.01	1.00	0.00	0	13.07	0.88	0.01	100.00	0
5	15	0.12	0.02	0.02	1.00	0.00	0	0.12	0.02	1.10	0.00	0	47.16	1.15	0.02	100.00	0
5	20	0.16	0.01	0.01	1.10	0.00	0	0.16	0.01	1.20	0.00	0	50.65	1.36	0.10	100.00	0
10	10	0.13	0.01	0.01	1.00	0.00	0	0.13	0.01	1.00	0.00	0	32.93	7.24	0.24	100.00	0
10	15	0.60	0.02	0.02	1.10	0.00	0	0.57	0.02	1.10	0.00	0	199.17	5.43	0.05	100.00	0
10	20	6.34	0.02	0.02	1.10	0.00	0	1.48	0.02	1.30	0.00	0	250.40	55.05	0.04	100.00	0
15	10	0.32	0.02	0.01	1.10	0.00	0	0.33	0.01	1.20	0.00	0	59.76	10.06	0.25	100.00	0
15	15	2.28	0.02	0.02	1.20	0.00	0	1.53	0.02	1.20	0.00	0	401.61	67.40	0.04	100.00	0
15	20	29.06	0.02	0.02	1.10	0.00	0	12.26	0.02	1.40	0.00	0	679.75	210.10	0.02	100.00	0
20	10	0.64	0.02	0.01	1.10	0.00	0	0.61	0.01	1.30	0.00	0	87.16	6.42	0.15	100.00	0
20	15	9.78	0.02	0.02	1.30	0.10	0	7.82	0.02	1.40	0.00	0	598.14	83.30	0.23	100.00	0
20	20	58.90	0.02	0.02	1.10	0.00	0	61.50	0.02	1.50	0.00	0	1518.80	237.82	0.13	100.00	0
30	10	2.34	0.02	0.01	1.10	0.00	0	2.63	0.01	1.70	0.00	0	242.88	8.80	0.05	100.00	0
30	15	67.49	0.02	0.02	1.20	0.00	0	46.06	0.02	1.60	0.00	0	1748.63	32.56	0.03	100.00	0
30	20	72.22	0.02	0.02	1.20	0.00	0	95.82	0.02	1.40	0.00	0	2795.26	898.38	0.17	100.00	0
40	10	5.94	0.02	0.01	1.10	0.00	0	6.15	0.01	1.80	0.10	0	461.99	251.90	0.02	100.00	0
40	15	3991.75	0.11	0.02	1.10	0.00	2	2290.37	0.01	2.00	0.00	1	6620.67	1069.92	0.02	100.00	0
40	20	1717.24	0.03	0.02	1.20	0.10	0	1999.05	0.02	1.60	0.00	0	6896.52	2645.82	0.12	98.30	1
50	10	25.92	0.02	0.01	1.10	0.00	0	32.16	0.01	1.70	0.20	0	890.14	129.26	0.03	100.00	0
50	15	6121.54	0.04	0.03	1.10	0.00	2	3995.28	0.06	1.80	0.00	1	11890.04	2119.33	0.13	91.20	2
50	20	7839.40	0.27	0.25	1.10	0.00	4	6965.02	0.28	1.50	0.00	3	11885.40	1491.24	0.31	78.10	5
60	10							93.74	0.01	1.70	0.10	0	1526.63	369.62	0.08	100.00	0
60	15							10059.44	0.62	1.40	0.00	5	15396.97	1623.85	0.63	58.00	7
60	20							11170.89	0.53	1.40	0.00	5	14497.18	2031.04	0.53	52.60	6

Élasticité non constante : $\beta_k = 5$ pour tout k

		EXACT-1						EXACT-2					Heuristique-1				
#OD	%t	Cpu	Gap	BGap	#it	NbCoupe	Nopt	Cpu	Gap	#it	NbCoupe	Nopt	Cpu	Cpu*	Gap	Nbpoint	Nopt
5	10	1.07	0.02	0.01	4.30	0.00	0	0.07	0.01	1.20	0.00	0	25.76	0.30	0.01	100.00	0
5	15	54.28	0.03	0.02	7.00	0.20	0	0.15	0.02	1.10	0.00	0	49.56	2.54	0.02	100.00	0
5	20	81.23	0.03	0.02	7.10	0.10	0	0.20	0.02	1.30	0.00	0	54.68	1.29	0.02	100.00	0
10	10	1475.61	0.03	0.02	5.10	0.20	0	0.14	0.02	1.20	0.00	0	42.27	2.66	0.02	100.00	0
10	15	9859.19	0.86	0.02	5.00	0.40	5	0.37	0.02	1.20	0.00	0	96.01	11.50	0.02	100.00	0
10	20	10959.99	1.49	0.03	4.40	0.30	6	0.74	0.02	1.60	0.00	0	117.33	1.17	0.02	100.00	0
15	10	8429.88	0.72	0.01	4.40	0.50	4	0.46	0.01	1.80	0.00	0	74.87	7.35	0.02	100.00	0
15	15	15541.68	7.51	0.06	2.20	0.00	8	0.70	0.02	1.20	0.00	0	139.88	12.88	0.03	100.00	0
15	20	16256.49	10.52	0.37	1.40	0.00	9	1.85	0.02	1.60	0.00	0	221.41	17.99	0.02	100.00	0
20	10	12817.26	1.64	0.07	3.30	0.30	7	0.82	0.01	1.80	0.00	0	97.35	5.31	0.01	100.00	0
20	15	18051.31	13.43	0.19	1.30	0.00	10	1.22	0.02	1.40	0.00	0	222.29	25.73	0.02	100.00	0
20	20	18044.32	16.56	0.47	1.40	0.00	10	4.90	0.02	1.60	0.00	0	348.12	110.98	0.02	100.00	0
30	10							1.74	0.02	1.70	0.00	0	136.83	23.04	0.02	100.00	0
30	15							3.14	0.02	1.50	0.00	0	384.65	3.80	0.02	100.00	0
30	20							21.18	0.02	1.70	0.00	0	680.10	59.56	0.02	100.00	0
40	10							3.44	0.02	1.80	0.00	0	192.59	25.94	0.02	100.00	0
40	15							8.76	0.02	1.67	0.11	0	1097.52	55.36	0.02	100.00	0
40	20							43.72	0.02	1.50	0.00	0	1334.02	133.03	0.02	100.00	0
50	10							8.83	0.01	2.00	0.00	0	393.78	185.41	0.06	100.00	0
50	15							26.72	0.02	1.90	0.10	0	1323.60	115.87	0.01	100.00	0
50	20							2069.65	0.05	1.56	0.11	1	3576.04	1065.25	0.05	100.00	0
60	10							17.83	0.01	2.00	0.00	0	601.17	212.09	0.01	100.00	0
60	15							84.35	0.02	1.78	0.00	0	3603.43	478.58	0.02	92.89	1
60	20							4889.55	0.31	1.40	0.00	2	4086.73	710.93	0.10	89.50	0

Tableau 3.5 Réseaux **Voronoï** : résultats obtenus avec demande non linéaire (élasticité non constante)

Élasticité non constante : $\beta_k = 2$ pour tout k

		EXACT-1						EXACT-2					Heuristique-1				
#OD	%t	Cpu	Gap	BGap	#it	NbCoupe	Nopt	Cpu	Gap	#it	NbCoupe	Nopt	Cpu	Cpu*	Gap	Nbpoint	Nopt
5	10	1.07	0.02	0.01	4.30	0.00	0	0.07	0.01	1.20	0.00	0	25.76	0.30	0.01	100.00	0
5	15	54.28	0.03	0.02	7.00	0.20	0	0.15	0.02	1.10	0.00	0	49.56	2.54	0.02	100.00	0
5	20	81.23	0.03	0.02	7.10	0.10	0	0.20	0.02	1.30	0.00	0	54.68	1.29	0.02	100.00	0
10	10	1475.61	0.03	0.02	5.10	0.20	0	0.14	0.02	1.20	0.00	0	42.27	2.66	0.02	100.00	0
10	15	9859.19	0.86	0.02	5.00	0.40	5	0.37	0.02	1.20	0.00	0	96.01	11.50	0.02	100.00	0
10	20	10959.99	1.49	0.03	4.40	0.30	6	0.74	0.02	1.60	0.00	0	117.33	1.17	0.02	100.00	0
15	10	8429.88	0.72	0.01	4.40	0.50	4	0.46	0.01	1.80	0.00	0	74.87	7.35	0.02	100.00	0
15	15	15541.68	7.51	0.06	2.20	0.00	8	0.70	0.02	1.20	0.00	0	139.88	12.88	0.03	100.00	0
15	20	16256.49	10.52	0.37	1.40	0.00	9	1.85	0.02	1.60	0.00	0	221.41	17.99	0.02	100.00	0
20	10	12817.26	1.64	0.07	3.30	0.30	7	0.82	0.01	1.80	0.00	0	97.35	5.31	0.01	100.00	0
20	15	18051.31	13.43	0.19	1.30	0.00	10	1.22	0.02	1.40	0.00	0	222.29	25.73	0.02	100.00	0
20	20	18044.32	16.56	0.47	1.40	0.00	10	4.90	0.02	1.60	0.00	0	348.12	110.98	0.02	100.00	0
30	10							1.74	0.02	1.70	0.00	0	136.83	23.04	0.02	100.00	0
30	15							3.14	0.02	1.50	0.00	0	384.65	3.80	0.02	100.00	0
30	20							21.18	0.02	1.70	0.00	0	680.10	59.56	0.02	100.00	0
40	10							3.44	0.02	1.80	0.00	0	192.59	25.94	0.02	100.00	0
40	15							8.76	0.02	1.67	0.11	0	1097.52	55.36	0.02	100.00	0
40	20							43.72	0.02	1.50	0.00	0	1334.02	133.03	0.02	100.00	0
50	10							8.83	0.01	2.00	0.00	0	393.78	185.41	0.06	100.00	0
50	15							26.72	0.02	1.90	0.10	0	1323.60	115.87	0.01	100.00	0
50	20							2069.65	0.05	1.56	0.11	1	3576.04	1065.25	0.05	100.00	0
60	10							17.83	0.01	2.00	0.00	0	601.17	212.09	0.01	100.00	0
60	15							84.35	0.02	1.78	0.00	0	3603.43	478.58	0.02	92.89	1
60	20							4889.55	0.31	1.40	0.00	2	4086.73	710.93	0.10	89.50	0

Élasticité non constante : $\beta_k = 5$ pour tout k

		EXACT-1						EXACT-2					Heuristique-1				
#OD	%t	Cpu	Gap	BGap	#it	NbCoupe	Nopt	Cpu	Gap	#it	NbCoupe	Nopt	Cpu	Cpu*	Gap	Nbpoint	Nopt
5	10	1.07	0.03	0.01	3.50	0.10	0	0.04	0.01	1.20	0.00	0	25.78	0.94	0.01	100.00	0
5	15	0.66	0.02	0.02	3.90	0.00	0	0.03	0.02	1.00	0.00	0	29.02	0.30	0.02	100.00	0
5	20	59.36	0.03	0.01	5.60	0.10	0	0.23	0.01	1.50	0.00	0	68.96	1.76	0.01	100.00	0
10	10	1880.22	0.06	0.01	3.70	0.00	1	0.13	0.01	1.20	0.00	0	49.95	1.22	0.01	100.00	0
10	15	29.37	0.02	0.02	4.10	0.00	0	0.11	0.02	1.00	0.00	0	71.80	3.16	0.02	100.00	0
10	20	5519.13	0.21	0.02	5.10	0.60	2	0.58	0.02	1.30	0.00	0	277.81	4.09	0.02	100.00	0
15	10	6216.73	0.67	0.05	3.00	0.20	3	0.32	0.02	1.20	0.00	0	78.93	12.24	0.02	100.00	0
15	15	6139.24	0.19	0.02	3.70	0.10	2	0.30	0.02	1.00	0.00	0	180.52	19.65	0.02	100.00	0
15	20	15137.70	3.39	0.15	2.70	0.40	8	16.49	0.02	1.50	0.00	0	1142.91	96.77	0.02	100.00	0
20	10	10915.82	2.50	0.09	2.10	0.10	6	1.34	0.01	1.30	0.00	0	143.59	19.84	0.01	100.00	0
20	15	12992.08	1.73	0.11	2.40	0.00	6	1.56	0.03	1.11	0.00	0	565.99	18.68	0.03	100.00	0
20	20	18054.27	11.38	0.50	1.30	0.20	10	60.65	0.02	1.60	0.00	0	3807.72	1524.08	0.02	100.00	0
30	10							4.49	0.02	1.40	0.10	0	383.37	80.22	0.02	100.00	0
30	15							24.18	0.01	1.60	0.00	0	3046.98	486.14	0.01	100.00	0
30	20							455.60	0.01	1.80	0.00	0	10427.90	1778.39	0.01	85.40	4
40	10							14.68	0.02	1.40	0.00	0	1061.66	100.74	0.02	100.00	0
40	15							45.63	0.02	1.40	0.10	0	6135.72	965.94	0.02	96.30	2
40	20							2074.54	0.11	1.50	0.00	1	12999.77	3235.77	0.11	68.10	6
50	10							67.74	0.01	1.60	0.20	0	3196.03	457.93	0.01	99.60	1
50	15							234.53	0.02	1.40	0.00	0	8166.61	227.00	0.02	85.30	3
50	20							4487.76	0.24	1.80	0.10	2	14967.34	3672.26	0.29	57.10	6
60	10							224.33	0.01	1.70	0.00	0	5558.49	1542.37	0.01	95.70	1
60	15							639.07	0.01	1.60	0.00	0	10925.23	1912.40	0.01	79.40	4
60	20							6535.68	0.27	1.60	0.00	3	15200.57	4655.41	0.27	55.70	7

Tableau 3.6 Réseaux **Grille** : résultats obtenus avec demande non linéaire (élasticité non constante)

Les remarques faites avec l'élasticité constante restent valables lorsque l'élasticité est non constante à la seule différence que le rapport des temps moyen d'exécution et la différence des erreurs relatives moyennes entre les méthodes diminuent. EXACT-2 est très efficace lorsque l'élasticité de la demande est non constante. L'heuristique 1 fournie des solutions situées en moyenne à 0.1% de la borne supérieure de EXACT-2. Les figures 3.12, 3.13 et 3.14 montrent l'évolution de EXACT-2 et l'heuristique 1.

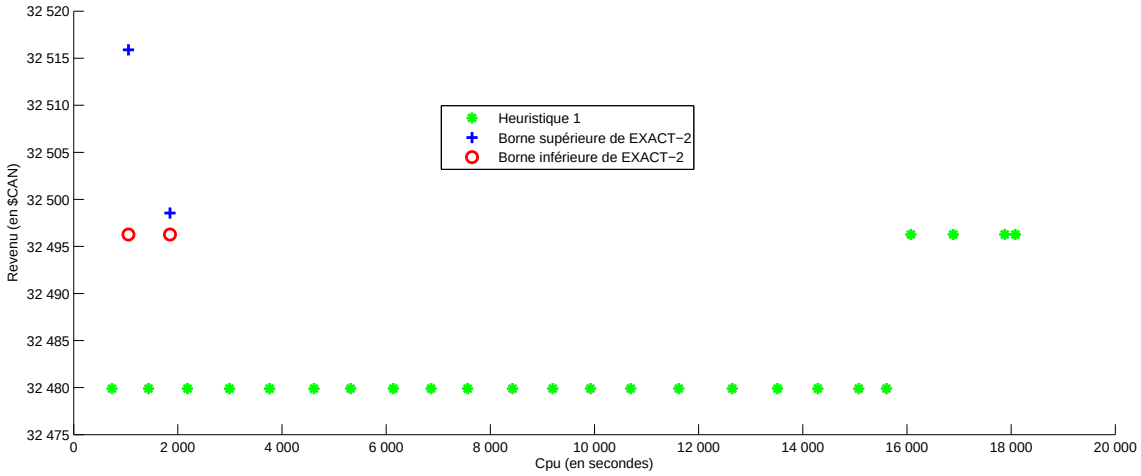


Figure 3.12 Évolution de la méthode EXACT-2 et l'heuristique en fonction du temps d'exécution sur une instance de **Grille** ayant 60 produits, 15% d'arcs taxables et $\beta_k = 5$

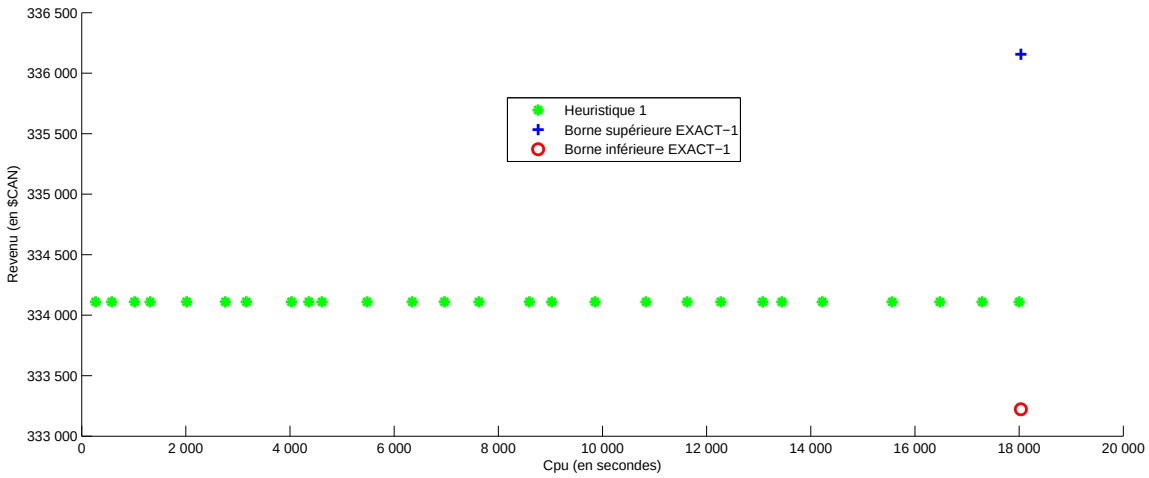


Figure 3.13 Évolution de la méthode EXACT-2 et l'heuristique en fonction du temps d'exécution sur une instance de **Voronoï** ayant 60 produits, 20% d'arcs taxables et $\beta_k = 2$

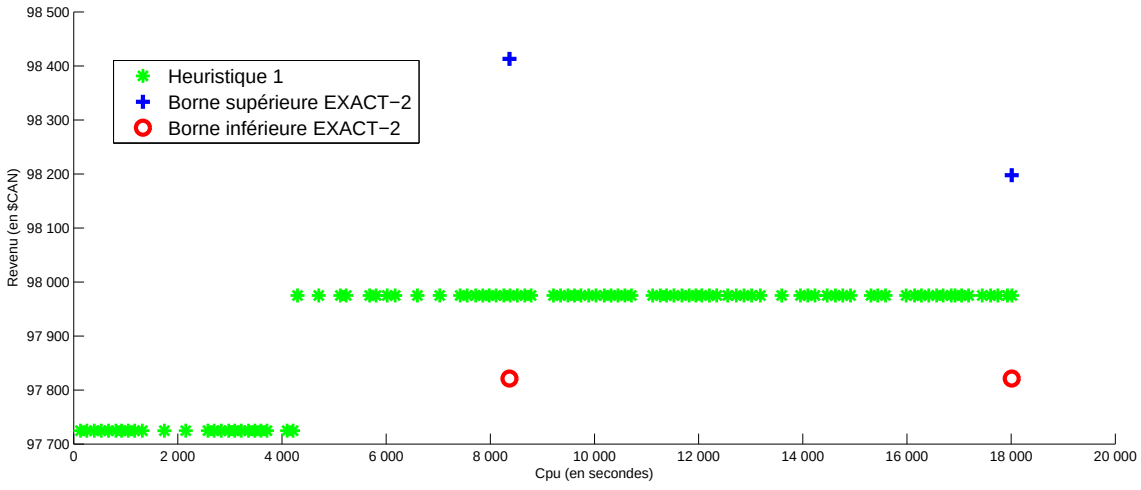


Figure 3.14 Évolution de la méthode EXACT-2 et l'heuristique en fonction du temps d'exécution sur une instance de **Voronoï** ayant 60 produits, 15% d'arcs taxables et $\beta_k = 2$

Finalement, nous illustrons à la figure 3.15 l'évolution du temps de calcul en fonction de l'élasticité de la demande lorsque la méthode EXACT-2 est utilisée. Cette figure est obtenue en fixant le temps d'exécution maximal à 10 heures sur les instances de Grille de 40 produits et 20% d'arcs taxables. Les valeurs $(x\%, y\%)$ sur la courbe représentent l'erreur relative moyenne $(x\%)$ et le pourcentage d'instances ayant atteint le temps d'exécution maximal $(y\%)$.

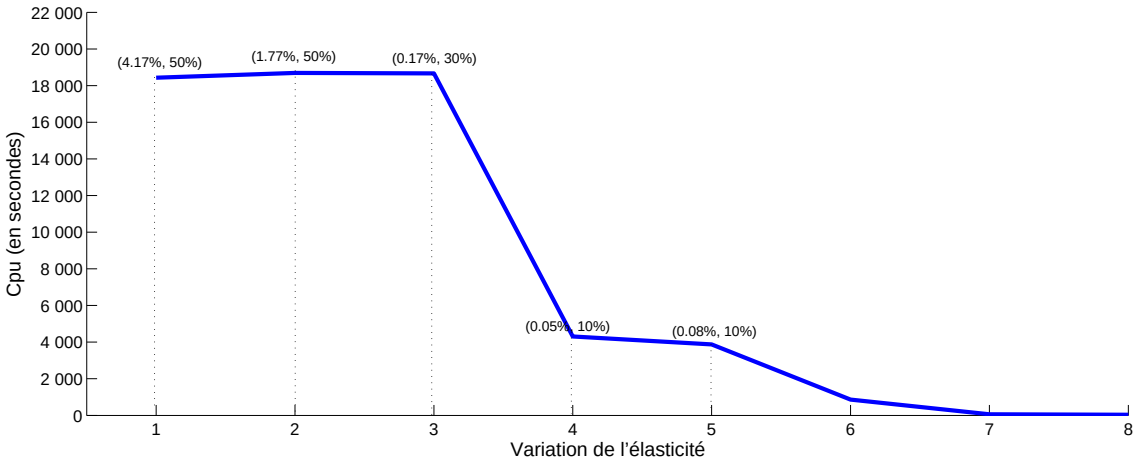


Figure 3.15 Variation du temps de calcul en fonction de l'élasticité de la demande

Cette courbe montre que le temps de calcul, l'erreur relative et le pourcentage d'instances ayant atteint le temps maximal diminue avec l'élasticité de la demande.

3.8 Conclusion

Dans ce chapitre, nous avons développé deux méthodes exactes pour résoudre le DNL. Ces méthodes exactes sont basées sur la surapproximation et la sous-approximation des fonctions non linéaires qui apparaissent dans le modèle DNL:CH. Puis ces méthodes exploitent la résolution du POI associé au DNL dans le but de déterminer le meilleur revenu et d'accélérer si possible la résolution du problème. Deux heuristiques basées sur une méthode de région de confiance ont également été développées. Pour les tests numériques, nous avons considéré deux classes de fonctions de demande : les fonctions de demande à élasticité constante et non

constante.

L'analyse de sensibilité montre que la difficulté de résolution du DNL en fonction de l'élasticité de la demande dépend de la méthode exacte utilisée. En utilisant EXACT-1, elle augmente avec l'élasticité tandis qu'en utilisant EXACT-2 elle diminue avec l'élasticité. La méthode EXACT-2 est plus efficace que EXACT-1. Elle est très efficace lorsque l'élasticité est forte. Cependant lorsque l'élasticité est faible, elle fournit des solutions de bonne qualité avec des temps de calcul très élevés.

Les heuristiques développées sont efficaces lorsque l'élasticité est faible et donnent des solutions situées en moyenne à 1% de la borne supérieure fournie par EXACT-2 avec des temps de calcul raisonnable. Lorsque l'élasticité est forte, elles donnent également des solutions de bonne qualité cependant leurs temps de calcul sont supérieurs à ceux de EXACT-2. Le temps de calcul des heuristiques diminue lorsque l'élasticité augmente.

CHAPITRE 4

PROBLÈME DE TARIFICATION SUR UN RÉSEAU AVEC DEMANDE ÉLASTIQUE ET CONTRAINTES DE CAPACITÉ

Nous étudions dans ce chapitre le problème de tarification sur un réseau avec demande élastique et contraintes de capacité (DE-CAP). Nous commençons par présenter à la section 4.1 la formulation mathématique du DE-CAP puis nous montrons que les contraintes de capacité peuvent être déplacées du second niveau vers le premier niveau. À la section 4.2, nous considérons le cas où la demande est linéaire et nous développons un programme quadratique en variables mixtes (MIP) pour le résoudre. La section 4.3 est consacré au problème avec demande non linéaire où nous développons deux méthodes exactes et une heuristique.

4.1 Formulation mathématique du DE-CAP

Nous considérons le problème où le meneur veut maximiser son revenu en imposant des tarifs sur un ensemble d'arcs et les usagers utilisent le réseau pour acheminer leur produit tout en minimisant leur coût de transport. Tel que mentionné à la section 1.3.3, le problème du second niveau est un problème d'équilibre-usager avec demande élastique et contraintes de capacité. La formulation mathématique est donnée en utilisant une formulation par chemins car nous avons vu au chapitre 2 que cette formulation donnait des meilleurs résultats que la formulation par arcs.

Soit $d_k = f_k(U_k)$ la demande du produit k, f_k une fonction non négative, continue et strictement décroissante et donc inversible. On suppose que les fonctions de demande sont séparables. U_k désigne le coût maximal des chemins utilisés pour acheminer le produit k. Dans le cas où il n'y a pas de contraintes de capacité, les chemins utilisés pour acheminer le produit k ont le même coût et dans ce cas U_k représente ce coût.

Dans le cas où nous introduisons les contraintes de capacité, Beckmann *et al.* [8] ont montré que les conditions d'optimalité du problème d'équilibre-usager avec demande élastique et contraintes de capacité sont données par :

$$h_p^k > 0 \Longrightarrow U_k = \sum_{a\in p} c_a + \sum_{a\in p\cap\mathcal{A}_1} t_a + \sum_{a\in p\cap\mathcal{A}_1} \theta_a \qquad \forall p \in P_k, \forall k \in \mathcal{K} \tag{4.1}$$

$$h_p^k = 0 \Longrightarrow U_k \leq \sum_{a\in p} c_a + \sum_{a\in p\cap\mathcal{A}_1} t_a + \sum_{a\in p\cap\mathcal{A}_1} \theta_a \qquad \forall p \in P_k, \forall k \in \mathcal{K} \tag{4.2}$$

$$d_k > 0 \Longrightarrow f_k^{-1}(d_k) = U_k \qquad \forall k \in \mathcal{K} \tag{4.3}$$

$$d_k = 0 \Longrightarrow f_k^{-1}(d_k) \leq U_k \qquad \forall k \in \mathcal{K} \tag{4.4}$$

où θ est le vecteur de variables duales associé aux contraintes de capacité. la variable $\theta_a, a \in \mathcal{A}_1$ mesure l'écart entre le coût des chemins saturés et le coût minimal des chemins disponibles « non saturés ». Les conditions (4.1) et (4.2) imposent que les chemins utilisés soient de plus petits coûts tandis que (4.3) et (4.4) impliquent que la demande d'un produit donné est une fonction du coût maximal des chemins utilisés dudit produit et qu'elle est nulle si le coût maximal des chemins utilisés est très élevé.

L'écriture de (4.1)-(4.4) sous la forme de contraintes nous conduit à :

$$h_p^k \left(\sum_{a\in p} c_a + \sum_{a\in p\cap\mathcal{A}_1} t_a + \sum_{a\in p\cap\mathcal{A}_1} \theta_a - U_k \right) = 0 \qquad \forall p \in P_k, \forall k \in \mathcal{K} \tag{4.5}$$

$$\sum_{a\in p} c_a + \sum_{a\in p\cap\mathcal{A}_1} t_a + \sum_{a\in p\cap\mathcal{A}_1} \theta_a - U_k \geq 0 \qquad \forall p \in P_k, \forall k \in \mathcal{K} \tag{4.6}$$

$$d_k \left(U_k - f_k^{-1}(d_k) \right) = 0 \qquad \forall k \in \mathcal{K} \tag{4.7}$$

$$U_k - f_k^{-1}(d_k) \geq 0 \qquad \forall k \in \mathcal{K}. \tag{4.8}$$

D'après les hypothèses énoncées ci-dessus sur les fonctions de demande f_k, les contraintes (4.7) et (4.8) peuvent être supprimées car $d_k = f_k(U_k) \Longleftrightarrow U_k = f_k^{-1}(d_k)$. Par la suite, nous considérons les conditions d'optimalité sans ces deux contraintes.

Beckmann *et al.* [8] ont par la suite montré que le problème d'optimisation qui correspond aux conditions d'optimalité (4.5)-(4.8) du problème du second niveau se formule comme suit :

$$\min_{h,d} \sum_{k\in\mathcal{K}} \left(\sum_{p\in P_k} \int_0^{h_p^k} \left(\sum_{a\in p} c_a + \sum_{a\in p\cap\mathcal{A}_1} t_a \right) sds - \int_0^{d_k} f_k^{-1}(s)ds \right)$$

$$\begin{aligned} \text{s.c} \quad & \sum_{p\in P_k} h_p^k = d_k && \forall\ k \in \mathcal{K} \\ & \sum_{k\in\mathcal{K}} \sum_{p\in P_k | a\in p} h_p^k \leq u_a && \forall a \in \mathcal{A}_1 \cup \mathcal{A}_2 \\ & h, d \geq 0 \end{aligned}$$

Finalement, le problème d'optimisation à deux niveaux du DE-CAP est :

CH:SN

$$\max_{t,h'} \sum_{k\in\mathcal{K}} \sum_{p\in P_k} \sum_{a\in p\cap\mathcal{A}_1} t_a h_p^{'k}$$

$$h' \in \arg\min_{h,U,d} \sum_{k\in\mathcal{K}} \left(\sum_{p\in P_k} \int_0^{h_p^k} \left(\sum_{a\in p} c_a + \sum_{a\in p\cap\mathcal{A}_1} t_a \right) sds - \int_0^{d_k} f_k^{-1}(s)ds \right)$$

$$\begin{aligned} \text{s.c} \quad & \sum_{p\in P_k} h_p^k = d_k && \forall\ k \in \mathcal{K} \\ & \sum_{k\in\mathcal{K}} \sum_{p\in P_k | a\in p} h_p^k \leq u_a && \forall a \in A_1 \\ & h, d \geq 0 \end{aligned}$$

En remplaçant le problème du second niveau par ses conditions d'optimalité présentées ci-

dessus ((4.5) et (4.6)), nous obtenons le programme à un niveau suivant :

CH:SN'

$$\max_{t,h,U,\theta} \sum_{k\in\mathcal{K}} \sum_{p\in P_k} \sum_{a\in p\cap\mathcal{A}_1} t_a h_p^k$$

$$\sum_{p\in P_k} h_p^k \;=\; f_k(U_k) \qquad \forall k \in \mathcal{K}$$

$$\sum_{k\in\mathcal{K}} \sum_{p\in P_k \;|\; a\in p} h_p^k \;\leq\; u_a \qquad \forall a \in \mathcal{A}_1 \cup \mathcal{A}_2$$

$$\left(\sum_{a\in p} c_a + \sum_{a\in p\cap\mathcal{A}_1} t_a + \sum_{a\in p\cap\mathcal{A}_1} \theta_a - U_k\right) h_p^k \;=\; 0 \qquad \forall p \in P_k,\ \forall k \in \mathcal{K}$$

$$\sum_{a\in p} c_a + \sum_{a\in p\cap\mathcal{A}_1} t_a + \sum_{a\in p\cap\mathcal{A}_1} \theta_a - U_k \;\geq\; 0 \qquad \forall p \in P_k,\ \forall k \in \mathcal{K}$$

$$h, \theta \;\geq\; 0$$

4.1.1 Déplacement des contraintes de capacité du second niveau vers le premier niveau

Brotcorne *et al.* [14] ont montré que, sous certaines hypothèses, le PTR avec contraintes de capacité est équivalent au problème obtenu en déplaçant les contraintes de capacité du second niveau vers le premier niveau. Nous montrons également que les contraintes de capacité du DE-CAP peuvent être déplacées vers le premier niveau.

En déplaçant les contraintes de capacité au premier niveau, nous déduisons que les chemins utilisés pour un produit donné ont le même coût. Dans ce cas, le problème du second niveau se réduit à un problème de plus court chemin par produit comme dans le cas du problème sans contraintes de capacité. Dès lors, la technique utilisée pour reformuler le problème sans contraintes de capacité en un programme à un niveau est utilisée pour obtenir un programme à un niveau du DE-CAP. Par la suite, les méthodes développées pour le problème sans contraintes de capacité sont adaptées pour résoudre le DE-CAP.

Supposons que tous les arcs non taxables ont une capacité supérieure ou égale à la demande

totale de tous les produits c'est-à-dire que les arcs de $\mathcal{A}_2$ peuvent supporter la demande totale des produits.

Proposition 4.1.1 *En supposant que la capacité des arcs de $\mathcal{A}_2$ sont supérieure ou égale à la demande totale, les contraintes de capacité peuvent être déplacées du second niveau vers le premier niveau.*

Preuve Soit CH:PN, le programme mathématique du DE-CAP où les contraintes de capacité sont déplacées du second niveau vers le premier niveau. CH:PN est une restriction de CH:SN car les contraintes sont contraignantes pour le meneur et pas pour les usagers. Nous commençons par écrire CH:PN sous la forme d'un programme mathématique à un niveau en remplaçant le problème du second niveau par ses conditions d'optimalité de Kuhn-Tucker. Le programme mathématique obtenu est :

CH:PN'

$$\max_{t,h,U} \sum_{k\in\mathcal{K}} \sum_{p\in P_k} \sum_{a\in p\cap\mathcal{A}_1} t_a h_p^k$$

$$\sum_{p\in P_k} h_p^k \;=\; f_k(U_k) \qquad \forall k \in \mathcal{K} \tag{4.9}$$

$$\sum_{k\in\mathcal{K}} \sum_{p\in P_k \;|\; a\in p} h_p^k \;\leq\; u_a \qquad \forall a \in \mathcal{A}_1 \tag{4.10}$$

$$\left(\sum_{a\in p} c_a + \sum_{a\in p\cap\mathcal{A}_1} t_a - U_k\right) h_p^k \;=\; 0 \qquad \forall p \in P_k,\ \forall k \in \mathcal{K} \tag{4.11}$$

$$\sum_{a\in p} c_a + \sum_{a\in p\cap\mathcal{A}_1} t_a - U_k \;\geq\; 0 \qquad \forall p \in P_k,\ \forall k \in \mathcal{K} \tag{4.12}$$

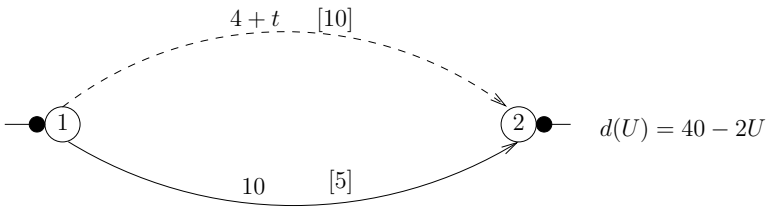
$$h \;\geq\; 0.$$

Soit (t^*, h^*) une solution optimale de CH:PN et U^* le vecteur de variables duales associées aux contraintes du second niveau de CH:PN. En posant $\theta^* = 0$, la solution $(t^*, h^*, U^*, 0)$ est réalisable pour CH:SN' avec la même valeur objectif que celle de CH:PN', dès lors la valeur optimale de CH:SN (ou CH:SN') est au moins égale à celle de CH:PN. Ce résultat confirme le

fait que CH:PN est une restriction de CH:SN. Par conséquent, il suffit de montrer que toute solution optimale de CH:SN peut être associée à une solution de CH:PN avec la même valeur objectif.

Soient (t^*, h^*) une solution optimale de CH:SN et $(t^*, h^*, U^*, \theta^*)$ la solution correspondante de CH:SN' où U^* et θ^* sont les vecteurs de variables duales associées aux contraintes du second niveau de CH:SN. Soit $(t^* + \theta^*, h^*, U^*)$ la solution construite à partir de $(t^*, h^*, U^*, \theta^*)$ en imposant à l'arc taxable a le tarif $t_a^* + \theta_a^*$. La solution $(t^* + \theta^*, h^*, U^*)$ est valide car d'après l'hypothèse les arcs non taxables ont une capacité supérieure à la demande totale et donc $\theta_a^* = 0$ pour tout $a \in \mathcal{A}_2$. $(t^* + \theta^*, h^*, U^*)$ est réalisable pour CH:PN' et sa valeur objectif est supérieure ou égale à celle de $(t^*, h^*, U^*, \theta^*)$ car $\theta^* \geq 0$. Puisque CH:SN' est une relaxation de CH:PN' (toute solution de CH:PN' est une solution de CH:SN'), alors la valeur objectif de $(t^*, h^*, U^*, \theta^*)$ est supérieure ou égale à celle de $(t^* + \theta^*, h^*, U^*)$, ce qui implique que $s1$ est optimale pour CH:PN'. Les solutions optimales $(t^*, h^*, U^*, \theta^*)$ et $(t^* + \theta^*, h^*, U^*)$ ont donc la même valeur, par conséquent $\theta^* = 0$ et (t^*, h^*) est optimale pour CH:PN. □

La proposition 4.1.1 n'est plus valide si certaines capacités sur les arcs de $\mathcal{A}_2$ ne sont pas supérieures ou égales à la demande totale des produits comme le montre l'exemple suivant où l'on considère le réseau ci-dessous où un gestionnaire (meneur) veut maximiser le revenu généré par le tarif de l'arc taxable (en pointillé).



Les usagers peuvent emprunter les chemins h_1 ou h_2. Le chemin h_1, celui du meneur, est constitué de l'arc taxable et le chemin h_2, celui de la compétition, est constitué de l'arc non taxable. Chaque arc a une capacité finie (valeur entre crochets sur les arcs). La demande est linéaire, $d(U) = 40 - 2U$, où U est le coût maximal des chemins utilisés. La variable associée à la contrainte de capacité de l'arc taxable (resp. non taxable) est θ_1 (resp. θ_2). Le programme

mathématique correspondant est :

$$\begin{aligned}
\max_{t,h_1,h_2,U,\theta_1,\theta_2} \quad & t \times h_1 \\
& h_1 + h_2 \;=\; 40 - 2U \\
& h_1 \;\leq\; 10 \\
& h_2 \;\leq\; 5 \\
& (4 + t + \theta_1)h_1 \;=\; 0 \\
& (10 + \theta_2)h_2 \;=\; 0 \\
& U \;\leq\; 4 + t + \theta_1 \\
& U \;\leq\; 10 + \theta_2 \\
& h_1, h_2, \theta_1, \theta_2 \geq 0.
\end{aligned}$$

La valeur maximale de $h_1 + h_2 (= 40 - 2U)$ est 15 ce qui implique que $U \geq 12.5$. Puisque $U \geq 12.5$ et que le coût du chemin de la compétition (h_2) est 10 alors $U = 4 + t \geq 12.5 \Longrightarrow t \geq 8.5$ et $\theta_1 = 0$ car h_1 est le chemin de coût maximal. On déduit qu'à l'optimalité, le chemin de la compétition est saturé ($h_2 = 5$) car ce chemin a un coût inférieur à celui du chemin h_1. En remplaçant h_1 par $35 - 2U$ et U par $4 + t$, la valeur optimale de t est donnée par le programme quadratique suivant :

$$\begin{aligned}
\max_{t} \quad & t(35 - 2(4 + t)) = 27t - 2t^2 \\
& t \;\geq\; 8.5.
\end{aligned}$$

La solution optimale du programme ci-dessus est obtenue pour la valeur $t = 8.5$. On déduit que $U = 12.5$, $\theta_2 = 2.5$, $h_1 = 10$. Les deux chemins sont saturés à l'optimalité. La valeur de $\theta_2 = 2.5 \neq 0$ et les coûts des chemins utilisés sont distincts.

Nous résolvons le DE-CAP où les arcs taxables (resp. non taxables) ont une capacité finie (resp. infinie). Le fait que les arcs taxables appartenant au meneur ont une capacité finie est

une caractéristique de la gestion du revenu car autrement le meneur pourrait imposer des tarifs qui inciteront les usagers à n'utiliser que ses arcs. La capacité des arcs non taxables, appartenant à la compétition, ont une capacité supérieure à la demande totale des produits car on suppose que les usagers doivent pouvoir acheminer toute leur demande en produits en n'utilisant que les arcs de $\mathcal{A}_2$ si le meneur imposait des tarifs élevés sur ses arcs. À partir de cette hypothèse, la propriété 4.1.1 est vraie et les contraintes de capacité sont déplacées vers le premier niveau. En déplaçant les contraintes de capacité au premier niveau, les chemins utilisés d'un produit donné ont des coûts égaux. Cette propriété est exploitée pour réécrire les contraintes de complémentarité et la fonction objectif du programme à un niveau CH:PN'. Ce nouveau programme mathématique est par la suite utilisé pour développer des algorithmes de résolution du DE-CAP.

4.1.2 Réécriture des contraintes de complémentarité et de la fonction objectif de CH:PN'

Soit γ_p^k la variable binaire qui indique si le chemin p du produit k est utilisé ou non c'est-à-dire

$$\gamma_p^k = \begin{cases} 1 & \text{si le chemin } p \text{ du produit } k \text{ est utilisé} \\ 0 & \text{sinon.} \end{cases}$$

La contrainte de complémentarité (4.11) et la contrainte (4.12) de CH:PN'

$$\left(\sum_{a \in p} c_a + \sum_{a \in p \cap \mathcal{A}_1} t_a - U_k \right) h_p^k \;=\; 0 \qquad \forall p \in P_k$$

$$\sum_{a \in p} c_a + \sum_{a \in p \cap \mathcal{A}_1} t_a - U_k \;\geq\; 0 \qquad \forall p \in P_k$$

peuvent être remplacées par l'ensemble de contraintes linéaires suivant :

$$\sum_{a\in p} c_a + \sum_{a\in p\bigcap \mathcal{A}_1} t_a - M_p^k(1-\gamma_p^k) \;\leq\; U_k \qquad \forall p \in P_k \tag{4.13}$$

$$\sum_{a\in p} c_a + \sum_{a\in p\in \mathcal{A}_1} t_a - U_k \;\geq\; 0 \qquad \forall p \in P_k \tag{4.14}$$

$$h_p^k \leq N_p^k \gamma_p^k \qquad \forall\; p \in P_k \tag{4.15}$$

$$\gamma_p^k \in \{0,1\} \qquad \forall\; p \in P_k \tag{4.16}$$

où M_p^k est une grande constante identique à celle définie par Dewez [27] et $N_p^k = \min\limits_{a\in p\cap\mathcal{A}_1}\{u_a\}$.

Les contraintes (4.13) et (4.14) assurent que les chemins utilisés du produit k aient le même coût et que ce coût soit inférieur aux coûts des chemins non utilisés du produit k. Les contraintes (4.15) imposent un flot nul aux chemins de coût supérieur à U_k car la variable γ_p^k associée au chemin p vaut 0 si son coût est supérieur à U_k (voir contrainte (4.13)).

Nous pouvons également réécrire la fonction objectif du produit k en utilisant la variable U_k et la demande d_k comme suit :

$$\begin{aligned}
\sum_{p\in P_k}\sum_{a\in p\bigcap\mathcal{A}_1} t_a h_p^k &= \sum_{p\in P_k}\left(\sum_{a\in p} c_a + \sum_{a\in p\bigcap\mathcal{A}_1} t_a\right) h_p^k - \sum_{p\in P_k} h_p^k \sum_{a\in p} c_a \\
&= \sum_{p\in P_k} U_k h_p^k - \sum_{p\in P_k} h_p^k \sum_{a\in p} c_a \qquad \text{car les coûts des chemins utilisés sont égaux} \\
&= U_k \sum_{p\in P_k} h_p^k - \sum_{p\in P_k} h_p^k \sum_{a\in p} c_a \qquad \text{or} \quad \sum_{p\in P_k} h_p^k = d_k \\
&= U_k d_k - \sum_{p\in P_k} h_p^k \sum_{a\in p} c_a. \qquad (4.17)
\end{aligned}$$

En remplaçant dans CH:PN' la fonction objectif et les contraintes (4.9)-(4.12) du produit k respectivement par l'expression (4.17) et les contraintes (4.13)-(4.16), nous obtenons le

programme non linéaire en variables mixtes suivant

DE-CAP:CH

$$\max_{t,h,U} \sum_{k\in\mathcal{K}} U_k d_k - \sum_{p\in P_k} h_p^k \sum_{a\in p} c_a$$

$$\sum_{p\in P_k} h_p^k \;=\; d_k \qquad \forall k \in \mathcal{K} \tag{4.18}$$

$$\sum_{k\in\mathcal{K}} \sum_{p\in P_k \;|\; a\in p} h_p^k \;\leq\; u_a \qquad \forall a \in \mathcal{A}_1 \tag{4.19}$$

$$\sum_{a\in p} c_a + \sum_{a\in p \bigcap \mathcal{A}_1} t_a - M_p^k(1-\gamma_p^k) \;\leq\; U_k \qquad \forall p \in P_k, \forall k \in \mathcal{K} \tag{4.20}$$

$$\sum_{a\in p} c_a + \sum_{a\in p\cap \mathcal{A}_1} t_a - U_k \;\geq\; 0 \qquad \forall p \in P_k, \forall k \in \mathcal{K} \tag{4.21}$$

$$h_p^k \leq N_p^k \gamma_p^k \qquad \forall\; p \in P_k, \forall k \in \mathcal{K} \tag{4.22}$$

$$h_p^k \geq 0 \qquad \forall\; p \in P_k, \forall k \in \mathcal{K}$$

$$\gamma_p^k \in \{0,1\} \qquad \forall\; p \in P_k, \forall k \in \mathcal{K}.$$

Le programme mathématique DE-CAP:CH est utilisé pour développer les algorithmes de résolution du DE-CAP.

4.2 Problème de tarification sur un réseau avec demande linéaire et contraintes de capacité

4.2.1 Programme mathématique

Nous supposons que la fonction de demande du produit k est $d_k = f_k(U_k) = a_k - b_k U_k$ où a_k, b_k sont des constantes positives. À partir de DE-CAP:CH, nous remplaçons la demande d_k du produit k par sa valeur $a_k - b_k U_k$ et nous obtenons le programme quadratique mixte

sous contraintes linéaires suivant

DL-CAP:CH

$$\max_{t,h,U} \sum_{k\in\mathcal{K}} \left(aU_k - b_k(U_k)^2 - \sum_{p\in P_k} h_p^k \sum_{a\in p} c_a \right)$$

$$\sum_{p\in P_k} h_p^k \;=\; a_k - b_k U_k \qquad \forall k \in \mathcal{K} \tag{4.23}$$

$$\sum_{k\in\mathcal{K}} \sum_{p\in P_k \;|\; a\in p} h_p^k \;\leq\; u_a \qquad \forall a \in \mathcal{A}_1 \tag{4.24}$$

$$\sum_{a\in p} c_a + \sum_{a\in p \bigcap \mathcal{A}_1} t_a - M_p^k(1-\gamma_p^k) \;\leq\; U_k \qquad \forall p \in P_k, \forall k \in \mathcal{K} \tag{4.25}$$

$$\sum_{a\in p} c_a + \sum_{a\in p\cap \mathcal{A}_1} t_a - U_k \;\geq\; 0 \qquad \forall p \in P_k, \forall k \in \mathcal{K} \tag{4.26}$$

$$h_p^k \leq N_p^k \gamma_p^k \qquad \forall\; p \in P_k, \forall k \in \mathcal{K} \tag{4.27}$$

$$h_p^k \geq 0 \qquad \forall\; p \in P_k, \forall k \in \mathcal{K}$$

$$\gamma_p^k \in \{0,1\} \qquad \forall\; p \in P_k, \forall k \in \mathcal{K}.$$

Soit DL:CH2, le meilleur MIP obtenu pour le DL. Le modèle DL-CAP:CH présenté ci-dessus contient le même nombre de variables binaires (associées aux chemins) que DL:CH2 et les mêmes contraintes à l'exception des contraintes de capacité. La différence majeure entre les deux modèles se situe au niveau de l'espace des solutions des variables binaires. En fait, l'espace des solutions des variables binaires du DL:CH2 est inclus dans celui du DL-CAP:CH car un seul chemin est utilisé pour chaque produit du DL:CH2 tandis qu'au moins un chemin est utilisé pour chaque produit du DL-CAP:CH.

4.2.2 Fonction objectif du meneur

Tel que présenté pour le DL, on montre que :

Proposition 4.2.1 *La fonction objectif du meneur du DL-CAP est quadratique concave par morceaux, discontinue et semi-continue supérieurement. Elle peut être linéaire sur certains*

morceaux (dès que certaines contraintes de capacité sont actives).

Preuve Supposons que, pour chaque produit k, l'on connaisse l'ensemble des chemins utilisés $p_1, \ldots, p_{n_k}$. Sans perte de généralité, supposons qu'il y ait au maximum un chemin non saturé parmi les chemins utilisés. Dans le cas contraire (solution dégénérée), il existe une réaffectation des flots sur ces chemins qui satisfait cette hypothèse. Supposons que les chemins saturés soient p_i, $i = 1, \ldots, n_k - 1$ et que leur flot soit donné par v_i^k. Le flot sur le chemin p_{n_k} est déterminé en utilisant la contrainte de conservation de flot (4.23) comme suit :

$$\begin{aligned}
\sum_{i=1}^{n_k} h_{p_i}^k = a_k - b_k U_k \Longleftrightarrow h_{p_{n_k}}^k &= a_k - b_k U_k - \sum_{i=1}^{n_k-1} h_{p_i}^k \\
&= a_k - b_k U_k - \sum_{i=1}^{n_k-1} v_i^k \\
&= a_k - b_k \left(\sum_{a \in p_{n_k}} c_a + \sum_{a \in p_{n_k} \cap \mathcal{A}_1} t_a \right) - \sum_{i=1}^{n_k-1} v_i^k \\
&= a_k - b_k \sum_{a \in p_{n_k}} c_a - b_k \sum_{a \in p_{n_k} \cap \mathcal{A}_1} t_a - \sum_{i=1}^{n_k-1} v_i^k.
\end{aligned}$$

La fonction objectif du produit k, $\sum_{i=0}^{n_k} h_i^k \sum_{a \in p_i \cap \mathcal{A}_1} t_a$, devient :

$$\begin{aligned}
&\sum_{i=0}^{n_k-1} h_i^k \sum_{a \in p_i \cap \mathcal{A}_1} t_a + h_{n_k}^k \sum_{a \in p_{n_k} \cap \mathcal{A}_1} t_a \\
&= \sum_{i=0}^{n_k-1} v_i^k \sum_{a \in p_i \cap \mathcal{A}_1} t_a + \left(a_k - b_k \sum_{a \in p_{n_k}} c_a - b_k \sum_{a \in p_{n_k} \cap \mathcal{A}_1} t_a - \sum_{i=1}^{n_k-1} v_i^k \right) \sum_{a \in p_{n_k} \cap \mathcal{A}_1} t_a \\
&= \sum_{i=0}^{n_k-1} v_i^k \left(\sum_{a \in p_i \cap \mathcal{A}_1} t_a - \sum_{a \in p_{n_k} \cap \mathcal{A}_1} t_a \right) + a_k \sum_{a \in p_{n_k} \cap \mathcal{A}_1} t_a - b_k \sum_{a \in p_{n_k}} c_a \sum_{a \in p_{n_k} \cap \mathcal{A}_1} t_a - b_k \left(\sum_{a \in p_{n_k} \cap \mathcal{A}_1} t_a \right)^2.
\end{aligned}$$

Cette fonction est continue et concave en $t = (t_a)_{a \in \mathcal{A}_1}$ car $b_k > 0$. Si le chemin p_{n_k} est saturé ou s'il ne contient aucun arc taxable, alors la fonction objectif du produit k est linéaire car $h_{n_k}^k = v_{n_k}^k$ ou $p_{n_k} \cap \mathcal{A}_1 = \emptyset$.

Nous concluons que la fonction objectif du meneur est concave (comme somme de fonctions concaves).

Le reste de la preuve est similaire à celle de la proposition 2.3.1 où on montre également que la fonction objectif du meneur est concave par morceau, discontinue et semi-continue supérieurement pour le DL. □

4.2.3 Exemple du DL-CAP

Le but de cet exemple est d'illustrer la propriété précédente et la variation des solutions du DL-CAP en fonction des contraintes de capacité.

Soit le DL-CAP où le réseau est décrit à la figure 4.1 et comporte trois produits.

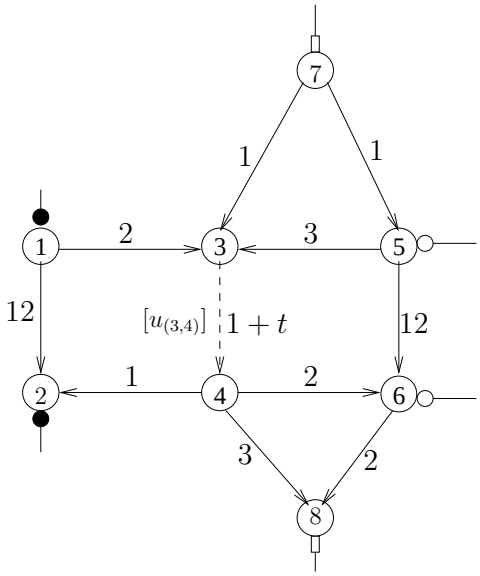


Figure 4.1 Réseau avec un seul arc taxable

Les informations sur les produits sont données au tableau 4.1. Seuls les chemins non dominés sont présentés dans le tableau 4.1 suivant :

	o - d	demande	chemins	coût	flot
produit 1	$1 \longrightarrow 2$	$d_1(U_1) = 42 - 3U_1$	$1 \to 3 \to 4 \to 2$ $1 \to 2$	$4 + t$ 12	h_1^1 h_2^1
produit 2	$5 \longrightarrow 6$	$d_2(U_2) = 35 - 2U_2$	$5 \to 3 \to 4 \to 6$ $5 \to 6$	$6 + t$ 12	h_1^2 h_2^2
produit 3	$7 \longrightarrow 8$	$d_3(U_3) = 25 - U_3$	$7 \to 3 \to 4 \to 8$ $7 \to 5 \to 6 \to 8$	$5 + t$ 15	h_1^3 h_2^3

Tableau 4.1 Données du réseau de la figure 4.1

Dans cette exemple, nous imposons le tarif de l'arc taxable $(3, 4)$ afin de maximiser le revenu généré, tout en sachant que le flot sur l'arc $(3, 4)$ ne doit pas dépasser la valeur $u_{(3,4)}$. Puis, nous faisons une analyse de sensibilité sur cette valeur.

Pour chaque produit, les usagers ont le choix entre deux chemins non dominés qu'ils peuvent emprunter pour acheminer ledit produit. L'un des chemins contient l'arc taxable $(3, 4)$ tandis que l'autre chemin ne contient pas l'arc $(3, 4)$. Puisque le réseau ne comporte qu'un seul arc taxable on peut donner, selon les valeurs du tarif t, les produits pour lesquels l'arc $(3, 4)$ est attractif. Ainsi, l'arc $(3, 4)$ est attractif pour les trois produits si $t \leq 6$, il est attractif seulement pour les produits 1 et 3 si $6 \leq t \leq 8$, il est attractif seulement pour le produit 3 si $8 \leq t \leq 10$ et n'est attractif pour aucun produit si $t > 10$.

La fonction objectif du meneur en fonction du tarif t est donnée par le tableau 4.2 :

$t \in [0, 6]$	$t \times \min\{u_{(3,4)}, 42 - 3(4 + t) + 35 - 2(6 + t) + 25 - (5 + t)\} = t \times \min\left\{u_{(3,4)}, 73 - 6t\right\}$
$t \in]6, 8]$	$t \times \min\{u_{(3,4)}, 42 - 3(4 + t) + 25 - (5 + t)\} = t \times \min\left\{u_{(3,4)}, 50 - 4t\right\}$
$t \in]8, 10]$	$t \times \min\{u_{(3,4)}, 25 - (5 + t)\} = t \times \min\left\{u_{(3,4)}, 20 - t\right\}$
$t \in]10, +\infty[$	0

Tableau 4.2 Fonction objectif dépendant de t et $u_{(3,4)}$

Résolvons le DL-CAP en attribuant les valeurs 75, 36, 15 et 10 à $u_{(3,4)}$.

Pour $u_{(3,4)} = 75$, la capacité de l'arc $(3,4)$ est supérieure à la somme des demandes maximales des trois produits. Le problème est donc équivalent à un problème sans contraintes de capacité. La fonction objectif du meneur en fonction du tarif t est donnée par :

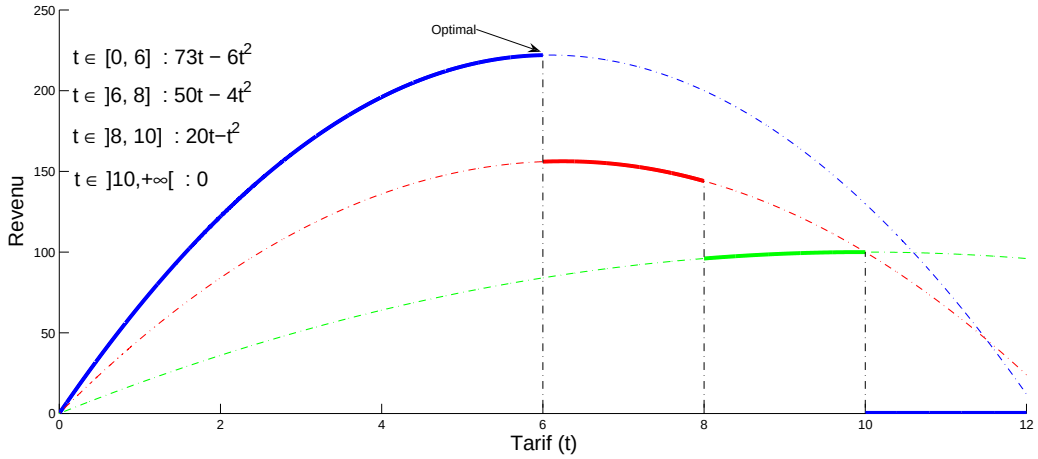


Figure 4.2 Fonction objectif du meneur avec $u_{(3,4)} = 75$

La solution optimale est obtenue pour la valeur de $t = 6$ pour un revenu de 222.

$u_{(3,4)}$	Solution optimale	Revenu optimal	Optimum local
75	$t = 6$ $h_1^1 = 12$ $h_2^1 = 0$ $h_1^2 = 11$ $h_2^2 = 0$ $h_1^3 = 14$ $h_2^3 = 0$	$(12 + 11 + 14) \times 6 = 222$	3 optima locaux avec $t = 6$, 6.25 et 10

Nous avons trois optima locaux pour les valeurs de $t = 6$, 6.25 et 10. La contrainte de capacité n'a aucun effet sur la solution du problème, donc elle peut être ignorée. Dans ce cas, nous dirons que les contraintes de capacité sont « *très faibles* ».

Pour $u_{(3,4)} = 36$, la fonction objectif du meneur en fonction du tarif t est donnée par :

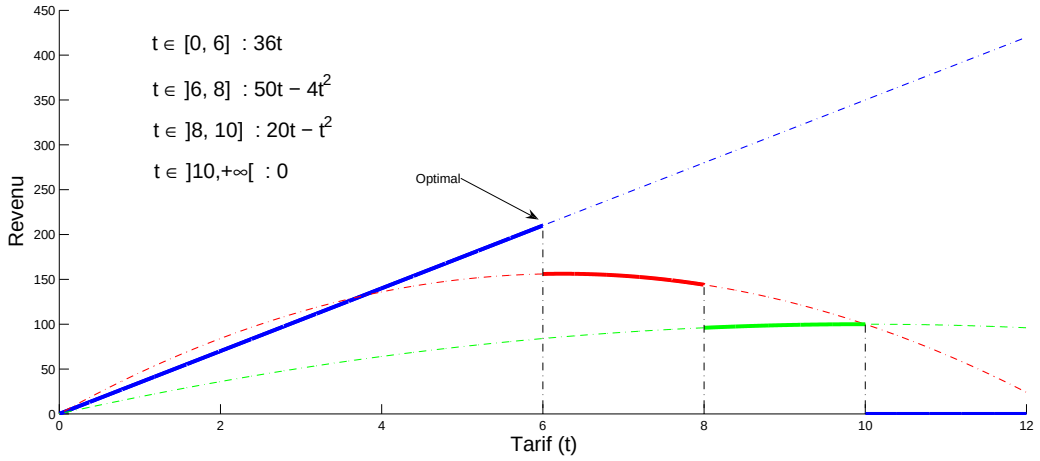


Figure 4.3 Fonction objectif du meneur avec $u_{(3,4)} = 36$

La solution optimale est obtenue pour la valeur de $t = 6$ pour un revenu de 216.

$u_{(3,4)}$	Solution optimale	Revenu optimal	Optimum local
36	$t = 6$ $h_1^1 = 12$ $h_2^1 = 0$ $h_1^2 = 10$ $h_2^2 = 1$ $h_1^3 = 14$ $h_2^3 = 0$	$(12 + 10 + 14) \times 6 = 216$	3 optima locaux avec $t = 6$, 6.25 et 10

Les optima locaux sont obtenus avec les valeurs de $t = 6$, 6.25 et 10. Avec l'incorporation de la contrainte de capacité, nous obtenons un objectif linéaire et croissant sur le sous-intervalle $t \in [0, 6]$ car l'arc $(3, 4)$ est saturé. Le sous-intervalle $t \in [0, 6]$ est le seul où l'arc $(3, 4)$ est saturé. Nous dirons dans ce cas que les contraintes de capacité sont « *faibles* ».

Pour $u_{(3,4)} = 15$, la fonction objectif du meneur en fonction du tarif t est donnée par :

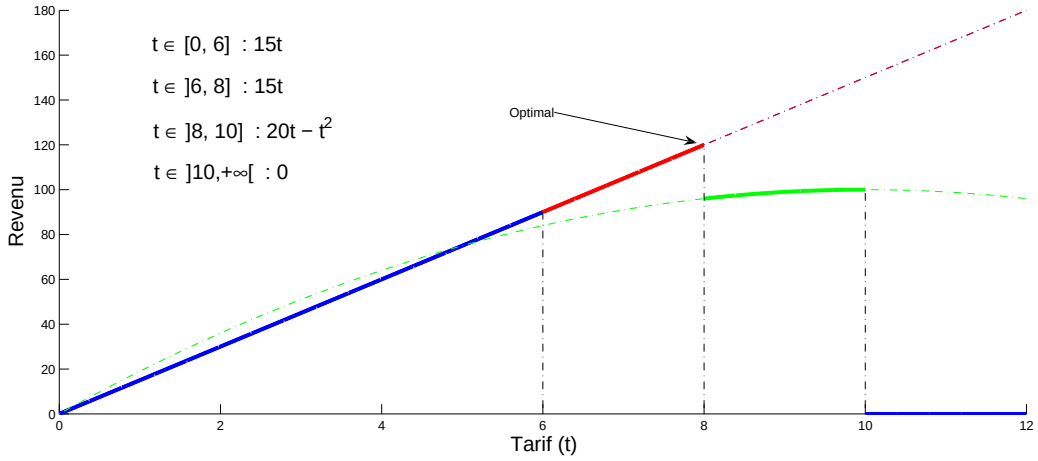


Figure 4.4 Fonction objectif du meneur avec $u_{(3,4)} = 15$

La solution optimale est obtenue pour la valeur de $t = 8$ pour un revenu de 120.

$u_{(3,4)}$	Solution optimale			Revenu optimal	Optimum local
15	$t = 8$	$h_1^1 = 3$	$h_2^1 = 3$	$(3 + 0 + 12) \times 8 = 120$	2 optima locaux
		$h_1^2 = 0$	$h_2^2 = 11$		avec $t = 8$ et 10
		$h_1^3 = 12$	$h_2^3 = 0$		

Pour $t \in [0, 8]$, l'objectif du meneur est linéaire de pente $u_{(3,4)}$ car l'arc $(3, 4)$ est saturé pour $t \in [0, 8]$. La solution correspondante à $t = 6$ n'est plus un optimum local car en augmentant le tarif à partir de cette valeur, le revenu augmente. Le nombre d'optima locaux est donc 2. Dans ce cas, nous dirons que les contraintes de capacité sont « *fortes* ».

Pour $u_{(3,4)} = 10$, la fonction objectif du meneur en fonction du tarif t est donnée par :

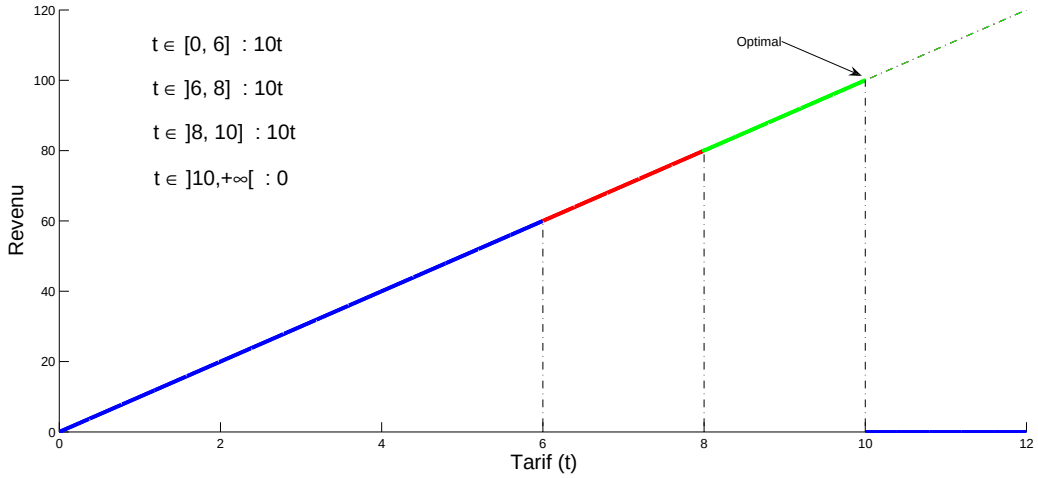


Figure 4.5 Fonction objectif du meneur avec $u_{(3,4)} = 10$

La solution optimale est obtenue pour la valeur de $t = 10$ pour un revenu de 100.

$u_{(3,4)}$	Solution optimale	Revenu optimal	Optimum local
10	$t = 10$ $h_1^1 = 0$ $h_2^1 = 6$ $h_1^2 = 0$ $h_2^2 = 11$ $h_1^3 = 10$ $h_2^3 = 0$	$(0 + 0 + 10) \times 10 = 100$	1 optimum local avec $t = 10$

Nous avons un seul minimum local (qui est global). Pour obtenir la solution optimale, il suffit de fixer t à 10 car l'arc est saturé pour toutes les valeurs de $t \in [0, 10]$. Nous dirons que les contraintes de capacité sont « *très fortes* ». Dans une telle situation où la fonction objectif du meneur est linéaire sur l'intervalle des tarifs, le DL-CAP est concave et toute méthode de recherche locale résout le problème.

Nous avons vu que le nombre d'optima locaux augmente avec les capacités sur les arcs. Intuitivement, nous l'expliquons par le fait qu'avec des contraintes de capacité faibles, la fonction objectif du meneur est définie sur plusieurs morceaux où sur chaque morceau l'objectif est soit concave, soit linéaire. En présence des contraintes de capacité fortes, le nombre de morceaux diminue considérablement. Cependant, nous savons que le nombre d'optima locaux dépend fortement du nombre de morceaux. Dans l'exemple ci-dessus, nous avons vu que pour les valeurs de $u_{(3,4)}$ égales à 75, 36, 15 et 10, la fonction objectif est définie respectivement

sur 4, 4, 3 et 2 morceaux. Le nombre d'optima locaux pour ces valeurs de $u_{(3,4)}$ sont respectivement 3, 3, 2 et 1 soit un de moins que le nombre de morceaux. De ce fait, la difficulté du DL-CAP augmente avec les capacités sur les arcs.

4.2.4 Expérimentations

Les expérimentations effectuées permettent de tester de façon quantitative le modèle proposé. Les paramètres de modélisation ayant servi aux tests numériques sont les mêmes que ceux présentés à la section 2.6.1.

Tel que présenté à la section 4.1.1, nous fixons à l'infini la capacité des arcs non taxables, c'est-à-dire pour tout $a \in \mathcal{A}_2$, $u_a = +\infty$. Pour définir les capacités sur les arcs taxables, nous nous sommes inspirés des stratégies proposées par Hearn *et al.* [39] et Yang *et al.* [70]. Ainsi, la capacité de l'arc $a \in \mathcal{A}_1$ est donnée par

$$u_a \;=\; \frac{1}{|\mathcal{K}_a|} \sum_{k \in \mathcal{K}_a} \left(a_k - b_k \gamma^k(\infty)\right) + \theta \left(\frac{c_a + 1}{\min\limits_{a' \in \mathcal{A}_1} c_{a'} + 1} \right)^{3/2}$$

où $\mathcal{K}_a$ est l'ensemble des produits ayant au moins un chemin qui contient l'arc a. L'expression $\frac{1}{|\mathcal{K}_a|} \sum_{k \in \mathcal{K}_a} \left(a_k - b_k \gamma^k(\infty)\right)$ représente la moyenne des demandes minimales des produits appartenant à $\mathcal{K}_a$, ce qui permet à l'arc a de satisfaire au moins la demande minimale d'un produit de $\mathcal{K}_a$. L'expression $\left[(c_a + 1) \,/\, \left(\min\limits_{a' \in \mathcal{A}_1} c_{a'} + 1 \right) \right]^{3/2}$ permet d'affecter la capacité sur l'arc a en fonction de son coût et du plus petit coût des arcs taxables. En d'autres termes, la capacité d'un arc est proportionnelle au rapport de son coût fixe et du plus petit coût fixe des arcs de $\mathcal{A}_1$. Enfin, θ est le coefficient qui permet de faire varier la capacité sur les arcs afin d'effectuer une analyse de sensibilité sur les contraintes de capacité.

Les expériences numériques ont été effectuées sur la famille d'instances du réseau *Voronoï* et *Grille*. Nous ne considérons que les tarifs positifs et nous varions le nombre de produits entre 10 et 60. La proportion d'arcs taxables est fixée à 10%, 15% et 20%. Afin d'étudier l'effet de l'élasticité de la demande et des contraintes de capacité sur le DL-CAP, une analyse

de sensibilité est faite simultanément sur l'élasticité de la demande et les contraintes de capacité. Pour cela, nous considérons dans un premier temps le DL-CAP avec des fonctions de demande à élasticité forte, moyenne et faible. Puis, nous faisons varier θ en lui attribuant des valeurs de manière à obtenir des contraintes de capacité faibles et fortes sur les arcs taxables du réseau. Tel que présenté dans l'exemple 4.2.3, les contraintes de capacité faibles (fortes) correspondent à l'attribution des grandes (petites) valeurs aux capacités sur les arcs. Les fonctions de demande à élasticité faible, moyenne et forte s'obtiennent en multipliant les pentes des fonctions de demande respectivement par 100, 50 et 1. Les résultats présentés ci-dessous sont une moyenne de 10 instances.

Les résultats numériques de la résolution du DL-CAP par le programme mathématique DL-CAP:CH sont reportés dans les tableaux 4.3 et 4.4 ci-dessous où EXACT-faible (resp. EXACT-forte) représente le MIP, DL-CAP:CH, obtenu avec les contraintes de capacité faibles (resp. fortes).

Élasticité forte $(100 \times b_k)$

Instance		EXACT-forte				EXACT-faible			
#OD	%t	Cpu	Gap	#it	Nopt	Cpu	Gap	#it	Nopt
10	10	0.03	0.00	1.85e+02	0	0.18	0.00	3.15e+03	0
10	15	0.56	0.00	4.37e+03	0	0.95	0.00	8.38e+03	0
10	20	7.83	0.00	5.63e+04	0	2.18	0.00	1.34e+04	0
20	10	0.06	0.00	3.59e+02	0	0.43	0.00	3.82e+03	0
20	15	3.45	0.00	2.09e+04	0	59.20	0.00	3.54e+05	0
20	20	1804.55	0.24	8.33e+06	1	3223.95	0.09	1.40e+07	1
30	10	0.14	0.00	8.58e+02	0	3.00	0.00	2.91e+04	0
30	15	10.31	0.00	4.70e+04	0	6263.99	0.42	3.29e+07	3
30	20	1839.45	0.22	1.39e+06	1	10924.18	1.16	3.74e+07	5
40	10	0.24	0.00	1.23e+03	0	4.96	0.00	4.49e+04	0
40	15	110.34	0.00	4.54e+05	0	4157.10	0.44	1.09e+07	2
40	20	2081.88	0.05	5.45e+06	1	16403.49	3.14	5.93e+07	9
50	10	0.54	0.00	2.65e+03	0	32.77	0.00	2.80e+05	0
50	15	413.39	0.00	1.36e+06	0	9842.15	2.44	3.31e+07	5
50	20	2839.76	0.01	8.25e+06	1	18019.54	5.21	4.92e+07	10
60	10	0.94	0.00	3.91e+03	0	104.77	0.00	7.06e+05	0
60	15	972.11	0.00	3.05e+06	0	13171.39	4.66	4.11e+07	7
60	20	812.47	0.00	2.14e+06	0	18016.60	7.21	4.07e+07	10

Élasticité moyenne $(50 \times b_k)$

Instance		EXACT-forte				EXACT-faible			
#OD	%t	Cpu	Gap	#it	Nopt	Cpu	Gap	#it	Nopt
10	10	0.06	0.00	5.21e+02	0	0.12	0.00	1.48e+03	0
10	15	1.72	0.00	1.14e+04	0	4.80	0.00	4.56e+04	0
10	20	5.03	0.00	3.05e+04	0	7.44	0.00	5.61e+04	0
20	10	0.10	0.00	4.09e+02	0	0.51	0.00	2.58e+03	0
20	15	2.95	0.00	8.08e+03	0	2926.76	0.47	1.03e+07	1
20	20	1806.42	0.41	1.10e+05	1	4006.23	0.45	1.69e+07	2
30	10	0.34	0.00	1.77e+03	0	1.56	0.00	1.51e+04	0
30	15	19.88	0.00	5.11e+04	0	3547.84	0.33	1.09e+07	1
30	20	1821.04	0.15	5.95e+06	1	11372.73	3.50	3.26e+07	6
40	10	0.34	0.00	1.11e+03	0	3.59	0.00	3.28e+04	0
40	15	25.90	0.00	9.57e+04	0	7510.77	1.90	3.72e+07	4
40	20	2098.20	0.94	3.05e+06	1	13787.91	2.73	3.70e+07	7
50	10	0.78	0.00	2.65e+03	0	40.55	0.00	2.31e+05	0
50	15	236.43	0.00	6.14e+05	0	11826.40	5.88	1.47e+07	6
50	20	5441.86	0.39	1.12e+07	3	18040.23	7.03	2.68e+07	10
60	10	1.89	0.00	1.10e+04	0	1018.37	0.00	2.81e+06	0
60	15	176.70	0.00	5.61e+05	0	12679.54	6.45	3.28e+07	7
60	20	5558.52	0.79	1.01e+07	3	18011.65	9.18	2.38e+07	10

Élasticité faible $(1 \times b_k)$

Instance		EXACT-forte				EXACT-faible			
#OD	%t	Cpu	Gap	#it	Nopt	Cpu	Gap	#it	Nopt
10	10	0.04	0.00	5.13e+02	0	0.18	0.00	1.45e+03	0
10	15	1.21	0.00	1.10e+04	0	12.84	0.00	8.22e+04	0
10	20	4.72	0.00	4.20e+04	0	9.93	0.00	6.09e+04	0
20	10	0.07	0.00	5.37e+02	0	0.52	0.00	3.38e+03	0
20	15	2.34	0.00	8.90e+03	0	2094.26	0.92	7.00e+05	1
20	20	3.01	0.00	9.19e+03	0	9031.84	2.68	2.77e+07	5
30	10	0.18	0.00	1.44e+03	0	1.17	0.00	8.23e+03	0
30	15	3.54	0.00	8.86e+03	0	7295.89	3.05	9.35e+06	4
30	20	1816.96	0.35	1.41e+06	1	11292.71	4.37	1.76e+07	5
40	10	0.26	0.00	1.34e+03	0	3.23	0.00	4.16e+04	0
40	15	63.18	0.00	2.07e+05	0	9187.73	2.20	7.55e+06	4
40	20	4015.91	1.62	3.78e+07	2	14539.47	4.62	7.99e+06	8
50	10	0.46	0.00	2.02e+03	0	8.08	0.00	7.65e+04	0
50	15	191.60	0.00	5.43e+05	0	16280.12	9.87	1.60e+07	9
50	20	5434.40	1.32	3.49e+07	3	16234.45	6.78	2.49e+07	9
60	10	2.42	0.00	2.13e+04	0	1865.03	0.88	5.45e+06	1
60	15	2140.70	0.11	1.78e+06	1	15208.20	10.48	1.43e+07	8
60	20	5709.48	1.43	6.79e+06	3	16354.50	8.77	1.03e+07	9

Tableau 4.3 Réseaux **Voronoi** : résultats obtenus avec demande linéaire et contraintes de capacité

Élasticité forte $(100 \times b_k)$

Instance		EXACT-forte				EXACT-faible			
#OD	%t	Cpu	Gap	#it	Nopt	Cpu	Gap	#it	Nopt
10	10	0.09	0.00	7.12e+02	0	0.17	0.00	1.52e+03	0
10	15	0.35	0.00	3.35e+03	0	0.32	0.00	2.70e+03	0
10	20	2.87	0.00	2.92e+04	0	1.69	0.00	7.39e+03	0
20	10	0.61	0.00	2.66e+03	0	3011.10	1.02	2.78e+07	1
20	15	3.48	0.00	2.20e+04	0	3739.50	0.37	1.48e+07	2
20	20	2064.69	0.15	6.86e+06	1	4213.28	1.39	1.25e+07	2
30	10	6.10	0.00	2.38e+04	0	1868.32	0.39	9.39e+06	1
30	15	2.80	0.00	7.86e+03	0	3701.95	1.66	1.36e+07	2
30	20	5316.84	0.36	7.48e+06	2	9485.99	1.34	2.56e+07	5
40	10	9.78	0.00	4.94e+04	0	4098.51	0.84	1.49e+07	2
40	15	33.23	0.00	1.80e+05	0	7595.38	1.38	3.30e+07	4
40	20	3706.92	0.46	6.04e+06	2	9500.44	2.13	2.37e+07	5
50	10	5.23	0.00	4.42e+04	0	2423.56	0.39	5.81e+06	1
50	15	768.61	0.00	7.45e+05	0	7732.89	1.08	1.80e+07	3
50	20	5540.07	0.89	9.84e+06	3	10381.42	2.46	3.44e+07	5
60	10	3.60	0.00	1.38e+04	0	3106.14	0.55	6.38e+06	1
60	15	1826.99	0.01	1.12e+06	1	13571.43	1.57	4.25e+07	7
60	20	5202.44	0.44	3.68e+06	2	14513.82	3.14	2.84e+07	8

Élasticité moyenne $(50 \times b_k)$

Instance		EXACT-forte				EXACT-faible			
#OD	%t	Cpu	Gap	#it	Nopt	Cpu	Gap	#it	Nopt
10	10	0.13	0.00	6.85e+02	0	0.39	0.00	2.86e+03	0
10	15	0.19	0.00	8.23e+02	0	0.49	0.00	3.74e+03	0
10	20	3.51	0.00	2.49e+04	0	41.86	0.00	1.62e+05	0
20	10	1.11	0.00	5.45e+03	0	1813.86	0.92	8.04e+05	1
20	15	7.20	0.00	3.70e+04	0	2574.46	0.98	8.29e+06	1
20	20	2564.76	0.28	7.38e+06	1	5034.30	0.35	1.06e+07	2
30	10	1806.32	0.02	4.62e+06	1	1908.05	0.08	1.08e+07	1
30	15	3.39	0.00	8.25e+03	0	6010.55	0.58	1.86e+07	3
30	20	5423.62	0.33	1.09e+07	3	7319.07	1.96	1.42e+07	4
40	10	140.48	0.00	3.22e+05	0	1860.57	0.27	7.34e+05	1
40	15	60.74	0.00	2.81e+05	0	3919.35	2.14	1.41e+06	2
40	20	3662.77	0.45	4.93e+06	2	7406.27	1.34	1.58e+07	4
50	10	1.78	0.00	1.84e+03	0	3604.94	0.41	4.13e+06	2
50	15	134.85	0.00	1.78e+05	0	8668.74	2.27	4.55e+07	4
50	20	5594.82	0.73	6.99e+06	3	9695.11	1.80	2.04e+07	5
60	10	2.81	0.00	3.20e+03	0	1971.79	0.20	5.14e+05	1
60	15	2072.87	0.04	4.15e+05	1	9139.78	1.66	5.52e+07	5
60	20	4183.71	0.20	5.12e+06	2	10862.94	1.87	2.06e+07	6

Élasticité faible $(1 \times b_k)$

Instance		EXACT-forte				EXACT-faible			
#OD	%t	Cpu	Gap	#it	Nopt	Cpu	Gap	#it	Nopt
10	10	0.14	0.00	6.91e+02	0	0.53	0.00	4.08e+03	0
10	15	0.29	0.00	1.65e+03	0	0.54	0.00	4.33e+03	0
10	20	11.91	0.00	8.33e+04	0	17.22	0.00	7.73e+04	0
20	10	1.24	0.00	5.58e+03	0	1812.45	0.32	4.53e+05	1
20	15	3.51	0.00	1.80e+04	0	4882.89	1.40	6.03e+06	2
20	20	3641.01	0.18	8.33e+06	2	3803.62	0.54	1.11e+07	2
30	10	1161.37	0.00	3.23e+06	0	1821.12	0.81	1.92e+06	0
30	15	3.17	0.00	7.47e+03	0	5441.41	1.25	7.87e+06	2
30	20	5420.87	0.43	1.00e+07	3	7211.77	1.11	1.27e+07	4
40	10	1802.21	0.03	1.90e+07	1	3601.75	0.16	1.08e+07	2
40	15	94.62	0.00	5.43e+05	0	4060.78	2.04	6.46e+06	1
40	20	3647.86	0.60	5.83e+06	2	5675.18	2.69	4.12e+06	2
50	10	1.18	0.00	1.16e+03	0	2874.37	0.22	3.31e+06	1
50	15	540.61	0.00	4.41e+05	0	7412.42	3.30	2.33e+07	4
50	20	5495.07	0.71	6.86e+06	3	10826.52	2.17	2.34e+07	6
60	10	2.02	0.00	3.86e+03	0	657.71	0.00	2.36e+06	0
60	15	907.41	0.00	1.12e+06	0	7971.30	2.32	2.06e+07	2
60	20	3780.73	0.31	8.46e+06	2	10820.56	2.00	2.43e+07	6

Tableau 4.4 Réseaux **Grille** : résultats obtenus avec demande linéaire et contraintes de capacité

Nous remarquons que le DL-CAP avec contraintes de capacité fortes est plus facile à résoudre que le DL-CAP avec contraintes de capacité faibles. En fait, la borne supérieure provenant de la relaxation linéaire de DL-CAP:CH est de meilleure qualité avec des contraintes de capacité fortes plutôt qu'avec des contraintes de capacité faibles car avec des contraintes de capacité fortes, la majorité des chemins contenant les arcs taxables sont saturés, ce qui entraîne les variables binaires du modèle relaxé correspondant à ces chemins à prendre des valeurs entières. De plus, cette borne supérieure décroît plus rapidement lorsque les contraintes de capacité sont fortes. Les résultats confirment cette remarque car le temps d'exécution, l'erreur relative et le pourcentage d'instances ayant atteint le temps maximal d'exécution avec EXACT-forte sont inférieurs à ceux obtenus avec EXACT-faible. Plus précisément, le temps d'exécution moyen obtenu avec EXACT-faible est environ 4 fois celui obtenu avec EXACT-forte pour les instances de 50 produits et moins. L'erreur relative maximale de EXACT-forte (resp. EXACT-faible) de tous instances obtenue à partir d'une moyenne de 10 instances est 1.62% (10.48%) et le pourcentage d'instances ayant atteint le temps maximal d'exécution avec EXACT-forte (resp. EXACT-faible) est 0.17% (resp. 18.34%). Ces valeurs augmentent avec la taille des instances. La figure 4.6 ci-dessous, présente l'évolution du temps de calcul en fonction des capacités sur les arcs. Elle est obtenue en fixant le temps d'exécution maximal à 10 heures sur les instances de Grille de 20 produits, 15% d'arcs taxables et $b_k = 100$. Les valeurs $(x\%, y\%)$ sur la courbe représentent l'erreur relative moyenne $(x\%)$ et le pourcentage d'instances ayant atteint le temps d'exécution maximal $(y\%)$.

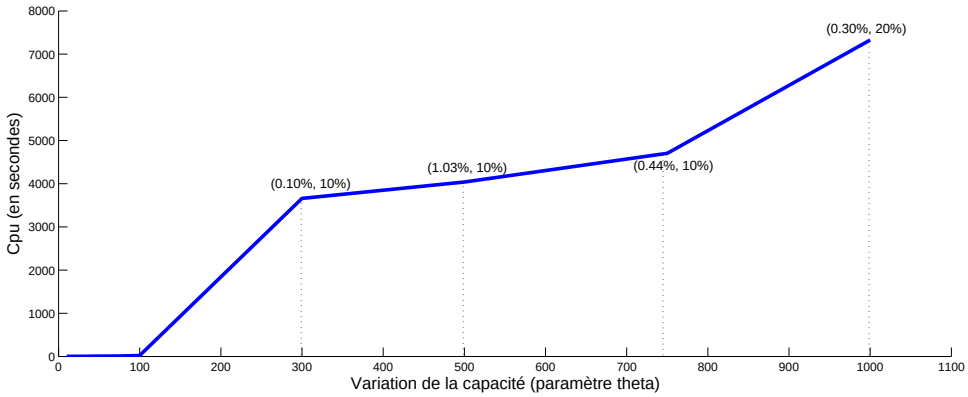


Figure 4.6 Variation du temps de calcul en fonction des capacités sur les arcs

Cette courbe montre que le temps de calcul, l'erreur relative et le pourcentage d'instances ayant atteint le temps maximal augmente avec les capacités sur les arcs.

Nous remarquons également que la difficulté du DL-CAP n'augmente pas avec l'élasticité de la demande. En fait, l'imposition des contraintes de capacité sur les arcs réduit la flexibilité des flots sur les arcs, ce qui entraîne dans la plupart des cas une variation des tarifs pour compenser la variation de la demande (annuler l'effet de l'élasticité). C'est pour cette raison que les bornes supérieures fournies par la relaxation linéaire des MIPs sont à peu près de même qualité pour les différents types d'élasticité. Les résultats numériques confirment cette remarque car l'erreur relative maximale obtenue avec l'élasticité forte, moyenne et faible est respectivement 10.48%, 9.18% et 7.21% avec le réseau de Voronoï et 3.14%, 2.27% et 2.69% avec la Grille. La figure 4.6 ci-dessous, présente l'évolution du temps de calcul en fonction de l'élasticité de la demande. Cette figure est obtenue en fixant le temps d'exécution maximal à 10 heures sur les instances de Voronoï de 20 produits, 15% d'arcs taxables avec les contraintes de capacité ni fortes, ni faibles.

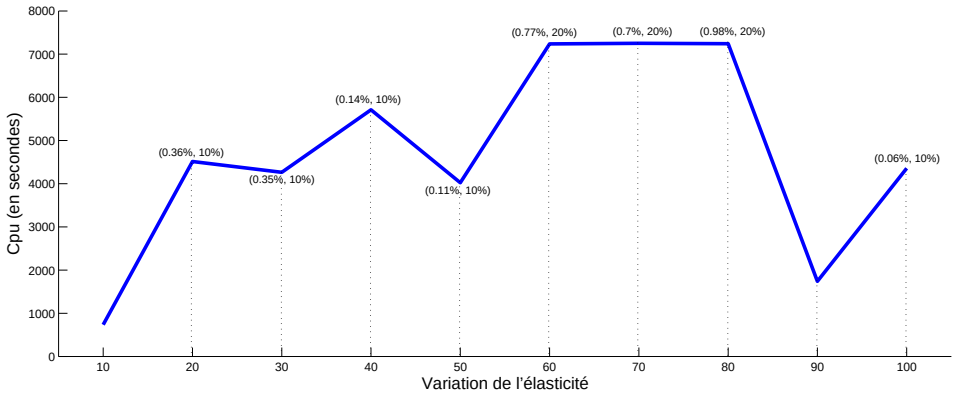


Figure 4.7 Variation du temps de calcul en fonction de l'élasticité de la demande

Nous remarquons que l'élasticité du problème n'a pas d'influence majeure sur la difficulté de résolution du DL-CAP.

Pour comparer le DL et le DL-CAP, nous comparons les résultats obtenus lorsque les contraintes de capacité sont faibles (EXACT-faible) et la formulation DL:CH2 du DL. Les erreurs relatives obtenues avec DL sont plus élevées que celles obtenues avec DL-CAP avec des contraintes de capacité faibles pour des instances de taille moyenne et grande dues à la meilleure qualité de la borne supérieure du DL-CAP comparée à celle du DL. À titre d'illustration, l'erreur relative moyenne sur les instances de 50 produits de Grille avec élasticité forte (resp. faible) est 112.25% (resp. 4.02) pour le DL contre 1.31% (resp. 1.89%). Le pourcentage d'instances ayant atteint le temps maximal d'exécution est quasi-identique pour les deux problèmes. Au vu de ce qui précède, on peut être tenté de dire que le DL est plus difficile à résoudre que le DL-CAP avec des contraintes de capacité faibles, ce qui n'est pas vraiment le cas. En fait, pour les instances de taille moyenne et grande, le DL fournit des bornes supérieures de très mauvaise qualité comparées à celles du DL-CAP due à l'introduction des contraintes de capacité qui améliorent ces bornes supérieures. Cependant, en considérant les instances de petite taille, le temps d'exécution du DL est presque négligeable comparé à celui du DL-CAP car les bornes supérieures du DL sont également de bonne qualité. Par exemple, avec les instances de 20 produits et 20% d'arcs taxables de Voronoï, le temps d'exécution

moyen obtenu avec l'élasticité faible (resp. forte) pour le DL est environ 3 minutes (resp. 34 minutes) contre 2 heures 30 minutes (resp. 54 minutes) pour le DL-CAP avec capacité faible. Notons que le DL-CAP avec des contraintes de capacité très fortes est facile car pour tout vecteur de tarifs qui génère du revenu, toutes les contraintes de capacité sont actives et par conséquent la fonction objectif du meneur en fonction des tarifs est linéaire. Dans ce cas, toute méthode de recherche locale résout le problème.

4.3 Problème de tarification sur un réseau avec demande non linéaire

Dans cette section, nous étudions le problème de tarification sur un réseau avec demande non linéaire et contraintes de capacité (DNL-CAP). Nous commençons par présenter la formulation mathématique utilisée pour résoudre le DNL-CAP (section 4.3.1). Puis, nous proposons deux méthodes exactes pour résoudre le DNL-CAP (section 4.3.2). À la section 4.3.3, nous développons une heuristique basée sur une méthode de région de confiance. Finalement, les expériences numériques sont présentées à la section 4.3.4.

4.3.1 Formulation mathématique

Nous considérons que la demande du produit k, $d_k = f_k(U_k)$, qui dépend de l'utilité U_k du produit k, est une fonction non linéaire, non négative et strictement décroissante où l'utilité U_k est le coût des chemins utilisés pour acheminer le produit k. En remplaçant d_k

dans DE-CAP:CH par sa valeur, la formulation mathématique correspondante est :

DNL-CAP:CH

$$\max_{t,h,U,\gamma} \sum_{k\in\mathcal{K}} f_k(U_k)U_k - \sum_{p\in P_k} h_p^k \sum_{a\in p} c_a$$

$$\sum_{p\in P_k} h_p^k \;=\; f_k(U_k) \qquad \forall k \in \mathcal{K} \tag{4.28}$$

$$\sum_{k\in\mathcal{K}} \sum_{p\in P_k \;|\; a\in p} h_p^k \;\leq\; u_a \qquad \forall a \in \mathcal{A}_1 \tag{4.29}$$

$$\sum_{a\in p} c_a + \sum_{a\in p\bigcap \mathcal{A}_1} t_a - M_p^k(1-\gamma_p^k) \;\leq\; U_k \qquad \forall p \in P_k, \forall k \in \mathcal{K} \tag{4.30}$$

$$\sum_{a\in p} c_a + \sum_{a\in p\cap \mathcal{A}_1} t_a - U_k \;\geq\; 0 \qquad \forall p \in P_k, \forall k \in \mathcal{K} \tag{4.31}$$

$$h_p^k \leq N_p^k \gamma_p^k \qquad \forall\; p \in P_k, \forall k \in \mathcal{K} \tag{4.32}$$

$$h_p^k \geq 0 \qquad \forall\; p \in P_k, \forall k \in \mathcal{K}$$

$$\gamma_p^k \in \{0,1\} \qquad \forall\; p \in P_k, \forall k \in \mathcal{K}$$

où M_p^k est une grande constante identique à celle définie par Dewez [27] et $N_p^k = \min_{a\in p\cap\mathcal{A}_1}\{u_a\}$.

Tel que présenté au chapitre 3, nous considérons deux classes de fonctions de demande : la fonction de demande à élasticité constante $f_k(U_k) = \alpha_k U_k^{-\beta_k}$ et la fonction de demande à élasticité non constante $f_k(U_k) = \alpha_k \exp(-\beta_k U_k)$. Les méthodes exactes que nous présentons à la section suivante sont basées sur la formulation DNL-CAP:CH. Le principe de ces méthodes est identique à celui présenté au chapitre 3 pour la résolution du DNL.

4.3.2 Méthode exacte

Elle possède une phase d'initialisation et une phase itérative. La phase d'initialisation consiste, à partir du DNL-CAP:CH, à surapproximer la fonction non linéaire $f_k(U_k)U_k$ et de sous-approximer la fonction non linéaire $f_k(U_k)$ dans le but d'obtenir un programme linéaire en variables mixtes (MIP). Quant à la phase itérative, elle résout le MIP et ajoute des coupes

au fur et à mesure afin d'améliorer les surapproximations et les sous-approximations des fonctions non-linéaires jusqu'à ce qu'elles soient jugées suffisantes.

Formellement, en remplaçant $f_k(U_k)U_k$ par W_k et en ajoutant les contraintes nécessaires aux différentes surapproximations et sous-approximations (pour plus de détails voir la section

3.4 du chapitre 3), on obtient le MIP suivant

DNL-CAP:App

$$\max_{t,h,U,T,\gamma,W,z,s} \sum_{k\in\mathcal{K}} W_k - \sum_{p\in P_k} h_p^k \sum_{a\in p} c_a$$

$$\sum_{p\in P_k} h_p^k \;\geq\; a_k^l + b_k^l U_k \qquad \forall l = 1,\ldots,L_k,\ \forall k\in\mathcal{K} \qquad (4.33)$$

$$\sum_{k\in\mathcal{K}} \sum_{p\in P_k \mid a\in p} h_p^k \;\leq\; u_a \qquad \forall a\in\mathcal{A}_1 \qquad (4.34)$$

$$\sum_{a\in p} c_a + \sum_{a\in p\bigcap \mathcal{A}_1} t_a - M_p^k(1-\gamma_p^k) \;\leq\; U_k \qquad \forall p\in P_k, \forall k\in\mathcal{K} \qquad (4.35)$$

$$\sum_{a\in p} c_a + \sum_{a\in \mathcal{A}_1\cap p} t_a - U_k \;\geq\; 0 \qquad \forall p\in P_k, \forall k\in\mathcal{K} \qquad (4.36)$$

$$h_p^k \leq N_p^k \gamma_p^k \qquad \forall\ p\in P_k, \forall k\in\mathcal{K} \qquad (4.37)$$

$$W_k \leq a_k^i + b_k^i U_k + M s_k \qquad \forall\ i = 1,\ldots,I_k,\ \forall k\in\mathcal{K} \qquad (4.38)$$

$$U_k \leq \tilde{\gamma}^k(1-s_k) + M s_k \qquad \forall k\in\mathcal{K} \qquad (4.39)$$

$$W_k \leq a_k^j + b_k^j U_k + M(1-z_k^j) + N(1-s_k) \qquad \forall l = 1,\ldots,J_k-1,\ \forall k\in\mathcal{K} \qquad (4.40)$$

$$U_k \geq U_k^j z_k^j \qquad \forall l = 1,\ldots,J_k-1,\ \forall k\in\mathcal{K} \qquad (4.41)$$

$$U_k \leq U_k^{j+1} z_k^j + M(1-z_k^j) \qquad \forall l = 1,\ldots,J_k-1,\ \forall k\in\mathcal{K} \qquad (4.42)$$

$$\sum_{j=1}^{J_k-1} z_k^j = s_k \qquad \forall k\ \in\mathcal{K} \qquad (4.43)$$

$$h_p^k \geq 0 \qquad \forall\ p\in P_k, \forall k\in\mathcal{K}$$

$$\gamma_p^k \in \{0,1\} \qquad \forall\ p\in P_k, \forall k\in\mathcal{K}$$

$$s_k\ \in\ \{0,1\} \qquad \forall k\in\mathcal{K}$$

$$z_k^j\ \in\ \{0,1\} \qquad \forall\ j = 1,\ldots,J_k-1,\ \forall k\in\mathcal{K}$$

où la sous-approximation de $f_k(U_k)$ se fait par l'ajout à DNL-CAP:CH des contraintes (4.33), la surapproximation de la partie concave (resp. convexe) de $f_k(U_k)U_k$ se fait par l'ajout à DNL-CAP:CH des contraintes (4.38) à (4.39) (resp. (4.40) à (4.43)). La variable binaire s_k

indique si l'on se trouve sur la partie concave ($s_k = 1$) ou convexe ($s_k = 0$) de la fonction $f_k(U_k)U_k$. La variable binaire z_j^k indique si on se trouve sur le sous-intervalle j de la partie convexe de $f_k(U_k)U_k$. Nous dénotons par EXACT-1 la méthode exacte issue de ces approximations.

De manière similaire à ce qui a été présenté au chapitre 3, nous réécrivons la fonction $f_k(U_k)U_k$ de telle sorte que la nouvelle fonction à surapproximer ait une partie convexe plus étroite. Pour ce faire, nous introduisons la variable T_k et nous imposons

$$U_k = T_k + \sum_{a \in p_0^k} c_a \tag{4.44}$$

où p_0^k est le chemin de plus petit coût fixe du produit k. En fait, T_k correspond au revenu généré par une unité de flot du chemin p_0^k. Puis, nous réécrivons la fonction objectif du produit k comme suit :

$$\begin{aligned}
f_k(U_k)U_k - \sum_{p \in P_k} \sum_{a \in p} c_a h_p^k &= f_k(U_k) \left(T_k + \sum_{a \in p_0^k} c_a \right) - \sum_{p \in P_k} h_p^k \sum_{a \in p} c_a \\
&= f_k(U_k)T_k + f_k(U_k) \sum_{a \in p_0^k} c_a - \sum_{p \in P_k} h_p^k \sum_{a \in p} c_a \\
&= f_k \left(T_k + \sum_{a \in p_0^k} c_a \right) T_k + \sum_{p \in P_k} h_p^k \sum_{a \in p_0^k} c_a - \sum_{p \in P_k} h_p^k \sum_{a \in p} c_a \\
&= f_k \left(T_k + \sum_{a \in p_0^k} c_a \right) T_k - \sum_{p \in P_k} h_p^k \left(\sum_{a \in p} c_a - \sum_{a \in p_0^k} c_a \right). \qquad (4.45)
\end{aligned}$$

La deuxième méthode exacte que nous dénotons EXACT-2 est basée sur le programme mathématique obtenu en remplaçant dans DNL-CAP:CH la fonction objectif du produit k par (4.45) et en ajoutant la contrainte (4.44). Pour la méthode EXACT-2, à la phase d'initialisation, nous surapproximons plutôt la fonction $f_k \left(T_k + \sum_{a \in p_0^k} c_a \right) T_k$. En fait, les coefficients des variables de flots de (4.45) sont négatifs car p_0^k étant le chemin de plus petit coût fixe du

produit k, $\sum_{a\in p} c_a - \sum_{a\in p_0^k} c_a \geq 0$ pour tout $p \in P_k$.

Les longueurs de l'intervalle des valeurs de U_k et T_k sont égales, et la partie convexe de $f_k(U_k)U_k$ est plus large que celle de $f_k\left(T_k + \sum_{a\in p_0^k} c_a\right) T_k$. De plus, $f_k(U_k)U_k$ est bornée inférieurement par $f_k\left(T_k + \sum_{a\in p_0^k} c_a\right) T_k$. Il en résulte que la surapproximation de $f_k(U_k)U_k$ augmente fortement l'aspect combinatoire du problème comparée à celle de $f_k\left(T_k + \sum_{a\in p_0^k} c_a\right) T_k$ car elle nécessite l'introduction de plus de variables binaires (variables z_j^k) pour une discrétisation identique de l'intervalle des valeurs de U_k et T_k. L'introduction des variables binaires z_j^k au modèle approximé, DNL-CAP:App, augmente l'aspect combinatoire du modèle et par conséquent son temps de calcul. Il s'ensuit que la méthode EXACT-2 aura un temps d'exécution inférieur à celui de la méthode EXACT-1.

Après la phase d'initialisation des deux méthodes exactes, nous obtenons le modèle DNL-CAP:App (MIP) qui surapproxime DNL-CAP:CH. La valeur de la fonction objectif de DNL-CAP:App est une borne supérieure du DNL-CAP. La phase itérative résout à chaque itération le modèle approximé DNL-CAP:App et lui ajoute des contraintes dans le but d'améliorer la surapproximation de $f_k(U_k)U_k$ ou $f_k\left(T_k + \sum_{a\in p_0^k} c_a\right) T_k$ et la sous-approximation de $f_k(U_k)$. La solution fournie par DNL-CAP:App à chaque itération n'est pas forcément réalisable pour le DNL-CAP. Cependant à partir de cette solution, nous pouvons déterminer (si possible) une solution réalisable du DNL-CAP en procédant de deux façons.

Détermination de la solution réalisable du DNL-CAP à partir d'une solution de DNL-CAP:App

Soit $(t^*, h^*, U^*, T^*, \gamma^*, W^*, z^*, s^*)$ une solution réalisable du DNL-CAP:App. Pour déterminer une solution réalisable du DNL-CAP, nous fixons dans DNL-CAP:CH la variable U à U^* puis nous résolvons le problème résultant. En fixant U à U^*, nous supposons connu le coût

des chemins utilisés. Le DNL-CAP:CH se réduit au programme linéaire en variables mixtes suivant :

DNL-CAP:CH(U^*)

$$\max_{t,h,\gamma} \sum_{k\in\mathcal{K}} f_k(U_k^*)U_k^* - \sum_{p\in P_k}\sum_{a\in p} c_a h_p^k$$

$$\sum_{p\in P_k} h_p^k \;=\; f_k(U_k^*) \qquad \forall k \in \mathcal{K}$$

$$\sum_{k\in\mathcal{K}}\sum_{p\in P_k \mid a\in p} h_p^k \;\leq\; u_a \qquad \forall a \in \mathcal{A}_1$$

$$\sum_{a\in p} c_a + \sum_{a\in p\bigcap \mathcal{A}_1} t_a - M_p^k(1-\gamma_p^k) \;\leq\; U_k^* \qquad \forall p \in P_k, \forall k \in \mathcal{K}$$

$$\sum_{a\in p} c_a + \sum_{a\in \mathcal{A}_1\cap p} t_a - U_k^* \;\geq\; 0 \qquad \forall p \in P_k, \forall k \in \mathcal{K}$$

$$h_p^k \leq N_p^k \gamma_p^k \qquad \forall\; p \in P_k, \forall k \in \mathcal{K}$$

$$h_p^k \geq 0 \qquad \forall\; p \in P_k, \forall k \in \mathcal{K}$$

$$\gamma_p^k \in \{0,1\} \qquad \forall\; p \in P_k, \forall k \in \mathcal{K}.$$

Le temps de résolution du DNL-CAP:CH(U^*) est négligeable comparé à celui de DNL-CAP. Cependant, DNL-CAP:CH(U^*) peut être irréalisable car étant donné que la solution du DNL-CAP:App vérifie $\sum_{p\in P_k} h_p^{*k} \leq f_k(U_k^*)$ pour tout k, les contraintes de capacité peuvent empêcher la contrainte de conservation de flots $\sum_{p\in P_k} h_p^k = f_k(U_k^*)$ d'être satisfaite car nous avons imposé que les variables duales associées aux contraintes de capacité soient nulles.

Une autre façon de déterminer une solution réalisable de meilleur revenu à partir d'une solution du DNL-CAP:App est d'utiliser l'optimisation inverse.

Optimisation Inverse

Le POI consiste à fixer la variable binaire γ qui indique les chemins utilisés pour un produit donné puis à déterminer les tarifs et les flots compatibles à ces chemins afin de maximiser

le revenu du meneur. Nous présentons le POI en utilisant le programme mathématique à un niveau utilisé pour développer la méthode EXACT-2.

Supposons que γ soit fixé à γ^* où γ^* provient de la solution du DNL-CAP:App. Le programme non linéaire obtenu est :

OPTINV(γ^*)

$$\max_{t,h,U} \sum_{k\in\mathcal{K}} \left[f_k \left(T_k + \sum_{a\in p_0^k} c_a \right) T_k - \sum_{p\in P_k} h_p^k \left(\sum_{a\in p} c_a - \sum_{a\in p_0^k} c_a \right) \right]$$

$$\sum_{p\in P_k} h_p^k \;=\; f_k(U_k) \qquad \forall k \in \mathcal{K}$$

$$\sum_{k\in\mathcal{K}} \sum_{p\in P_k \;|\; a\in p} h_p^k \;\leq\; u_a \qquad \forall a \in \mathcal{A}_1$$

$$\sum_{a\in p} c_a + \sum_{a\in p \bigcap \mathcal{A}_1} t_a - M_p^k(1-\gamma_p^{*k}) \;\leq\; U_k \qquad \forall p \in P_k, \forall k \in \mathcal{K}$$

$$\sum_{a\in p} c_a + \sum_{a\in \mathcal{A}_1 \cap p} t_a - U_k \;\geq\; 0 \qquad \forall p \in P_k, \forall k \in \mathcal{K}$$

$$U_k \;=\; T_k + \sum_{a\in p_0^k} c_a \qquad \forall k \in \mathcal{K}$$

$$h_p^k \leq N_p^k \gamma_p^{*k} \qquad \forall\; p \in P_k, \forall k \in \mathcal{K}$$

$$h_p^k \geq 0 \qquad \forall\; p \in P_k, \forall k \in \mathcal{K}.$$

La technique utilisée pour résoudre OPTINV(γ^*) est la même que celle développée pour la méthode EXACT-2 du DNL-CAP car la difficulté provient de la présence des deux fonctions non linéaires $f_k \left(T_k + \sum_{a\in p_0^k} c_a \right) T_k$ et $f_k(U_k)$ du produit k. Le temps de résolution de OPTINV(γ^*) est négligeable comparé à celui du DNL-CAP car la seule difficulté liée à la résolution de OPTINV(γ^*) est la surapproximations des fonctions non linéaires. OPTINV(γ^*) peut être irréalisable car il peut ne pas exister un vecteur de tarifs t qui fasse en sorte que l'ensemble de chemins γ^* puisse être utilisé.

Algorithme exact

Trois paramètres d'entrée sont nécessaires : la tolérance ε sur la surapproximation et la sous-approximation des fonctions $f_k(U_k)U_k$ (resp. $f_k\left(T_k + \sum_{a \in p_0^k} c_a\right) T_k$) et $f_k(U_k)$ de la méthode EXACT-1 (resp. EXACT-2), le nombre de points L_k de la discrétisation de l'intervalle des valeurs de U_k pour la sous-approximation de la fonction $f_k(U_k)$ et le nombre de points I_k (resp. J_k) de la discrétisation de l'intervalle des valeurs de U_k ou T_k sur la partie concave (resp. convexe) pour la surapproximation de la fonction $f_k(U_k)U_k$ ou $f_k\left(T_k + \sum_{a \in p_0^k} c_a\right) T_k$. L'algorithme $\varepsilon-$exact se présente comme suit :

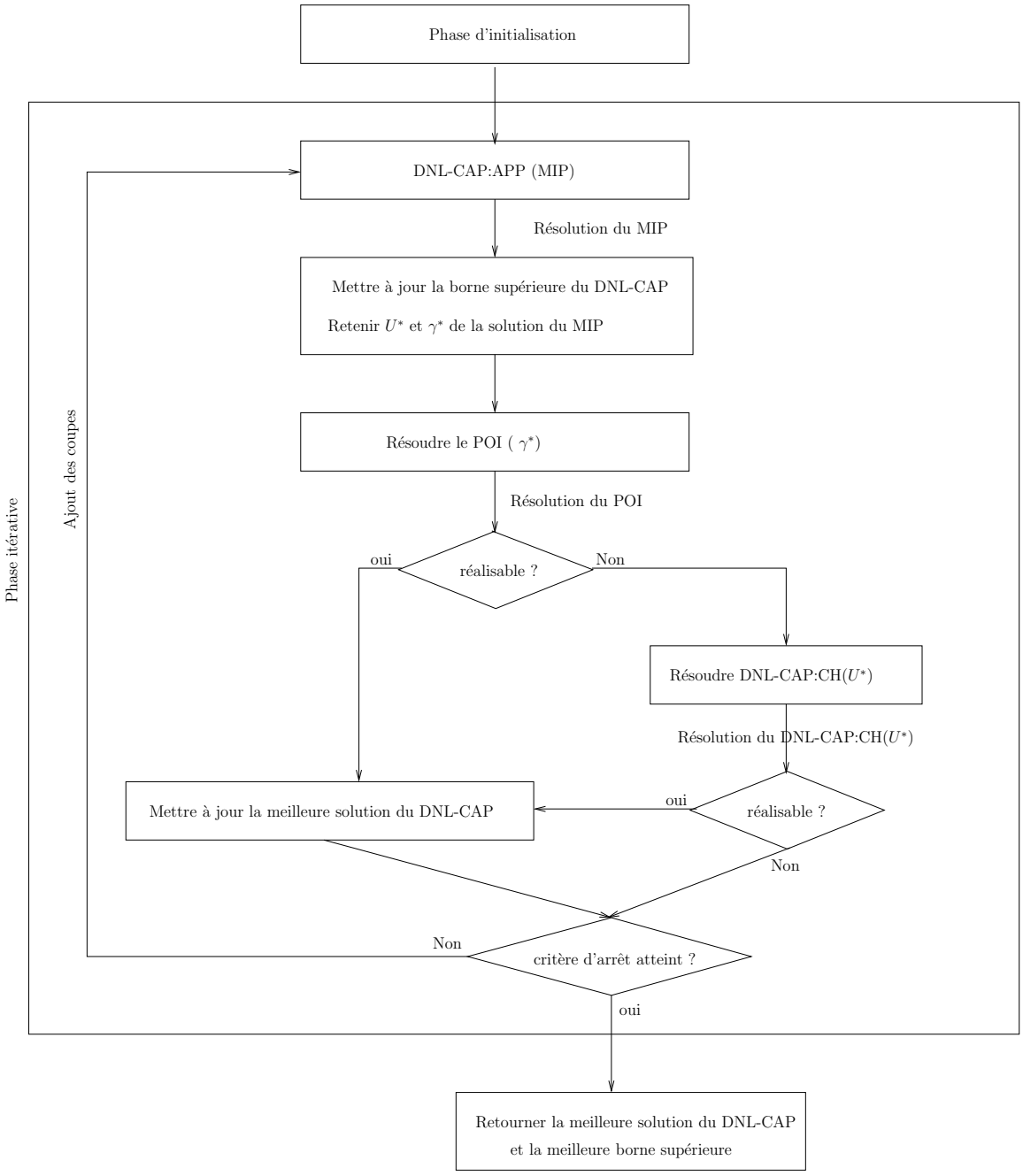


Figure 4.8 Algorithme $\varepsilon-$exact

4.3.3 Heuristique

Cette heuristique est basée sur une méthode de région de confiance comme celles développées pour le DNL. Étant donné que la non linéarité de la fonction de demande augmente la difficulté du DNL-CAP, le modèle de région de confiance est obtenu en linéarisant les fonctions de demande autour d'une solution courante, ce qui nous mène au DL-CAP. Puis, une région de confiance est définie autour du vecteur de tarifs sur les arcs afin qu'ils restent proches de ceux de la solution courante.

Formellement, soit $(\overline{t}, \overline{h}, \overline{U}, \overline{\gamma})$ une solution courante de DNL-CAP:CH, en linéarisant la fonction demande $f_k(U_k)$ de chaque produit k autour de $\overline{U}_k$, nous obtenons une approximation de la fonction demande $f_k(U_k) \approx a_k - b_k U_k$ où a_k et b_k sont les coefficients de la tangente à la fonction $f_k(U_k)$ au point $\overline{U}_k$. Dans un premier temps, nous remplaçons dans la fonction objectif de DNL-CAP:CH, chaque fonction $f_k(U_k)$ par $a_k - b_k U_k$, puis nous ajoutons la contrainte $\overline{t} - \Delta t \leq t \leq \overline{t} + \Delta t$ où Δt est le rayon de la région de confiance.

Le modèle de la méthode à région de confiance se formule alors comme un suit :

QUAD:CAP

$$\max_{t,h,U} \sum_{k\in\mathcal{K}} \left(aU_k - b_k(U_k)^2 - \sum_{p\in P_k} h_p^k \sum_{a\in p} c_a \right)$$

$$\sum_{p\in P_k} h_p^k \;=\; a_k - b_k U_k \qquad \forall k \in \mathcal{K} \tag{4.46}$$

$$\sum_{k\in\mathcal{K}} \sum_{p\in P_k \;|\; a\in p} h_p^k \;\leq\; u_a \qquad \forall a \in \mathcal{A}_1 \tag{4.47}$$

$$\sum_{a\in p} c_a + \sum_{a\in p \bigcap \mathcal{A}_1} t_a - M_p^k(1-\gamma_p^k) \;\leq\; U_k \qquad \forall 0p \in P_k, \forall k \in \mathcal{K} \tag{4.48}$$

$$\sum_{a\in p} c_a + \sum_{a\in \mathcal{A}_1 \cap p} t_a - U_k \;\geq\; 0 \qquad \forall p \in P_k, \forall k \in \mathcal{K} \tag{4.49}$$

$$h_p^k \leq N_p^k \gamma_p^k \qquad \forall\; p \in P_k, \forall k \in \mathcal{K} \tag{4.50}$$

$$\overline{t}_a - \Delta t_a \leq t_a \leq \overline{t}_a + \Delta t_a \qquad \forall a \in \mathcal{A}_1. \tag{4.51}$$

$$h_p^k \geq 0 \qquad \forall\; p \in P_k, \forall k \in \mathcal{K}$$

$$\gamma_p^k \in \{0,1\} \qquad \forall\; p \in P_k, \forall k \in \mathcal{K}.$$

QUAD:CAP est la formulation DL-CAP:CH présentée à la section 4.2.1 où l'on ajoute la contrainte (4.51) pour délimiter la taille du pas du vecteur de tarifs.

Pour déterminer une solution du DNL-CAP à partir d'une solution du modèle (QUAD:CAP), on résout OPTINV(γ^*) ou DNL-CAP:CH(U^*) où γ^* et U^* proviennent de la solution du QUAD:CAP.

L'heuristique est exécutée en utilisant plusieurs solutions de départ pour diversifier l'espace de recherche des optima locaux. Les solutions de départ sont déterminées en générant uniformément sur un intervalle les valeurs des tarifs sur les arcs. À partir de ces tarifs, nous cherchons une solution S du DNL-CAP correspondant à ces tarifs. Cependant la solution S ne vérifie pas toujours la propriété selon laquelle les variables duales associées aux contraintes de capacité sont nulles (exploitée pour obtenir la formulation DNL-CAP:CH). Ainsi, à par-

tir de S, on détermine une solution correspondante où les variables duales associées aux contraintes de capacité sont nulles en recalculant les tarifs et les flots par une résolution de DNL-CAP:CH(U) où U est le vecteur de coût maximal des chemins utilisés de la solution S.

4.3.4 Expérimentations

Les expérimentations effectuées permettent d'évaluer de façon quantitative les méthodes proposées pour la résolution du DNL-CAP.

Les fonctions de demande utilisées sont celles présentées à la section 3.7 pour le DNL. Pour affecter les capacités sur les arcs du réseau, le même principe utilisé lorsque la demande est linéaire est répété à la seule différence que l'on considère plutôt des fonctions de demande non linéaires. Nous fixons à l'infini la capacité des arcs non taxables, c'est-à-dire pour tout $a \in \mathcal{A}_2$, $u_a = +\infty$. Pour les arcs taxables, la capacité de l'arc a est définie en tenant compte du type de la fonction de demande.

Pour les fonctions de demande à élasticité constante, la capacité sur l'arc taxable a est définie comme suit

$$u_a = \frac{1}{|\mathcal{K}_a|} \sum_{k \in \mathcal{K}_a} \frac{\alpha_k}{\left(\gamma^k(\infty)\right)^{\beta_k}} + \theta \left(\frac{c_a + 1}{\min\limits_{a' \in \mathcal{A}_1} c_{a'} + 1} \right)^{3/2}$$

où $\mathcal{K}_a$ est l'ensemble des produits ayant au moins un chemin contenant l'arc a. L'expression $\frac{1}{|\mathcal{K}_a|} \sum_{k \in \mathcal{K}_a} \frac{\alpha_k}{\left(\gamma^k(\infty)\right)^{\beta_k}}$ est la moyenne des demandes minimales des produits appartenant à $\mathcal{K}_a$, ce qui permet à l'arc taxable a de satisfaire au moins la demande minimale d'un produit de $\mathcal{K}_a$. Pour les fonctions de demande à élasticité non constante, la capacité sur l'arc taxable a est définie par

$$u_a = \frac{1}{|\mathcal{K}_a|} \sum_{k \in \mathcal{K}_a} \alpha_k \exp\left(-\frac{\beta_k}{C^k_{moy}} \gamma^k(\infty) \right) + \theta \left(\frac{c_a + 1}{\min\limits_{a' \in \mathcal{A}_1} c_{a'} + 1} \right)^{3/2}$$

où $\frac{1}{|\mathcal{K}_a|}\sum_{k\in\mathcal{K}_a}\alpha_k \exp\left(-\frac{\beta_k}{C^k_{moy}}\gamma^k(\infty)\right)$ est la moyenne des demandes minimales des produits appartenant à $\mathcal{K}_a$, ce qui permet à l'arc taxable a de satisfaire au moins la demande minimale d'un produit de $\mathcal{K}_a$. L'expression $[(c_a+1)/\left(\min_{a'\in\mathcal{A}_1} c_{a'}+1\right)]^{3/2}$ permet d'affecter la capacité sur l'arc a en fonction de son coût fixe et du plus petit coût fixe des arcs taxables. Le coefficient θ permet de varier la capacité sur les arcs et est utilisé pour l'analyse de sensibilité des contraintes de capacité.

Les expériences numériques sont effectuées sur la famille d'instances des réseaux *Voronoï* et *Grille* présentée à la section 2.6.1. Nous ne considérons que les tarifs positifs et nous varions le nombre de produits entre 5 et 40. La proportion d'arcs taxables est fixée à 10%, 15% et 20%. Nous utilisons les fonctions de demande à élasticité constante et non constante. Nous fixons les valeurs de β_k à 2 et 5 afin de varier l'élasticité de la demande. Dans le même ordre d'idée, les valeurs de θ sont choisis de manière à imposer des contraintes de capacité fortes et faibles. Les résultats numériques sont obtenus en fixant ε à 10^{-2}. L'intervalle des valeurs de U_k et T_k est discrétisé en 20 points pour la surapproximation et la sous-approximation des fonctions non linéaires à la phase d'initialisation des méthodes exactes. Pour la méthode approchée, nous utilisons 50 solutions de départ et le nombre d'itérations maximal de l'heuristique est fixé à 50. Le temps d'exécution maximal des méthodes est fixé à 5 heures (18000 secondes). Les résultats numériques sont une moyenne de 10 instances.

Les résultats numériques sont reportés aux tableaux 4.5, 4.6 pour l'élasticité constante et 4.7 et 4.8 pour l'élasticité non constante où Bgap défint l'écart relatif (en %) entre la borne supérieure fournie par EXACT-2 et la solution trouvée par EXACT-1. La colonne Cpu* représente le temps auquel l'heuristique trouve la meilleure solution.

Élasticité constante : $\beta_k = 2$ pour tout k

		Capacité forte												Capacité faible											
		EXACTE-1					EXACTE-2				Heuristique				EXACTE-1					EXACTE-2				Heuristique			
#OD	%t	Cpu	Gap	BGap	#it	Nopt	Cpu	Gap	#it	Nopt	Cpu	Cpu*	Gap	Nopt	Cpu	Gap	BGap	#it	Nopt	Cpu	Gap	#it	Nopt	Cpu	Cpu*	Gap	Nopt
5	10	0.03	0.00	0.00	1.00	0	0.03	0.00	1.00	0	2.14	0.14	0.00	0	0.71	0.00	0.00	1.89	0	0.03	0.00	1.30	0	3.04	0.23	0.00	0
5	15	1.24	0.00	0.00	1.60	0	0.27	0.00	1.30	0	16.50	0.69	0.00	0	18.17	0.01	0.00	3.30	0	0.22	0.00	1.70	0	28.62	3.33	0.03	0
5	20	0.84	0.01	0.00	1.40	0	0.28	0.00	1.50	0	20.32	2.40	0.00	0	8.34	0.01	0.00	2.30	0	0.22	0.00	1.40	0	38.88	6.83	0.26	0
10	10	0.14	0.00	0.00	1.50	0	0.09	0.00	1.40	0	5.58	0.31	0.00	0	1964.15	1.01	0.00	2.10	1	0.25	0.00	1.80	0	17.51	2.10	0.12	0
10	15	28.81	0.00	0.00	1.80	0	0.97	0.00	1.70	0	46.24	2.00	0.00	0	9512.98	0.37	0.02	3.30	4	1.84	0.00	2.00	0	81.34	14.55	0.04	0
10	20	156.10	0.00	0.00	1.90	0	7.34	0.00	1.50	0	825.62	39.38	0.00	0	7225.12	0.30	0.02	3.00	3	7.72	0.00	1.90	0	147.68	25.09	0.06	0
15	10	0.25	0.00	0.00	1.50	0	0.10	0.00	1.40	0	10.83	0.60	0.00	0	11064.85	0.48	0.01	2.10	6	0.46	0.01	2.10	0	32.30	3.54	0.01	0
15	15	20.95	0.00	0.00	1.40	0	1.01	0.00	1.40	0	82.58	17.83	0.00	0	15299.39	1.80	0.22	3.30	8	116.52	0.00	2.50	0	509.04	57.97	0.00	0
15	20	2028.67	0.14	0.00	1.89	1	121.42	0.00	2.10	0	722.81	14.38	0.00	0	9465.25	0.82	0.06	1.38	4	58.41	0.00	2.20	0	254.06	79.78	0.00	0
20	10	2.52	0.00	0.00	1.40	0	0.20	0.00	1.30	0	22.96	0.58	0.00	0	14040.05	2.06	0.08	1.22	7	0.95	0.00	1.90	0	45.41	2.95	0.00	0
20	15	498.94	0.00	0.00	1.25	0	218.56	0.00	1.40	0	807.75	29.66	0.00	0	18040.24	18.74	15.09	1.00	9	2186.76	0.13	2.10	1	713.97	192.32	0.25	0
20	20	3854.36	0.00	0.00	1.90	1	18.24	0.00	2.10	0	375.34	26.46	0.23	0	14705.23	15.72	13.98	1.00	7	3322.53	0.23	2.00	1	1341.22	381.11	0.32	0
30	10						0.18	0.00	1.30	0	47.93	2.07	0.00	0						1.95	0.01	1.60	0	165.93	38.51	0.01	0
30	15						1805.29	0.00	1.30	1	1881.26	164.28	0.01	1						7506.99	1.12	1.30	4	2841.88	602.67	1.21	0
30	20						557.29	0.02	1.70	0	421.98	23.79	0.02	0						3701.16	0.28	1.50	2	2786.91	2086.04	0.33	0
40	10						0.27	0.00	1.50	0	38.40	2.37	0.00	0						35.08	0.01	1.60	0	244.19	47.58	0.01	0
40	15						1812.44	0.02	1.20	1	1417.29	116.92	0.09	0						9254.33	6.14	1.30	5	6386.58	3062.14	2.62	3
40	20						602.69	0.01	1.50	0	710.85	212.61	0.01	0						7624.06	0.92	1.50	3	4787.19	2390.11	1.60	1

Élasticité constante : $\beta_k = 5$ pour tout k

		EXACTE-1					EXACTE-2				Heuristique				EXACTE-1					EXACTE-2				Heuristique			
#OD	%t	Cpu	Gap	BGap	#it	Nopt	Cpu	Gap	#it	Nopt	Cpu	Cpu*	Gap	Nopt	Cpu	Gap	BGap	#it	Nopt	Cpu	Gap	#it	Nopt	Cpu	Cpu*	Gap	Nopt
5	10	0.14	0.00	0.00	2.00	0	0.03	0.00	1.40	0	0.92	0.09	0.00	0	397.82	0.00	0.02	5.90	0	0.07	0.01	1.70	0	6.21	0.44	0.01	0
5	15	22.01	0.01	0.00	4.10	0	1.49	0.00	3.00	0	21.69	1.44	0.00	0	364.08	0.01	0.02	8.80	0	230.74	0.00	3.10	0	72.62	8.62	0.01	0
5	20	9.46	0.04	0.02	2.60	0	1.25	0.00	2.10	0	16.87	1.18	0.00	0	598.47	0.01	0.04	8.80	0	162.10	0.00	3.30	0	273.03	52.71	0.00	0
10	10	0.28	0.03	0.00	2.10	0	0.07	0.00	1.70	0	13.48	0.83	0.00	0	7314.96	0.54	0.00	5.00	3	0.46	0.00	2.70	0	54.20	10.15	0.05	0
10	15	75.24	0.00	0.02	2.50	0	21.99	0.01	2.60	0	92.93	15.96	0.00	0	19676.49	3.20	2.14	3.22	8	3014.31	1.65	3.90	1	498.24	76.18	1.85	0
10	20	281.54	0.00	0.01	3.10	0	5.24	0.00	2.60	0	87.98	7.54	0.00	0	16199.22	5.40	5.94	3.80	8	5425.92	1.23	3.10	2	2535.25	1157.28	2.79	0
15	10	0.50	0.00	0.01	3.50	0	0.10	0.00	2.40	0	4.00	0.38	0.00	0	12874.33	2.30	0.18	4.60	7	1.67	0.13	3.20	0	157.85	16.20	0.12	0
15	15	2645.61	0.00	1.49	3.20	1	1843.44	1.43	2.50	1	92.72	13.14	2.48	0	18064.73	173.23	166.58	1.10	10	5252.94	2.46	3.60	2	6788.90	2740.50	3.33	3
15	20	3690.56	0.15	0.06	3.10	2	392.08	0.00	3.20	0	94.78	6.99	0.01	0	18261.93	266.09	186.32	1.20	10	7505.17	1.13	2.60	3	7640.21	1109.53	3.55	3
20	10	0.71	0.00	0.01	3.56	0	0.14	0.00	2.30	0	8.01	0.74	0.00	0	15885.90	80.07	71.63	1.38	7	30.70	0.12	3.20	0	565.94	47.86	0.12	0
20	15	3372.37	0.03	1.17	2.80	1	2097.86	0.99	2.30	1	85.69	13.24	1.16	0	18033.51	215.03	187.45	1.10	10	10590.30	4.81	2.20	5	9268.02	5181.55	8.59	4
20	20	4475.05	0.15	0.07	2.80	2	1846.74	0.04	3.10	1	120.44	8.27	0.01	0	18027.91	300.50	211.73	1.00	10	11342.88	3.90	2.60	4	12675.03	7858.57	11.30	6
30	10						0.45	0.01	2.10	0	65.55	18.51	0.00	0						3366.59	0.26	2.80	1	565.24	46.89	0.27	0
30	15						3651.23	1.98	2.40	2	209.70	37.42	2.74	0						12087.59	13.78	1.80	6	12672.40	7967.93	8.97	6
30	20						5718.93	0.60	2.00	3	459.60	36.03	1.20	0						18395.43	10.37	1.30	10	10481.35	9325.41	9.32	4
40	10						0.44	0.00	2.30	0	53.97	1.72	0.01	0						7991.62	0.92	2.70	4	3242.89	1230.11	0.92	1
40	15						3668.53	1.14	2.10	2	439.16	67.24	1.36	0						16535.83	20.08	1.20	9	13715.92	11285.19	21.27	7
40	20						6620.16	1.29	2.50	3	557.31	57.82	1.29	0						18358.97	20.02	1.00	10	13476.97	8814.43	21.55	6

Tableau 4.5 Réseaux **Voronoï** : résultats obtenus avec demande non linéaire et contraintes de capacité (élasticité constante)

Élasticité constante : $\beta_k = 2$ pour tout k

		Capacité forte												Capacité faible											
		EXACTE-1					EXACTE-2				Heuristique				EXACTE-1					EXACTE-2				Heuristique			
#OD	%t	Cpu	Gap	BGap	#it	Nopt	Cpu	Gap	#it	Nopt	Cpu	Cpu*	Gap	Nopt	Cpu	Gap	BGap	#it	Nopt	Cpu	Gap	#it	Nopt	Cpu	Cpu*	Gap	Nopt
5	10	0.02	0.00	0.00	1.00	0	0.02	0.00	1.00	0	0.52	0.06	0.00	0	0.15	0.00	0.00	1.90	0	0.02	0.00	1.00	0	0.30	0.07	0.00	0
5	15	0.06	0.00	0.00	1.11	0	0.03	0.00	1.00	0	1.13	0.14	0.00	0	0.15	0.00	0.00	1.22	0	0.04	0.00	1.00	0	0.59	0.17	0.00	0
5	20	0.50	0.00	0.00	1.20	0	0.12	0.00	1.00	0	10.20	0.64	0.00	0	0.64	0.00	0.00	2.20	0	0.11	0.00	1.10	0	3.64	0.40	0.00	0
10	10	0.05	0.00	0.00	1.00	0	0.03	0.00	1.00	0	1.70	0.16	0.00	0	132.12	0.00	0.01	2.70	0	0.15	0.01	1.00	0	1.90	0.56	0.01	0
10	15	0.17	0.00	0.00	1.11	0	0.09	0.00	1.00	0	6.95	0.37	0.00	0	18.43	0.00	0.01	1.56	0	0.19	0.01	1.00	0	2.63	0.72	0.01	0
10	20	16.33	0.00	0.00	1.20	0	0.58	0.00	1.00	0	72.58	7.80	0.00	0	315.19	0.00	0.01	1.90	0	0.71	0.01	1.10	0	58.27	6.30	0.01	0
15	10	0.07	0.00	0.00	1.00	0	0.05	0.00	1.00	0	3.68	0.36	0.00	0	10690.96	1.82	0.01	2.10	5	0.45	0.01	1.00	0	15.03	1.19	0.04	0
15	15	0.36	0.00	0.02	1.11	0	0.18	0.00	1.00	0	38.15	1.07	0.00	0	8600.62	0.83	0.02	1.80	4	1.07	0.01	1.00	0	36.66	3.22	0.01	0
15	20	1708.95	0.00	0.02	1.20	0	878.17	0.00	1.00	0	2631.55	1826.90	0.00	1	13966.43	4.03	1.12	1.29	5	29.31	0.01	1.10	0	329.16	17.04	0.01	0
20	10	0.12	0.00	0.00	1.00	0	0.06	0.00	1.00	0	4.83	0.38	0.00	0	17060.35	7.28	0.04	1.00	9	2.18	0.01	1.00	0	41.32	4.42	0.01	0
20	15	1.05	0.00	0.00	1.00	0	0.35	0.00	1.00	0	48.18	3.37	0.00	0	16064.07	5.56	0.20	1.00	8	16.76	0.01	1.00	0	546.76	26.41	0.01	0
20	20	208.84	0.00	0.02	1.00	0	40.06	0.00	1.00	0	5693.96	2352.36	0.00	3	18037.13	8.44	1.84	1.00	7	92.80	0.01	1.00	0	2611.16	291.58	0.01	0
30	10						0.08	0.00	1.00	0	5.60	0.50	0.00	0						149.59	0.00	1.50	0	3071.60	124.90	0.00	1
30	15						0.88	0.00	1.00	0	392.46	94.08	0.00	0						2360.07	0.05	1.30	1	6313.70	3507.15	0.06	3
30	20						26.44	0.00	1.00	0	4013.72	1908.64	0.00	1						6213.30	0.35	1.70	2	13525.54	6446.57	0.41	7
40	10						0.33	0.00	1.00	0	17.48	1.20	15.05	0						654.06	0.01	1.00	0	7450.67	2033.14	0.01	2
40	15						53.65	0.00	1.00	0	1839.33	147.87	0.00	1						5854.36	0.33	1.00	3	11110.74	5857.90	0.35	6
40	20						6.68	0.00	1.00	0	2235.01	139.24	0.00	1						12039.47	2.61	1.10	6	16237.23	13781.08	4.37	9

Élasticité constante : $\beta_k = 5$ pour tout k

		EXACTE-1					EXACTE-2				Heuristique				EXACTE-1					EXACTE-2				Heuristique			
#OD	%t	Cpu	Gap	BGap	#it	Nopt	Cpu	Gap	#it	Nopt	Cpu	Cpu*	Gap	Nopt	Cpu	Gap	BGap	#it	Nopt	Cpu	Gap	#it	Nopt	Cpu	Cpu*	Gap	Nopt
5	10	0.07	0.00	0.00	1.10	0	0.04	0.00	1.00	0	0.65	0.08	0.00	0	33.67	0.01	0.00	5.00	0	0.07	0.00	1.90	0	0.66	0.20	0.00	0
5	15	0.47	0.00	0.00	2.11	0	0.10	0.00	1.30	0	0.98	0.16	0.00	0	205.47	0.00	0.01	7.56	0	0.13	0.00	1.70	0	1.07	0.28	0.00	0
5	20	1.27	0.00	0.00	2.50	0	0.34	0.00	1.40	0	10.42	0.54	0.00	0	228.48	0.03	0.01	8.20	0	0.55	0.00	2.40	0	11.51	2.60	0.12	0
10	10	0.23	0.00	0.00	1.30	0	0.05	0.00	1.00	0	0.98	0.16	0.00	0	8145.02	1.50	0.00	2.90	4	0.31	0.00	1.80	0	2.88	0.96	0.10	0
10	15	0.85	0.01	0.00	1.44	0	0.10	0.00	1.10	0	3.36	0.27	0.00	0	12518.26	0.85	0.01	5.50	4	0.53	0.00	1.80	0	13.99	3.54	0.02	0
10	20	7.77	0.01	0.00	2.30	0	0.75	0.00	1.40	0	23.34	2.66	0.00	0	18132.46	1.08	0.07	1.50	10	12.35	0.00	2.90	0	272.95	15.15	0.04	0
15	10	0.25	0.00	0.00	1.00	0	0.07	0.00	1.00	0	2.03	0.24	0.00	0	16003.81	4.08	0.80	2.56	7	0.91	0.00	2.20	0	25.95	2.02	0.00	0
15	15	6.94	0.01	0.02	1.33	0	0.24	0.00	1.10	0	26.84	1.83	0.00	0	18046.44	6.24	0.44	1.00	10	20.83	0.00	1.80	0	80.71	5.69	0.00	0
15	20	2968.38	0.09	0.04	3.00	1	1855.96	0.00	1.40	1	2873.71	1848.13	0.72	1	18036.30	24.85	13.82	1.00	8	2039.33	0.01	2.90	1	2600.28	322.07	0.01	1
20	10	0.56	0.00	0.00	1.10	0	0.10	0.00	1.00	0	3.01	0.28	0.00	0	17400.02	36.15	28.31	1.12	7	8.90	0.00	2.20	0	158.50	10.90	0.00	0
20	15	14.10	0.04	0.03	1.67	0	0.47	0.00	1.20	0	65.05	2.58	0.00	0	18046.55	46.06	31.25	1.00	8	1820.46	1.77	1.90	1	256.89	53.60	1.92	0
20	20	3989.82	0.12	0.05	2.50	2	2148.52	0.01	1.50	1	3842.02	1923.99	0.03	2	18032.51	37.54	18.66	1.00	7	6342.66	0.82	2.20	3	5739.59	2165.20	1.12	2
30	10						0.14	0.00	1.00	0	5.38	0.49	0.00	0						3349.28	0.62	1.70	1	2273.76	1815.00	1.22	1
30	15						4.19	0.01	1.00	0	1837.90	7.38	0.01	1						5738.25	4.46	1.20	3	5035.32	1141.97	4.65	2
30	20						1566.20	0.00	1.20	0	1976.60	126.83	0.00	1						13717.79	8.40	1.50	7	13072.63	8967.55	17.42	7
40	10						1.31	0.00	1.20	0	18.51	1.88	0.00	0						5452.81	2.05	1.60	3	4481.07	3662.43	8.73	2
40	15						3.77	0.01	1.30	0	1867.82	31.16	0.01	1						12617.83	7.90	1.30	7	10274.20	7516.80	13.43	4
40	20						17.33	0.01	1.40	0	2104.97	109.60	0.01	1						14742.06	10.84	1.40	8	16216.68	13801.96	17.96	9

Tableau 4.6 Réseaux **Grille** : résultats obtenus avec demande non linéaire et contraintes de capacité (élasticité constante)

Élasticité non constante : $\beta_k = 2$ pour tout k

		Capacité forte												Capacité faible											
		EXACTE-1					EXACTE-2				Heuristique			EXACTE-1					EXACTE-2				Heuristique		
#OD	%t	Cpu	Gap	BGap	#it	Nopt	Cpu	Gap	#it	Nopt	Cpu	Cpu*	Gap	Nopt	Cpu	Gap	BGap	#it	Nopt	Cpu	Gap	#it	Nopt	Cpu	Cpu*	Gap	Nopt
5	10	0.03	0.00	0.00	1.00	0	0.01	0.00	1.00	0	2.17	0.17	0.00	0	0.03	0.00	0.00	1.00	0	0.05	0.00	1.20	0	3.67	0.15	0.00	0
5	15	0.24	0.00	0.01	1.00	0	0.08	0.01	1.00	0	17.15	0.94	0.01	0	0.32	0.00	0.00	1.00	0	0.23	0.00	1.50	0	15.18	1.65	0.00	0
5	20	0.33	0.00	0.00	1.00	0	0.09	0.00	1.00	0	21.21	4.08	0.00	0	0.85	0.00	0.00	1.00	0	0.50	0.00	1.70	0	20.75	1.00	0.01	0
10	10	0.06	0.00	0.00	1.00	0	0.02	0.00	1.10	0	8.47	0.37	0.00	0	0.23	0.00	0.00	1.10	0	0.25	0.00	1.50	0	11.25	2.10	0.00	0
10	15	0.66	0.00	0.01	1.10	0	0.26	0.01	1.00	0	39.35	5.48	0.01	0	4.50	0.00	0.00	1.00	0	2.53	0.00	1.50	0	75.68	15.28	0.00	0
10	20	2.47	0.00	0.01	1.00	0	1.48	0.01	1.00	0	67.87	5.03	0.01	0	354.41	0.00	0.00	1.20	0	19.23	0.00	1.50	0	96.97	12.89	0.00	0
15	10	0.07	0.00	0.00	1.00	0	0.03	0.00	1.00	0	5.96	0.28	0.00	0	0.28	0.00	0.00	1.00	0	0.38	0.00	1.60	0	24.25	7.10	0.00	0
15	15	5.89	0.00	0.00	1.00	0	0.32	0.00	1.00	0	53.01	5.33	0.00	0	1429.23	0.00	0.00	1.10	0	19.03	0.00	1.90	0	132.58	44.82	0.02	0
15	20	395.71	0.00	0.01	1.00	0	12.16	0.01	1.00	0	148.41	8.11	0.07	0	510.89	0.00	0.00	1.20	0	141.63	0.00	1.90	0	410.64	21.41	0.00	0
20	10	0.11	0.00	0.00	1.00	0	0.05	0.00	1.10	0	14.01	0.68	0.00	0	0.60	0.00	0.00	1.10	0	0.78	0.00	1.70	0	31.13	2.66	0.00	0
20	15	40.86	0.00	0.00	1.10	0	19.32	0.00	1.10	0	926.78	33.98	0.00	0	315.17	0.00	0.00	1.00	0	24.31	0.00	1.70	0	560.32	25.11	0.00	0
20	20	184.03	0.00	0.01	1.00	0	4.42	0.01	1.10	0	259.16	13.92	0.01	0	230.12	0.00	0.00	1.20	0	276.37	0.00	1.90	0	1904.62	63.58	0.00	0
30	10						0.07	0.01	1.00	0	18.06	0.90	0.01	0						2.20	0.00	1.70	0	90.05	10.96	0.00	0
30	15						55.64	0.00	1.10	0	1862.80	49.98	0.00	1						37.43	0.00	1.70	0	2001.20	198.00	0.00	1
30	20						6.40	0.01	1.00	0	812.99	78.45	0.01	0						3224.40	0.09	1.60	1	6069.66	703.67	0.11	3
40	10						0.09	0.00	1.00	0	33.55	3.00	0.00	0						5.28	0.01	1.00	0	92.77	6.50	0.01	0
40	15						28.84	0.01	1.10	0	690.66	35.00	0.09	0						69.17	0.02	1.00	0	2448.01	126.58	0.30	1
40	20						37.58	0.01	1.10	0	737.17	271.31	0.01	0						1090.77	0.01	1.20	0	7494.43	1624.49	0.01	3

Élasticité non constante : $\beta_k = 5$ pour tout k

		EXACTE-1					EXACTE-2				Heuristique				EXACTE-1					EXACTE-2				Heuristique			
#OD	%t	Cpu	Gap	BGap	#it	Nopt	Cpu	Gap	#it	Nopt	Cpu	Cpu*	Gap	Nopt	Cpu	Gap	BGap	#it	Nopt	Cpu	Gap	#it	Nopt	Cpu	Cpu*	Gap	Nopt
5	10	0.04	0.00	0.00	1.10	0	0.01	0.00	1.00	0	0.89	0.07	0.00	0	1.39	0.00	0.00	1.60	0	0.04	0.00	1.30	0	3.07	0.21	0.00	0
5	15	1.07	0.00	0.00	2.10	0	0.23	0.00	1.70	0	11.78	1.49	0.00	0	61.09	0.01	0.01	5.40	0	24.24	0.00	2.90	0	698.62	299.28	0.29	0
5	20	13.82	0.00	0.00	2.50	0	7.87	0.01	2.10	0	16.80	1.60	0.00	0	188.39	0.02	0.01	4.70	0	214.85	0.01	3.10	0	122.57	31.81	0.02	0
10	10	0.34	0.00	0.01	2.60	0	0.06	0.01	1.40	0	2.99	0.20	0.00	0	743.50	0.00	0.00	3.00	0	0.32	0.00	2.10	0	32.42	8.31	0.00	0
10	15	14.08	0.00	0.01	2.80	0	1.69	0.03	2.40	0	45.05	8.15	0.01	0	9895.82	56.61	51.86	3.44	4	44.75	0.03	3.70	0	432.57	49.68	0.00	0
10	20	87.90	0.00	0.00	2.20	0	3.28	0.04	2.00	0	50.16	7.31	0.00	0	9820.01	9.07	7.42	3.30	5	1997.54	0.06	3.50	0	378.99	22.15	0.06	0
15	10	0.20	0.00	0.00	2.50	0	0.07	0.00	1.60	0	2.81	0.25	0.00	0	5291.71	0.55	0.01	2.22	2	1.04	0.01	3.50	0	37.94	7.31	0.00	0
15	15	438.08	0.00	0.01	2.80	0	3.80	0.01	2.40	0	69.14	3.64	0.01	0	16340.38	88.78	70.59	1.40	9	3619.75	1.90	3.20	2	355.97	70.80	2.02	0
15	20	233.44	0.00	0.01	2.70	0	6.25	0.01	2.40	0	58.28	5.37	0.01	0	15087.29	59.17	56.75	1.44	6	5564.31	0.62	2.80	3	2722.95	383.52	1.06	1
20	10	0.22	0.00	0.01	2.70	0	0.08	0.01	1.50	0	12.63	0.62	0.00	0	4839.54	13.18	12.63	2.00	2	0.82	0.33	2.10	0	60.81	10.60	0.02	0
20	15	3625.62	0.00	0.02	3.00	2	6.95	0.02	1.90	0	67.57	6.16	0.20	0	18040.25	77.99	56.76	1.25	8	5124.61	0.80	2.70	1	1560.96	396.90	1.03	0
20	20	4020.73	0.40	0.39	2.50	2	24.56	0.02	3.33	0	172.78	33.83	0.01	0	14089.08	39.00	55.23	2.44	7	6937.59	1.65	2.20	3	5376.08	1418.36	2.03	2
30	10						0.15	0.01	1.78	0	26.46	2.06	0.01	0						6.27	0.02	2.30	0	124.33	8.07	0.03	0
30	15						1893.87	0.18	2.10	1	130.81	7.76	0.20	0						6418.65	2.10	1.89	3	4291.59	1592.26	2.50	2
30	20						3319.00	0.99	2.00	1	376.06	30.12	1.00	0						10879.05	2.53	1.40	6	7742.11	3527.45	3.53	4
40	10						0.24	0.01	1.78	0	20.45	4.60	0.01	0						46.70	0.01	2.90	0	147.27	9.94	0.01	0
40	15						1840.38	0.91	2.20	1	122.24	44.28	0.91	0						13046.71	4.17	1.80	7	6241.30	3830.69	4.94	2
40	20						2813.56	0.40	2.00	1	454.81	77.55	1.45	0						16411.53	5.15	1.10	9	10313.74	5680.50	4.84	5

Tableau 4.7 Réseaux **Voronoï** : résultats obtenus avec demande non linéaire et contraintes de capacité (élasticité non constante)

Élasticité non constante : $\beta_k = 2$ pour tout k

		Capacité forte												Capacité faible											
		EXACTE-1					EXACTE-2				Heuristique				EXACTE-1					EXACTE-2				Heuristique			
#OD	%t	Cpu	Gap	BGap	#it	Nopt	Cpu	Gap	#it	Nopt	Cpu	Cpu*	Gap	Nopt	Cpu	Gap	BGap	#it	Nopt	Cpu	Gap	#it	Nopt	Cpu	Cpu*	Gap	Nopt
5	10	0.02	0.00	0.00	1.00	0	0.03	0.00	1.00	0	0.97	0.10	0.00	0	0.04	0.00	0.00	1.00	0	0.03	0.00	1.00	0	0.84	0.13	0.01	0
5	15	0.03	0.00	0.00	1.00	0	0.05	0.00	1.00	0	0.97	0.11	0.00	0	0.05	0.00	0.00	1.00	0	0.05	0.00	1.00	0	1.25	0.17	0.00	0
5	20	0.13	0.00	0.00	1.00	0	0.22	0.00	1.00	0	8.58	0.58	0.00	0	0.13	0.00	0.00	1.00	0	0.16	0.00	1.00	0	9.56	0.90	0.00	0
10	10	0.03	0.00	0.00	1.00	0	0.04	0.00	1.00	0	1.20	0.11	0.00	0	0.23	0.00	0.01	1.00	0	0.23	0.01	1.00	0	6.49	0.58	0.01	0
10	15	0.06	0.00	0.00	1.00	0	0.08	0.00	1.00	0	12.42	0.26	0.00	0	0.34	0.00	0.00	1.00	0	0.26	0.00	1.00	0	10.91	0.98	0.00	0
10	20	0.51	0.00	0.00	1.00	0	0.85	0.00	1.00	0	237.05	6.86	0.00	0	1.37	0.00	0.00	1.00	0	1.26	0.00	1.00	0	142.21	4.18	0.00	0
15	10	0.04	0.00	0.00	1.00	0	0.06	0.00	1.00	0	1.66	0.15	0.00	0	1.65	0.00	0.01	1.00	0	1.15	0.00	1.00	0	37.56	11.11	0.00	0
15	15	0.14	0.00	0.02	1.00	0	0.16	0.00	1.00	0	34.32	0.83	0.00	0	11.20	0.00	0.01	1.00	0	2.26	0.00	1.00	0	114.94	14.38	0.00	0
15	20	3.09	0.00	0.02	1.00	0	4.98	0.00	1.00	0	2996.74	210.62	0.00	0	575.91	0.00	0.01	1.00	0	103.04	0.00	1.00	0	4020.84	1833.82	1.40	2
20	10	0.05	0.00	0.00	1.00	0	0.07	0.00	1.00	0	2.99	0.25	0.00	0	35.59	0.00	0.01	1.00	0	44.47	0.01	1.00	0	1751.26	98.10	0.01	0
20	15	0.39	0.00	0.00	1.00	0	0.32	0.00	1.00	0	44.33	1.80	0.00	0	76.73	0.00	0.01	1.00	0	62.38	0.01	1.00	0	2824.39	359.18	0.01	1
20	20	6.45	0.00	0.02	1.00	0	7.80	0.00	1.00	0	3773.11	548.10	0.00	2	3683.18	0.48	0.33	1.00	2	2645.51	0.32	1.00	1	9131.50	4506.68	0.64	4
30	10						0.10	0.00	1.00	0	6.53	0.62	0.00	0						58.76	0.01	1.00	0	4054.71	1893.88	0.01	1
30	15						0.60	0.00	1.00	0	529.92	8.48	0.00	0						1822.78	0.58	1.00	1	7241.26	2095.18	0.58	2
30	20						8.37	0.00	1.00	0	2227.97	413.82	0.00	1						2223.75	0.14	1.00	1	14060.11	6109.72	0.14	6
40	10						0.13	0.00	1.00	0	6.64	0.58	0.00	0						618.50	0.01	1.00	0	5543.01	2212.52	0.01	3
40	15						6.16	0.00	1.00	0	1825.51	141.56	0.00	1						1299.64	0.01	1.00	0	6096.83	2592.41	0.01	3
40	20						2.93	0.00	1.00	0	217.57	9.49	0.00	0						5560.74	1.09	1.00	3	10915.79	7504.43	2.39	5

Élasticité non constante : $\beta_k = 5$ pour tout k

		EXACTE-1					EXACTE-2				Heuristique				EXACTE-1					EXACTE-2				Heuristique			
#OD	%t	Cpu	Gap	BGap	#it	Nopt	Cpu	Gap	#it	Nopt	Cpu	Cpu*	Gap	Nopt	Cpu	Gap	BGap	#it	Nopt	Cpu	Gap	#it	Nopt	Cpu	Cpu*	Gap	Nopt
5	10	0.04	0.00	0.00	1.10	0	0.02	0.00	1.00	0	0.69	0.08	0.00	0	0.83	0.00	0.01	3.50	0	0.07	0.01	1.10	0	1.14	0.13	0.01	0
5	15	0.08	0.00	0.00	1.60	0	0.03	0.00	1.00	0	0.70	0.13	0.00	0	2.82	0.00	0.01	2.60	0	0.07	0.01	1.00	0	1.37	0.20	0.01	0
5	20	0.41	0.00	0.00	1.90	0	0.18	0.00	1.10	0	14.51	0.51	0.00	0	17.18	0.01	0.02	4.50	0	0.34	0.02	1.20	0	19.66	1.49	0.02	0
10	10	0.09	0.00	0.00	1.30	0	0.03	0.00	1.00	0	1.43	0.16	0.00	0	17.25	0.00	0.01	2.30	0	0.21	0.01	1.10	0	11.42	1.14	0.01	0
10	15	0.14	0.00	0.00	1.30	0	0.06	0.00	1.00	0	6.10	0.31	0.00	0	63.50	0.00	0.01	3.00	0	0.30	0.01	1.00	0	21.54	1.67	0.01	0
10	20	9.88	0.00	0.00	2.40	0	0.57	0.00	1.10	0	1468.76	142.24	0.00	0	1939.73	0.01	0.01	2.50	1	10.61	0.01	1.30	0	214.52	11.94	0.01	0
15	10	0.09	0.00	0.00	1.10	0	0.04	0.00	1.00	0	2.35	0.27	0.00	0	4111.79	0.02	0.01	1.90	2	0.63	0.01	1.10	0	40.20	6.75	0.01	0
15	15	0.51	0.00	0.02	1.50	0	1.18	0.00	1.00	0	39.64	2.11	0.00	0	5449.47	0.29	0.03	1.50	3	13.32	0.01	1.00	0	1993.35	991.77	0.01	1
15	20	921.77	0.00	0.03	2.90	0	84.00	0.01	1.00	0	3102.71	930.08	0.01	1	5850.18	1.16	0.26	1.50	3	1813.10	0.23	1.00	1	333.98	41.89	0.01	0
20	10	0.14	0.00	0.00	1.10	0	0.06	0.00	1.00	0	2.07	0.28	0.00	0	3949.43	6.77	6.28	2.00	2	71.50	0.01	1.20	0	1845.46	1806.74	0.02	1
20	15	52.84	0.00	0.01	1.50	0	0.27	0.00	1.00	0	59.03	1.70	0.00	0	2346.74	0.28	0.06	3.50	1	803.97	0.02	1.00	0	331.93	19.23	0.02	0
20	20	2182.36	0.23	0.25	3.00	1	284.55	0.01	1.10	0	4108.92	2099.45	0.01	1	9766.79	1.63	0.49	3.50	5	2856.08	0.19	1.00	1	9498.38	5891.81	0.46	5
30	10						0.11	0.00	1.00	0	5.00	0.56	0.00	0						45.51	0.02	1.20	0	2412.47	449.28	0.02	1
30	15						2.60	0.00	1.00	0	1390.74	63.65	0.00	0						397.10	0.01	1.20	0	6010.85	2664.67	0.01	3
30	20						56.22	0.00	1.00	0	1907.33	617.24	0.00	1						2135.72	0.01	1.40	0	9435.07	7902.43	0.01	5
40	10						0.14	0.00	1.00	0	6.63	0.64	0.00	0						44.89	0.01	1.20	0	3940.42	1732.51	0.01	2
40	15						21.71	0.01	1.00	0	1671.23	55.94	0.01	0						317.55	0.01	1.30	0	7336.41	3365.92	0.28	3
40	20						2.89	0.01	1.00	0	174.84	16.31	0.01	0						5501.62	0.40	1.20	3	7549.21	6415.08	0.42	3

Tableau 4.8 Réseaux **Grille** : résultats obtenus avec demande non linéaire et contraintes de capacité (élasticité non constante)

Les deux méthodes exactes diffèrent l'une de l'autre par les fonctions non linéaires utilisées au niveau de la fonction objectif. Ces fonctions jouent un rôle très important dans l'efficacité de la méthode exacte à cause de leur surapproximation qui augmente la difficulté de résolution du DNL-CAP. Les fonctions utilisées pour EXACT-2 sont plus faciles à surapproximer que celles utilisées pour EXACT-1. Il en résulte donc un temps d'exécution élevé de EXACT-1 comparé à celui de EXACT-2. La même remarque a été faite pour le DNL. Dans la suite, nous utilisons EXACT-2 pour commenter les résultats de l'analyse de sensibilité sur l'élasticité et les contraintes de capacité.

Le DNL-CAP avec élasticité constante est plus difficile à résoudre que celui avec élasticité non constante car les fonctions non linéaires à surapproximer au niveau de l'objectif sont toujours convexes lorsque l'élasticité est constante tandis qu'elles sont plutôt pseudo-concaves lorsque l'élasticité est non constante. À titre d'exemple, en considérant le réseau Voronoï (resp. Grille), $\beta_k = 5$ et les contraintes de capacité fortes, le temps moyen d'exécution et l'erreur relative moyenne est 2 heures et 4.49% (resp. 1 heure et 2.04%) avec l'élasticité constante contre 1 heure 10 minutes et 1.05% (resp. 13 minutes et 0.05%) avec l'élasticité non constante. Pour ces instances, on note que le temps d'exécution avec élasticité constante est 1.5 (resp. 4.8) fois celui avec élasticité non constante. Dans la suite, nous utilisons les résultats obtenus avec élasticité constante pour étayer nos remarques.

La méthode EXACT-2 est très efficace pour les instances à contraintes de capacité fortes où l'on est capable de résoudre les instances de 40 produits en moins de 1 heure. Par contre, lorsque les contraintes de capacité sont faibles, elle est performante en général sur les instances de 20 produits et moins indépendamment de l'élasticité. Cependant, lorsque la taille des instances augmente, elle devient moins efficace comme en témoignent les instances de Voronoï de 40 produits, 20% d'arcs taxables et $\beta_k = 5$ où toutes les instances ont atteint le temps d'exécution maximal et l'erreur relative moyenne est 20%. Il s'ensuit que le DNL-CAP est plus difficile avec les contraintes de capacité faibles qu'avec les contraintes de capacité fortes, en d'autres termes la difficulté du DNL-CAP augmente avec la capacité sur les arcs. À

titre d'illustration, en considérant la Grille, le temps moyen d'exécution des instances de 20 produits et moins avec contraintes de capacité faibles est 2.1 fois le temps d'exécution avec contraintes de capacité fortes. Ce facteur passe à 50 lorsque le nombre de produits se situe entre 30 et 40. La figure 4.9 ci-dessous présente l'évolution du temps moyen d'exécution en fonction des capacités sur les arcs. Elle est obtenue en fixant le temps d'exécution maximal à 10 heures sur les instances de Voronoï de 20 produits, 20% d'arcs taxables et $\beta_k = 2$.

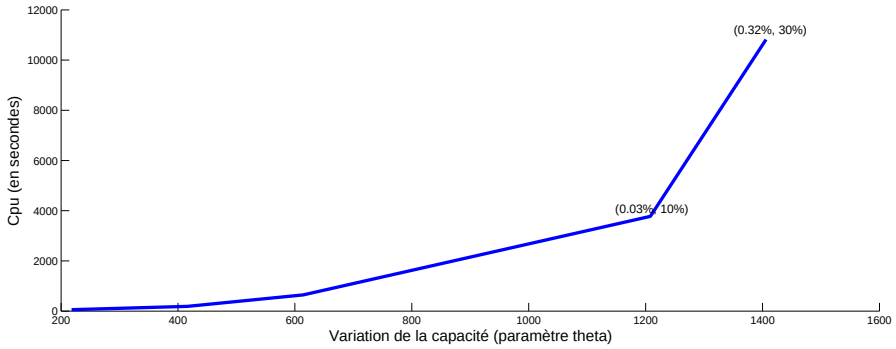


Figure 4.9 Variation du temps moyen d'exécution en fonction des capacités sur les arcs

Cette courbe montre que le temps moyen d'exécution, l'erreur relative moyenne et le pourcentage d'instances ayant atteint le temps maximal augmente avec les capacités sur les arcs.

Nous constatons que la difficulté du DNL-CAP augmente avec l'élasticité de la demande. Par exemple, en considérant les instances du réseau de Voronoï (resp. Grille), le temps moyen d'exécution et l'erreur relative moyenne est de 27 minutes et 0.42% (resp. 5 minutes et 0.002%) lorsque $\beta_k = 2$ contre 2 heures et 4.49% (resp. 1 heure et 2.04%) lorsque $\beta_k = 5$. La figure 4.10 ci-dessous présente l'évolution du temps moyen d'exécution en fonction de l'élasticité de la demande. Elle est obtenue en fixant le temps d'exécution maximal à 10 heures sur les instances de Grille avec 20 produits, 20% d'arcs taxables et des capacités moyennes.

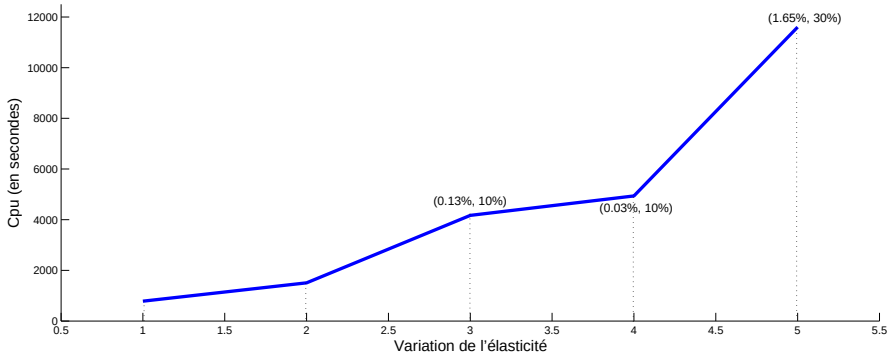


Figure 4.10 Variation du temps moyen de calcul en fonction de l'élasticité de la demande

Cette courbe montre que le temps moyen d'exécution, l'erreur relative moyenne et le pourcentage d'instances ayant atteint le temps maximal augmente avec l'élasticité de la demande.

L'heuristique développée fournit des solutions de bonne qualité avec un temps de calcul raisonnable. Pour les instances où la solution optimale a été trouvée, l'erreur relative moyenne des solutions fournies par l'heuristique est 0.5% et, pour les autres instances, les solutions de l'heuristique sont situées à au plus 3.26% de la borne supérieure fournie par la méthode EXACT-2. Nous pouvons donc conclure que l'heuristique fournie des solutions situées à au plus 2% de la borne supérieure du DNL-CAP. De manière générale, l'heuristique trouve rapidement une solution de bonne qualité comparée à la méthode EXACT-2 comme le montrent les figures 4.11, 4.12 et 4.13 ci-dessous.

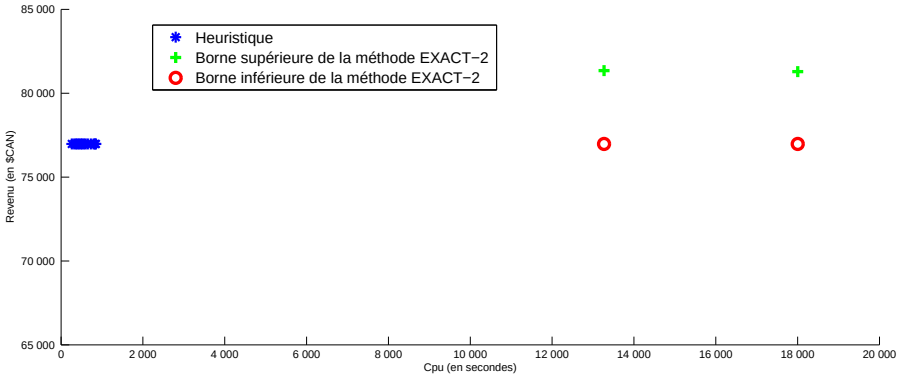


Figure 4.11 Évolution de la méthode EXACT-2 et l'heuristique en fonction du temps d'exécution sur une instance de **Voronoï** de 40 produits et 20% d'arcs taxables avec élasticité constante ($\beta_k = 5$) et contraintes de capacité fortes

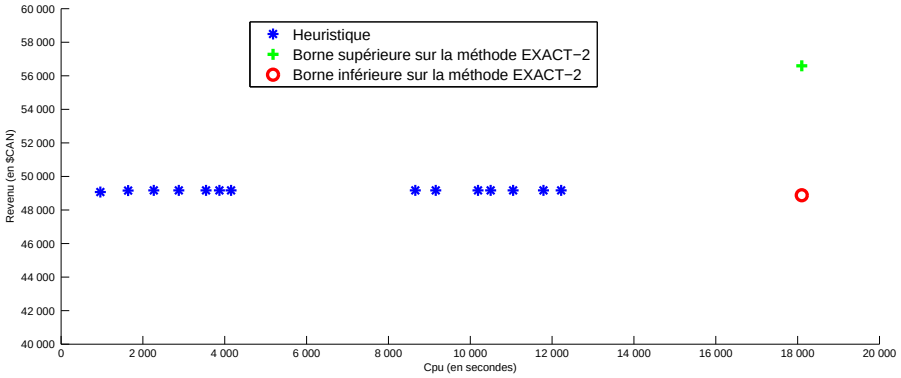


Figure 4.12 Évolution de la méthode EXACT-2 et l'heuristique en fonction du temps d'exécution sur une instance de **Voronoï** de 40 produits et 20% d'arcs taxables avec élasticité constante ($\beta_k = 5$) et contraintes de capacité faibles

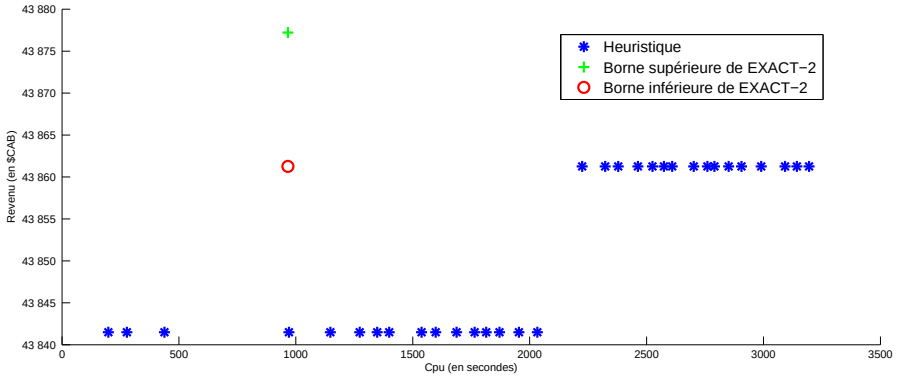


Figure 4.13 Évolution de la méthode EXACT-2 et l'heuristique en fonction du temps d'exécution sur une instance de **Voronoï** de 40 produits et 20% d'arcs taxables avec élasticité non constante ($\beta_k = 5$) et contraintes de capacité faibles

La moyenne du temps d'exécution de la méthode EXACT-2 est 1.6 fois celle de l'heuristique. Lorsque les contraintes de capacité sont fortes, l'erreur relative moyenne de la méthode EXACT-2 est légèrement inférieure à celle de l'heuristique bien que la différence soit inférieure à 0.5%. Pour certaines de ces instances, la méthode EXACT-2 trouve rapidement une solution de bonne qualité comparée à l'heuristique (voir figures 4.14 et 4.15 ci-dessous).

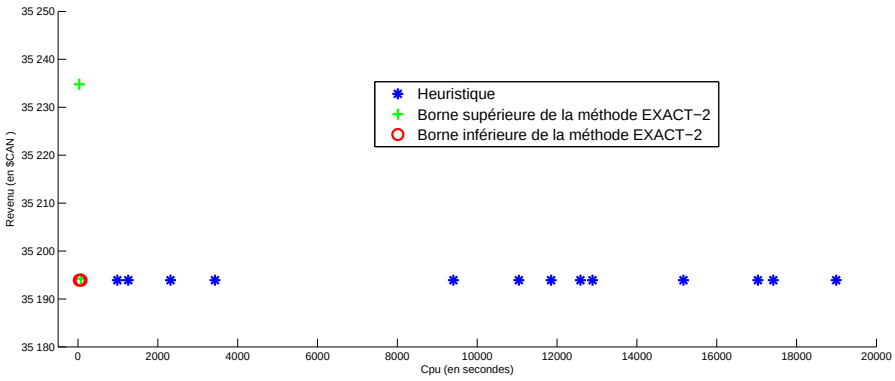


Figure 4.14 Évolution de la méthode EXACT-2 et l'heuristique en fonction du temps d'exécution sur une instance de **Grille** de 40 produits et 20% d'arcs taxables avec élasticité constante ($\beta_k = 5$) et contraintes de capacité fortes

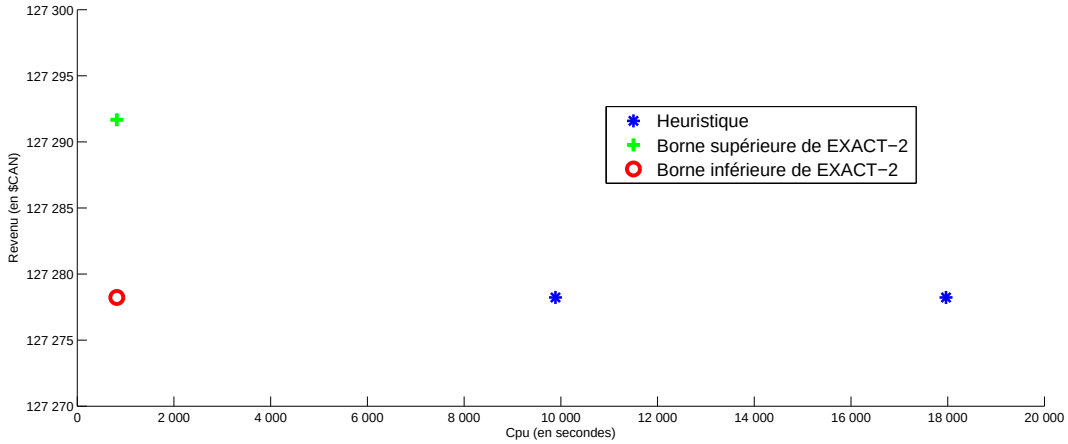


Figure 4.15 Évolution de la méthode EXACT-2 et l'heuristique en fonction du temps d'exécution sur une instance de **Grille** de 40 produits et 20% d'arcs taxables avec élasticité non constante ($\beta_k = 5$) et contraintes de capacité faibles

Enfin, nous remarquons que le temps de calcul de l'heuristique augmente avec l'élasticité de la demande et les contraintes de capacité.

Nous comparons DNL et le DNL-CAP sur les instances de 40 produits et moins afin de présenter l'impact de l'incorporation des contraintes de capacité au DNL. La formulation DNL-CAP:CH contient le même nombre de variables binaires (associées aux chemins) que DNL:CH. Les deux modèles possèdent exactement les mêmes contraintes à l'exception des contraintes de capacité. Cependant l'espace des solutions des variables binaires du DNL:CH est moins combinatoire comparé à celui des variables binaires du DNL-CAP:CH car un seul chemin est utilisé pour chaque produit du DNL:CH tandis qu'au moins un chemin est utilisé pour chaque produit du DNL-CAP:CH. Par conséquent, l'aspect combinatoire du DNL augmente lorsqu'on incorpore les contraintes de capacité, et donc le DNL est moins difficile que le DNL-CAP. À titre d'exemple, en considérant les instances de 40 produits et moins, le temps moyen d'exécution, l'erreur relative moyenne et le pourcentage d'instances ayant atteint le temps d'exécution maximal avec le DNL (resp. DNL-CAP avec contraintes de capacité fortes et faibles) sont 4 minutes, 0.03% et 0.09% (resp. 9 minutes, 0.10% et 0.26% et 1 heure, 1.60%

et 1.80%). Ces valeurs augmentent lorsque l'on considère seulement les instances à 30 et 40 produits et deviennent 14 minutes, 0.06% et 0.29% (resp. 20 minutes, 0.21% et 0.54% et 2 heures, 4.64% et 4.04%) pour le DNL (resp. DNL-CAP avec contraintes de capacité fortes et faibles). Notons par ailleurs que la borne supérieure du MIP de la méthode EXACT-2 pour le DNL-CAP est de bonne qualité comparée à celle de la méthode EXACT-2 du DNL bien qu'elle décroît très lentement que celle du DNL. Tel que mentionné pour le DL-CAP, le DNL-CAP est facile lorsque les contraintes de capacité sont très fortes car pour tout vecteur de tarifs qui génère du revenu, toutes les contraintes de capacité sont actives et la fonction objectif du meneur est linéaire en fonction des tarifs. Dans ce cas, le DNL devient concave et toute méthode de recherche locale résout le problème.

CONCLUSION

Dans sa formulation originelle à deux niveaux, le problème de tarification optimale sur un réseau ne tient pas compte de l'élasticité de la demande par rapport aux coûts de transport. Bien sûr, ceci peut être modélisé implicitement par le biais de chemins artificiels permettant d'absorber une demande fictive, mais ce procédé ne permet pas une représentation fine du processus. Dans cette thèse, nous avons intégré explicitement au modèle des fonctions de demande. Bien que celui-ci devienne non linéaire, l'affectation « tout-ou-rien » des usagers aux chemins de coût minimal lui confère un caractère combinatoire que nous avons exploité dans diverses formulations impliquant à la fois des variables continues et binaires. En nous basant sur ces formulations, nous avons proposé des algorithmes de trois types : exact, asymptotiquement exact (basé sur des approximations) et heuristique, soit dans l'espace des flots d'arcs, soit dans celui des flots de chemins. À partir de tests numériques impliquant des réseaux générés aléatoirement, il en ressort que la difficulté majeure de résolution demeure combinatoire. En particulier, les instances où la demande est linéaire ne sont pas beaucoup plus difficiles à résoudre que celles impliquant une demande constante. Par contre, lorsque les fonctions de demande sont non linéaires et faiblement élastiques, le problème devient plus difficile d'un point de vue numérique.

Dans un deuxième temps, nous avons intégré à notre modèle des capacités sur les arcs tarifés. Lorsque ces capacités sont élevées, elles jouent un rôle minimal et influencent peu les temps de résolution. Lorsque qu'elles sont faibles, le problème devient fortement contraint et les flots ont tendance, à l'optimum, à se concentrer sur les chemins de plus faibles coûts fixes. Comme, pour un ensemble de chemins prédéterminés, le problème est concave pour une grande classe de fonctions de demande, des algorithmes basés sur une recherche locale convergent fréquemment vers l'optimum global.

Cette thèse était principalement consacrée à la modélisation et à la résolution numé-

rique du problème de tarification avec demande variable. Au cours de cette étude, un certain nombre de questions se sont posées naturellement. La plus intrigante concerne la complexité du problème de tarification sur des réseaux où tous les arcs peuvent être tarifés. Nous conjecturons que cette variante est NP-difficile mais n'avons pas pu le prouver. Nous terminons en mentionnant des extensions naturelles du problème à des situations plus réaliste où le comportement du consommateur n'est pas uniforme dans la population. Deux cas nous viennent à l'esprit, d'abord celui de l'affectation des usagers aux chemins du réseau selon un modèle de choix discret (logit, probit, etc.), puis celui de la partition des usagers en classes caractérisées par leur « valeur du temps », cette partition étant représentée par le biais d'une loi de probabilité discrète ou continue.

RÉFÉRENCES

[1] F.A. Al-Khayyal(1992). Generalized Bilinear Programming, Part I : Models, Applications and Linear Programming Relaxation. *European Journal of Operational Research, Vol. 60 ,N° 3, p 306–314.*

[2] C. Audet (1997). Optimisation Globale Structurée : Propriétés, Équivalences et Résolution. *Thèse de Doctorat, École Polytechnique de Montréal.*

[3] C. Audet, P. Hansen, B. Jaumard and G. Savard (1999). A Branch and Cut Algorithm for Nonconvex Quadratically Constrained Quadratic Programming. *Mathematical Programming, Vol. 87 ,N° 1, p 131-152.*

[4] C. Audet, P. Hansen, B. Jaumard and G. Savard (1997). Links between the Linear Bilevel and Mixed 0-1 Programming Problems. *Journal of Optimization Theory an Applications. Vol. 93, N° 2, p 273-300.*

[5] F. Babonneau and J.P. Vial (2008). An Efficient Method to compute Traffic Assignment Problems with Elastic Demands. *Transportation Science, Vol. 42, N° . 2, p 249-260.*

[6] J.F. Bard (1991). Some Properties of the Bilevel Programming Problem. *Journal of Optimization Theory and Applications, Vol. 68, N° 2, p 371–378.*

[7] M. S. Bazaraa, H. D. Sherali and C.H. Shetty (2006). Nonlinear Programming : Theory and Algorithms. *Third Edition, book.*

[8] M. J. Beckmann and T. F. Golob (1974). Traveler decisions and traffic flows : a behavioral theory of network equilibrium. *Proc. 6th Int. Symp. on Transportation and Traffic Theory (Buckley D. J. Ed., p. 453-482). Elsevier, New York.*

[9] M. J. Beckmann, C. B. McGuire and C. B. Winsten (1956). Studies in the Economics of Transportation. *Yale University Press, New Haven, CT.*

[10] O. Ben-Ayed (1993). Bilevel Linear Programming. *Computer and Operations Research, Vol. 20, N° 5, p 485-501.*

[11] M. Bouhtou, S.v. Hoesel, A.F.v.d. Kraaij and J.L. Lutton (2007). Tariff Optimization in Networks. *Informs Journal on Computing, summer 2007, Vol. 19, N° 3, p 458-469.*

[12] L. Brotcorne, M. Labbé, P. Marcotte and G. Savard (2000). A Bilevel Model and Solution Algorithm for Freight Tariff-Setting Problem. *Transportation Science, Vol. 34, N° 3, p 289-302.*

[13] L. Brotcorne, M. Labbé, P. Marcotte and G. Savard (2001). A Bilevel Model for Toll Optimization on a Multicommodity Transportation Network. *Transportation Science, Vol. 35, N° 4, p 345-358.*

[14] L. Brotcorne, P. Marcotte, G. Savard and M. Wiart. (2008) Joint Pricing and Network Capacity Setting Problem. *Operations Research, sept 2008, Vol. 56, N° 5, p 1105-1115.*

[15] D. G. Cantor and M. Gerla (1974). Optimal routing in a packet-switched computer network. *IEEE Transactions on computers, Vol. 23, N° 3, p 1062-1069.*

[16] M. Chouman, P. Marcotte and G. Savard (2005). Bilevel Model for Pricing Telecommunications Services. *INFORMS Annual Meeting, San Francisco, 13-16 novembre 2005.*

[17] F. Cirinei (2007). Problème de Tarification sur un Réseau. *Thèse de doctorat, Université de Valenciennes et du Hainaut-Cambrésis, France.*

[18] B. Colson, P. Marcotte, Savard, G (2005). An implementable Trust-Region Method for Nonlinear Bilevel Programming. *Computational Optimization and Applications, Vol. 30, p 211-227.*

[19] B. Colson, P. Marcotte, Savard, G (2005). Bi-level Programming : A survey. *Business and Economics, Vol. 3, N° 2, p 87-107.*

[20] A.R. Conn, N.I.M. Gould, and Ph.L. Toint (2000). Trust-Region Methods. *SIAM Publications : Philadelphia, Pennsylvania, USA*

[21] J.P Côté, P. Marcotte and G. Savard (2003). A Bi-level Model Approach to Pricing and Fare Optimisation in Airline Industry. *Journal of Revenue and Pricing Management, Vol. 2, p 24-36.*

[22] Cplex 2010. *Using the Cplex Callable Library and Cplex Mixed Integer Programming.*

[23] S. Dafermos (1980). Traffic Equilibrium and Variational Inequalities. *Transportation Science, Vol. 14, N° 1, p 42-54.*

[24] S. Dempe (1992). A Necessary and a Sufficient Optimality Condition for Bilevel Programming Problems.*Optimization, Vol. 25, N° 4, p 341-354.*

[25] S. Dempe and J. F. Bard (2001). Bundle Trust-Region Algorithm for Bilinear Bilevel Programming. *In Journal of Optimization Theory and Applications, Vol. 110, N° 2, p 265-288.*

[26] S. Dempe (2002). Foundations of bilevel programming. *In Nonconvex Optimization and its Applications, Vol. 61, 320 p.*

[27] S. Dewez (2004). On the Toll Setting Problem. *Ph.D thesis, Université Libre de Bruxelles, Institut de Statistique et de Recherche Opérationnelle, Belgique.*

[28] S. Dewez, M. Labbé, P. Marcotte, and G. Savard (2008). New Formulations and Valid Inequalities for a Bilevel Pricing Problem. *Operations Research Letters, Vol. 36, N° 2, p 141-149.*

[29] M. Didi B., P. Marcotte and G. Savard (2006). Path-based Formulations of a Bilevel Toll Setting Problem. *Optimization and its Application, Vol. 2, N° 1, p 29-50.*

[30] M. Florian and D.W. Hearn (2003). Network Equilibrium and Pricing. *Handbook of transportation science, 2nd edition, R.W. Hall (Editor), Kluwer academic publisher, Norwell, MA, Vol. 56, N° 5, p 373-412.*

[31] M. Fortin (2004). Tarification avec Segmentation de la Demande et Congestion. *Mémoire de Maîtrise, École Polytechnique de Montréal, Canada.*

[32] M. Frank and P. Wolfe (1956). An Algorithm for Quadratic Programming. *Naval Research Logistics Quarterly, Vol. 3, N° 1-2, p 95-110.*

[33] N.H. Gartner (1980). Optimal Traffic with Elastic Demand. *Review Part I : Analysis Framework, Transportation Science, Vol. 14, N° 2, p 174-207.*

[34] N.H. Gartner (1980). Optimal Traffic with Elastic Demand. *A Review ; Part II : Algorithmic Approaches, Transportation Science, Vol. 14, N° 2, p 192-208.*

[35] M. Gendreau, P. Marcotte and G. Savard (1996). A Hybrid Tabu-Ascent Algorithm for the Linear Bilevel Programming Problem. *Journal of Global Optimization, Vol. 8, N° 3, p 217–233.*

[36] A. Grigoriev, S. v. Hoesel, F. v. d. Kraaij, M. Uetz and M. Bouhtou (2005). Pricing Network Edges to Cross a River. *Lecture Notes in Computer Science 3351, p 140-153.*

[37] J. Han, G. Liu and W. Shouyang (2000). A New Descent Algorithm for Soving Quadratic Bilevel Programming Problems. *Mathematics and Statistics, Vol. 16, N° 3, p 235-244.*

[38] D.W. Hearn, B.M. Yildirim (2002). A Toll Pricing Framework for Traffic Assignment Problem with Elastic Demand. *Center for Applied Optimization. Industrial and System Engeneering Department, University of Florida.*

[39] D.W. Hearn, B.M. Yildirim (2005). A First Best Toll Pricing Framework for Variable demand traffic Assignment Problems. *Transportation Research Part B : Methodological. Vol. 39, N° 8, p 659-678.*

[40] D.W. Hearn and S. Lawphongpanich (2004). An MPEC approach to second-best pricing. *Mathematical Programming, Vol. 101, N° 1, p 33-55.*

[41] G. Heilporn (2008). Network Pricing Problem : Complexity, Polyhedral study and Solutions Approaches. *Thèse de Doctorat, Université Libre de Bruxelles, Belgique et Université de Montréal, Canada.*

[42] S. v. Hoesel, F. v. d. Kraaij, C. Mannino, G. Oriolo, and M. Bouhtou (2003). Polynomial Cases of the Tarification Problem. *Technical report, Maastricht Economic Research School on Technology and Organisation, 10 p.*

[43] C.H. Holloway (1974). An extension of the Frank and Wolfe method of the feasible directions. *Mathematical Programming, Vol. 6, N° 1, p 14-27.*

[44] B. Jaumard, G. Savard and X. Xiong (1995). An Exact Algorithm for Convex Bilevel Programming. *Les Cahiers du GERAD G-95-33, Montréal.*

[45] J.D. Kant (2008). Cours Li 352 : Industrie Informatique et son Environnement Économique. *Cours de Licence Informatique, UPMC.*

[46] M. Labbé, P. Marcotte and G. Savard (1998). A Bilevel Model of Taxation and its Application to Optimal Highway Pricing. *Management Science, Part 1, Vol. 44, N° 12, p 1608-1622.*

[47] J. T. Lafrance (1986). The Structure of Constant Elasticity Demand Models. *American Journal of Agricultural Economics, Vol. 68, N° 3, p 543-552.*

[48] Q. S. Lei and C. Jian (2004). An Algorithm for Transit Assignment with Elastic Demand under Capacity Constraints. *Intelligent Control and Automation, Vol 6, N° 55, p 5245-5247.*

[49] P. Marcotte and S. Nguyen (1998). Hyperpath Formulations of Traffic Assignment Problems. *Equilibrium and Advanced Transportation Modeling, Kluwer Academic Publishers, Boston, MA, p 175-199.*

[50] P. Marcotte, G. Savard, D.L Zhu (2001). A Trust Region Algorithm for Nonlinear Bilevel Programming. *Operations Research Letters, Vol. 29, N° 4, p 171-179.*

[51] P. Marcotte, S. Nguyen and A. Schoeb (2004). A Strategic Flow Model of Traffic Assignment in Static Capacitated Networks. *Operations Research, Vol. 52, N° 2, p 191-212.*

[52] P. Marcotte, G. Savard, and A. Schoeb (2007). A Hybrid Approach to the Solution of a Pricing Model with Continuous Demand Segmentation. *Submitted to Operations Research.*

[53] P. Marcotte and M. Patriksson (2007). "Traffic Equilibrium" Transportation. *Handbooks in Operations Research and Management Science, C. Barnhardt and G. Laporte eds., North-Holland, p 623-714.*

[54] H. L. Moore. Partial Elasticity of Demand (1926). *The Quarterly Journal of Economics, Vol. 40, N° 3, p 393-401.*

[55] M. Patriksson (1994). The Traffic Assignment Problem - Models and Algorithms. *Topics in Transportation, VSP.*

[56] A. Poirier (2007). Gestion de la Congestion d'un Centre Ville par Péages. *Mémoire de Maîtrise, École Polytechnique de Montréal, Canada.*

[57] S. Roch, P. Marcotte, and G. Savard (2005). Design and Analysis of an Approximation Algorithm for Stackelberg Network Pricing. *Networks, Vol. 46, N° 1, p 57-67.*

[58] G. Savard (1989). Contribution à la Programmation à Deux Niveaux. *Thèse de Doctorat, École Polytechnique de Montréal, Canada.*

[59] G. Savard and J. Gauvin (1994). The Steepest Descent Direction for the Nonlinear Bilevel Programming Problem. *Operations Research Letters, Vol. 15, N° 5, p 265–272.*

[60] A. Schoeb (1999). Une Étude Théorique et Algorithmique d'un Modèle d'Affectation de Trafic avec Capacités Rigides. *Mémoire de Maîtrise, Université de Montréal.*

[61] H. D. Sherali and A. Alameddine (1992). A New Reformulation-Linearization Technique for Bilinear Programming Problems. *Journal of Global Optimization, Vol. 2, N° 3 , p 379–410.*

[62] L.N. Vicente, G. Savard and J.J. Júdice (1992). Descent Approaches for Quadratic Bilevel Programming. *Les Cahiers du GERAD G-92-36, Montréal.*

[63] L.C. Vicente and P.H. Calamai (1994). Bilevel and Multilevel Programming : a Bibliography Review. *Journal of Global Optimization, Vol. 5, N° 3, p 291–306.*

[64] L.N. Vicente, G. Savard and J.J. Júdice (1996). Discrete Linear Bilevel Programming Problem. *Journal of Optimization Theory and Application, Vol. 89. N° 3, p 597-614.*

[65] J.G. Wardrop (1952). Some Theoretical Aspects for Road Traffic Research *Proceeding of the Institute of Civil Engineers, Part 2, Vol. 1, N° 36, p 325-378.*

[66] U.P. Wen and Y.H. Yang (1990). Algorithms for Solving the Mixed Integer Two-Level Linear Programming Problem. *Computers and Operations Research, Vol. 17, Nº 2, p 133–142.*

[67] D.J. White and G. Anandalingam (1993). A Penalty Function Approach for Solving Bi-Level Linear Programs. *Journal of Global Optimization, Vol. 3, Nº 4 , p 397–419.*

[68] L.A. Wosley (1998). Integer Programming. *book.*

[69] H. Yang, H. Huang (1998). Principle of Marginal Cost Pricing. *Transportation Research, Vol. 32, Nº 1, p 45-55.*

[70] H. Yang, F. Xiao and H. Huang (2006). Competition and Equilibria of Private Toll Roads with Elastic Demand. *Transportation Research Board, Paper 06-0183.*

[71] J.Y. Yen (1971). Finding the K-Shortest Loopless Paths in Network. *Management Science, Vol. 17, Nº 11, p 712-716.*

[72] S. Zuyev, P. Desnogues and H. Rakotoarisoa (1996). Simulations of large Telecommunications Network based on Probabilistic Modeling. *Tech. rept. INRIA Sophia Antipolis.*