Titre: Title:	Tarification optimale : complexité et approximation
Auteur: Author:	Sébastien Roch
Date:	2003
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Roch, S. (2003). Tarification optimale : complexité et approximation [Mémoire de maîtrise, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/7151/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/7151/
Directeurs de recherche: Advisors:	Gilles Savard, & Patrice Marcotte
Programme: Program:	Non spécifié

UNIVERSITÉ DE MONTRÉAL

TARIFICATION OPTIMALE : COMPLEXITÉ ET APPROXIMATION

SÉBASTIEN ROCH

DÉPARTEMENT DE MATHÉMATIQUES ET DE GÉNIE INDUSTRIEL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION

DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES

(MATHÉMATIQUES APPLIQUÉES)

MAI 2003

Your file Votre référence
ISBN: 0-612-86431-6
Our file Notre référence
ISBN: 0-612-86431-6

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé:

TARIFICATION OPTIMALE : COMPLEXITÉ ET APPROXIMATION

présenté par: ROCH Sébastien

en vue de l'obtention du diplôme de: Maîtrise ès sciences appliquées

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

M. HERTZ Alain, Doct. ès Sc., président

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

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

M. AUDET Charles, Ph.D., membre

На Инка

REMERCIEMENTS

Je tiens à surtout remercier mes superviseurs, Gilles Savard et Patrice Marcotte, pour leur aide, leur soutien et leur disponibilité ainsi que les membres du jury, Alain Hertz et Charles Audet, pour leurs commentaires et leurs suggestions. Merci aussi à mes collègues de bureau Léandre Ratsirahonana, Éric Rancourt et Sylvain Crouzet. Finalement, je voudrais souligner le support financier du CRSNG.

RÉSUMÉ

Ce mémoire présente une approche combinatoire à la résolution d'un problème de tarification optimale dont les applications s'étendent des péages routiers à l'industrie aérienne. Plus précisément, on considère le problème de la maximisation du revenu engendré par le prélèvement de péages sur les arcs d'un réseau de transport, sous la contrainte que les usagers sont assignés aux chemins les plus courts pour les niveaux de péages choisis. D'abord, on montre que le problème est fortement *NP*-difficile, répondant ainsi à une question ouverte depuis quelques années. Ensuite, on présente un algorithme en temps polynomial qui fournit une solution approchée avec une précision – dans le pire des cas – de $\frac{1}{2}\log(m_T) + 1$, où m_T est le nombre d'arcs à péages. Finalement, on montre que cette approximation est optimale par rapport à une relaxation naturelle en construisant une famille d'instances pour lesquelles le saut de relaxation est atteint.

ABSTRACT

We consider the problem of maximizing the revenue raised from tolls set on the arcs of a transportation network, under the constraint that users are assigned to toll-compatible shortest paths. We first prove that this problem is strongly *NP*-hard. We then provide a polynomial-time algorithm with a worst-case precision guarantee of $\frac{1}{2}\log m_T + 1$, where m_T denotes the number of toll arcs. Finally we show that the approximation is tight with respect to a natural relaxation by constructing a family of instances for which the relaxation gap is reached.

TABLE DES MATIÈRES

LISTE DES FIGURES

LISTE DES NOTATIONS ET DES SYMBOLES

P	classe des problèmes de décision dont la nature positive ou négative peut être déterminée en temps polynomial
NP	classe des problèmes de décision dont la nature positive (le cas échéant) peut être vérifiée en temps polynomial
SAT	satisfaisabilité d'une expression booléenne
$3\text{-}SAT$	SAT en forme normale conjonctive avec trois littéraux par clause
OPT	valeur optimale d'un problème d'optimisation
APP	valeur retournée par un algorithme d'approximation
I	instance d'un problème d'optimisation
$\mathbf{N}$	ensemble des nombres naturels (incluant 0)
$\mathbf{R}$	ensemble des nombres réels
G	multigraphe
V	sommets de G
A	arcs de G
A_T, A_U	arcs taxables et non taxables respectivement de G
m_T, m_U	nombre d'arcs taxables et non taxables respectivement dans G
c, d	coûts fixes sur A_T et A_U respectivement
T	vecteur de taxe sur A_T (appelé aussi péage)

s, t	origine et destination respectivement
$\mathcal{N}_T$	réseau taxé dont les péages sont donnés par le vecteur T
$\mathcal{SP}[\mathcal{N}_T]$	ensemble des plus courts chemins sur $\mathcal{N}_T$
P	un chemin de s à t (ne pas confondre avec la notation plus haut; cela devrait être clair d'après le contexte)
$\mathcal{L}(P)$	longueur du chemin P avec $T = 0$
$\mathcal{L}_\infty$	longueur du plus court chemin non taxable
$B(P)$	borne supérieure sur le profit associé à P
LP	borne supérieure sur le profit total
$\log(x)$	logarithme en base 2 de x
$\mathcal{K}$	ensemble de paires origine-destination
h^k	demande sur la paire k
x^k, y^k	flots sur les arcs taxables et non taxables respectivement de la paire k
A_1, A_2	matrices de conservation de flot
b^k	vecteur de droite dans le système de conservation de flot
M	constante arbitrairement grande
λ^k	vecteur dual
$[n]$	ensemble $\{1, \ldots n\}$
MaxToll	problème de tarification optimale sur un réseau à une paire origine-destination

LISTE DES ANNEXES

INTRODUCTION

Ce mémoire est consacré à un problème de tarification optimale sur un réseau, dont les applications sont nombreuses en transport, ainsi que dans l'industrie aérienne et celle des télécommunications. Le modèle étudié ici a été introduit par Labbé et al. [19; 20] et plusieurs travaux y ont été consacrés depuis sa publication. Notre approche se distingue par une perspective résolument théorique et une démarche empruntée à l'informatique. Le fil conducteur du mémoire est une volonté d'élucider la difficulté du problème d'un point de vue algorithmique. Il s'agit d'une tâche aux ramifications multiples et, par conséquent, nous nous sommes limités à deux axes principaux : la complexité algorithmique du problème dans sa forme la plus simple, et son approximabilité.

La complexité du problème de tarification a été abordée, mais laissée partiellement ouverte, par Labbé et al. Ces auteurs ont montré qu'une version peu naturelle du problème est *NP*-difficile. Dans ce mémoire, nous complétons leur travail en confirmant la nature *NP*-difficile du modèle canonique. Puis nous concevons un algorithme d'approximation, c'est-à-dire une heuristique dont on garantit un temps de calcul polynomial et une précision bornée. Le facteur d'approximation obtenu, optimal par rapport à la relaxation employée, dépend de la taille de l'instance, ce qui suggère que le problème est difficile à résoudre approximativement.

Les résultats de ce mémoire sont contenus dans un article soumis pour publication. Cet article est placé en annexe alors que le reste du mémoire lui sert d'introduction. Ainsi, le chapitre 1 présente une revue critique de littérature plus élaborée que celle retrouvée dans l'article. Le chapitre 2 décrit les principales étapes du travail ayant mené aux résultats. Une courte introduction à certaines notions nécessaires à la compréhension de la démarche utilisée est contenu dans le chapitre 3. Finalement, le chapitre 4 comprend une synthèse des résultats obtenus.

CHAPITRE 1

REVUE DE LITTÉRATURE

The purpose of models is not to fit
the data but to sharpen the questions.
S. KARLIN

La littérature qui concerne directement le modèle de tarification étudié dans ce mémoire est relativement restreinte puisque l'histoire de cette modélisation est assez courte. Des travaux pertinents en programmation à deux niveaux, en économie et en informatique théorique sont aussi mentionnés.

1.1 Programmation à deux niveaux

La programmation mathématique à deux niveaux concerne l'étude de problèmes d'optimisation de forme générale

$$\min_{(u,v)} \; F(u, v)$$
$$\begin{cases} u \in U \subseteq \mathbf{R}^{n_u} \\ v \in \arg\min_{v \in V(u) \subseteq \mathbf{R}^{n_v}} f(u, v), \end{cases}$$

où n_u et n_v sont des entiers positifs et, pour chaque $u \in \mathbf{R}^{n_u}$, $V(u)$ est un sous-ensemble de $\mathbf{R}^{n_v}$. Ces programmes sont un outil de modélisation pour des situations où un joueur (le *meneur*) intègre dans son plan d'optimisation la réaction d'un second joueur (le *suiveur*) à ses actions. Ils sont intimement liés aux jeux de Stackelberg statiques [27] ainsi qu'aux programmes mathématiques avec contraintes d'équilibre [21]. La programmation à deux niveaux permet de modéliser une grande variété de situations en recherche opérationnelle, en économie et en ingénierie. Elle a été introduite par Bracken et al. [4]. Un traitement élémentaire se trouve dans la *Encyclopedia of Optimization* [11] alors qu'une revue de littérature malheureusement peu récente est disponible dans Vicente et al. [30].

1.2 Complexité algorithmique

Les programmes à deux niveaux sont généralement non convexes et non différentiables, donc par nature "intractables". En particulier, Jeroslow [16] a montré que la programmation biniveau linéaire est *NP*-difficile et Hansen et al. [13] ont montré qu'elle est en fait fortement *NP*-difficile. Ce résultat a été raffiné par Vicente et al. [31], qui ont montré que l'obtention d'un simple certificat d'optimalité locale est fortement *NP*-difficile. En fait, Audet et al. [1] ont mis en évidence une relation intime entre la programmation à deux niveaux et la programmation en nombres entiers. Cette "intractabilité" a incité les spécialistes à développer des heuristiques

adaptées à chaque problème. La première analyse d'une telle heuristique a été faite par Marcotte [22] pour un problème de réseau. La référence standard sur la *NP*-complétude est Garey et al. [12]. Une introduction détaillée aux heuristiques avec analyse dans le pire des cas , ou algorithmes d'approximation, est donnée dans Vazirani [28].

1.3 Tarification optimale

Dans ce mémoire, on étudie plus précisément un algorithme d'approximation pour un problème de tarification optimale sur un réseau, formulé et analysé par Labbé et al. [19]. Il s'agit de l'optimisation de péages sur un réseau routier dans le but de maximiser le profit du gestionnaire. Labbé et al. ont montré qu'une version générale du problème est *NP*-difficile. Ils ont toutefois laissé ouverte la question de déterminer si la version la plus canonique du problème, c'est-à-dire sans borne inférieure sur les péages, est *NP*-difficile. Ces auteurs ont aussi donné une formulation à deux niveaux du problème de tarification puis ont montré comment transformer celle-ci en un programme en nombres entiers, ce qui leur a permis de résoudre le problème par des techniques bien connues d'énumération. Plusieurs auteurs [5; 6; 3] ont proposé d'autres formulations et développé des heuristiques (sans analyse dans le pire des cas). Un lien remarquable avec le problème du voyageur de commerce a également été mis en évidence par Marcotte et al. [23].

1.4 Autres travaux pertinents

Un problème différent en tarification optimale consiste à maximiser le bien-être social plutôt que le profit du gestionnaire. Ce point de vue est pertinent lorsque des effets de congestion sont présents dans le réseau. Si les usagers choisissent eux-mêmes les chemins qu'ils empruntent, ces choix "égoistes" ne minimisent pas nécessairement le délai total de l'ensemble des usagers. Un gestionnaire peut alors ajouter des taxes sur les arcs pour influencer les décisions prises par les utilisateurs. Dans ce cas, il est bien connu [24] que la tarification au coût marginal maximise l'utilité globale lorsque les usagers sont homogènes (c'est-à-dire qu'ils ont tous la même perception des taxes). Le cas plus général a été traité par Cole et al. [8] et Yang et al. [32] qui ont démontré l'existence de schémas de tarification optimaux même si les usagers sont hétérogènes. Toutefois, lorsque certaines ressources ne sont pas sous le contrôle du gestionnaire (par exemple certains arcs sont non-taxables), l'optimum social peut être hors d'atteinte (voir Verhoef, Nijkamp et Rietveld [29], Hearn et Ramana [14], et Larsson et Patriksson [18] pour des exemples en transport). Dans ce cas, le problème de déterminer les taxes qui maximisent le bien-être social est similaire, dans sa structure, à celui étudié dans sa mémoire.

Plus récemment, des analyses de problèmes de Stackelberg ont été appliquées à l'ordonnancement et au design de réseau par Roughgarden [25; 26], au routage par

Korilis, Lazar et Orda [17] ainsi qu'à la tarification des réseaux informatiques par Cocchi et al. [7].

CHAPITRE 2

PRÉSENTATION DE L'ARTICLE

It can be of no practical use to know that π is irrational;
but if we can know, it surely would be intolerable not to know.
E.C. TITCHMARSH

Dans ce chapitre, on développe les principales étapes du travail. On présente aussi brièvement le contenu de l'article.

2.1 Organisation du travail

Deux objectifs principaux sous-tendent le travail présenté en annexe :

1. déterminer la complexité algorithmique du problème de tarification optimale;
2. caractériser l'"approximabilité" du problème.

Ces objectifs sont liés et deux cas sont possibles : soit la complexité du problème est polynomiale, ce qui peut être établi en exhibant un algorithme polynomial *exact*, répondant ainsi aux questions soulevées par les deux objectifs; soit le problème est

NP-complet, auquel cas un algorithme polynomial ne peut être qu'approximatif (sauf si $P = NP$) et le deuxième objectif consiste alors à obtenir le meilleur facteur d'approximation possible en temps polynomial.

La démarche théorique est similaire dans les deux éventualités. D'abord, une étape simple mais fondamentale consiste à formuler le problème d'une façon précise et appropriée. Ensuite, une étude en profondeur de la structure combinatoire du problème permet de faire apparaître quelques idées algorithmiques.

C'est cette analyse qui nous a permis de construire une réduction *NP*-complète et de répondre ainsi à la question soulevée par le premier objectif. Par la suite, l'essentiel du travail a consisté à concevoir un algorithme polynomial d'approximation et à en faire l'analyse détaillée. En particulier, en utilisant une relaxation naturelle du problème, un facteur d'approximation garanti par l'algorithme a été établi. En fait, la construction d'instances spéciales nous a permis de démontrer l'impossibilité d'améliorer ce facteur de précision par rapport à la relaxation, atteignant ainsi partiellement le deuxième objectif. Du même coup, nous avons déterminer la précision de la relaxation, un résultat qui s'inscrit aussi naturellement dans l'atteinte du second objectif.

2.2 Contenu de l'article

Les deux objectifs fixés au début du projet ont été atteints. Tous les résultats pertinents sont décrits dans l'article en annexe. Cet article intitulé *Design and analysis of an approximation algorithm for Stackelberg network pricing* a été soumis à la revue *Networks*. Les auteurs de l'article sont Patrice Marcotte, Gilles Savard et l'auteur de ce mémoire. L'atteinte du premier objectif, par l'entremise d'une réduction *NP*-complète, est détaillée à la section 2.2 de l'article. L'étude de la structure combinatoire du problème sur laquelle repose l'algorithme est présentée à la section 3.1. Puis, l'algorithme est décrit et analysé aux sections 3.2 à 4.3.

2.3 Sections additionnelles

Pour faciliter la compréhension de l'article, le chapitre 3 détaille certaines notions théoriques essentielles. En particulier, on rappelle brièvement la propriété de *NP*-complétude. On développe aussi la notion d'algorithme d'approximation, qui fait l'objet d'une vaste littérature en informatique théorique mais qui est peut-être un peu moins connue en recherche opérationnelle.

CHAPITRE 3

COMPLEXITÉ ET APPROXIMATION

Technical skill is mastery of complexity
while creativity is mastery of simplicity.
E.C. ZEEMAN

Deux notions sont importantes pour comprendre l'article inséré en annexe. À la section 3.1, on rappelle la théorie des problèmes *NP*-complets. Puis, la notion d'algorithme d'approximation est développée à la section 3.2

3.1 Complexité algorithmique

On s'intéresse d'abord à des *problèmes de décision*, qui consistent à déterminer si un certain objet combinatoire possède une propriété donnée[1]. Un exemple pourrait consister à déterminer si un graphe est connexe, ou 3-coloriable. La réponse est simplement *oui* (instances positives) ou *non* (instances négatives).

Parmi ces problèmes, les classes P et NP sont les plus importantes en pratique. Un problème de décision est dans la classe P s'il existe un algorithme qui détermine

[1]En simplifiant un peu. On évitera d'être formel dans cette section.

la nature positive ou négative de l'instance en un temps de calcul borné par un polynôme de la taille de l'instance. Dans le cas de la connexité d'un graphe par exemple, la taille de l'instance est le nombre d'arcs m. Évidemment, plus la taille de l'instance est grande (plus m est grand), plus le temps de calcul est potentiellement long. Il est souhaitable que ce temps de calcul ne croisse pas trop rapidement avec l'augmentation de la taille de l'instance. Une croissance polynomiale est considérée satisfaisante.

D'autre part, un problème est dans la classe NP si pour chaque instance positive, il existe une preuve vérifiable en temps polynomial que l'instance possède la propriété voulue[2]. Par exemple, chaque graphe 3-coloriable possède évidemment un 3-coloriage. Étant donné ce 3-coloriage, il est facile de vérifier que l'instance est positive.

Alors qu'il est clair que $P \subseteq NP$ (dans P, l'instance elle-même est une preuve), le statut de l'inclusion inverse n'est pas établi. En fait, pour un très grand nombre de problèmes dans NP, par exemple le 3-coloriage de graphe, on ne connaît pas d'algorithme polynomial, malgré d'innombrables tentatives échelonnées sur plusieurs décennies. Ainsi, la conjecture généralement admise est que $P \neq NP$.

Une propriété fondamentale de la classe NP est qu'elle possède un sous-ensemble

[2]Et aucune preuve n'existe lorsque l'instance est négative.

de problèmes *complets*. Un problème de décision dans *NP* est *NP-complet* s'il est *NP-difficile*, c'est-à-dire qu'un algorithme polynomial pour celui-ci se convertit en un algorithme polynomial pour tous les problèmes dans *NP*. Si l'on croit la conjecture précédente, montrer qu'un problème est *NP*-difficile constitue une preuve d'intractabilité. Le premier problème *NP*-complet connu a été celui de la satisfaisabilité de formules booléennes (ou *SAT*). Ainsi, le point de départ de la théorie de la *NP*-complétude est le théorème de Cook [9].

Théorème 1 *Le problème* SAT *est NP-complet.*

Il est à noter que la notion de problème *NP*-difficile ne s'applique pas seulement aux problèmes de décision. En particulier, un *problème d'optimisation* peut être *NP*-difficile (et donc intractable). Pour montrer qu'un nouveau problème (de décision ou d'optimisation) est *NP*-difficile, il suffit de réduire un problème *NP*-complet connu à celui-ci, c'est-à-dire montrer que le problème *NP*-complet est un cas particulier du nouveau problème. De plus, le passage de la solution du nouveau problème à celle du problème *NP*-complet doit se faire en temps polynomial. Un exemple de réduction est donné dans l'article. On y utilise le problème *NP*-complet *3-SAT*, un cas particulier de *SAT*. Le problème *3-SAT* consiste à déterminer si une formule booléenne en forme normale conjonctive avec 3 littéraux par clause est satisfaisable. Illustrons par un exemple. Soient x_1, x_2, x_3 et x_4 des variables booléennes,

qui prennent les valeurs VRAI ou FAUX. Soit aussi la formule booléenne

$$F(x_1, x_2, x_3, x_4) = (x_1 \vee x_2 \vee \overline{x}_3) \wedge (\overline{x}_2 \vee x_3 \vee \overline{x}_4) \wedge (\overline{x}_1 \vee x_3 \vee x_4)$$

où $\overline{x}_i$ est la négation de x_i. Chaque parenthèse est appelée clause. Chaque clause est une disjonction de 3 littéraux (variable ou sa négation) et les clauses sont reliées par des conjonctions. C'est la forme normale conjonctive avec 3 littéraux par clause (ou *3-CNF*). Ici, F est satisfaisable avec $x_1 = x_2 = x_3 = x_4 =$ VRAI.

Finalement, un problème est *fortement NP*-difficile si la réduction du problème *NP*-complet connu à celui-ci n'emploie que des constantes numériques polynomiales en la taille du problème. Ce résultat est plus fort puisqu'il implique qu'il n'existe pas d'algorithme *pseudopolynomial* pour le problème, c'est-à-dire qu'il n'existe pas d'algorithme dont le temps de calcul est un polynôme de la taille et des constantes numériques de l'instance (alors que la taille de ces constantes est proportionnelle à leur logarithme).

3.2 Algorithmes d'approximation

Sauf si $P = NP$, un problème d'optimisation *NP*-difficile n'admet pas d'algorithme polynomial – autrement dit, efficace. Certains se contentent alors d'algorithmes exponentiels. Typiquement, le problème est mis sous la forme d'un programme

linéaire avec variables binaires puis une méthode d'énumération classique est employée. Il est aussi possible de concevoir des heuristiques, qui en général n'offrent aucune garantie en terme de complexité et d'approximation. Par ailleurs dans les années 80 et 90, les théoriciens de l'informatique ont développé un intérêt marqué pour une autre solution, celle des algorithmes d'approximation. Comme les heuristiques populaires en recherche opérationnelle, ces algorithmes fournissent une solution réalisable approchée, mais ils garantissent en plus que le calcul se termine en temps polynomial et que le rapport entre la valeur optimale et la valeur de la solution obtenue est borné.

Définition 1 *Un α-algorithme d'approximation pour un problème d'optimisation*[3] *est un algorithme polynomial qui garantit ($\alpha \geq 1$)*

$$APP[I] \geq \frac{1}{\alpha} OPT[I]$$

où $APP[I]$ et $OPT[I]$ sont les valeurs approchée (retournée par l'algorithme) et optimale respectivement de l'instance I. En général, α peut dépendre de la taille de I.

La motivation première des théoriciens de l'informatique en cette matière est de poursuivre la classification des problèmes algorithmiques entamée par Cook, Karp

[3] *Dans la suite, tous les problèmes d'optimisation sont des problèmes de maximisation.*

et Levin dans les années 70. Sachant qu'un problème donné est NP-complet, une question naturelle est la suivante : puisque, selon toute vraisemblance, il n'existe pas d'algorithme efficace pour résoudre ce problème, quelle précision peut atteindre un algorithme polynomial ?

Tous les problèmes NP-complets ne sont pas égaux du point de vue de l'approximation. Certains possèdent des algorithmes efficaces avec une précision aussi bonne que l'on souhaite. D'autres, beaucoup plus difficiles, n'admettent que des facteurs qui dépendent logarithmiquement ou linéairement de la taille du problème. En fait pour de nombreux problèmes, il est possible de montrer qu'il est NP-difficile de faire mieux qu'un facteur de précision donné.

Un ingrédient important dans l'analyse d'un algorithme d'approximation est l'existence d'une borne supérieure (disons LP) sur la valeur optimale

$$LP[I] \geq OPT[I], \quad \forall I.$$

Dans le cas de la programmation en nombres entiers par exemple, une relaxation linéaire est généralement employée, c'est-à-dire que les contraintes $\{0, 1\}$ sont relaxées dans la formulation linéaire avec variables binaires.

Cette borne supérieure sert d'estimation de la valeur optimale. Ainsi dans l'analyse d'un algorithme, on montre en général

$$APP[I] \geq \frac{1}{\alpha} LP[I] \geq \frac{1}{\alpha} OPT[I].$$

Pour une relaxation donnée, il existe une borne inférieure sur le facteur d'approximation qui peut être obtenue en utilisant cette relaxation comme estimation de la valeur optimale. Cette quantité est appelée saut de relaxation.

Définition 2 *Le saut de relaxation est donné par*

$$\gamma = \max_I \left\{ \frac{LP[I]}{OPT[I]} \right\}.$$

En effet, puisque l'algorithme d'approximation fournit une solution réalisable au problème, on a

$$\frac{LP[I]}{OPT[I]} \leq \frac{LP[I]}{APP[I]}, \quad \forall I$$

donc

$$\gamma = \max_I \left\{ \frac{LP[I]}{OPT[I]} \right\} \leq \max_I \left\{ \frac{LP[I]}{APP[I]} \right\}.$$

Le terme de droite est le meilleur facteur d'approximation qui peut être obtenu en utilisant LP comme estimation de OPT. Si le facteur d'approximation d'un

algorithme est γ, la précision de cet algorithme est donc optimale par rapport à la relaxation utilisée. Cependant, ceci ne constitue pas une preuve qu'il est impossible de faire mieux qu'un facteur γ : une relaxation plus serrée pourrait mener à un algorithme plus précis.

CHAPITRE 4

DISCUSSION DES RÉSULTATS

A computation is a temptation that
should be resisted as long as possible.
J.P. BOYD, INSPIRÉ PAR T.S. ELIOT

Les principaux résultats de l'article sont présentés brièvement dans ce chapitre. À la section 4.1, on rappelle l'approche de type biniveau au problème de tarification. Puis, notre approche combinatoire est décrite aux section suivantes.

4.1 Approche classique

Le point de départ du projet est un article de Labbé et al.[19; 20] où est introduit un modèle de tarification optimale basé sur la programmation mathématique à deux niveaux. Cette classe de programmes mathématiques est de la forme générale

$$\min_{(u,v)} \; F(u, v)$$
$$\begin{cases} u \in U \\ v \in \arg\min_{v \in V(u)} f(u, v), \end{cases}$$

où $u \in U \subseteq \mathbf{R}^{n_u}$ est la variable de premier niveau et $v \in V(u) \subseteq \mathbf{R}^{n_v}$ est la variable de deuxième niveau. Labbé et al. ont spécialisé ce programme à une formulation appropriée pour un problème de taxation : les fonctions objectifs y sont bilinéaires et les contraintes, linéaires. On suppose une structure sous-jacente de réseau comportant un ensemble $\mathcal{K}$ de paires origine-destination. Les usagers de la paire $k \in \mathcal{K}$ se déplacent de s_k à t_k. Chaque arc est affecté d'un coût fixe et certains arcs, dits taxables, voient leurs coûts augmentés de péages[1]. On suppose que les usagers empruntent un chemin qui minimise leur coût total (coût fixe plus taxe) alors que le gestionnaire du réseau, quant à lui, maximise son profit en choisissant des taxes appropriées. Le programme biniveau ainsi obtenu est

$$\max_{T,\, x^k,\, y^k} \; T' \sum_{k \in \mathcal{K}} h^k x^k$$
$$\begin{cases} \min\limits_{x^k,\, y^k} \; (c+T)' \sum\limits_{k \in \mathcal{K}} x^k + d' \sum\limits_{k \in \mathcal{K}} y^k \\ \forall k \in \mathcal{K} \begin{cases} A_1 x^k + A_2 y^k = b^k \\ x^k,\, y^k \geq 0, \end{cases} \end{cases}$$

où $T \in \mathbf{R}^{m_T}$ est le vecteur des taxes, $x^k \in \mathbf{R}^{m_T}$ est le flot de la paire k sur les arcs taxables et $y^k \in \mathbf{R}^{m_U}$ est le flot de la paire k sur les arcs non taxables. Les vecteurs $c \in \mathbf{R}^{m_T}$ et $d \in \mathbf{R}^{m_U}$ sont les coûts fixes sur les arcs taxables et non taxables respectivement et les constantes h^k donnent la demande sur chaque paire. Les

[1]Dans la suite, on utilise les expressions péage et taxe sans distinction.

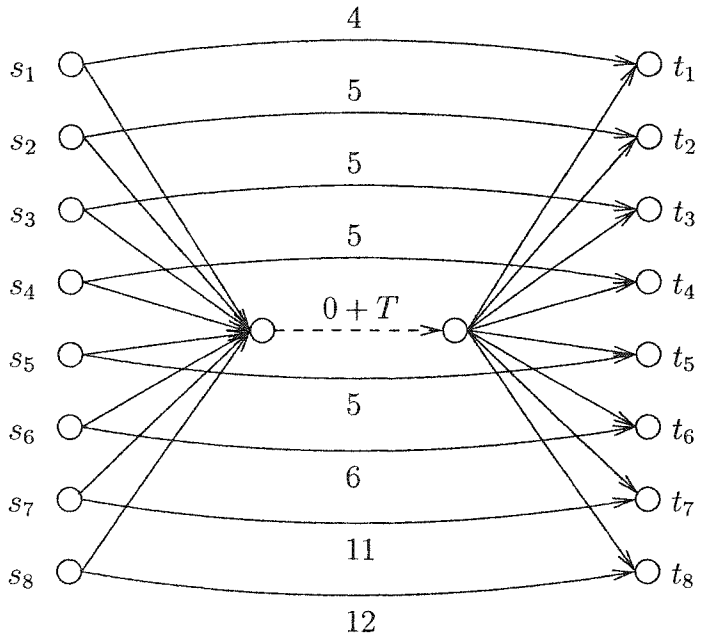

Figure 4.1 Une instance avec 8 paires origine-destination.

contraintes linéaires au deuxième niveau imposent la conservation du flot. On aura compris que le programme ci-dessus emploie une notation simplifié qui signifie que les variables (x^k, y^k) sont solutions optimales du problème d'optimisation détaillé dans l'espace réservé aux contraintes.

La figure 4.1 présente un exemple. Il s'agit d'une instance avec 8 paires origine-destination (s_k, t_k), $k = 1, \ldots, 8$, et un arc taxable (en pointillés). Les coûts fixes apparaissent près de chaque arc (le coût est nul si rien n'est indiqué). L'affectation optimale est $T = 5$ avec un revenu de 35.

Labbé et al. ont aussi montré comment calculer les solutions optimales de ce

problème à l'aide d'une formulation à un seul niveau en variables mixtes $\{0,\ 1\}$

$$\max_{T,\, x^k,\, y^k,\, \lambda^k,\, t^k} \sum_{k \in \mathcal{K}} h^k \sum_{i=1}^{m_x} t_i^k$$

$$\forall k \in \mathcal{K} \begin{cases} A_1 x^k + A_2 y^k = b^k,\ A_1' \lambda^k \leq c + T,\ A_2' \lambda^k \leq d \\ c' x^k + d' y^k + \sum_{i=1}^{m_x} t_i^k = b^{k\prime} \lambda^k \\ -M x_i^k \leq t_i^k \leq M x_i^k,\ -M(1 - x_i^k) \leq t_i^k - T_i \leq M(1 - x_i^k) \quad \forall\ i \in [m_x] \\ y^k \geq 0,\ x^k \in \{0,\ 1\}^{m_x}, \end{cases}$$

où M est une constante arbitrairement grande et $[n]$ dénote l'ensemble $\{1, \ldots, n\}$. Cette formulation s'obtient en remplaçant le problème de second niveau par ses conditions d'optimalité, qui sont ensuite linéarisées par l'introduction des variables t^k.

4.2 Approche combinatoire et algorithmique

Notre objectif dans ce mémoire est de présenter une approche purement combinatoire et algorithmique. Aussi, la formulation suivante du problème de tarification optimale paraît plus pertinente pour notre point de vue. Il est à noter qu'il s'agit du même problème que celui de la section précédente, avec toutefois une simplification supplémentaire : les paires origine-destination sont limitées à une seule. De plus, les taxes sont supposées non négatives (mais l'article traite également le cas

général).

Soit $G = (V, A)$ un multigraphe orienté avec deux sommets particuliers : l'ori-gine $s \in V$ et la destination $t \in V$. L'ensemble des arcs A est divisé en deux sous-ensembles A_T et A_U d'arcs *taxables* et *non taxables* dont les cardinalités respectives sont m_T et m_U. À chaque arc est assigné un *coût fixe* $c : A_T \rightarrow \mathbf{N}^{m_T}$ ou $d : A_U \rightarrow \mathbf{N}^{m_U}$ (dans la suite, $\mathbf{N} = \{0, 1, \ldots\}$). Lorsque des taxes $T : A_T \rightarrow \mathbf{N}^{m_T}$ sont ajoutées aux coûts fixes sur A_T, un *réseau taxé* $\mathcal{N}_T = (G, c+T, d, s, t)$ est obtenu. L'ensemble des chemins les plus courts entre s et t est noté $\mathcal{SP}[\mathcal{N}_T]$. Le problème de tarification optimale, appelé MAXTOLL dans l'article, se formule de la façon suivante (voir aussi la figure 4.2) :

$$\max_{\substack{T \geq 0 \\ P \in \mathcal{SP}[\mathcal{N}_T]}} \sum_{e \in A_T \cap P} T(e).$$

Dans ce cadre, le gestionnaire du réseau doit trouver le bon équilibre entre des péages faibles, qui génèrent un revenu faible, et des péages élevés, qui génèrent aussi un revenu faible puisqu'ils incitent les usagers à emprunter peu d'arcs taxables. Notons que s'il existe plusieurs chemins de coût minimal, l'usager emprunte celui qui rapporte le plus grand profit au gestionnaire. On peut montrer facilement que cette hypothèse n'affecte pas la solution du problème (voir l'article).

Une instance de MAXTOLL est illustrée à la figure 4.3. Ce réseau contient deux

INSTANCE:	– un graphe orienté $G = (V,\, A_T \cup A_U)$ – des vecteurs de coûts fixes $c : A_T \rightarrow \mathbf{N}^{m_T}$ et $d : A_U \rightarrow \mathbf{N}^{m_U}$ – deux sommets $s, t \in V$ tels qu'il existe un chemin simple de s à t dans $(V,\, A_U)$
SOLUTION:	– un vecteur de taxes non négatives T sur A_T – un chemin P de s à t de coût minimal par rapport à la structure de coût $(c + T,\, d)$
MESURE:	– maximiser $\sum_{e \in A_T \cap P} T(e)$

Figure 4.2 Définition de MAXTOLL.

arcs taxables. Le chemin optimal est $(s,\, v_2,\, v_5,\, t)$ avec un revenu de 5 quand $T_1 = 5$ et $T_2 = +\infty$.

4.3 Borne sur le revenu

Une borne naturelle sur le revenu a été obtenue par Labbé et al. par des arguments de dualité linéaire. Cette borne découle aussi du théorème 4 à la section 3 de l'article. Soient $\mathcal{L}_\infty$ la longueur du plus court chemin non taxable[2] entre s et t, et $\mathcal{L}(P)$ la longueur du chemin P lorsque $T = 0$.

Théorème 2 *Soit P un chemin de s à t. Le revenu le plus élevé qui peut être*

[2]C'est-à-dire qui ne contient aucun arc taxable.

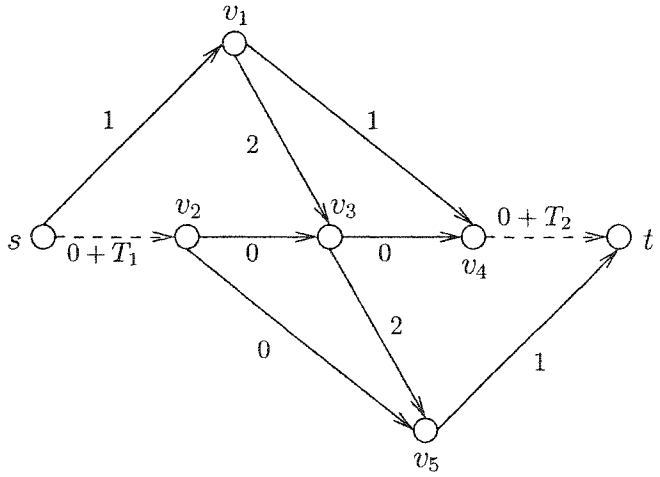

Figure 4.3 Un réseau avec deux arcs taxables.

récolté lorsque l'usager emprunte le chemin P est borné par

$$B(P) \equiv \mathcal{L}_\infty - \mathcal{L}(P). \tag{4.1}$$

Puisque $\mathcal{L}_\infty$ ne dépend pas de P, il s'ensuit que la borne la plus élevée correspond au chemin avec la plus petite valeur de $\mathcal{L}(P)$ c'est-à-dire que P est un plus court chemin lorsque les taxes sont nulles. On note $\mathcal{L}_0$ le coût d'un tel chemin et

$$LP = \mathcal{L}_\infty - \mathcal{L}_0, \tag{4.2}$$

la valeur de la plus grande borne. Cette borne est simplement la différence entre les coûts des plus courts chemins correspondant à des taxes infinies et nulles

respectivement. Il est à noter que la borne peut toujours être atteinte lorsque le réseau ne contient qu'un seul arc taxable.

4.4 *NP*-complétude de MAXTOLL

Labbé et al. ont montré qu'une version un peu plus générale de MAXTOLL, soit avec des coûts fixes négatifs et des bornes inférieures sur les taxes, est *NP*-difficile. Pour ce faire, ces auteurs ont utilisé une réduction à partir du problème de l'existence d'un chemin hamiltonien dans un graphe. La question de la *NP*-complétude de MAXTOLL dans sa version la plus simple est demeurée ouverte jusqu'à récemment. En effet, il s'agit du premier résultat important de l'article en annexe. La réduction employée est fort différente de celle de Labbé et al. Le problème de décision *3-SAT* y est utilisé.

Théorème 3 *Le problème* MAXTOLL *est fortement NP-difficile.*

4.5 Algorithme d'approximation

Le théorème précédent implique donc que tout algorithme efficace ne peut résoudre MAXTOLL qu'approximativement, sauf si $P = NP$. Trouver un tel algorithme d'approximation n'est pas une tâche facile : un algorithme qui semble bien fonc-

tionner en pratique ne suffit pas. Pour garantir un facteur d'approximation dans le pire des cas, l'algorithme doit non seulement donner de bons résultats sur *toutes* les instances mais doit aussi se *laisser analyser* facilement.

L'algorithme EXPLOREDESCENDANTS – deuxième résultat de l'article – répond à ces deux critères. Il est basé sur un point de vue de type *chemin* : plutôt que d'explorer l'espace des taxes (c'est de cette façon qu'a été décrit le problème à la section 4.2), on recherche un *bon* chemin que l'on oblige l'usager à emprunter en choisissant des taxes qui en font le chemin le plus court. Par exemple, le chemin le plus court lorsque $T = 0$ semble en général être un bon chemin potentiel puisque c'est sur ce chemin qu'il y a la plus grande marge pour ajouter des taxes entre le coût initial et la borne supérieure. C'est d'ailleurs avec ce chemin que commence EXPLOREDESCENDANTS.

Il existe plusieurs algorithmes efficaces pour calculer le profit maximal sur un chemin donné. Labbé et al. en ont suggéré un; l'article en annexe en fournit un autre qui donne des informations supplémentaires utiles. Cet algorithme, MAXREV dans l'article, considère tous les arcs taxables du chemin dans l'ordre de passage et choisit pour chacun la taxe la plus élevée compatible avec le statut de plus court chemin du chemin considéré.

Malheureusement, il est très facile de construire un exemple pour lequel le plus

court chemin à $T = 0$ mène à un profit décevant. Il n'est donc pas possible de se contenter de ce chemin. Typiquement, certaines taxes sont contraintes à être faibles par des sous-chemins non-taxables courts (voir l'article pour plus de détails à ce sujet). L'idée de l'algorithme EXPLOREDESCENDANTS est d'éviter ces arcs improductifs. Deux nouveaux chemins sont construits qui comportent chacun un sous-ensemble des arcs taxables du chemin précédent[3] puis le nouveau profit est calculé. L'analyse consiste à montrer par induction qu'un profit "satisfaisant" est éventuellement obtenu en procédant de cette façon.

Théorème 4 *L'algorithme* EXPLOREDESCENDANTS *est un* $(\frac{1}{2}\log(m_T) + 1)$*-algorithme d'approximation.*

4.6 Optimalité de l'approximation

Le facteur d'approximation obtenu à la section précédente peut paraître insatisfaisant à première vue. En effet, si le nombre d'arcs taxables m_T n'est que d'une soixantaine, l'algorithme garantit seulement 25% de la solution optimale – un résultat qui ferait sourire un ingénieur. Il est important de mettre cela en perspective. Garantir un pourcentage donné de la solution optimale sur toute instance, aussi difficile soit-elle, est un petit exploit en soi. De plus, cela n'affecte pas

[3] Ces nouveaux chemins sont appelés *descendants*.

le comportement en moyenne de l'algorithme, qui pourrait être aussi bon que mauvais. D'ailleurs, le lecteur de l'article n'y trouvera pas de résultats expérimentaux. Transformer l'algorithme EXPLOREDESCENDANTS en un bon algorithme en pratique et l'adapter à des problèmes plus intéressants d'un point de vue appliqué (plusieurs paires origine-destination, capacités, etc.) est un projet de recherche en soi. L'objectif principal de ce projet est plutôt d'explorer d'un point de vue théorique la structure combinatoire de MAXTOLL et d'en déterminer la complexité algorithmique. Un résultat important dans ce sens est fourni à la section 4.2 de l'article. On y montre qu'un algorithme efficace – basé sur la relaxation naturelle de MAXTOLL – ne peut pas garantir un facteur d'approximation meilleur que celui obtenu pour EXPLOREDESCENDANTS. Cela suggère que MAXTOLL est difficile à résoudre non seulement exactement, mais aussi approximativement, d'où la conjecture :

Conjecture 1 *Pour* MAXTOLL, *il est NP-difficile d'obtenir un facteur d'approximation asymptotiquement meilleur que* $O(\log m_T)$.

CONCLUSION

Les résultats obtenus dans le cadre de ce mémoire confirment que la tarification optimale d'un réseau est un problème algorithmique difficile. La version la plus épurée du problème, MAXTOLL, est *NP*-complète et le meilleur algorithme polynomial d'approximation (en un certain sens) ne garantit qu'un facteur de précision de l'ordre de $O(\log m_T)$. Par surcroît, il est entendu que le modèle étudié dans ce mémoire est beaucoup trop simple pour présenter un véritable intérêt pratique. Son utilité première réside plutôt dans l'aisance avec laquelle il se laisse analyser d'un point de vue théorique.

Néanmoins, MAXTOLL est le noyau dur d'une vaste classe de problèmes beaucoup plus réalistes. En guise de conclusion, on présente dans ce chapitre des extensions naturelles du problème, qui restent à être explorées autant sur le plan théorique que pratique. Une chose est sûre, la *NP*-complétude de MAXTOLL implique que toutes les généralisations qui suivent sont aussi des problèmes algorithmiques difficiles.

1. *Paires origine-destination multiples* : L'algorithme EXPLOREDESCENDANTS s'adapte à la généralisation du problème MAXTOLL muni d'un ensemble $\mathcal{K}$ de plusieurs paires origine-destination. Dans ce cas, les usagers de chaque paire $k \in \mathcal{K}$ minimisent indépendamment leur coût total. Si des péages distincts

T^k pouvaient être associés à chaque paire $k \in \mathcal{K}$, ce nouveau problème se décomposerait simplement en $|\mathcal{K}|$ copies du problème de tarification original[4]. Toutefois, l'imposition d'un péage unique T complique considérablement la situation en raison des interactions existant entre les paires. Il est toutefois possible d'appliquer EXPLOREDESCENDANTS sur chaque paire séparément puis de choisir les taxes qui mènent au profit total le plus élevé. Un facteur d'approximation de l'ordre de $O(|\mathcal{K}| \log m_T)$ est ainsi obtenu.

2. *Capacités et congestion* : Une autre généralisation naturelle consiste à ajouter sur les arcs du réseau des capacités ou des délais dépendant du flot – c'est-à-dire de la congestion. Dans ce cas, notre approche n'est plus valide : se contenter de considérer des chemins n'est plus suffisant, il faut considérer des flots d'usagers. Malgré tout, il est peut-être possible de recycler les idées mises de l'avant dans ce mémoire, mais une intuition nouvelle est clairement nécessaire.

3. *Usagers hétérogènes* : Plusieurs auteurs, en particulier Côté et al. [10], Cole et al. [8] et Yang et al. [32], ont considéré le cas d'usagers hétérogènes. Dans la version de MAXTOLL étudiée dans ce mémoire, tous les usagers perçoivent les coûts fixes et les péages de la même façon : ce sont des usagers homogènes. La réalité est tout à fait différente : un usager riche souhaitera voyager le plus rapidement possible de son origine à sa destination même

[4] $|S|$ dénote la cardinalité de l'ensemble S.

s'il doit pour cela payer des péages élevés; un usager pauvre, au contraire, désirera sans doute minimiser les péages encourus même si cela implique emprunter un chemin plus long. Pour modéliser de tels usagers hétérogènes, une fonction de perception des péages intervient dans le coût total qui devient $c + \beta T$, où β caractérise l'usager. Dans le cas d'un nombre fini de classes d'usagers (et si l'on oublie pour le moment la congestion et les paires origine-destination multiples), il est évidemment possible d'appliquer EXPLOREDESCENDANTS à chaque classe d'usagers séparément et d'obtenir le même facteur d'approximation que dans le cas homogène. Malheureusement, ce genre de modèle ne devient véritablement intéressant qu'avec des capacités sur les arcs, auquel cas les difficultés mentionnées au paragraphe précédent resurgissent.

4. *Oligopoles* : Qu'advient-il lorsque plusieurs entreprises en compétition contrôlent des sous-ensembles distincts d'arcs taxables ? Le lecteur de ce mémoire devra se contenter de la question. Une meilleure compréhension des généralisations précédentes est nécessaire avant d'aborder cette situation oligopolistique complexe, qui est par ailleurs d'un intérêt pratique incontestable.

RÉFÉRENCES

AUDET, C., HANSEN, P., JAUMARD, B., SAVARD, G. (1997). Links between Linear Bilevel and Mixed 0-1 Programming Problems, *Journal of Optimization Theory and Applications* 93, 273–300.

AUSIELLO, G., CRESCENZI, P., KANN, V., MARCHETTI-SPACCAMELA, A., PROTASI, M. (1999). *Complexity and Approximation : Combinatorial Optimization Problems and Their Approximability Properties*, Springer, Berlin.

BOUHTOU, M., VAN HOESEL, S., VAN DER KRAAIJ, A.F., LUTTON, J.-L. (2003). Tariff optimization in networks, soumis pour publication.

BRACKEN, J., McGILL, J. (1973). Mathematical programs with optimization problems in the constraints, *Operations Research* 21, 37–44.

BROTCORNE, L., LABBÉ, M., MARCOTTE, P., SAVARD, G. (2000). A bilevel model and solution algorithm for a freight tariff-setting problem, *Transportation Science* 34, 289–302.

BROTCORNE, L., LABBÉ, M., MARCOTTE, P., SAVARD, G. (2001). A bilevel model for toll optimization on a multicommodity transportation network, *Transportation Science* 35, 345–358.

COCCHI, R., SHENKER, S., ESTRIN, D., ZHANG, L. (1993). Pricing in Computer Networks: Motivation, Formulation, and Example, *IEEE/ACM Transactions on Networking* 1, 614–627.

COLE, R., DODIS, Y., ROUGHGARDEN, T. (2003). Pricing Network Edges for Heterogeneous Selfish Users, à paraître dans *STOC 2003.*

COOK, S.A. (1971). The complexity of theorem-proving procedures, *Proceedings of the Third Annual ACM Symposium on the Theory of Computing*, 151–158.

CÔTÉ, J.-PH., MARCOTTE, P., SAVARD, G. (2002). A Bilevel Modeling to Pricing and Fare Optimization in the Airline Industry, *Cahiers du GERAD*, G-2002-71, HEC Montreal.

FLOUDAS, C.A., PARDALOS, P.M., Éditeurs (2001). *Encyclopedia of Optimization*, Kluwer.

GAREY, M.R., JOHNSON, D.S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman, New York.

HANSEN, P., JAUMARD, B., SAVARD, G. (1992). New branch-and-bound rules for linear bilevel programming, *Journal Of Optimization Theory and Applications* 22, 1194–1217.

HEARN, D., RAMANA, M.V. (1998). Solving congestion toll pricing models, dans: *Equilibrium and Advanced Transportation Modelling*, Marcotte and Nguyen (éditeurs), Kluwer, pp. 109–124.

HOCHBAUM, D.S., Éditeur (1997). *Approximation Algorithms for NP-hard Problems*, PWS Publishing Company, Boston.

JEROSLOW, R.G. (1985). A polynomial hierarchy and a simple model for competitive analysis, *Mathematical Programming* 32, 146–164.

KORILIS, Y.A., LAZAR, A.A., ORDA, A. (1997). Achieving Network Optima Using Stackelberg Routing Strategies, *IEEE/ACM Transactions on Networking* 1, 161–173.

LARSSON, T., PATRIKSSON, M. (1998). Side constrained traffic equilibrium models – Traffic management through link tolls, dans: *Equilibrium and Advanced Transportation Modelling*, Marcotte and Nguyen (Éditeurs), Kluwer, 125–151.

LABBÉ, M., MARCOTTE, P., SAVARD, G. (1998). A bilevel model of taxation and its application to optimal highway pricing, *Management Science* 44, 1595–1607.

LABBÉ, M. MARCOTTE, P., SAVARD, G. (1999). On a class of bilevel programs, dans: *Nonlinear Optimization and Related Topics*, Di Pillo and Giannessi (Editors), Kluwer Academic Publishers, 183–206.

LUO, Z.-Q., PANG, J.-S., RALPH, D. (1996). *Mathematical Programming with Equilibrium Constraints*, Cambridge University Press, UK.

MARCOTTE, P. (1986). Network design problem with congestion effects: a case of bilevel programming, *Mathematical Programming* 34, 142–162.

MARCOTTE, P., SAVARD, G., SEMET, F. (2002). A Bilevel Programming Approach to the Travelling Salesman Problem, *Cahiers du GERAD*, G-2002-73, HEC Montréal.

PIGOU, A.C. (1920). *The Economics of Welfare*, Macmillan and Co., London.

ROUGHGARDEN, T. (2001). Stackelberg Scheduling Strategies, *Proceedings of the 33rd ACM Symposium on Theory of Computing*, Crete, pp. 104–113.

ROUGHGARDEN, T. (2001). Designing Networks for Selfish Users is Hard, *Proceedings of the the 42nd Annual Symposium on Foundations of Computer Science*, Las Vegas, pp. 472–481.

STACKELBERG, H. (1952). *The Theory of the Market Economy*, Oxford University Press, Oxford.

VAZIRANI, V.V. (2001). *Approximation Algorithms*, Springer, Berlin.

VERHOEF, E.T., P. NIJKAMP, P. RIETVELD (1996). Second-best congestion pricing: the case of an untolled alternative, *Journal of Urban Economics* 40, 279–302.

VICENTE, L.N., CALAMAI, P.H. (1994). Bilevel and Multilevel Programming: A Bibliography Review, *Journal of Global Optimization* 5, 291–306.

VICENTE, L., SAVARD, G., JÚDICE, J. (1994). Descent Approaches for Quadratic Bilevel Programming, *Journal of Optimization Theory and Applications* 81, 379–399.

YANG, H., HUANG, H.J. (2003). The multi-class, multi-criteria traffic network equilibrium and systems optimum problem, à paraître dans *Transportation Reasearch, part B.*

ANNEXE I

ARTICLE INTITULÉ *DESIGN AND ANALYSIS OF AN APPROXIMATION ALGORITHM FOR STACKELBERG NETWORK PRICING*

Design and analysis of an approximation algorithm for Stackelberg network pricing

Sébastien Roch and Gilles Savard
Département de mathématiques
et de génie industriel
École Polytechnique de Montréal
Montréal, Québec, Canada

Patrice Marcotte
Département d'informatique
et de recherche opérationnelle
Université de Montréal
Montréal, Québec, Canada

16th June 2003

Abstract

We consider the problem of maximizing the revenue raised from tolls set on the arcs of a transportation network, under the constraint that users are assigned to toll-compatible shortest paths. We first prove that this problem is strongly NP-hard. We then provide a polynomial time algorithm with a worst-case precision guarantee of $\frac{1}{2}\log m_T + 1$, where m_T denotes the number of toll arcs. Finally we show that the approximation is tight with respect to a natural relaxation by constructing a family of instances for which the relaxation gap is reached.

Keywords: network pricing, approximation algorithms, Stackelberg games, combinatorial optimization, NP-hard problems.

1 Introduction

This paper focuses on a class of bilevel problems that arise naturally when tariffs, tolls, or devious taxes are to be determined over a network. This class of problems encompasses several important optimization problems encountered in the transportation, telecommunication, and airline industries. Our aim is twofold: first, we show that the problem is NP-hard; then we present a polynomial time algorithm with a tight worst-case guarantee of performance.

Bilevel programming is a modelling framework for situations where one player (the "leader") integrates within its optimization schedule the reaction of a second player (the "follower") to its own course of action. These problems are closely related to static Stackelberg games and mathematical programs with equilibrium constraints (or MPECs, see Luo, Pang and Ralph [13]), in which the lower level solution characterizes the equilibrium state of a physical or social system. Bilevel programs allow the modelling of a variety of situations that occur in operations research, economics, finance, etc. For instance, one may consider the maximization of social welfare, taking into account the selfish behavior of consumers. It is well-known that the taxation of resources and services at marginal cost (Pigovian taxes [16]) maximizes global welfare. However, when some resources fall outside the control of the leader, the social optimum might not be reachable, yielding a "second-best" problem of true Stackelberg nature (see Verhoef, Nijkamp and Rietveld [20], Hearn and Ramana [6], and Larsson and Patriksson [10] for traffic examples). In constrast

with these studies, we adopt the point of view of a firm involved in the management of the network but oblivious to social welfare; the firm's only goal is to maximize its own revenue.

Bilevel programs are generically nonconvex and nondifferentiable, i.e., to all practical extent, intractable. In particular, it has been shown by Jeroslow [8] that linear bilevel programming is NP-hard. This result has been refined by Vicente, Savard and Júdice [21], who proved that obtaining a mere certificate of local optimality is strongly NP-hard. Actually, Audet et al. [1] unveiled the close relationship between bilevel programming and integer programming. This "intractability" has prompted the development of heuristics that are adapted to the specific nature of the instance under consideration, together with their worst-case analysis. Such analysis was first performed for a network design problem with user-optimized flows by Marcotte [14], who proved worst-case bounds for convex optimization based heuristics. More recently, worst-case analysis of Stackelberg problems has been applied to job scheduling and to network design by Roughgarden [17, 18], to network routing by Korilis, Lazar and Orda [9] and to pricing of computer networks by Cocchi et al. [3]. All these works focus on "soft" Stackelberg games, where the objectives of both players are non-conflicting, and where heuristics are expected to perform well in practice, although their worst-case behavior may turn out to be bad.

In this paper, we analyze an approximation algorithm for the toll optimization problem (MAXTOLL in the sequel) formulated and analyzed by Labbé, Marcotte

and Savard [11]. In this game, which is almost zero-sum, a leader sets tolls on a subset of arcs of a transportation network, while network users travel on shortest paths with respect to the cost structure induced by the tolls. Labbé et al. [11] proved that the Hamiltonian path problem can be reduced to a version of MAX-TOLL involving *negative* arc costs and *positive* lower bounds on tolls[1]. In this paper, we improve this result by showing that MAXTOLL, without lower bound constraints on tolls, is strongly NP-hard. Next, in the single-commodity case, we provide a polynomial time algorithm with a performance guarantee of $\frac{1}{2}\log m_T + 1$, where m_T denotes the number of toll arcs in the network. We then use this result as well as specially constructed instances to prove the tightness of our analysis, as well as the optimality of the approximation factor obtained with respect to a natural upper bound.

The rest of the paper is organized as follows. In Section 2, we state the problem and prove that it is NP-hard. In Section 3 we introduce an approximation algorithm whose performance is analyzed in Section 4.

2 The model and its complexity

2.1 The model

The generic *bilevel toll problem* can be expressed as

[1]It was also shown recently by Marcotte et al.[15] that the TSP is a special case of MAXTOLL.

$$\max_{T} Tx$$

where x is the partial solution of the parametric linear program

$$\begin{aligned} \min_{x,y} \quad & (c_1 + T)x + c_2 y \\ \text{s.t.} \quad & A_1 x + A_2 y = b \\ & x, y \geq 0. \end{aligned}$$

In the above, T represents a *toll vector*, x the vector of *toll commodities* and y the vector of *toll-free commodities.*

We shall consider a combinatorial version of this problem. Let $G = (V, A)$ be a directed multigraph with two distinguished vertices: the origin $s \in V$ and the destination $t \in V$. The arc set A is partitioned into subsets A_T and A_U of *toll* and *toll-free* arcs, of respective cardinalities m_T and m_U. Arcs are assigned *fixed costs* $c : A_T \to \mathbf{N}^{m_T}$ and $d : A_U \to \mathbf{N}^{m_U}$ (in the sequel, $\mathbf{N} = \{0, 1, \ldots\}$). Once *tolls* are added to the fixed costs of A_T, we obtain a *toll network* $\mathcal{N}_T = (G, c + T, d, s, t)$. Denoting by $\mathcal{SP}[\mathcal{N}_T]$ the set of shortest paths from s to t, we can then formulate MAXTOLL as the combinatorial mathematical program (see also Figure 1):

$$\max_{\substack{T \geq 0 \\ P \in \mathcal{SP}[\mathcal{N}_T]}} \sum_{e \in A_T \cap P} T(e). \tag{1}$$

This is a single-commodity instance of the toll setting problem analyzed in [11]. In this framework, the leader must strike the right balance between low toll levels, which generate low revenue, and high levels, which could also result in low revenue,

INSTANCE: – a directed graph $G = (V,\ A_T \cup A_U)$

– fixed cost vectors $c : A_T \rightarrow \mathbf{N}^{m_T}$ and $d : A_U \rightarrow \mathbf{N}^{m_U}$

– distinguished vertices $s, t \in V$ such that there exists a simple path from s to t in $(V,\ A_U)$

SOLUTION: – a nonnegative toll vector T on A_T

– a simple $s - t$ path P of minimal length w.r.t. the cost structure $(c + T,\ d)$

MEASURE: – maximize $\sum_{e \in A_T \cap P} T(e)$

Figure 1: MAXTOLL

as the follower would select a path with few toll arcs, or even none. An instance of MAXTOLL is illustrated in Figure 2.

Several remarks are in order. First, to avoid a trivial situation, we posit the existence of at least one toll-free path from s to t. Second, our formulation implies that, given ties at the lower level (the user's level), the leader chooses among the toll-compatible shortest paths the one travelled by the follower. Note that a risk-averse leader could always force the use of the most profitable path by subtracting a small amount from every toll on that path, thus yielding a revenue as close as

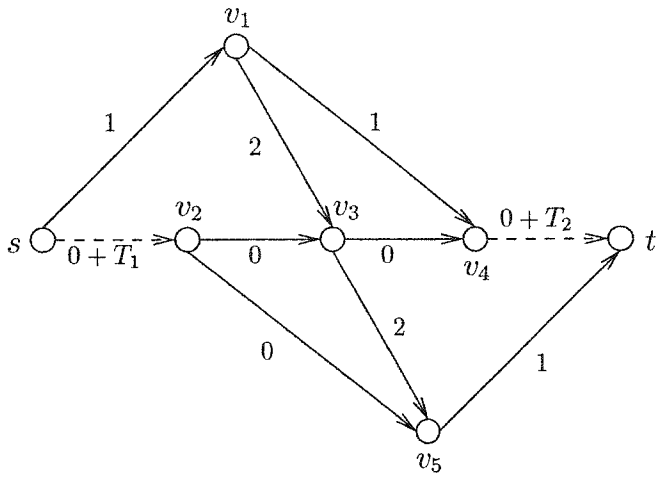

Figure 2: This network contains two toll arcs (represented by dashed arcs). Fixed costs are given by numbers close to each arc. The optimal path is (s, v_2, v_5, t) with a revenue of 5 when $T_1 = 5$ and $T_2 = 1000$.

desired to the revenue generated by the "cooperative" solution. Thirdly, once a path P has been selected by the leader, toll arcs outside P become irrelevant. In practice, the removal of these arcs can be achieved by setting tolls to an arbitrarily large value on toll arcs outside P. We denote by $\mathcal{N}_T(P)$ the network where these arcs have been removed. Finally, our central results hold also in a version of MAXTOLL where T is unconstrained; in this case, negative tolls can be interpreted as subsidies. Actually, Labbé, Marcotte and Savard [12] have constructed instances where the optimal solution involves negative tolls. Nevertheless, throughout most of the paper we focus on nonnegative tolls because (i) this case is interesting in its own sake, (ii) intermediate results are easier to interpret when tolls are thought to be nonnegative.

A natural upper bound on the leader's revenue has been derived by Labbé et al. [11] using duality arguments from linear programming theory. It also follows from Theorem 4 of Section 3.1. Let $\mathcal{L}_\infty$ be the length of a shortest toll-free path and let $\mathcal{L}(P)$ be the length of a given path P with $T = 0$.

Theorem 1 *Let P be a path. Then the optimal revenue associated with P is bounded by*

$$B(P) \equiv \mathcal{L}_\infty - \mathcal{L}(P). \tag{2}$$

Since $\mathcal{L}_\infty$ does not depend on P, it follows that the largest upper bound corresponds to the path with smallest value of $\mathcal{L}(P)$; that is, P is a shortest path when tolls

are set to 0. We denote the length of such a path by $\mathcal{L}_0$ and by

$$LP = \mathcal{L}_\infty - \mathcal{L}_0 \tag{3}$$

the value of a path-independent upper bound. This bound is simply the difference between the costs of shortest paths corresponding to infinite and null tolls, respectively. Note that, if the set of toll arcs is a singleton, the upper bound can always be achieved.

2.2 NP-hardness of MAXTOLL

The purpose of this section is to show that MAXTOLL is strongly NP-hard. We also prove that a version of MAXTOLL where the toll vector is unconstrained shares this property, thus settling a conjecture about the complexity status of the generic toll setting problem.

Theorem 2 MAXTOLL *is strongly NP-hard.*

Proof: Let C denote the sum of all fixed costs. It is not difficult to show that there exists an optimal toll vector T that is integer-valued and less than $C + 1$; in particular, optimal solutions are of polynomial size.

Now, consider a reduction from 3-SAT to MAXTOLL (see [4]). Let $x_1, \ldots, x_n$ be n Boolean variables and

$$F = \bigwedge_{i=1}^{m} (l_{i1} \vee l_{i2} \vee l_{i3}) \tag{4}$$

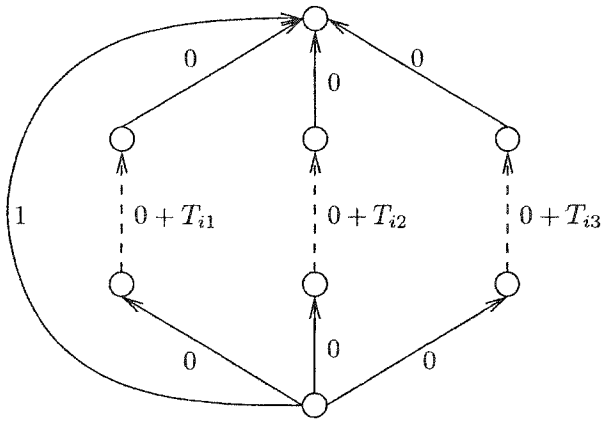

Figure 3: Sub-network for clause $(l_{i1} \vee l_{i2} \vee l_{i3})$.

be a 3-CNF formula consisting of m clauses with literals (variables or their negations) l_{ij}. For each clause, we construct a sub-network comprising one toll arc for each literal as shown in Figure 3.

The idea is the following: if the optimal path goes through toll arc T_{ij}, then the corresponding literal l_{ij} is TRUE (note: if $l_{ij} = \overline{x}_k$, then x_k =FALSE). The sub-networks are connected by two arcs, a toll-free arc of cost 2 and a toll arc of cost 0, as shown in Figure 4.

If F is satisfiable, we want the optimal path to go through a single toll arc per sub-network (i.e., one TRUE literal per clause) and simultaneously want to make sure that the corresponding assignment of variables is consistent; i.e., paths that include a variable and its negation must be ruled out. For that purpose, we assign to every pair of literals corresponding to a variable and its negation an inter-clause toll-free arc between the corresponding toll arcs (see Figure 4). As we will see, this implies

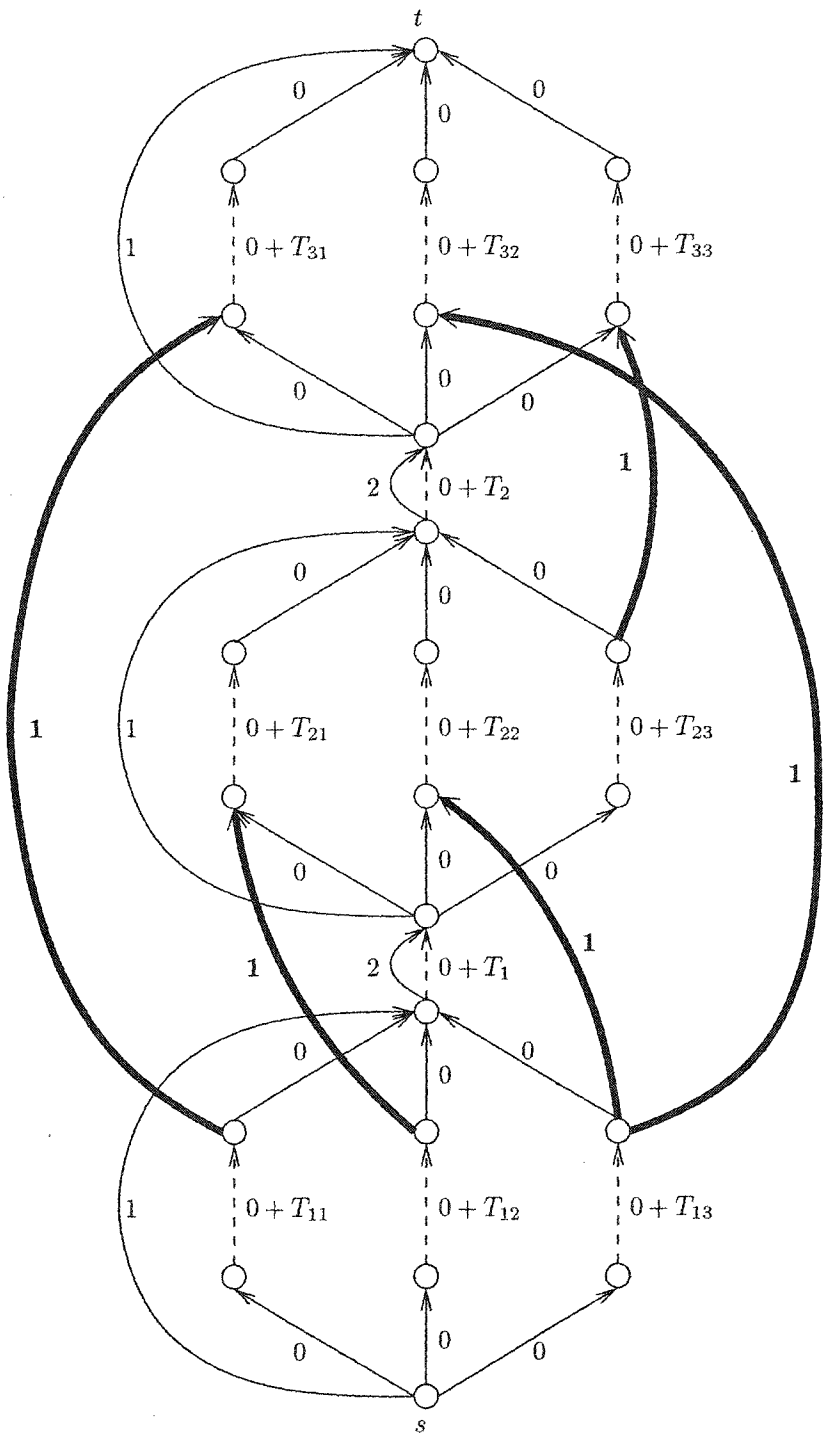

Figure 4: Network for the formula $(x_1 \vee x_2 \vee \overline{x}_3) \wedge (\overline{x}_2 \vee x_3 \vee \overline{x}_4) \wedge (\overline{x}_1 \vee x_3 \vee x_4)$. Inter-clause arcs are bold. Path through T_{12}, T_{22}, T_{32} is optimal ($x_2 = x_3 =$TRUE).

that *inconsistent* paths, involving a variable and its negation, are suboptimal.

Since the length of a shortest toll-free path is $m + 2(m-1) = 3m - 2$ and that of a shortest path with zero tolls is 0, $3m - 2$ is an upper bound on the revenue. We claim that F is satisfiable if and only if the optimal revenue is equal to that bound.

Assume that the optimal revenue is equal to $3m - 2$. Obviously, the length of the optimal path when tolls are set to 0 must be 0, otherwise the upper bound cannot be reached. To achieve this, the optimal path has to go through one toll arc per sub-network (it cannot use inter-clause arcs) and tolls have to be set to 1 on selected literals, $C + 1$ on other literals and 2 on tolls T_k, $\forall\ k$. We claim that the optimal path does not include a variable and its negation. Indeed, if that were the case, the inter-clause arc joining the corresponding toll arcs would impose a constraint on the tolls between its endpoints. In particular, the toll T_k immediately following the initial vertex of this inter-clause arc would have to be set at most to 1, instead of 2. This yields a contradiction. Therefore, the optimal path must correspond to a consistent assignment, and F is satisfiable (note: if a variable and its negation do not appear on the optimal path, this variable can be set to any value).

Conversely if F is satisfiable, at least one literal per clause is TRUE in a satisfying assignment. Consider the path going through the toll arcs corresponding to these literals. Since the assignment is consistent, the path does not simultaneously include a variable and its negation, and no inter-clause arc limits the revenue. Thus, the upper bound of $3m - 2$ is reached on this path.

Finally, note that the number of arcs in the reduction is less than

$$10m + 2(m-1) + (3m)^2$$

and that all constants are polynomially bounded in m.

□

It is not difficult to prove that the same NP-hardness reduction works when negative tolls are allowed.

Theorem 3 MAXTOLL *is still strongly NP-hard when negative tolls are allowed.*

Proof: We use the same reduction as in the nonnegative case. The proof rests on two results proved in [11]. First, the upper bound is valid when T is unrestricted. Second, there exists optimal solutions of polynomial size. The latter result follows from a polyhedral characterization of the feasible set.

From the first result, we know that the $3m-2$ upper bound on the revenue is unchanged. On the other hand, if F is satisfiable, the feasible solution considered in the nonnegative case still yields a $3m-2$ revenue. We only have to make sure that negative tolls cannot produce a $3m-2$ revenue when F is not satisfiable. Again, to reach the upper bound, one has to use a path of length 0 when tolls are set to 0. Consequently the optimal path comprises exactly one literal per clause. Now the toll-free arcs of length 1 and 2 limit the T_{ij}'s on the path to 1 and the T_k's to 2, so negative tolls are useless in this case. Indeed, inconsistent paths will

see their revenue limited by inter-clause arcs without any possibility to make for the loss incurred from negative tolls.

□

3 An approximation algorithm

In this section, we devise a polynomial time approximation algorithm for MAX-TOLL. Such algorithm is guaranteed to compute a feasible solution with objective at least OPT/α, where OPT is the optimal revenue and α, which depends on the number m_T of toll arcs, denotes the approximation factor. For a survey of recent results on approximation algorithms, the reader is referred to [7] and [19].

3.1 Preliminaries: characterizing consistent tolls

The leader is only interested in paths that have the potential of generating positive revenue. This remark warrants the following definitions. Recall that $\mathcal{N}_T(P)$ is the network $\mathcal{N}_T$ in which toll arcs outside the path P have been removed.

Definition 1 *Let m_T^P denote the number of toll arcs in a path P from s to t. We say that P is* **valid** *if $m_T^P \geq 1$ and P is a shortest path with respect to a null toll vector T, i.e., $P \in \mathcal{SP}[\mathcal{N}_0(P)]$.*

It is clear that non-valid paths cannot generate revenue. Any valid path P can be expressed as a sequence

$$P = \left(v_{0,1},\ \tau_1,\ v_{1,2},\ \tau_2, \ldots, \tau_{m_T^P},\ v_{m_T^P, m_T^P+1}\right) \tag{5}$$

where τ_i is the i-th toll arc of P (in the order of traversal) and $v_{i,i+1}$ is the toll-free subpath of P from the terminal vertex $\text{TERM}(\tau_i)$ of τ_i to the initial vertex $\text{INIT}(\tau_{i+1})$ of τ_{i+1}. According to this notation, $v_{0,1}$ starts in s and $v_{m_T^P, m_T^P+1}$ ends in t. Since $P \in \mathcal{SP}[\mathcal{N}_0(P)]$, $v_{i,i+1}$ is a shortest toll-free path from $\text{TERM}(\tau_i)$ to $\text{INIT}(\tau_{i+1})$. We extend this notation to $v_{i,j}$, a shortest toll-free path from $\text{TERM}(\tau_i)$ to $\text{INIT}(\tau_j)$, with length $\mathcal{U}_{i,j}$. For $k < l$, let $\mathcal{L}_{k,l}$ be the length of P from $\text{TERM}(\tau_k)$ to $\text{INIT}(\tau_l)$ with tolls set to 0, and $\mathcal{T}_{k,l}$ be the sum of tolls between $\text{TERM}(\tau_k)$ and $\text{INIT}(\tau_l)$ on P,

$$\mathcal{L}_{k,l} = \sum_{i=k}^{l-1} \mathcal{U}_{i,i+1} + \sum_{i=k+1}^{l-1} c(\tau_i) \qquad \mathcal{T}_{k,l} = \sum_{i=k+1}^{l-1} T(\tau_i), \tag{6}$$

with the convention that $\sum_{i=k}^{l} x_i = 0$ if $l < k$.

Definition 2 *Let P be a valid path. The toll vector T is* **consistent** *with P if $P \in \mathcal{SP}[\mathcal{N}_T(P)]$; that is, P remains the shortest path when tolls outside P are removed and tolls on P are set according to the vector T.*

The following result, which characterizes consistent tolls, is the starting point of our algorithm.

Theorem 4 *Let P be a valid path. Then, the toll vector T is consistent with P if and only if*

$$\mathcal{L}_{i,j} + \mathcal{T}_{i,j} \leq \mathcal{U}_{i,j} \qquad \forall\ 0 \leq i < j \leq m_T^P + 1. \tag{7}$$

Proof : $\Longrightarrow$) Obvious from the very definition of consistent tolls. $\Longleftarrow$) The converse looks equally obvious. However, one must be careful about paths that borrow the toll arcs of P in a *different* sequence. Assume, by contradiction, that such a path $\widetilde{P}$ is strictly shorter than P in $\mathcal{N}_T(P)$, and that conditions (7) are satisfied. We note

$$\widetilde{P} = \left(\tilde{v}_{0,1},\ \tilde{\tau}_1,\ \tilde{v}_{1,2},\ \ldots,\ \tilde{\tau}_{m_T^{\widetilde{P}}},\ \tilde{v}_{m_T^{\widetilde{P}},m_T^{\widetilde{P}}+1}\right) \tag{8}$$

with corresponding $\widetilde{\mathcal{U}}_{i,j}$, $\widetilde{\mathcal{L}}_{k,l}$ and $\widetilde{\mathcal{T}}_{k,l}$ for all i, j, k, l with $k < l$. For all i, $\tilde{\tau}_i = \tau_{\delta(i)}$ for some injective function δ (toll arcs outside P are irrelevant). To derive a contradiction, we will construct a path that is shorter than $\widetilde{P}$ by modifying the "backward" toll-free subpaths of $\widetilde{P}$ to obtain a new path that is shorter than $\widetilde{P}$. This improved path will happen to be P.

Let $j_1 = 1$ and $j_k = \min\{j > j_{k-1} : \delta(j) > \delta(j_{k-1})\}$, as long as such j_k exists, the last one being denoted j_K. We further define $\delta(0) = j_0 = 0$ and $\delta(m_T^{\widetilde{P}}+1) = j_{K+1} = m_T^{\widetilde{P}}+1$. By the definition of this increasing subsequence, we have $\delta(j_k-1) \leq \delta(j_{k-1})$ for all $k > 1$. Thus, if $j_k - 1 = j_{k-1}$,

$$\widetilde{\mathcal{L}}_{j_{k-1},j_k} + \widetilde{\mathcal{T}}_{j_{k-1},j_k} = \mathcal{U}_{\delta(j_{k-1}),\delta(j_k)} \geq \mathcal{L}_{\delta(j_{k-1}),\delta(j_k)} + \mathcal{T}_{\delta(j_{k-1}),\delta(j_k)},$$

(note that in this case, there are no toll arcs on $\widetilde{P}$ between TERM($\tilde{\tau}_{j_{k-1}}$) and INIT($\tilde{\tau}_{j_k}$)), otherwise

$$\begin{aligned}
\widetilde{\mathcal{L}}_{j_{k-1},j_k} + \widetilde{\mathcal{T}}_{j_{k-1},j_k} &= \widetilde{\mathcal{L}}_{j_{k-1},j_k-1} + \widetilde{\mathcal{T}}_{j_{k-1},j_k-1} + c(\tau_{\delta(j_k-1)}) \\
&\quad + T(\tau_{\delta(j_k-1)}) + \mathcal{U}_{\delta(j_k-1),\delta(j_k)} \\
&\geq \mathcal{U}_{\delta(j_k-1),\delta(j_k)} \geq \mathcal{L}_{\delta(j_k-1),\delta(j_k)} + \mathcal{T}_{\delta(j_k-1),\delta(j_k)} \\
&\geq \mathcal{L}_{\delta(j_{k-1}),\delta(j_k)} + \mathcal{T}_{\delta(j_{k-1}),\delta(j_k)}
\end{aligned}$$

and we can replace the subpath of $\widetilde{P}$ between TERM($\tilde{\tau}_{j_{k-1}}$) and INIT($\tilde{\tau}_{j_k}$), by the subpath of P between TERM($\tau_{\delta(j_{k-1})}$) and INIT($\tau_{\delta(j_k)}$) without increasing its length. But after doing this for all k, we obtain P which is a contradiction.

□

Remark. There always exists an optimal solution with tolls less than $C + 1$. Indeed, $\mathcal{U}_{i,j} < C + 1$, $\forall i, j$ implies that optimal tolls on P have to be lower than $C + 1$. Therefore, fixing tolls to $C + 1$ outside P generates no additional active constraints on the tolls of P.

Figure 5 provides an equivalent representation of the example of Figure 2 according to Theorem 4.

3.2 The general idea underlying the algorithm

EXPLOREDESCENDANTS is an approximation algorithm motivated by the characterization of consistent tolls. Initially, the algorithm computes the optimal revenue

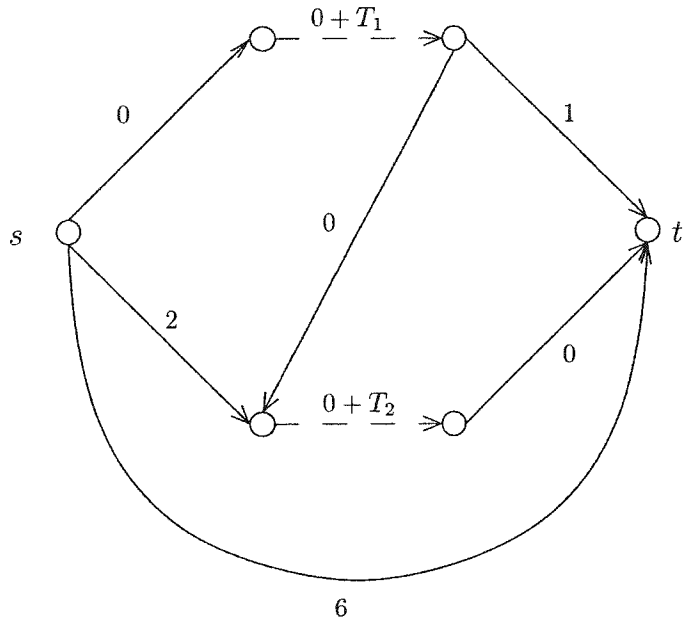

Figure 5: Equivalent representation of the example of Figure 2. All arcs have been replaced by shortest paths between toll arcs and origin/destination.

associated with a shortest path in $\mathcal{SP}[\mathcal{N}_0]$, i.e, a path with the largest upper bound $B(P)$[2]. If revenue is smaller than the upper bound LP, Theorem 4 implies that there exists a toll-free subpath $v_{k,l}$ $k < l$ whose short length forces some tolls in P to be small. To relax this constraint, it makes sense to skip the subpath of P between TERM(τ_k) and INIT(τ_l) and to replace it by $v_{k,l}$. This yields a new path whose length will not be much larger than the length of P, and for which some constraints (7) have been removed. This path is a natural candidate for improved revenue. By repeating this process, we will show that Algorithm EXPLOREDESCENDANTS can uncover, in polynomial time, a path with good approximation properties.

Algorithm EXPLOREDESCENDANTS is streamlined in Figure 6. It comprises two subroutines, MAXREV and TOLLPARTITION. MAXREV computes the largest revenue compatible with the shortest path status of P. Starting from P, TOLLPARTITION generates two descendants of path P. The algorithm is initialized with P_0 in $\mathcal{N}_0$, a shortest path of length $\mathcal{L}_0$.

[2]As shown in the next section, this can be achieved in polynomial time.

Algorithm EXPLOREDESCENDANTS

Input: a path P

Output: a path $\overline{P}$, tolls $\overline{T}$ and objective value $\overline{V}$

- Compute maximum revenue V_P achievable on P and corresponding toll vector T_P: $(V_P, T_P) \leftarrow$ MAXREV(P)
- If $V_P < B(P)$ then
  - Derive new paths from P :

    $(P_1, P_2) \leftarrow$ TOLLPARTITION(P)
  - For $i = 1, 2$:

    $(\overline{V}_i, \overline{T}_i, \overline{P}_i) \leftarrow$ EXPLOREDESCENDANTS(P_i)
  - Return $\overline{V} \leftarrow \max\{V_P, \overline{V}_1, \overline{V}_2\}$ and corresponding toll vector $\overline{T}$ and path $\overline{P}$
- Else return $(\overline{V}, \overline{T}, \overline{P}) \leftarrow (V_P, T_P, P)$

Figure 6: Algorithm EXPLOREDESCENDANTS

3.3 Maximizing path-compatible revenue

Let P be a valid path in $\mathcal{N}$ denoted by

$$P = \left(v_{0,1},\, \tau_1,\, v_{1,2},\, \ldots,\, \tau_{m_T^P},\, v_{m_T^P, m_T^P+1}\right) \tag{9}$$

with corresponding values of $\mathcal{U}_{i,j}$, $\mathcal{L}_{k,l}$ and $\mathcal{T}_{k,l}$ for all i, j, k, l, with $k < l$. The optimal tolls compatible with P's shortest path status are obtained from the recursion

$$T(\tau_k) := t_k \equiv \min_{0 \le i < k < j \le m_T^P + 1} \left\{ \mathcal{U}_{i,j} - \mathcal{L}_{i,j} - \mathcal{T}_{i,k} \right\} \tag{10}$$

where $T(\tau_1) = t_1, \ldots, T(\tau_{k-1}) = t_{k-1}$. The validity of the above formula rests on Theorem 5 where we construct a sequence of toll-free paths such that (i) every toll of P is bounded from above by at least one path in the sequence (condition (13) below), (ii) the sum of the tolls defined by (10) is equal to the sum of the bounds imposed by the paths of the sequence (condition (14) below).

Labbé et al. [11] give another polynomial time algorithm for this task but it does not provide the information we need regarding the active toll-free subpaths. This information will be instrumental in generating new paths from P and in obtaining an approximation guarantee.

Theorem 5 *Let $v_{i,j}$, $0 \le i < j \le m_T^P + 1$, be the toll-free subpaths defined in Section 3.1 and let t_k, $1 \le k \le m_T^P$ be as in (10). Then, there exists a sequence of paths*

$$v_{i(1),j(1)},\, v_{i(2),j(2)}, \ldots,\, v_{i(q),j(q)} \tag{11}$$

with $i(1) = 0$ *and* $j(q) = m_T^P + 1$ *such that for all* h *(see Figure 7)*

$$i(h+1) < j(h) \leq i(h+2) < j(h+1) \tag{12}$$

and for all k

$$|\{h \ : \ i(h) + 1 \leq k \leq j(h) - 1\}| \geq 1 \tag{13}$$

with equality if $t_k \neq 0$. *This implies that for all* h', h'' *with* $h' \leq h''$

$$\sum_{k=i(h')+1}^{j(h'')-1} t_k = \sum_{h=h'}^{h''} \left[\mathcal{U}_{i(h),j(h)} - \mathcal{L}_{i(h),j(h)}\right]. \tag{14}$$

Proof: In the recursive formula (10), let $\{(i'(k), j'(k))\}_{k=1}^{m_T^P}$ denote the indices for which minima are attained. These indices exist for all k since we assumed that there exists a toll-free path from s to t. In case of nonuniqueness, select the largest index j and the corresponding smallest index i. Actually, we will introduce a slightly different construction for $\{(i'(k), j'(k))\}_{k=1}^{m_T^P}$. Note that for all k, there holds

$$t_l = 0, \quad \forall \ k+1 \leq l \leq j'(k) - 1. \tag{15}$$

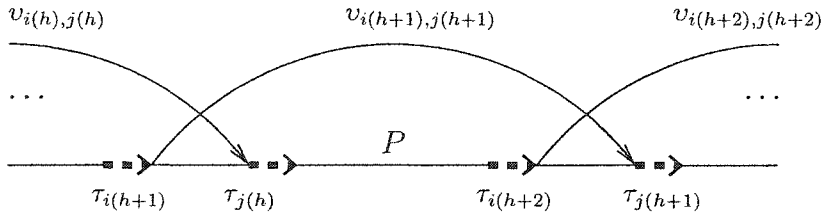

Figure 7: A section of a path P, showing toll arcs τ_k with $k = i(h+1)$, $j(h)$, $i(h+2)$, $j(h+1)$.

Rather than evaluating (10) from t_k to t_{k+1}, we jump from t_k to $t_{j'(k)}$ and set

$$(i'(l),\, j'(l)) = (i'(k),\, j'(k)), \quad \forall\ k+1 \le l \le j'(k)-1 \tag{16}$$

where $(i'(k),\, j'(k))$ is chosen according to the rule stated above. This modification is necessary in order to ensure that $j(h) \le i(h+2)$, as will become apparent shortly.

In view of Theorem 4, which implies that

$$\sum_{k=i(h)+1}^{j(h)-1} T(\tau_k) \le \mathcal{U}_{i(h),j(h)} - \mathcal{L}_{i(h),j(h)}, \tag{17}$$

we say that $v_{i(h),j(h)}$ *covers* the toll arcs τ_k for $i(h)+1 \le k \le j(h)-1$. To derive (14), we look for a subset of $\{v_{i'(k),j'(k)}\}_{k=1}^{m_T^P}$ that covers all toll arcs of P and such that $t_k = 0$ for all arcs τ_k covered by more than one subpath.

We proceed backwards. Select $l_1 = m_T^P$ and recursively compute $l_k = i'(l_{k-1})$ until $l_{q+1} = 0$ for some index q. There follows:

$$j'(l_1) = m_T^P + 1 \quad \text{and} \quad i'(l_q) = 0. \tag{18}$$

Now, reverse the sequence by setting $(i(k),\, j(k)) = (i'(l_{q+1-k}),\, j'(l_{q+1-k}))$, $1 \le k \le q$, to obtain a sequence that satisfies the assumptions of Theorem 4:

1. $i(1) = 0$, $j(q) = m_T^P + 1$: This follows from (18).

2. $i(h) < i(h+1)$: This follows from the construction of the backwards sequence $\{l_i\}_{i=1}^{q}$.

3. $i(h+1) < j(h)$: By contradiction, assume that $j(h) \leq i(h+1)$. This implies that $\tau_{j(h)}$ is not covered by a toll-free subpath. This is impossible by the very construction of $\{(i(k), j(k))\}_{k=1}^{q}$.

4. $j(h) \leq i(h+2)$: By contradiction, assume that $j(h) > i(h+2)$. Then $(i'(l_{q+1-h}), j'(l_{q+1-h})) = (i(h), j(h))$ implies that $i'(l) = i(h)$ for all indices $l_{q+1-h} < l < j'(l_{q+1-h}) = j(h)$ and in particular for index $l = l_{q+1-h-1} = i'(l_{q+1-h-2}) = i(h+2)$ ($< j(h)$ by assumption). This implies the contradiction $i'(l_{q+1-h-1}) = i(h+1) = i(h)$.

5. (13) and (14): By construction, every arc is covered by at least one path. By the preceding inequalities, the only arcs covered by more than one path must belong to the interval $k = i(h+1)+1, \ldots, j(h)-1$, for some index h. By (15), their tolls must be zero, and (13) is satisfied; (14) then follows from (13).

□

Corollary 1 *The toll assignment defined by (10) is optimal for P.*

Proof: For every toll assignment T' consistent with P

$$\begin{aligned} \sum_{k=1}^{m_T^P} T'(\tau_k) &\leq \sum_{h=1}^{q} \left[\mathcal{U}_{i(h),j(h)} - \mathcal{L}_{i(h),j(h)}\right], && \text{by (7) and (13)} \\ &= \sum_{k=1}^{m_T^P} t_k, && \text{by (14).} \end{aligned}$$

□

3.4 Partitioning the set of toll arcs

The approximation algorithm progressively removes toll arcs from the network. It makes use of the following definition.

Definition 3 *A* **descendant** P' *of a valid path* $P \in \mathcal{N}_0$ *is a simple path from* s *to* t *in* $\mathcal{N}_0(P)$ *that traverses the toll arcs of* P *(actually only a subset of them) in the same order as does* P.

Let P be a valid path in $\mathcal{N}_0$. Theorem 5 suggests a way of constructing a descendant of P that stands a chance of achieving a high revenue, whenever P's revenue is low. Let us consider the set

$$v_{i(1),j(1)},\ v_{i(2),j(2)},\ \ldots,\ v_{i(q),j(q)} \tag{19}$$

of toll-free paths such that

$$\begin{aligned} 0 = i(1) < i(2) < j(1) \leq i(3) < j(2) \leq i(4) < j(3) \leq \cdots \\ \cdots \leq i(q) < i(q-1) < j(q) = m_T^P + 1 \end{aligned} \tag{20}$$

and equality (14) holds. If the maximum revenue on P is $B(P)$ then descendants need not be considered, their upper bounds are smaller or equal to $B(P)$. This happens in particular if $q = 1$. We can therefore assume that q is larger than 1.

We now consider two descendants: P_1 contains all $v_{i(h),j(h)}$ with h odd and is composed of arcs of P between them; P_2 is constructed in a similar manner, with even values of the index h. For instance, P_1 may start in s, borrow $v_{i(1),j(1)}$, take

path P between INIT($\tau_{j(1)}$) and TERM($\tau_{i(3)}$), bifurcate on $v_{i(3),j(3)}$, return to P, and so on. Such pattern, which is allowed by (20), is performed by procedure TOLLPARTITION. Note that P_1 and P_2 have no toll arcs in common and that some toll arcs of P belong neither to P_1 nor to P_2. The rationale behind this construction is the relationship between the maximum revenue achievable on P and the upper bounds on the tolls of P_1 and P_2 given by Theorem 5.

Both these descendants are valid paths. If this were not the case, there would exist a subpath $v_{i(h),j(h')}$ ($h < h'$ both odd) such that $\mathcal{U}_{i(h),j(h')}$ is strictly smaller than the length of P_1 between TERM($\tau_{i(h)}$) and INIT($\tau_{j(h')}$) (indeed, a path is valid if and only if the null toll vector is consistent with it). Since this length is equal to $\mathcal{L}_{i(h),j(h')} + \mathcal{T}_{i(h),j(h')}$ by (14) we obtain that the toll assignment on P is not consistent, a contradiction. A similar argument applies to P_2.

3.5 A detailed example

We apply the algorithm to the example of Figure 2. Here $LP = 6$. We start with the shortest path when tolls are set to 0

$$P_0 = (s, v_2, v_3, v_4, t).$$

Because of the toll-free subpaths (s, v_1, v_4) and (v_2, v_5, t), optimal tolls on P_0 are $T_1 = 2$ and $T_2 = 1$. Since $V_{P_0} = 3 < B(P_0) = 6$, the algorithm splits the tolls of P_0

into two subsets, forming the paths

$$P_1 = (s, v_2, v_5, t)$$

and

$$P_2 = (s, v_1, v_4, t).$$

The optimal revenue on P_1 is 5 and that on P_2 is 4. EXPLOREDESCENDANTS finally returns path P_1. Note that in this example, the algorithm returns the optimal path, this is not always the case.

4 Analysis of the approximation factor and running time

4.1 Worst-case guarantee

We shall prove that EXPLOREDESCENDANTS is an $\frac{1}{2}\log m_T + 1$-approximation algorithm for MAXTOLL. The *exact* approximation factor is given by the recursion

$$\alpha(k) = \frac{1}{2} \max_{\substack{i+j \leq k \\ 0<i \leq j<k}} \{1 + \alpha(i) + \alpha(j)\}, \tag{21}$$

with $\alpha(1) = 1$. It can be shown by induction that for all k,

$$\alpha(k) \leq \frac{1}{2}\log k + 1$$

Definition 4 *The maximum revenue V_P induced by path P is* **sufficient** *if*

$$V_P \geq \frac{1}{\alpha(m_T^P)} B(P). \tag{22}$$

Theorem 6 *Let P be a valid path on $\mathcal{N}_0$. If the maximum revenue achievable on P is not sufficient, then either path P_1 or P_2 (say P') returned by* TOLLPARTITION *satisfies*

$$\frac{1}{\alpha(m_T^{P'})}B(P') \geq \frac{1}{\alpha(m_T^{P})}B(P) \tag{23}$$

Proof: Let

$$P = \left(v_{0,1},\ \tau_1,\ v_{1,2},\ \ldots, \tau_{m_T^P},\ v_{m_T^P, m_T^P+1}\right)$$

with corresponding values of $\mathcal{U}_{i,j}$, $\mathcal{L}_{k,l}$ and $\mathcal{T}_{k,l}$ for all i, j, k, l, $k < l$. Similarly, for $i = 1, 2$,

$$P_i = \left(v^i_{0,1},\ \tau^i_1,\ v^i_{1,2},\ \ldots, \tau^i_{m_T^{P_i}},\ v^i_{m_T^{P_i}, m_T^{P_i}+1}\right)$$

with corresponding values of $\mathcal{U}^i_{i,j}$, $\mathcal{L}^i_{k,l}$ and $\mathcal{T}^i_{k,l}$ for all i, j, k, l, $k < l$. Let

$$v_{i(1),j(1)},\ v_{i(2),j(2)}, \ldots,\ v_{i(q),j(q)}$$

be the sequence of toll-free paths obtained from Theorem 5.

For all $h > 1$, one of the paths P_i contains $v_{i(h-1),j(h-1)}$, while the other contains $v_{i(h),j(h)}$. Therefore, the subpath of P between $\text{TERM}(\tau_{i(h)})$ and $\text{INIT}(\tau_{j(h-1)})$ is in neither P_1 nor P_2. The remaining arcs of P belong to either P_1 or P_2. This implies that

$$\begin{aligned}
\mathcal{L}^1_{0,m_T^{P_1}+1} + \mathcal{L}^2_{0,m_T^{P_2}+1} &= \sum_{h=1}^{q} \mathcal{U}_{i(h),j(h)} + \mathcal{L}_{0,m_T^P+1} - \sum_{h=2}^{q} \mathcal{L}_{i(h),j(h-1)} \\
&= \sum_{h=1}^{q} \left[\mathcal{U}_{i(h),j(h)} - \mathcal{L}_{i(h),j(h)}\right] + 2\mathcal{L}_{0,m_T^P+1}
\end{aligned}$$

By Theorem 5, the first term on the right-hand side is equal to $\sum_{k=1}^{m_T^P} t_k$ and is strictly smaller than $B(P)/\alpha(m_T^P)$ by hypothesis. Multiplying by (-1) and adding $2\mathcal{U}_{0,m_T^P+1} = 2\mathcal{U}_{0,m_T^{P_i}+1}$ on both sides, we get

$$B(P_1) + B(P_2) > \left(2 - \frac{1}{\alpha(m_T^P)}\right) B(P).$$

By definition of α:

$$\begin{aligned} & \alpha(m_T^P) \geq \frac{1}{2}[1 + \alpha(m_T^{P_1}) + \alpha(m_T^{P_2})] \\ \Longrightarrow \quad & \left(2 - \frac{1}{\alpha(m_T^P)}\right) \geq \frac{\alpha(m_T^{P_1}) + \alpha(m_T^{P_2})}{\alpha(m_T^P)}. \end{aligned}$$

By substituting in the preceding inequality, we obtain

$$\begin{aligned} & \alpha(m_T^{P_1})\left(\tfrac{1}{\alpha(m_T^{P_1})} B(P_1)\right) + \alpha(m_T^{P_2})\left(\tfrac{1}{\alpha(m_T^{P_2})} B(P_2)\right) \\ & \qquad \geq (\alpha(m_T^{P_1}) + \alpha(m_T^{P_2}))\left(\tfrac{1}{\alpha(m_T^P)} B(P)\right), \end{aligned}$$

which yields the desired result. Indeed, if we had $\frac{1}{\alpha(m_T^{P_i})} B(P_i) < \frac{1}{\alpha(m_T^P)} B(P)$ for $i = 1, 2$, this would imply the opposite inequality.

□

As a corollary, we obtain the main result of this section.

Corollary 2 *Let APP denote the revenue obtained from the application of the procedure* EXPLOREDESCENDANTS *to a path P_0 of shortest length $\mathcal{L}_0$ in $\mathcal{N}_0$. Then,*

$$APP \geq \frac{1}{\alpha(m_T^{P_0})} LP \geq \frac{1}{\alpha(m_T)} OPT. \tag{24}$$

Proof: Let P be a valid path and $(\overline{V}, \overline{T}, \overline{P})$ the output of EXPLOREDESCENDANTS(P). It is sufficient to show, by induction on m_T^P that

$$\overline{V} \geq \frac{1}{\alpha(m_T^P)} B(P). \tag{25}$$

This statement is true if $m_T^P = 1$ since the upper bound $B(P)$ is always achievable on a path with a single toll arc. Now assume that the property holds when the number of toll arcs is less than $m_T^P > 1$. Let $(V_P, T_P) :=$ MAXREV(P). If V_P is sufficient, (25) is satisfied. If V_P is not sufficient then, by Theorem 6, TOLLPARTITION returns a path P' with $\frac{1}{\alpha(m_T^{P'})} B(P') \geq \frac{1}{\alpha(m_T^P)} B(P)$. Since the number of toll arcs in P' is less than that in P, property (25) is satisfied for P', and the preceding inequality implies that (25) is satisfied for P as well.

□

Note that this result applies to the case of negative tolls as well since the upper bound (2) is the same. Indeed, EXPLOREDESCENDANTS allows only nonnegative tolls but it computes a feasible solution with a revenue at least LP/α, where LP is unchanged in the unbounded case.

4.2 Tightness of the approximation

The approximation algorithm determines, in a constructive manner, an upper bound $\alpha(m_T)$ on the ratio OPT/LP. In this section, we show through a family of instances that this bound is tight.

Theorem 7 *Let $\mathcal{I}(m_T)$ denote the set of instances of* MAXTOLL *correponding to a fixed number of toll arcs m_T. Then for all $m_T \geq 1$, the relaxation gap on $\mathcal{I}(m_T)$ is $\alpha(m_T)$, that is*

$$\alpha(m_T) = \max_{I \in \mathcal{I}(m_T)} \left\{ \frac{LP[I]}{OPT[I]} \right\}. \tag{26}$$

Proof: Let us consider a two-node and two-arc network $Z(1)$, with origin s_1 and destination t_1. The first arc (a toll arc) has fixed cost 0 and the second arc (a toll-free arc) has fixed cost 2. We recursively construct a network $Z(k)$ made up of a copy of $Z(\lceil k/2 \rceil)$ and a copy of $Z(\lfloor k/2 \rfloor)$ (if k is even, there are two copies of $Z(k/2)$), and two distinguished nodes, the origin s_k and the destination t_k. These are linked by five toll-free arcs, as illustrated in Figure 8. Strictly speaking, the vertices $s_{\lceil k/2 \rceil}$, $t_{\lceil k/2 \rceil}$, $s_{\lfloor k/2 \rfloor}$, $t_{\lfloor k/2 \rfloor}$ should be re-labeled, otherwise many vertices will share the same label even though we consider them all distinct. The parameters a_k and b_k of $Z(k)$ are set to

$$a_k = 1 + \alpha\Big(\Big\lfloor \frac{k}{2} \Big\rfloor\Big) - \alpha\Big(\Big\lceil \frac{k}{2} \Big\rceil\Big) \quad b_k = 1 - \alpha\Big(\Big\lfloor \frac{k}{2} \Big\rfloor\Big) + \alpha\Big(\Big\lceil \frac{k}{2} \Big\rceil\Big), \tag{27}$$

with $a_1 = b_1 = 1$ and α is defined in (21). Note that a_k, $b_k \geq 0$ because $0 \leq \alpha(\lceil k/2 \rceil) - \alpha(\lfloor k/2 \rfloor) \leq 1$.

It is not difficult to show, by induction, that

$$\alpha(k) = \frac{1}{2}\left[1 + \alpha\Big(\Big\lceil \frac{k}{2} \Big\rceil\Big) + \alpha\Big(\Big\lfloor \frac{k}{2} \Big\rfloor\Big)\right]. \tag{28}$$

Let $LP(k)$ and $OPT(k)$ denote, respectively, the relaxed and optimal revenue values on $Z(k)$. Let $\mathcal{U}(k)$ be the length of the shortest toll-free path from s_k

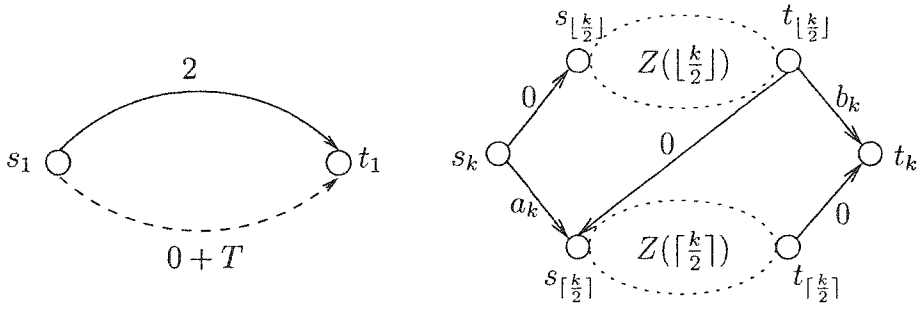

Figure 8: Networks $Z(1)$, left, and $Z(k)$, right. Toll arcs are dashed.

to t_k in $Z(k)$. We claim that $\mathcal{U}(k) = 2\alpha(k)$. Indeed, we have $\mathcal{U}(1) = 2\alpha(1) = 2$ and, assuming that $\mathcal{U}(k') = 2\alpha(k')$ for all $k' < k$, it follows from (28) that:

$$\mathcal{U}(k) = \min\left\{\mathcal{U}\left(\left\lfloor\frac{k}{2}\right\rfloor\right) + b_k, \mathcal{U}\left(\left\lceil\frac{k}{2}\right\rceil\right) + a_k, \mathcal{U}\left(\left\lceil\frac{k}{2}\right\rceil\right) + \mathcal{U}\left(\left\lfloor\frac{k}{2}\right\rfloor\right)\right\} = 2\alpha(k). \quad (29)$$

Clearly, $OPT(1) = LP(1) = 2$ and $LP(1)/OPT(1) = \alpha(1) = 1$. For $k > 1$, the shortest path in $Z(k)$ with tolls set to 0 has length 0. This implies that $LP(k) = \mathcal{U}(k) = 2\alpha(k)$. To conclude, we need to show (by induction) that $OPT(k) = 2$. We consider two cases. If the optimal path on $Z(k)$ goes through the arc joining $t_{\lfloor k/2 \rfloor}$ and $s_{\lceil k/2 \rceil}$, then $OPT(k) \leq 2$ because $a_k + b_k = 2$. Otherwise, the optimal path belongs entirely to $Z(\lfloor k/2 \rfloor)$ or $Z(\lceil k/2 \rceil)$ in which case $OPT(k) = 2$ by the induction hypothesis. [3]

□

[3]It is straightforward to check that the same argument works in the negative tolls case.

We have just proved the optimality of our approximation factor with respect to the upper bound. We can prove more than that: the same family of instances, under a slight modification, can be used to show that our analysis of EXPLORE-DESCENDANTS is tight. For this purpose, we require instances where APP is much smaller than OPT. This is not the case in the examples of Figure 8 where, indeed, $APP = OPT = 2$. However, our aim is attained if we add to $Z(k)$ a toll arc of fixed cost 1 from s^k to t^k. Then $OPT = LP - 1 = 2\alpha(k) - 1$ (the optimal path being the new toll arc) yet APP is still 2 since the algorithm starts with a path of length zero and misses the optimal path of length 1.

4.3 Complexity analysis

MAXTOLL is initialized with a shortest path P_0, which can be computed in $O(n^2)$ time. The toll arcs of the descendants constitute a subset of the toll arcs in P_0, and their traversal order is the same. Therefore, the values $\mathcal{U}_{i,j}$, $i < j$ computed for P_0 can be reused for all descendants, under an appropriate renumbering. This operation is achieved in $O(m_T^{P_0} n^2) = O(m_T n^2)$ time. The time required to evaluate $\mathcal{L}_{i,j}$, $i < j$, as the algorithm proceeds, is much smaller.

Within EXPLOREDESCENDANTS, MAXREV requires at most $O((m_T^P)^3)$ to compute the maximal revenue induced by path P. Based on the indices $\{(i'(k), j'(k))\}_{k=1}^{m_T^P}$ obtained from MAXREV, TOLLPARTITION generates two descendants in $O(m_T^P)$ time. It follows that the running time of EXPLOREDESCENDANTS on a path P is

determined by the recursion

$$\mathbb{T}(m_T^P) = \mathbb{T}(m_T^{P_1}) + \mathbb{T}(m_T^{P_2}) + O((m_T^P)^3) \tag{30}$$

where $m_T^{P_1} + m_T^{P_2} \leq m_T^P$. Therefore, the worst-case complexity, achieved when $m_T^{P_1}$ is always equal to $m_T^P - 1$, is $O((m_T^P)^4)$. The worst-case running-time of the entire algorithm is $O(m_T(m_T^3 + n^2))$.

5 Concluding remarks

Our algorithm can also be applied to the multi-commodity extension of MAX-TOLL considered in [11], where each commodity $k \in \mathcal{K}$ is associated with an origin-destination pair. Given a demand matrix, users solve shortest path problems parameterized by the toll vector T. If distinct tolls T^k could be assigned to distinct commodities, the multi-commodity extension would reduce to a $|\mathcal{K}|$-fold version of the basic problem. Otherwise, the interaction between commodity flows on the arcs of a common transportation network complicates the problem, both from a theoretical and algorithmical point of view. Of course, we can obtain an $O(|\mathcal{K}| \log m_T)$ guarantee by applying MAXTOLL to each commodity separately and then selecting, among the $|\mathcal{K}|$ commodity toll vectors, the one that generates the highest revenue. However bad this bound is, we conjecture that it is tight with respect to the relaxation, which is the sum of the single-commodity bounds, weighted by their respective demands. Indeed, we believe that the instances of Figure 8 can

be generalized to the multi-commodity case.

Other generalizations of MAXTOLL involve capacity constraints and lower bounds on tolls. In the latter case, the relaxation gap becomes infinite for any value of m_T, and our approach fails, as procedure EXPLOREDESCENDANTS becomes irrelevant. A completely different line of attack is then required.

Finally, we raise the following important issue: Can our $\frac{1}{2}\log m_T + 1$ guarantee be improved? Such result would obviously require a tighter upper bound than the one used in this paper.

Acknowledgements

This research was partially supported by NSERC and NATEQ.

References

[1] AUDET, C., HANSEN, P., JAUMARD, B., SAVARD, G. (1997). Links between Linear Bilevel and Mixed 0-1 Programming Problems, *Journal of Optimization Theory and Applications* 93, 273–300.

[2] AUSIELLO, G. ET AL. (1999). *Complexity and Approximation*, Springer, Berlin.

[3] COCCHI, R., SHENKER, S., ESTRIN, D., ZHANG, L. (1993). Pricing in Computer Networks: Motivation, Formulation, and Example, *IEEE/ACM Transactions on Networking* 1, 614–627.

[4] GAREY, M.R., JOHNSON, D.S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H. Freeman, New York.

[5] HANSEN, P., JAUMARD, B., SAVARD, G. (1992). New branch-and-bound rules for linear bilevel programming, *Journal Of Optimization Theory and Applications* 22, 1194–1217.

[6] HEARN, D., RAMANA, M.V. (1998). Solving congestion toll pricing models, dans: *Equilibrium and Advanced Transportation Modelling*, Marcotte and Nguyen (éditeurs), Kluwer, 109–124.

[7] HOCHBAUM, D.S., Éditeur (1997). *Approximation Algorithms for NP-hard Problems*, PWS Publishing Company, Boston.

[8] JEROSLOW, R.G. (1985). A polynomial hierarchy and a simple model for competitive analysis, *Mathematical Programming* 32, 146–164.

[9] KORILIS, Y.A., LAZAR, A.A., ORDA, A. (1997). Achieving Network Optima Using Stackelberg Routing Strategies, *IEEE/ACM Transactions on Networking* 1, 161–173.

[10] LARSSON, T., PATRIKSSON, M. (1998). Side constrained traffic equilibrium models – Traffic management through link tolls, dans: *Equilibrium and Advanced Transportation Modelling*, Marcotte and Nguyen (Éditeurs), Kluwer, 125–151.

[11] LABBÉ, M., MARCOTTE, P., SAVARD, G. (1998). A bilevel model of taxation and its application to optimal highway pricing, *Management Science* 44, 1595–1607.

[12] LABBÉ, M. MARCOTTE, P., SAVARD, G. (1999). On a class of bilevel programs, dans: *Nonlinear Optimization and Related Topics*, Di Pillo and Giannessi (Editors), Kluwer Academic Publishers, 183–206.

[13] LUO, Z.-Q., PANG, J.-S., RALPH, D. (1996). *Mathematical Programming with Equilibrium Constraints*, Cambridge University Press, UK.

[14] MARCOTTE, P. (1986). Network design problem with congestion effects: a case of bilevel programming, *Mathematical Programming* 34, 142–162.

[15] MARCOTTE, P., SAVARD, G., SEMET, F. (2002). A Bilevel Programming Approach to the Travelling Salesman Problem,. *Cahiers du GERAD*, G-2002-73, HEC Montréal.

[16] PIGOU, A.C. (1920). *The Economics of Welfare*, Macmillan and Co., London.

[17] ROUGHGARDEN, T. (2001). Stackelberg Scheduling Strategies, *Proceedings of the 33rd ACM Symposium on Theory of Computing*, Crete, 104–113.

[18] ROUGHGARDEN, T. (2001). Designing Networks for Selfish Users is Hard, *Proceedings of the the 42nd Annual Symposium on Foundations of Computer Science*, Las Vegas, 472–481.

[19] VAZIRANI, V.V. (2001). *Approximation Algorithms*, Springer, Berlin.

[20] VERHOEF, E.T., NIJKAMP, P., RIETVELD, P. (1996). Second-best congestion pricing: the case of an untolled alternative, *Journal of Urban Economics* 40, 279–302.

[21] VICENTE, L., SAVARD, G., JÚDICE, J. (1994). Descent Approaches for Quadratic Bilevel Programming, *Journal of Optimization Theory and Applications* 81, 379–399.